Forschungsgruppe für Geodäsie und Geoinformation
Forschungsgruppe für Ingenieurgeodäsie
HOFER Hannes 0227132

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Kombinierte Indoor/Outdoor Positionierung mit Smartphones

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur/in

im Rahmen des Studiums

Wirtschaftsingenieurwesen Informatik

eingereicht von

Hannes Hofer
Matrikelnummer 0227132

an der

Fakultät für Informatik der Technischen Universität Wien

Betreuung

Betreuer/in: Ao.Univ.Prof. Privatdoz. Dipl.-Ing. Dr.techn. Retscher, Günther

Wien, 22.11.2015 _____ _____

(Unterschrift Verfasser/in) (Unterschrift Betreuer/in)

Erklärung zur Verfassung der Arbeit

HOFER Hannes

Eichenstraße 46

A-1120 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien am 22.11.2015

Kurzfassung

In unserer heutigen mobilen Welt kann nahezu jeder Punkt der Erde innerhalb von einigen Stunden erreicht werden. Mithilfe von GPS-Receivern, die wir in unseren Smartphones finden, ist die Navigation von A nach B sehr einfach geworden. Im urbanen Raum und in Gebäuden ist die Positionierung jedoch oft nicht ausreichend genau oder gar nicht möglich, da die Positionsbestimmung per GPS ungenauer wird, wenn das Signal gestreut, reflektiert oder abgeschwächt wird. Da wir die Welt aber so leicht bereisen können, ist auch die Navigation in einer lokalen, unbekannten Umgebung wichtiger geworden. Neben dem GPS-Receiver verfügen Smartphones auch über Bauteile wie z. B. Bewegungssensoren, digitalen Kompass und WLAN-Modul, die für das Navigieren herangezogen werden können. Das Ziel dieser Diplomarbeit ist es, die Daten der genannten Komponenten zu kombinieren, um die Genauigkeit der Positionierung sowohl im Outdoor- als auch im Indoor-Bereich zu verbessern. Die Grundidee ist, wegentscheidende Positionen intelligent zu wählen und mithilfe des WLAN-Moduls zu erkennen sowie Positionsveränderungen mithilfe des Bewegungssensors und des digitalen Kompasses zu ermitteln. Zur Erfassung der Daten wurde eine App erstellt, die in einem Framework von programmierten MATLAB-Funktionen zur Simulation der Navigation verwendet wurde. So konnte gezeigt werden, dass durch die Kombination der Sensoren und Module, die Positionen für die untersuchten Testläufe, im Mittel auf 2,0 m genau errechnet werden konnten, während die Abweichung der mittels GPS ermittelten Koordinaten bei 16,7 m lag, sofern GPS verfügbar war.

Abstract

In the mobile world we live in today, almost every point on earth can be reached within a few hours. GPS receivers on our smartphones allow for easy navigation from point A to point B. In urban areas and in buildings, the positioning is often insufficiently accurate or not possible at all, because positioning via GPS becomes less accurate if the signal is dispersed, reflected or weakened. The fact that we can easily travel the world has made navigation in a local but unknown environment more important. In addition to the GPS receiver, smartphones are also equipped with components such as motion sensors, a digital compass and a wireless module, which can be used for navigating. The aim of this thesis is to combine the data of these components in order to improve the accuracy of positioning in both outdoor and indoor areas. The idea is to intelligently choose checkpoints of a track and recognize them by using the wireless module. The motion sensor and the digital compass were used to detect changes in positioning. In order to capture the data, an app was developed, and the data was used in a framework of programmed MATLAB functions to simulate navigation. It could be proven that the combination of sensors and modules allows for higher accuracies compared with only using the GPS receiver. When sensors and modules were combined, test results showed an accuracy of 2.0 m on average. In contrast, the accuracy achieved with GPS was 16.7 m on average, if GPS was available.

Begriffsdefinition und Erläuterungen

Im Anschluss sind die wichtigsten Begriffe und deren Bedeutung in der Arbeit definiert. Die Erläuterungen beziehen sich auch auf die Verwendung der Begriffe im Zusammenhang mit der Diplomarbeit.

Begriff	*Erläuterung*
Accelerometers	Siehe Beschleunigungssensor bzw. 2.3.1
App	Android Applikation. In der Arbeit ist das CPS-App gemeint. Siehe auch 3.2
APs	Wireless Access Point – „drahtloser Zugangspunkt“
Beschleunigungssensor	Erfasst die Bewegungen eines Smartphones. Siehe 2.3.1
DAAS	Datenerfassungs- und Analysesystem. Siehe *Kapitel 3*
digitaler Kompass	Azimut-Wert des Orientierungssensors.
euklidischer Abstand	Abstand zwischen zwei n-dimensionalen Vektoren im euklidischen Raum.
GPS	Global-Positioning-System
GPS-Receiver	GPS Empfänger des Smartphones
GSM-Netz	Global System for Mobile Communications – Standard für Mobilfunknetze.
KNN-Algorithmus	k-Nearest-Neighbor-Algorithmus. Siehe 2.2.2
Magnefeldsensor	Smartphonesensor, der Magnetische Flussdichte misst. Siehe 2.3.2
MATLAB-Framework	Sammlung von für die Arbeit erstellten MATLAB Funktionen zur Simulation.
NN-Algorithmus	Nearest-Neighbor-Algorithmus WLAN-Fingerprinting -"nächste-Nachbarn-Algorithmus". Siehe 2.2.1
Orientierungssensor	Ermittelt aus Beschleunigungssensor und Magnefeldsensor die Orientierung des Smartphones im Raum. Siehe 2.3.3
RSS	Received Signal Strength. Siehe auch Signalstärke.
RSSI	Received Signal Strength Indication - RSS Werte mit NaN bzw. -101dBm.
RSSI-Vektor	RSSI-Werte eines WLAN-Scans als Vektor dargestellt. Siehe 4.3.1
Signalstärke	Leistungspegel des WLAN-Signals (in dBm Dezibel Milliwatt). Siehe auch RSS.
WLAN	Wireless Local Area Network – „drahtloses lokales Netzwerk“
WLAN-Modul	Sender und Empfänger für das WLAN.
WLAN-Scan	Datensatz eines Scans nach WLAN-APs. Siehe Structure Variable WIFISCANS.
S2	Kurzbezeichnung für das Testgerät Samsung Galaxy S2. Siehe 8.2
HTC	Kurzbezeichnung für das Testgerät HTC EVO 3D. Siehe 8.2
TL	Testlauf – In der Structure Variable `TESTRUNS` werden WLAN-Scans und Sensordaten erfasst. Siehe auch 6.1 und 8.5
Testgebiet	Es wurden in zwei Testgebieten Daten erfasst. Diese sind in 8.3 definiert.
EI-Testgebiet	Befindet sich vor und im Neuen Elektrotechnik Institutsgebäude der TU Wien.
Heim-Testgebiet	Befindet sich um den Häuserblock des Studentenheims in der Eichenstraße 46.

1 Einleitung

1.1 Problemstellung

Präzise Navigation war seit jeher eine große Herausforderung. Einen zentralen Fortschritt stellte das vom US-Verteidigungsministerium entwickelte Global-Positioning-System (GPS) dar. Durch die Laufzeitenmessung ausgesendeter Satellitensignale ist eine genaue und schnelle Bestimmung der Position auf wenige Meter möglich geworden. [50]

Leistungsfähige GPS-Empfänger sind heute durch die Nutzung von Smartphones einer breiten Masse zugänglich. Wenn auch die in den modernen Smartphones verbauten GPS-Receiver immer leistungsfähiger werden, ist die Lokalisierung mittels GPS nicht überall gleich zuverlässig. Im urbanen Raum können Reflexion oder Streuung der Signale zu Abweichungen bis zu 100m führen. Abschattung bzw. komplette Abblockung des GPS-Signals machen eine Standortbestimmung mit GPS gar unmöglich. [87; 88]

Innerhalb von Gebäuden ist die Zuverlässigkeit von GPS am geringsten. Um jedoch durch große Gebäudekomplexe wie z. B. Flughäfen, Amtsgebäude oder Einkaufszentren mithilfe eines Smartphones navigieren zu können, muss der Standort auf wenige Meter bekannt sein.

Moderne Handys bieten eine Vielzahl von Bauteilen, welche zur Positionsbestimmung nutzbar sind. Zum einen kann das WLAN-Modul zur Ortsbestimmung herangezogen werden [39]. Beim sogenannten WLAN-Fingerprinting werden die Signalstärken von mehreren WLAN-Access-Points (APs) gemessen und einer Position zugeordnet. Des Weiteren können die Daten des Beschleunigungssensors bzw. Accelerometers, Magnetfeldsensors und Barometers ausgewertet werden, um Veränderungen in der Position wahrzunehmen. In [70] wird ein Ansatz beschrieben, der den Beschleunigungssensor und den digitalen Kompass benutzt, um Schritte und deren Richtung zu erkennen. Es ist auch möglich, über den Beschleunigungssensor mittels Integration der Sensordaten auf die zurückgelegte Strecke zu schließen [62].

Zu Beginn der Arbeit stand die Frage, wie die Positionierung mittels GPS, WLAN-Fingerprinting und Sensoren sinnvoll kombiniert werden kann. So entwickelte sich eine Idee für einen Ansatz, der bestimmte Kontrollpunkte verwendet und durch Erfassung der Positionsveränderung zwischen diesen Punkten eine durchgehende Navigation möglich macht. Bei diesem Ansatz sollen mittels WLAN-Fingerprinting diese Kontrollpunkte erkannt und die Veränderung der Position mithilfe der Sensoren festgestellt werden.

1.2 Ansatz und Zielsetzungen

Wie zuvor angeführt, ist es möglich, mit dem WLAN-Modul eines Handys die Signalstärken von Access Points zu ermitteln und so auf eine bestimmte Position rückzuschließen [38]. Ein weitverbreiteter Ansatz ist das WLAN-Fingerprinting [22], bei welchem an Referenzpunkten die Signalstärken der vorhandenen APs gemessen und in einer Datenbank (DB) gespeichert werden. Zur Lokalisation werden die APs erneut Vorort gescannt und mit den Daten in der Fingerprinting-DB verglichen. Aus der DB wird dann jene Position gewählt, welche den geringsten euklidischen Abstand zu den Signalstärken des aktuellen Scans aufweist. In [71] wurde ein Fingerprint-Algorithmus in den Räumlichkeiten der “Engineering Geodesy Group” an der TU Wien bereits getestet. Dabei konnte nach Betrachtung aller Tests in fast 50% der

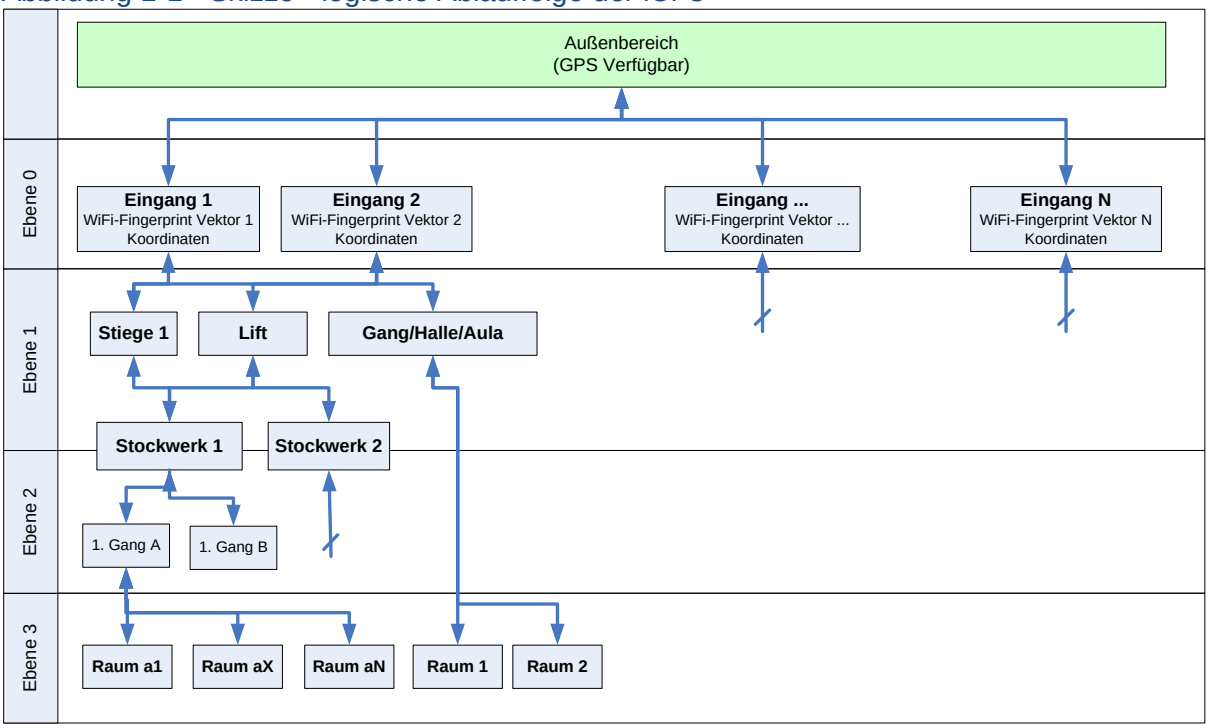
Fälle entweder der korrekte Raum oder zumindest der Nachbarraum erkannt werden. Das ist allerdings für eine zuverlässige Navigation durch das Gebäude noch zu ungenau.

Bei dem in [70] vorgestellten Ansatz werden der Beschleunigungssensor und der digitale Kompass kombiniert, um Schritte und deren Richtung zu erkennen. Mit dieser Kombination konnten gute Ergebnisse mit wenigen Metern Abweichung erzielt werden. Auf diese Art lassen sich allerdings nur Veränderungen in der Position bestimmen, wobei bei jeder Veränderung eine weitere, wenn auch nur kleine, Abweichung. Das liegt daran, dass Sensoren und Algorithmen nur bis zu einem bestimmten Maß zuverlässig arbeiten. Da sich die Fehler kumulieren, wird bei einer längeren Strecke der Fehler irgendwann zu groß werden. Neben der Möglichkeit, Schritte zu zählen [46], können die Daten des Beschleunigungssensors auch intergiert werden, um so auf die zurückgelegte Strecke zu schließen [62]. Ein Ziel der Arbeit wird es sein, folgendes zu untersuchen:

Wie können GPS, WLAN-Fingerprinting und Schritterkennung kombiniert werden?

Wie bereits erwähnt ist die Verwertbarkeit von GPS-Signalen mit einem Smartphone in Gebäuden meist nicht bzw. nur mit großen Abweichungen möglich. Aber gerade in einem Gebäude ist es wichtig, die Position auf wenige Meter bestimmen zu können. Führt man sich vor Augen, wie man in einen bestimmten Raum eines Gebäudes gelangt, so müssen gewisse Punkte passiert werden, um dort hinzugelangen. Zuerst wird ein Gebäudeeingang gewählt. In vielen Fällen wird man dort eine Aula oder Ähnliches betreten. Um in einen anderen Stock zu gelangen, muss entweder der Lift oder die Treppe benutzt werden. Bevor man den Raum betreten kann, muss erst ein Gang gewählt und abgegangen werden. Diese Punkte stehen in Abhängigkeit zueinander und müssen nach einer logischen Abfolge abgeschritten werden, um von A nach B zu gelangen. Türen, Stiegen und Gänge usw. können als Knotenpunkte oder Vektoren eines Graphen betrachtet werden.

Abbildung 1-1 - Skizze - logische Ablauffolge der iCPs



Je weiter man in das Gebäude vordringt, eine desto tiefere Ebene des Graphen erreicht man. Aus diesen Ebenen lassen sich Kategorien ableiten, die bereits eine einfache logische

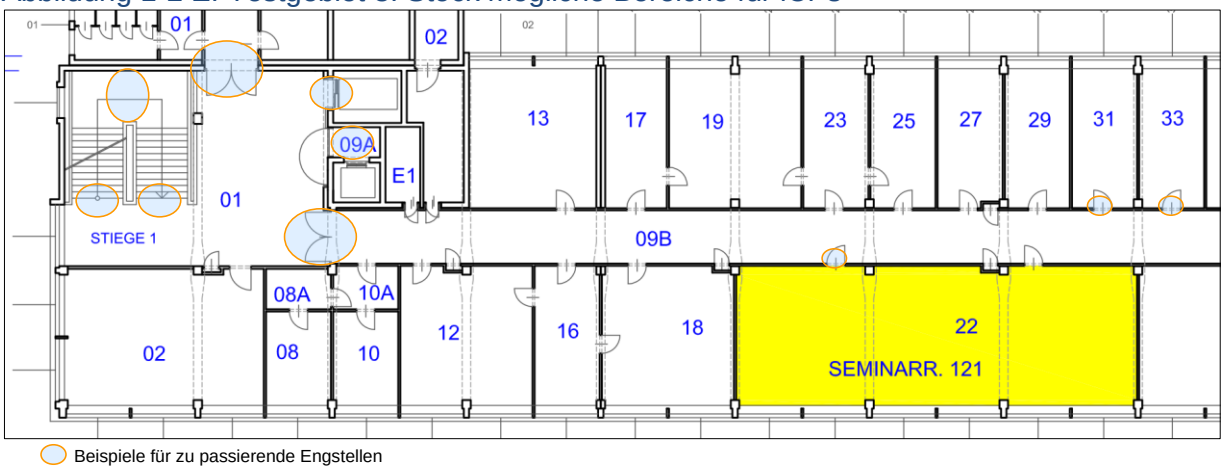
Abfolge beschreiben. In Abbildung 1-1 ist eine mögliche Einteilung als Skizze dargestellt. Bei diesem Beispiel befinden sich alle Eingänge auf Ebene 0. Auf der letzten Ebene befinden sich alle Räume. Die Ebene steht also dafür, wie weit man bereits in ein Gebäude vorgedrungen ist. Dies ist nur eine mögliche Darstellungsform, die die Zusammenhänge von Gängen und Räumen in einem Gebäude beschreibt. Folgt man von Beginn an dem Schema in Abbildung 1-1, so steht immer nur eine bestimmte Auswahl von Punkten zur Verfügung.

Um diese logischen Zusammenhänge nutzen zu können, ist es aber nötig, bestimmte Knotenpunkte in einem Gebäude zu erkennen. In der vorliegenden Arbeit sollte untersucht werden, ob diese wichtigen Punkte mit einem auf WLAN-Fingerprinting basierenden Algorithmus erkannt werden können.

Bei der Wahl dieser Check-Points sind die Strukturen des Gebäudes nutzbar. Denn in Bauwerken müssen immer wieder unterschiedliche Engpässe passiert werden, um voranzukommen. Diese Engpässe bieten oft auch Strukturen (wie Wände, Ecken oder Türen), welche die Signalstärken zu den vorhandenen APs beeinflussen können. Die Idee ist es also, nicht wie üblich einen Raster von Positionen zu erstellen, an denen die Signalstärken der APs gemessen werden, sondern die Positionen bzw. Punkte auf intelligente Weise so zu wählen, dass diese als gut unterscheidbare Knotenpunkte verwendet werden können. Diese „intelligente-Check-Points“ (iCPs) sollen dann für die Navigation durch das Gebäude genutzt werden.

Abbildung 1-2 stellt einen Teil des Gebäudeplans des EI-Testgebietes dar. Darauf sind interessante Bereiche für die Wahl von iCPs markiert. Dabei handelt es sich um Stellen, die passiert werden müssen und zu neuen Abschnitten im Gebäude führen. Diese iCPs wären also in zweifacher Weise "intelligent" – einerseits durch die Wahl ihres Standpunktes und anderseits durch die Logik ihrer möglichen Ablaufreihenfolge.

Abbildung 1-2 EI-Testgebiet 3. Stock mögliche Bereiche für iCPs



Um zu analysieren, ob dieser Ansatz zu realisieren ist, wurden folgende Punkte untersucht:

Wie sind die iCPs zu wählen, um sich gut unterscheiden zu lassen?

Kann erkannt werden, wann und ob ein Punkt passiert wurde?

Wie kann die Position zwischen diesen Punkten bestimmt werden?

Mit den genannten Fragen wird sich diese Arbeit in weiterer Folge auseinandersetzen. Da die iCPs mit einem WLAN-Fingerprinting-Ansatz erkannt und unterschieden werden sollen, wird auf die Entwicklung dieses Ansatzes besonderer Wert gelegt. Um diesen Ansatz analysieren

und testen zu können, wurde ein System geschaffen, das es möglich macht, Ideen und Ansätze schnell zu testen und miteinander zu vergleichen. Es wurde eine Umgebung entwickelt, die reale Daten von Sensoren und WLAN-Modul aufzeichnet und für Simulationen bzw. Tests verwendet. Dafür wurde eine Android-Applikation zur Datenaufzeichnung entwickelt (CPS-App 3.2) und in MATLAB eine Simulationsumgebung (MWF 0), die diese Daten importiert und analysiert, angefertigt.

Als Testgebiet dienten die Räumlichkeiten der Forschungsgruppe für Ingenieurgeodäsie bzw. dem Department für Geodäsie und Geoinformation an der TU Wien und der Häuserblock um das Studentenwohnheim ÖJAB-Haus Meidling in der Eichenstraße 46 (1120 Wien). Die Testgebiete und Referenzpunkte sind in „*Anhang A - Code, Testgeräte, Testgebiete und Testläufe*“ in Abschnitt 8.3 beschrieben.

1.3 Überblick über den Inhalt

In **Kapitel 2** werden die technischen Grundlagen erläutert, um die Zusammenhänge der in weiterer Folge beschriebenen Algorithmen deutlich zu machen. Beschrieben werden in diesem Kapitel unterschiedliche Methoden zur Positionsbestimmung mit Smartphones. Im Detail wird dabei auf das WLAN-Fingerprinting eingegangen und erläutert, wie die Algorithmen funktionieren. Weiter werden die verwendeten Sensoren in den Geräten vorgestellt und deren Funktionsweise erklärt.

Im **dritten Kapitel** wird das entwickelte System zur Datenerfassung beschrieben. Dieses besteht aus einer speziellen für die Arbeit entwickelten Android Applikation (**CPS-App**) und einem MATLAB-Framework (**MWF**).

Mit dem **CPS-App** können WLAN-Scans durchgeführt und einer Positions-ID und einer Richtung zugeordnet werden. Außerdem bietet es die Möglichkeit, Sensordaten parallel mit einer Reihe von WLAN-Scans aufzuzeichnen. All diese Daten können in MATLAB importiert und dort analysiert werden.

Das **MWF** bietet eine Reihe von Funktionen für den Import und die Auswertung von WLAN-Scans. Die aufgezeichneten Sensordaten können verwendet werden, um Teststrecken abzugehen und die entstandenen Daten als Testläufe zu speichern. Aus den Testläufen können mit verschiedenen Ansätzen zurückgelegte Wege berechnet und mit deren Referenzstrecken verglichen werden. Diese Wege sind in den Karten der jeweiligen Testgebiete darstellbar.

Kapitel 4 erläutert die Tests und die Ergebnisse zum WLAN-Fingerprinting. Beschrieben wird, wie die für die Analyse verwendeten WLAN-Scans erhoben und bestimmten Positionen zugeordnet wurden. Analysiert wurde, ob sich für eine WLAN-Fingerprinting-Datenbank das arithmetische Mittel oder der Median besser als Mittelwert eignet und wie mit APs umgangen wird, die an einer Position nicht immer erreichbar sind. Dafür wurden verschiedene Methoden und Szenarien getestet.

Das **fünfte Kapitel** führt die „*intelligente Check Points“* (iCPs) ein. Dabei werden Metadaten genutzt, um das WLAN-Fingerprinting zu unterstützen. Verwendet werden die Orientierungsinformationen der Smartphones, um das Fingerprinting richtungsabhängig zu betreiben. Außerdem wird das Fingerprinting anhand einer logischen sequenziellen Abfolge analysiert, die von Zusammenhängen zwischen den iCPs hergeleitet wurde. Die erzielten Ergebnisse werden mit denen von **Kapitel 4** verglichen.

Kapitel 6 zeigt die schrittweise Entwicklung eines auf GPS-Daten, Schritterkennung, Richtungserkennung und iCPs basierenden Algorithmus. Im MFW wird der iCP-INS-Algorithmus genutzt, um die aufgezeichneten Testläufe nachzuverfolgen und in Karten einzuzeichnen. Präsentiert wird dabei, wie iCPs mit Metainformationen genutzt werden können, um die ermittelte Strecke präziser zu machen. Verglichen wird der entwickelte Algorithmus mit den GPS-Aufzeichnungen und dem INS-Algorithmus, der nur auf Erkennung der Positionsveränderung basiert.

Kapitel 7 fasst die Ergebnisse der Arbeit zusammen und reflektiert Schwächen und Stärken des Ansatzes.

Anhang A definiert die Speicherorte der erfassten Daten und beschreibt die verwendeten Testgeräte, Testgebiete, deren Referenzpunkte und iCPs sowie die ausgewählten Teststrecken und durchgeführten Testläufe.

Anhang B enthält die Ergebnisse der durchgeführten Analysen der entwickelten Algorithmen zur Erkennung von Schritten und Positionsveränderung.

In **Anhang C** sind die Ergebnisse des vorgeschlagenen iCP-INS-Algorithmus angeführt, welcher WLAN, GPS- und Sensordaten verbindet.

2 Technische Grundlagen

In diesem Kapitel werden technische Grundlagen für die in der Arbeit relevante Hardware und Software beschrieben. Angeführt sind jene Bauteile von Smartphones, die zur Positionierung verwendet werden können. Der erste Abschnitt gibt einen Überblick über Techniken und Ansätze, welche auf die Hardware von Smartphones anwendbar sind.

Im zweiten Teil wird auf die Positionierung mittels WLAN-Fingerprinting ausführlicher eingegangen. Zuletzt werden die Funktionsprinzipien der verwendeten Sensoren erläutert.

In der Zusammenfassung wird Resümee gezogen und erklärt, welche der Techniken sich für den Ansatz mit iCP eignen.

2.1 Überblick an Technologien zur Positionierung

Moderne Smartphones verfügen über eine Reihe von Sensoren und Modulen, die zur Positionierung verwendet werden können. Auch wenn bei den meisten die Positionierung nicht deren Primärfunktion ist, kann mit unterschiedlichen Techniken auf eine Position rückgeschlossen werden. Durch die Kombination verschiedener erfassten Daten können mit verschiedenen Techniken hohe Genauigkeiten erreicht werden.

Dieser Abschnitt liefert einen Überblick über Techniken und Ansätze zur Positionierung, die auf Smartphones anwendbar sind. Als Erstes werden die wesentlichen Möglichkeiten zur Positionsbestimmung mithilfe des klassischen GSM-Netzes erklärt. Weiter wird auf die GPS Positionierung mit Mobiltelefonen eingegangen. Im Anschluss werden generelle in WLAN Netzen anwendbare Techniken erklärt. Im vierten Abschnitt werden Methoden erklärt, die auf der Erkennung der Positionsänderung basieren. Zur Vervollständigung wird im letzten Abschnitt auf weitere Ansätze verwiesen, die mit moderneren dazu fähigen Smartphones möglich sind.

2.1.1 Positionierung im Mobilfunknetz

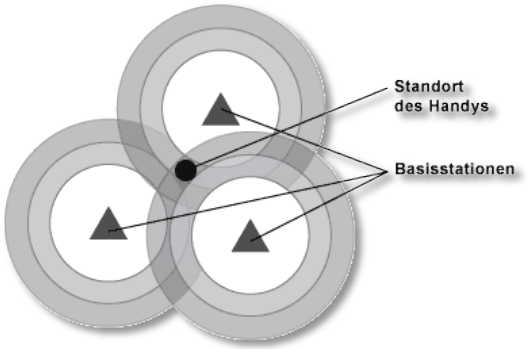
Es ist möglich, das Mobilfunknetz zur Positionsbestimmung zu nutzen. Je nach Umgebung und Methode kann die Position mit unterschiedlicher Genauigkeit bestimmt werden. Die Spanne reicht dabei von unter 100 Metern bis zu mehreren Kilometern. Dieser Abschnitt fasst die Literatur in [30; 64; 79] zusammen.

Die **Cell-ID-** oder **Cell-of-Origin-Methode (COO)** ist eine einfache Möglichkeit, die Position in einem GSM-Netz abzuschätzen. Jedes eingewählte Telefon ist bei einem Mobilfunkmast angemeldet, der seine Position mitsendet. Das ganze Mobilfunknetz ist in Zellen aufgebaut, in deren Zentren jeweils ein Funkmast bzw. eine Basisstation stehen. Je nachdem, wie groß die Zelle einer Basisstation ist, an der das Gerät angemeldet ist, kann über die Cell-ID auf eine ungefähre Position geschlossen werden. Die Dichte bzw. Größe der Zellen ist abhängig von der Anzahl der Geräte, die bedient werden müssen. In ländlichen Regionen stehen die Masten meist einige Kilometer auseinander. Um alle Geräte in dicht besiedelten Städten bedienen zu können, stehen hingegen die Basisstationen meist nur wenige 100 m auseinander. Je kleiner die Zellen sind, umso genauer kann die Bestimmung der Position erfolgen.

Wenn zu zwei weiteren Basisstationen Kontakt besteht, kann eine ungefähre Position bestimmt werden. Außerdem kann auch genutzt werden, dass Mobilfunkzellen in drei Sektoren eingeteilt werden. Wie in Abbildung 2-1 zu sehen ist, kann aus der Schnittmenge

der Sektoren der Bereich ermittelt werden, in dem sich das Gerät befindet. So kann die Position noch etwas genauer bestimmt werden.

Abbildung 2-1 Cell-of-Origin-Methode (COO) [64]



Verbessert kann die Methode werden, indem diese mit dem **Timing Advance (TA)** Parameter kombiniert wird. Dieser ist für die Synchronisierung von Handy und Basisstation gedacht. Der Parameter sagt etwas über die Signallaufzeit aus. Er ist in 64 Stufen zu je 3,7 µs (entspricht ca. 1,1 km) eingeteilt und kann zur Abschätzung der Entfernung zur Basisstation genutzt werden.

Beim **EOTD-Verfahren** (Enhanced Observed Time Difference) wird die Laufzeit direkt gemessen. Benötigt werden mehrere EOTD-fähige Basisstationen und dafür ausgelegte Endgeräte. Mit dem Verfahren sind Genauigkeiten bis zu 30 Meter möglich, jedoch ist dies mit höheren Kosten bei den Basisstationen und Endgeräten verbunden.

Für die Indoor-Positionierung sind diese Verfahren nicht geeignet, da sie zu ungenau sind. Die Information über die ungefähre Position könnte aber z. B. dazu genutzt werden, um in der Nähe befindlichen Gebäude, für die eine WLAN-Fingerprinting-Datenbank existiert, zu finden. Diese Datenbanken könnten im Hintergrund heruntergeladen werden und sofort bereitstehen, wenn diese benötigt werden. Im Gegensatz zu GPS ist das Mobilnetz in einem Gebäude fast immer erreichbar und die Position lässt sich nahezu ohne Zeitverzögerung ermitteln.

2.1.2 Global Positioning System - GPS

Das zurzeit aus 31 Satelliten [50] bestehende Navigationssystem wurde ursprünglich für das US-Militär entwickelt und ist seit Anfang der 1980er Jahre für die zivile Nutzung freigegeben. Durch die im Jahre 2000 abgeschaffte *„Selective Availability",* wo den Signalen künstlich ein Rauschen hinzugefügt wurden, wurden auf einige Meter genaue Positionierungen ermöglicht [72]. Mit präzisen GPS-Receiver und Differential -GPS [45] sind sogar Genauigkeiten im Bereich von wenigen Zentimetern möglich.

Die in den heutigen Handys verbauten Receiver erreichen je nach Umgebung bestenfalls eine Genauigkeit von einigen Metern. Wenn keine weiteren Informationen bekannt sind, werden zumindest 3 Satelliten benötigt, um Breitengrad (lat. latitude) und Längengrad (lon. longitude) zu bestimmen. Mit einem zusätzlichen Satelliten wird auch die Bestimmung der Seehöhe (altitude) möglich. Mit A-GPS [76] kann die Dauer, bis der erste GPS Wert (TTFF – time to first fix“) ermittelt wird, verkürzt werden. Nötig ist dazu eine Netzverbindung durch das Smartphone, womit die Daten der Satellitenpositionen aus dem Netzwerk geladen werden können und nicht über die Signale der Satelliten vom Gerät berechnet werden müssen.

Zudem kann die Genauigkeit der berechneten Position durch die Kombination mit der im Mobilfunknetz ermittelten Position erhöht werden. [70]

GPS Signale werden im urbanen Raum oft verfälscht. Im Indoor-Bereich sind diese entweder zu ungenau oder mit den in den Smartphones verbauten Receivern gar nicht erreichbar. Damit ist GPS für präzise Indoor-Navigation nicht geeignet.

2.1.3 Techniken zur Positionsbestimmung mit WLAN

Das in der Einleitung erwähnte WLAN-Fingerprinting gehört zu den Techniken, bei denen aus der empfangenen Signalstärke (RSS - Received Signal Strength) eine Position abgeleitet wird. Diese auf **RSS-basierende-Techniken** (siehe dazu [24]) haben sich gegenüber traditionellen Positionierungsmethoden auf Grund ihre Einfachheit und den guten Ergebnissen als geeignete Alternative zur herkömmlichen Methoden erwiesen. Auf WLAN-Fingerprinting wird in Abschnitt 2.2 im Detail eingegangen.

Klassisch wird die Laufzeit gemessen, um die zurückgelegte Strecke eines Signals aufgrund der benötigten Zeit zu berechnen. Da es sich bei einem Funksignal um eine elektromagnetische Welle handelt, die eine konstante Geschwindigkeit (Lichtgeschwindigkeit) hat, kann daraus die Länge der zurückgelegten Strecke ermittelt werden.

Bei einem **Time-of-Arrival-**Ansatz **(TOA)** müssen gesendete Signale mit einem Zeitstempel versehen werden, um deren Laufzeit zu ermitteln. Um die Position mittels Triangulation eindeutig bestimmen zu können, werden Signale von mindesten 3 WLAN-Basisstationen bzw. APs zu einem Receiver benötigt. Dabei müssen die Uhren von Sendern und Empfängern mit hoher Genauigkeit synchronisiert sein, denn schon kleinste Abweichungen führen zu großen Fehlern. Außerdem müssen die Positionen der APs bekannt sein bzw. vermessen werden. Mit diesem Ansatz werden zwar hohe Genauigkeiten erzielt, aber aufgrund der Komplexität und spezifischer Anforderungen an das System ist dieser nicht immer ohne weiteres einsetzbar. [1; 40]

Neben der Möglichkeit, die Laufzeit direkt zu messen, kann diese auch indirekt abgeleitet werden. Bei einem **Time-Difference-of-Arrival**-Ansatz **(TDOA**) [32] werden jeweils die Signale gleichzeitig von den Basisstationen ausgesendet. Aus den unterschiedlichen Ankunftszeiten und den bekannten Koordinaten der APs, können die Entfernungen zu den APs in Abhängigkeit zueinander gesetzt werden. Daraus lässt sich dann auch die Position bestimmen. Der Vorteil ist, dass nur die WLAN-Basisstationen miteinander synchronisiert werden müssen. Bereits bestehende Netze müssenden jedoch auch modifiziert bzw. angepasst werden.

2.1.4 Inertial Navigation System (INS)

In der Einleitung wurde bereits erklärt, dass mithilfe der Sensoren eines Smartphones auf die Änderung der Position geschlossen werden kann. Dabei wird eine bekannte Startposition benötigt, um z. B. mit den Daten von Beschleunigungssensor und Magnetfeldsensor die Änderung der Position zu errechnen. Dies wird in der Literatur als „inertial navigation“ bezeichnet. Eine ausführliche Einführung dazu ist in [86] zu finden. Die nötigen technischen Grundlagen zur Funktionsweise der verwendeten Sensoren sind in Abschnitt 2.3 beschrieben. Die für die Arbeit verwendeten INS-Algorithmen werden in Abschnitt 6.4 im Detail erklärt.

2.1.5 Sonstige Ansätze

Neben den beschriebenen Ansätzen gibt es noch eine Reihe anderer Techniken, die Potenzial zu Positionierung haben. So kann die **Bluetooth-Technology** benutzt werden, um auf die Position eines bluetoothfähigen Geräts zu schließen. Bluetooth eignet sich allerdings nur für kurze Distanzen, hat aber den Vorteil, dass es wenig Energie verbraucht. Ein Überblick zur Positionierung mittels Bluetooth ist in [21] zu finden.

Bei einem auf **Radio Frequency Identification (RFID)** basierenden Ansatz [31] werden an definierten Stellen RFID-Chips angebracht. Eine Position kann dann mit einem entsprechenden Lesegerät auf kurze Distanzen ermittelt werden. Neuere Handys wie z. B. das HTC One X oder das Samsung Galaxy S3 haben Lesegeräte verbaut [18], welche Chips, die mit NFC (**Near Field Communication)** Standard arbeiten, auslesen können. Der NFC-Standard baut auf der RFID Technologie auf, das Auslesen ist jedoch nur auf sehr kurzen Distanzen möglich. Diese Technologie könnte z B. verwendet werden, um Ausgangs- bzw. Startpositionen in Gebäuden zu definieren, von denen aus navigiert werden kann.

Zusätzlich existieren auf die Nutzung von Ultraschall und Infrarot [39] basierende Ansätze.

2.2 Grundlagen zum WLAN-Fingerprinting

Wie schon in Kapitel 2.1.3 beschrieben, ist das WLAN-Fingerprinting ein auf Erfassung der RSS basierender Ansatz. Dabei werden Positionen und WLAN-Scans über eine Fingerprinting-Datenbank in Verbindung gebracht. Aus den gemessenen RSS-Werten wird ein sogenannter RSSI-Wert (Received Signal Strength Indication) formuliert. Da eine exakte Definition für diese RSSI-Werte nicht existiert, können sich diese von System zu System unterscheiden. Beim WLAN-Fingerprinting kann zwischen deterministischen und statistischen Algorithmen unterschieden werden [39].

Für die Arbeit wurde ein deterministischer Ansatz verwendet. Die Grundlagen zu diesen Algorithmen sind in den anschließenden zwei Abschnitten erläutert. Am Ende wird noch auf einen statistischen Ansatz verwiesen.

2.2.1 Nearest Neighbor Algorithmus

In [19] wird ein Nearest Neighbor (NN) Ansatz beschrieben. Bei diesem deterministischen Algorithmus werden WLAN-Scans mit der dazugehörigen Position in einer Datenbank gespeichert. Durch die Ermittlung von Mittelwerten wird jeder gemessenen Position ein RSSI-Vektor zugewiesen. Diese Zuweisung wird in der Arbeit als Kalibrierungsphase bzw. Trainingsphase bezeichnet. Am Ende dieser Phase verfügt jede Position i über einen Vektor $RSSI_i$ (F. 2-2). Dieser Vektor enthält die empfangenen Signalstärken Si_{APx} (RSS-Werte) der erfassten Access-Points $AP1$ bis APn. In der Trainingsphase kann eine beliebige Anzahl von Positionen gemessen und gespeichert werden. Meist erfolgt eine rasterhafte Anordnung dieser Positionen, wie dies in [71] bereits für das EI-Testgebiet im 3. Stock durchgeführt wurde.

Zur Positionierung kann an einer beliebigen Stelle ein WLAN-Scan durchgeführt werden. Daraus wird der $RSSI_m$-Vektor (F. 2-1) erstellt und die euklidischen Abstände [85] $d_{m,i}$ (F. 2-3) zu allen Vektoren $RSSI_i$ in der Datenbank berechnet. Als Position wird jene gewählt, die den kleinsten Abstand $d_{m,i}$ hat. Gewählt wird also jene Position, deren RSSI-Vektor den nächsten Nachbarn (Nearest Neighbor NN) zum Vektor $RSSI_m$ darstellt.

$$RSSI_m = [Sm_{AP1}, Sm_{AP2}, \dots, Sm_{APn}] \dots RSSI\text{-}Vektor\ zur\ Positionierung \quad (F.\ 2\text{-}1)$$

$$RSSI_i = [Si_{AP1}, Si_{AP2}, \dots, Si_{APn}] \dots RSSI\text{-}Vektoren\ i\ der\ Kalibrierungsphase \quad (F.\ 2\text{-}2)$$

$$d_{m,i} = \sqrt{(Sm_{AP1} - Si_{AP1})^2 + (Sm_{AP2} - Si_{AP2})^2 + \dots + (Sm_{APn} - Si_{APn})^2} \quad (F.\ 2\text{-}3)$$

2.2.2 Erweiterungen des Nearest Neighbor Algorithmus

Eine Erweiterung des NN-Ansatzes ist der **K-Nearest-Neighbor (KNN)** Algorithmus. Bei diesem Algorithmus werden k Positionen, welche die minimalen euklidischen Abstände $d_{m,i}$ zu den RSS-Vektoren ergeben, aus der Datenbank ausgewählt. Aus den Koordinaten dieser Positionen wird die mittlere Position bestimmt. Dabei muss k≥2 sein, ansonsten würde es sich um einen NN-Algorithmus handeln. Durch die Mittelung von mehreren Referenzpositionen lassen sich mit dem KNN- Algorithmus exaktere Ergebnisse erzielen. [44]

Beim **Weighted KNN** werden ebenfalls die k minimalen $d_{m,i}$ ermittelt. Jedoch werden diese je nach der Größe der euklidischen Abstände verschieden stark gewichtet. [39]

2.2.3 Bayesian Algorithmus

Ein Problem ist, dass die RSS-Werte der APs an einer Position meist Schwankungen unterlegen sind. Daher werden beim Bayesian Algorithmus die gemessenen RSSI-Vektoren benutzt, um ein statistisches Modell zu erstellen, das aus dem Satz von Bayes [20] abgeleitet wurde. Bei der Positionierung wird für einen erfassten $RSSI_m$-Vektor die Wahrscheinlichkeit der Zugehörigkeit zu jeder Position in der Datenbank berechnet. Dem $RSSI_m$-Vektor wird jene Position mit der höchsten Wahrscheinlichkeit zugewiesen. [39]

2.3 Verwendete Sensoren in Smartphones

Wie schon besprochen verfügen aktuelle Smartphones über eine Reihe von Sensoren, die zur Positionierung herangezogen werden können. In diesem Abschnitt sind jene Sensoren beschrieben, welche im Zuge der Arbeit Anwendungen fanden.

In Android werden die Sensoren eines Gerätes mittels SensorManger [14] angesprochen und verwaltet. Wird ein Sensor initialisiert [77], liefert dieser Daten an das System. Standardmäßig verfügt nahezu jedes Gerät über einen Beschleunigungssensor und Magnetfeldsensor. Aus den Werten dieser Sensoren stellt Android einen Orientierungssensor virtuell zur Verfügung. Ein Datensatz eines Sensors besteht im Allgemeinen aus drei Sensorwerten und einem Wert der die Genauigkeit der Daten beschreibt. Die Sensorwerte bezeichnen die gemessenen Daten in den jeweiligen Achsen des Geräte-Koordinatensystems, wie in Abbildung 2-2 dargestellt.

Abbildung 2-2 Geräte-Koordinatensystem [63]

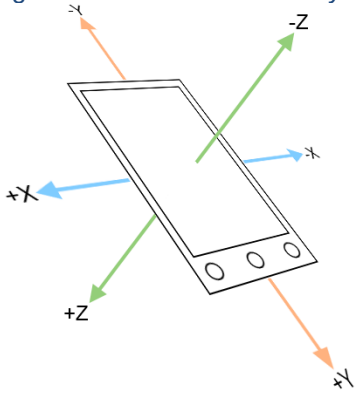
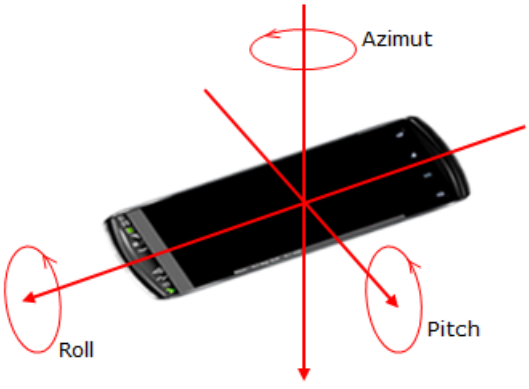


Abbildung 2-3 Weltkoordinaten des Orientierungssensor [43]

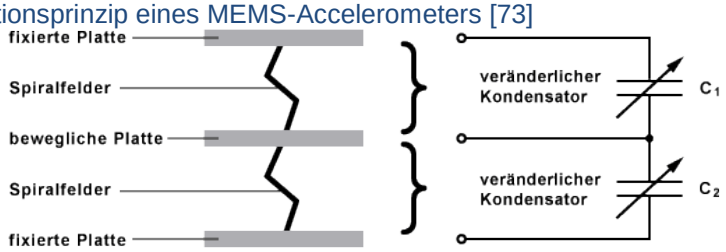


Beim Orientierungssensor sind die Achsen jedoch abhängig von der Lage des Handys, diese bilden die Werte in einem Weltkoordinaten-System ab, wie in Abbildung 2-3. Für das bessere Verständnis der Funktionsprinzipien der Sensoren sind diese im Anschluss beschrieben.

2.3.1 Beschleunigungssensor (Accelerometer)

Der Beschleunigungssensor in einem Smartphone ist ein Micro-Electro-Mechanical-System (MEMS). [74] Das sind kleinste Bauelemente, die elektronische Schaltungen kombiniert mit mikromechanischen Strukturen in einem Chip vereinen. Im Prinzip wird beim Beschleunigungssensor eine Platte (Masse) zwischen zwei Federn aufgehängt. Diese Masse befindet sich zwischen zwei fixierten Platten. Werden diese unter Spannung gesetzt, entsteht eine Reihenschaltung von zwei Kondensatoren mit veränderbaren Kapazitäten. Da die Kapazität eines Kondensators proportional zum Abstand der Platten ist, kann aus Messung der Kapazitätsänderung die Bewegung der mittleren Platte hergeleitet werden. Die Abstände zur mittleren Platte verändern sich dann, wenn diese eine Beschleunigung erfährt. Aus der Kapazitätsveränderung kann daher die Beschleunigung berechnet werden. Dieses Prinzip ist in Abbildung 2-4 skizziert und wird für jede Achse des Gerätes separat angewandt. [73]

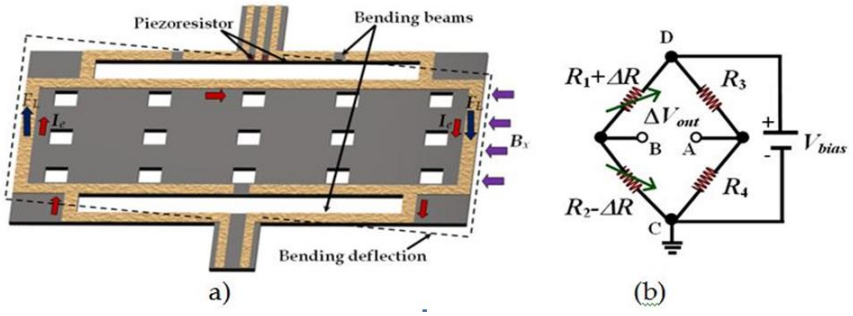
Abbildung 2-4 Funktionsprinzip eines MEMS-Accelerometers [73]



2.3.2 Magnetfeldsensor

Mit dem Magnetfeldsensor wird das lokale Magnetfeld gemessen. Dieser beruht auf der Messung der Lorentzkraft mittels des piezoresistiven Effektes. Durch die Veränderung des elektrischen Widerstands eines Materials, auf welches Druck oder Zug ausgeübt wird, kann auf die ausgeübte Kraft geschlossen werden. Der prinzipielle Aufbau (a) und Schaltplan (b) sind in Abbildung 2-5 dargestellt. Das magnetische Feld übt eine Kraft auf den Sensor aus, welche den elektrischen Widerstand beeinflusst, der wiederum gemessen werden kann. [35]

Abbildung 2-5 Prinzip und Schaltplan eines Magnetfeldsensors [35]



Das gemessene Magnetfeld beziehungsweise die magnetische Flussdichte, welche ein Maß für die Stärke ist, wird allgemein in der Einheit Tesla [T] gemessen. Das Erdmagnetfeld hat in etwa eine Flussdichte von 30-60 µT. Jedes elektrisch betriebene Gerät erzeugt selbst ein magnetisches Feld, das durchaus in die Größenordnung des Erdmagnetfeldes gelangen kann und somit den Sensor beeinflusst. Daher sind diese Sensordaten gerade in Gebäuden mit vielen elektronischen Geräten oft unzuverlässig. [35]

2.3.3 Orientierungssensor

Der von Android bereitgestellte Orientierungssensor ermöglicht es, die Ausrichtung des Gerätes im Raum abzufragen. Dieser Sensor ist kein realer Sensor, zumal er nicht als Hardware im Gerät existent ist. Der Orientierungssensor wird jedoch, wie alle anderen Sensoren, mithilfe des *SensorManager* initialisiert. Android nutzt die Daten des Magnetfeldsensors und Beschleunigungssensors, um die Werte für Azimut (Richtung bzw. digitale Kompass), Pitch (Vorwärtsneigung) und Roll (Seitwärtsneigung) zu berechnen. Dabei wird das Vorhandensein der überall wirkenden Erdbeschleunigung genutzt, um über die Sensorwerte der 3-Achsen des Beschleunigungssensors die Lage des Gerätes abzuschätzen.

Kombiniert wird diese Information mit den Daten des Magnetfeldsensors, um auf die Richtung bzw. den Azimut-Wert zu schließen. Die zuverlässigsten Werte für den digitalen Kompass liefert der Sensor, wenn das Handy eine waagerechte Position (Roll-Wert ist null) zur Erdoberfläche einnimmt, wie in Abbildung 2-3 skizziert. Das Verwenden dieses Sensors ist ab der Version Android 2.2 veraltet, da dieser rechenintensiv ist und nur genaue Werte liefert, wenn der Roll-Wert nicht null ist. [43]

Dadurch, dass der Orientierungssensor die Werte des Magnetfeldsensors verwendet, wird auch dieser von magnetischen Feldern beeinflusst. In modernen Gebäuden befinden sich oft viele elektronische Geräte und Kabel, die ein eigenes Magnetfeld erzeugen, weswegen dort größere Abweichungen entstehen können. Konkrete Beispiele werden in Kapitel 6.3 gezeigt.

2.4 Konklusion, Zusammenfassung

Von den vorgestellten Positionierungstechniken eignet sich das WLAN-Fingerprinting für den Ansatz, der in dieser Arbeit verfolgt werden soll, am besten. In anderen Arbeiten wurde gezeigt, dass damit im Indoor-Bereich gute Ergebnisse erzielt werden können. Weiter kann auf ein bestehendes WLAN Netz zugegriffen werden und es müssen keine spezifischen Anforderungen, wie z. B. das Synchronisieren der APs, erfüllt sein. Im einfachsten Fall wird lediglich ein Gerät benötigt, das RSSI-Vektoren in einer Datenbank speichert, um später mittels eines WLAN-Scans auf Positionen schließen zu können.

Positionierung über GPS und das Mobilfunknetz eigenen sich nicht für die Anwendung im Indoor-Bereich, da diese zu unzuverlässig bzw. zu ungenau sind. Jedoch könnten diese Techniken mit einem WLAN-Fingerprinting-System kombiniert werden.

Der NN-Algorithmus ist der einfachste Fingerprinting-Algorithmus und eignet sich, um eine zuvor vermessene Position zu identifizieren. Da in dieser Arbeit bestimmte Positionen bzw. iCPs untersucht werden sollen, wurde mit diesem als Basis gearbeitet.

Um von einer bekannten Positionen aus zu navigieren, bietet sich ein auf einem „*Inertial Navigation System*“ (INS) basierender Ansatz an. Daher war es Ziel der Arbeit, mittels Fingerprinting iCPs zu erkennen und zwischen diesen mit einem INS-Algorithmus zu navigieren. Zu beachten ist dabei, dass die Sensoren oft ungenaue Werte liefern, wobei das Navigieren in Gebäuden eine besondere Herausforderung darstellt.

3 Datenerfassungs- und Analysesystem (DAAS)

Ein Ziel der Arbeit war es, eine Test- und Simulationsumgebung zu schaffen, in der der vorgeschlagene Ansatz Schritt für Schritt erarbeitet werden kann. Mögliche Methoden und Ansätze sollen damit getestet, analysiert und verglichen werden. Die dahinterstehende Idee ist, alle für die Positionierung verwendbaren Daten aufzuzeichnen und diese für die Simulationen zu verwenden.

Dieses Datenerfassungs- und Analysesystem (DAAS) wurde aus zwei Teilkomponenten realisiert. Für die Erfassung der Daten wurde eine Android-Applikation (App) in Eclipse [28] mit den auf Java basierten Android Developer Tools [3] programmiert.

In MATLAB wurde ein Framework (MFW) geschrieben, das die erfassten Daten importiert und analysiert. Mit den importierten Datensätzen können Analysen und Simulationen unterschiedlichster Art durchgeführt werden.

Erfasst können grundsätzlich zwei Arten von Datensätzen werden, WLAN-Scans und Testläufe. Die WLAN-Scans werden auf einer definierten Position erstellt. Dabei werden die empfangenen Signalstärken (RSS) der Access Points (APs) sowie der Azimut-Wert des Orientierungssensors (2.3.3) erfasst. Zur Aufzeichnung der Testläufe muss eine Teststrecke abgegangen werden. Die dabei auftretenden Sensordaten, GPS-Koordinaten und WLAN-Scans werden mit einem Zeitstempel versehen und abgespeichert. Bei den Testläufen kann das Erreichen von Referenzpunkten manuell festgehalten werden.

Das Erfassen der beschriebenen Datensätze hat einige Vorteile. So können verschiedene Methoden mit unterschiedlichen Parametern schnell und einfach getestet und gegenübergestellt werden. Dadurch, dass die Algorithmen mit den gleichen Datensätzen arbeiten, werden diese unter denselben Bedingungen miteinander verglichen. Einflussfaktoren wie z. B. Positionen oder Verfügbarkeit von Satelliten oder Störfaktoren für die Signalstärke der AP [39], usw. bleiben unverändert. Somit steigt auch die Aussagekraft einer Gegenüberstellung zweier Methoden, da diese exakt dieselben Datensätze benutzen.

Da Teststrecken nicht immer wieder von Neuem abgegangen werden müssen, wird Zeit gespart. Testläufe können einmal aufgezeichnet und mit verschiedenen Methoden untersucht werden. Es können auch verschiedene Parameter untersucht bzw. optimiert werden.

3.1 DAAS – Systemanordnung

Wie bereits in der Einleitung beschrieben, besteht das DAAS aus zwei Teilen, aus einer App zur Datenerfassung bzw. -aufzeichnung und aus einer Reihe von MATLAB-Funktionen für den Import und die Analyse dieser Daten. So können beliebige Simulationen mit den erfassten Daten durchgeführt werden. Da diese Funktionen den Rahmen für Simulationen und Analysen bieten, werden diese in der Arbeit als MATLAB-Framework (MFW) bezeichnet.

In Abbildung 3-1 ist die bei der Arbeit verwendete Systemanordnung skizziert. Als Testgeräte standen das Samsung Galaxy S2 und das HTC Evo 3D zu Verfügung. Diese sind im Anhang A in 8.2 genauer beschrieben. Auf den Geräten wurde jeweils das CPS-App [37, CPS-APP\InstallDatei] zur Datenerfassung installiert. Die Abkürzung CPS ist ein Hinweis auf das letztlich angestrebte Ziel und steht für Combined-Positioning-System. Das CPS-App speichert die erfassten Daten lokal auf den SD-Karten der Handys. Der Funktionsumfang und Details zum Programmaufbau des CPS-App sind im Abschnitt 3.2 beschrieben.

Zum Sammeln und Synchronisieren der erfassten Daten wurden die im Google Play Store frei verfügbaren Apps DropBox [26] und DropBox Sync [60] verwendet. Die Daten werden auf einem Dropbox-Konto [25] gespeichert. Ein mit dem Dropbox-Konto verbundener PC kann auf die erfassten Daten nach dem Synchronisieren zugreifen. Das System ist so ausgelegt, dass beliebig viele Nutzer und Geräte verwendet werden können, ohne dass es zu Inkonsistenz in den Daten kommen kann.

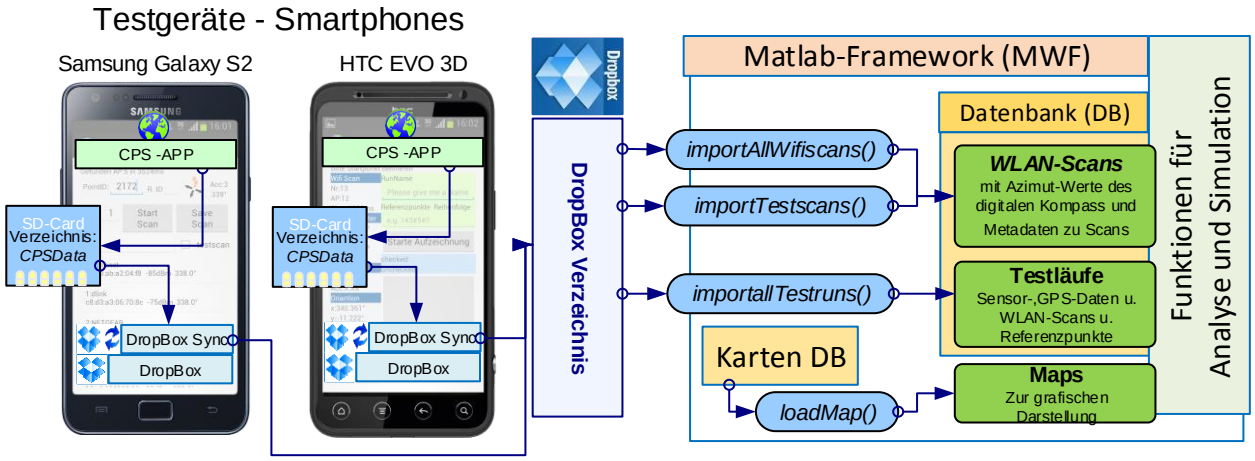
Die Daten werden in Textfiles gespeichert und von den Funktionen des MFW interpretiert. Das Framework wird im Abschnitt 3.3 erörtert. Die Hauptfunktionen sind in der Abbildung 3-1 eingezeichnet. Mit der Funktion *importAllWiFiscans()* [37, MFW\functions\wifi] werden die WLAN-Scans, die für die Kalibrierungsphase vorgesehen sind, importiert. Die Funktion *importTestscans()* [37, MFW\functions\wifi] lädt jene WLAN-Scans, die als Testscans für die Positionierungsphase erzeugt wurden. Detaillierte Ausführungen zu den Funktionen und deren Zusammenhänge sind dem Kapitel 3.3.2 zu entnehmen.

Die aufgezeichneten Sensordaten und die dazu gehörigen WLAN-Scans werden mit *importallTestruns() [37, MFW\functions\testruns]* in die Datenbank geladen. In 3.3.3 wird beschrieben, wie die Testlauf-Datensätze importiert und interpretiert werden.

Für die grafische Darstellung ist es möglich, Bilder von Karten zu laden und einzupassen. Berechnete Positionen können so auf diesen Karten eingetragen werden. Wie diese Funktionen arbeiten, wird in Abschnitt 3.3.4 gezeigt.

Die im späteren Verlauf für die Analyse verwendeten Funktionen bauen auf den Datensätzen der hier beschrieben Funktionen auf.

Abbildung 3-1 Schematische Darstellung des gesamten DAAS



3.2 Android Applikation (CPS-App)

Die Android App besteht neben dem Startmenü aus zwei weiteren Benutzeroberflächen (GUIs) bzw. Tools. Ein Tool, der *Kompass-WLAN-Scanner,* dient zur Ermittlung der WLAN-Scan-Datensätze. Der *WLAN-Sensor-Recorder*, das zweite Tool, dient zum Aufzeichnen der Testlauf-Datensätze. Detaillierte Angaben zum Funktionsumfang der GUIs sind im nachfolgenden *Abschnitt 3.2.1* erläutert.

Im *Abschnitt 3.2.2* wird in den Aufbau und die Lebenszyklen einer Android Applikation eingeführt. Das *Kapitel 3.2.3* beschreibt die Struktur der CPS-App und erläutert, welche Klassen implementiert wurden, sowie deren Zusammenhänge. In dem Kapitel 3.2.4 wird der Aufbau der erzeugten Datensätze erläutert.

In 3.2.5 werden die Methoden der Klassen, welche die Tools implementieren, näher erklärt. Die Methoden und Programmabläufe werden darin skizziert. Detaillierte Beschreibungen sind den Kommentaren im JAVA-Code des Eclips-Projekts [37, CPS-APP\Eclipse-Projekt] zu entnehmen.

3.2.1 Funktionsumfang und Userinterface

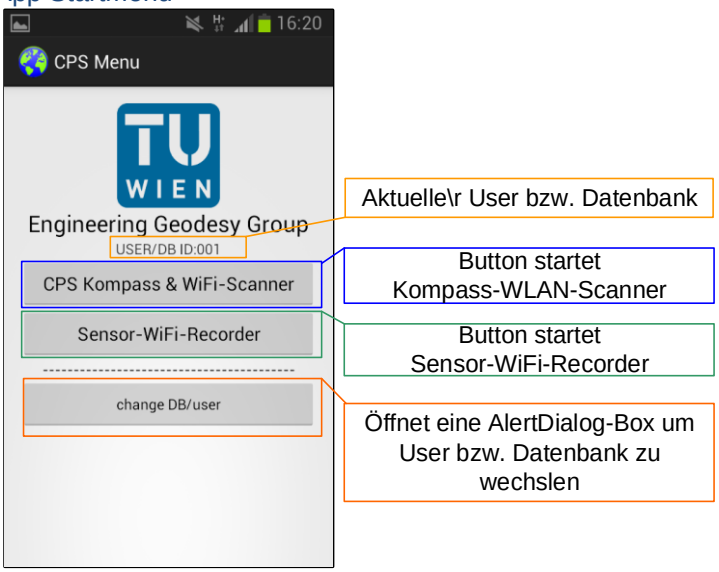
Das Startmenü

Wird das CPS-App gestartet, so gelangt man in das in Abbildung 3-2 dargestellte Startmenü. Beim ersten Start der App erscheint eine DialogBox [8], welche zur Eingabe einer User- oder Projekt-ID (*USER/DB-ID*) auffordert.

Mit dieser *UserID* wird ein Verzeichnis auf der SD-Karte des Handys erstellt. So können Daten von verschiedenen Personen, Projekten, bzw. Testgebieten auf einem Handy gespeichert und unterschieden werden. Die Unterscheidung von Personen kann z. B. bei der Schritterkennung (z. B. Schrittgröße) eine Rolle spielen. Die aktuelle UserID wird im Startmenü angezeigt.

Im Startmenü ist es möglich, den User bzw. das Projekt zu wechseln und den Kompass-WiFi-Scanner oder den Sensor-WiFi-Recorder zu starten.

Abbildung 3-2 CPS-App Startmenü

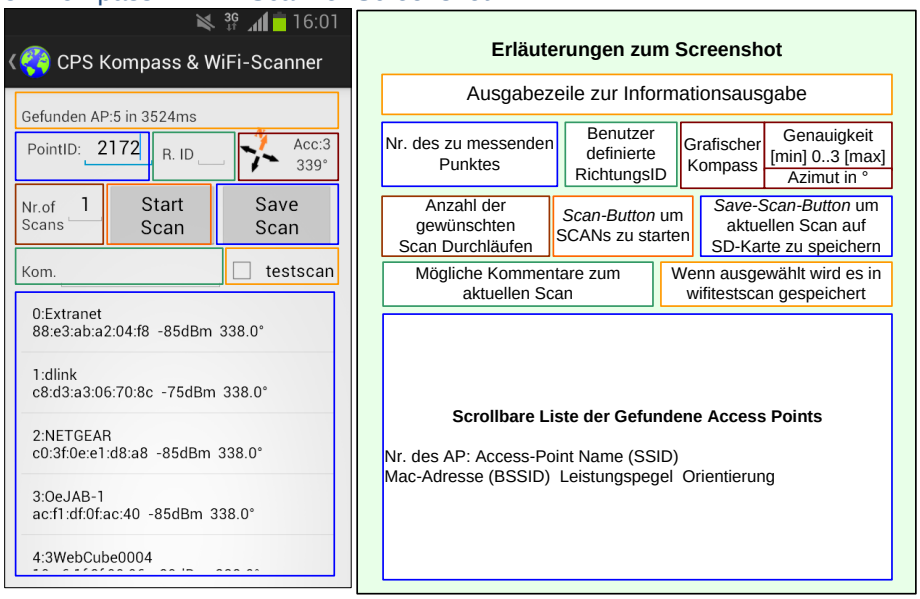


Kompass-WLAN-Scanner

Die Abbildung 3-3 stellt den **Kompass-WLAN-Scanner** dar. Mit diesem Tool ist es möglich, die RSS-Werte der APs zu ermitteln und diese mit dem Richtungswert (Azimut-Wert [81]) des Handys in Verbindung zu bringen. Dazu muss das Handy horizontal zum Boden, also so wie in Abbildung 6-1, gehalten werden, um den präzisesten Azimut-Wert zu erhalten [11].

Der **Kompass-WLAN-Scanner** verfügt im obersten Bereich über eine Ausgabezeile, welche die Informationen über den aktuellen Status des Tools anzeigt. Um einen WLAN-Scan-Datensatz einer Position zuordnen zu können, muss dieser mit einer Positions-ID (*PointID*) versehen werden. Daher kann der Datensatz erst gespeichert werden, wenn die *PointID* eingegeben wurde. Somit ist sichergestellt, dass jeder WLAN-Scan über eine *PointID* verfügt. Die Zuordnung der PointID zu einer konkreten Position bzw. Koordinaten erfolgt im MFW.

Abbildung 3-3 Kompass-WLAN-Scanner Screenshot



Rechts vom Textfeld *PointID* kann vom Benutzer die *RichtungsID* definiert werden, denn beim WLAN-Fingerprinting werden WLAN-Scans an einer Position meist in vier Richtungen [19] durchgeführt. Gemessen wird üblicherweise in vier um 90° versetzte Richtungen. Der Grund ist, dass der menschliche Körper Signale abschwächen bzw. verfälschen kann [39]. Die in dem Feld *RichungsID* eingegebene Zahl kann später als Referenz verwendet werden. Im MFW dient sie auch dazu, die WLAN-Datensätze in Richtungskategorien einzuteilen. Für die Arbeit wurden vier Kategorien verwendet, gewählt werden kann jedoch eine beliebige ganze Zahl. Neben dem Textfeld für die Richtungs-ID wird der aktuelle Azimut-Wert bzw. der Kompass grafisch und numerisch angezeigt.

Der Kompass-WLAN-Scanner bietet die Funktion, die gewünschte *Anzahl von WLAN-Scans* zu definieren. So können mehrere Scans hintereinander durchführt werden. Die Anzahl ist standardmäßig auf eins eingestellt. So wird nur ein Scan durchgeführt, der manuell mit dem *Save-Scan-Button* gespeichert werden muss. Wird die Anzahl der durchzuführenden WLAN-Scans erhöht, ändert sich der Text des *Scan-Buttons* zu „*Scan&Save*“. Wenn danach der Button gedrückt wird, wird automatisch die eingegebene Anzahl an WLAN-Scans durchgeführt. Die erfassten WLAN-Scan-Datensätze werden unmittelbar nach jedem Scan gespeichert.

Ein WLAN-Scan-Datensatz kann außerdem mit einem Kommentar versehen werden. So könnten z. B. mögliche Einflussfaktoren, wie etwa die Anzahl der Menschen in einem Raum oder verbunden Geräte, vermerkt werden.

Gespeichert werden die WLAN-Scan-Datensätze auf der SD-Karte im Unterverzeichnis der aktuellen *UserID*. Für jede PointID wird eine Datei erstellt, in der alle WLAN-Scans des Punktes gespeichert werden. Um die Kalibrierungsdaten und Testdaten zu trennen, kann die Checkbox *testscan* genutzt werden.

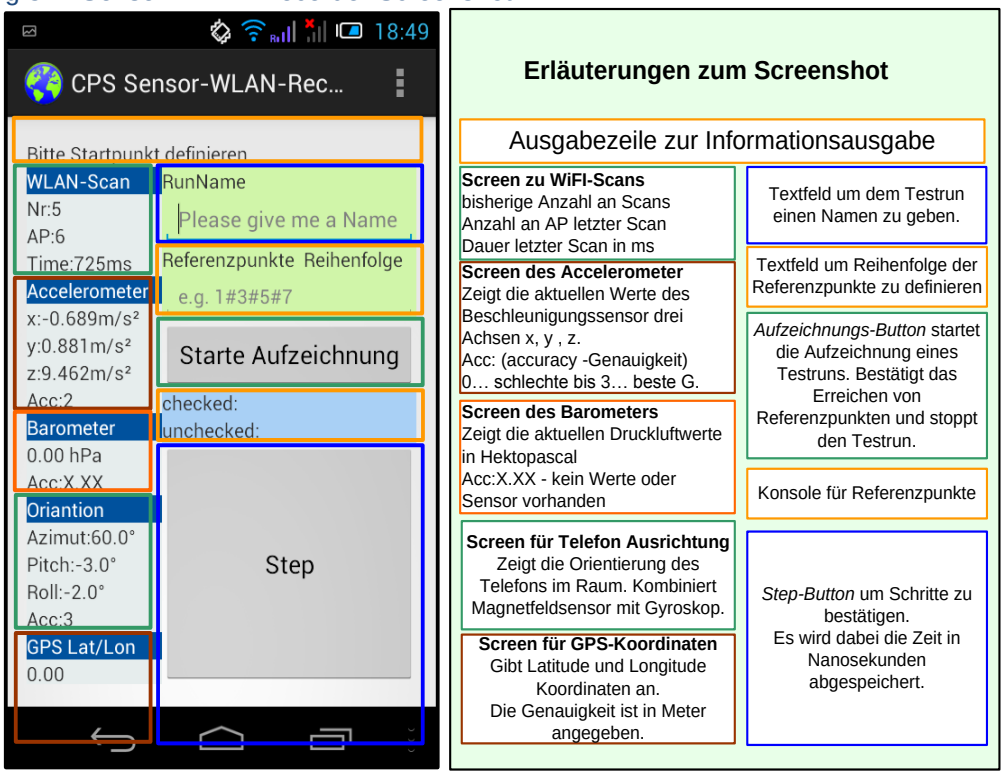
Im unteren Bereich der GUI werden die RSS-Werte der Access Points und entsprechende Informationen dazu in einer Liste dargestellt. In der Liste sind die gefundenen APs durchnummeriert. Für jeden AP wird der Netzwerkname (SSID - Service Set Identification), die Mac-Adresse (BSSID - Basic Service Set Identification), die Signalstärke (RSS-Wert in dBm) und der zur Startzeit des Scans gemessene Azimut-Wert dargestellt. Alle RSS-Werte eines Scans haben den gleichen Azimut-Wert.

Sensor-WLAN-Recorder

Die Abbildung 3-4 zeigt den Sensor-WLAN-Recorder. Mit diesem Tool können Sensor-, GPS-daten und eine Folge von WLAN-Scans aufgezeichnet werden. Die Daten werden mithilfe von Zeitstempel miteinander in Verbindung gebracht. Diese Datensätze dienen in weiterer Folge zur Simulation der zu entwickelnden und zu testenden Positionierungsalgorithmen. Wie der Kompass-WLAN-Scanner verfügt auch der Sensor-WLAN-Recorder über eine Ausgabezeile, welche über den aktuellen Status des Tools informiert.

Die GUI ist in zwei Bereiche geteilt: Links befindet sich die Datenleiste, welche die aktuellen Werte der Sensoren und Informationen zum WLAN-Scanner anzeigt. Rechts sind die Elemente, die zur Konfiguration und Durchführung der Aufzeichnungen dienen, angeordnet. Jeder Testlauf kann mit einer Bezeichnung im Feld *RunName* versehen werden.

Abbildung 3-4: Sensor-WLAN-Recorder Screenshot



In der linken Datenleiste werden im Abschnitt *„WLAN Scan“* Informationen zu den WLAN-Scans gegeben. Neben *„Nr:“* steht die Anzahl der bisher durchgeführten Scans. Wie viele RSS-Werte bzw. APs im letzten WLAN-Scan gemessen wurden, ist neben *„AP:“* abzulesen. Neben dem Textfeld *„Time:“* ist die Scandauer des letzten WLAN-Scans in Millisekunden festgehalten.

Die Sensor-Leiste zeigt weiter die Daten von Accelerometer, Barometer, Orientierung und GPS-Receiver an. Sollte ein Sensor nicht vorhanden sein oder keine Daten liefern, zeigt dieser bei „Acc:“ (eng. accuracy; Genauigkeit) den String *„X.XX“* an. Ansonsten wird dort der erhaltene Genauigkeitswert dargestellt.

Das *Accelerometer* liefert für die 3 Achsen des Gerätes (siehe Abbildung 2-2) Daten in m/s². Die Daten des *Orientierungssensors* beziehen sich auf Weltkoordinaten (siehe Abbildung 2-3) und stellen den Grad-Wert für den Azimut-Wert (digitalen Kompass), den Pitch-Wert (Vorwärtsneigung) und den Roll-Wert (Seitwärtsneigung) dar.

Die mittels *GPS-Modul* ermittelten Koordinaten für Breitengrad (lat.: *latitude φ)* und Längengrad (lon.: *longitude, λ*) sind in Dezimalgrad angegeben. Die Abweichung, die vom GPS-Modul ermittelt wurde, ist in der Zeile „Acc:“ in Metern dargestellt.

Bevor eine Aufzeichnung gestartet wird, können Referenzpunkte und somit eine Teststrecke definiert werden. Die Referenzpunkte werden durch Zahlen repräsentiert, damit diese im MATLAB-Framework mit einer Position in Verbindung gebracht werden können. In Kapitel 3.3.4 wird erläutert, wie diese Referenzpunkte in ein lokales Koordinatensystem eingepasst wurden. Im **Anhang A** sind die für die Arbeit verwendeten Referenzpunkte und Teststrecken definiert. Um eine komplette Teststrecke einzugeben, werden die Referenzpunkte durch das Raute–Symbol (#) getrennt. Es müssen mindesten zwei Referenzpunkte für den Start- und Endpunkt der Testrecke eingegeben werden.

Die Aufzeichnung der Daten wird mit Drücken des *Aufzeichnungs-Buttons* begonnen. Bei der Aufzeichnung werden nur neu entstandene Daten in die Dateien geschrieben. Wenn die Aufzeichnung läuft und Referenzpunkte eingegeben wurden, erscheint der nächste Referenzpunkt als Text für den *Aufzeichnungs-Button.* Durch das erneute Drücken auf den Button mit der Referenznummer wird dieser Punkt als erreicht registriert.

Die bereits erreichten Referenzpunkte werden in der Box darunter, in der Zeile „checked“, aufgelistet. Die noch zu passierenden Punkte können der Zeile „unchecked“ entnommen werden. Mit diesem System können während eines Testlaufes erreichte Positionen mit einem bestimmten Zeitpunkt in Verbindung gebracht werden. Diese Daten können als Referenz-Wege verwendet werden. Im MATLAB-Framework wurden Referenz-Schritte (siehe 6.1) ermittelt, um die berechneten Positionen zu evaluieren.

Wenn Referenzpunkte eingeben werden, müssen zumindest zwei Punkte definiert werden. Der Erste und Letzte definieren zugleich den Start und das Ende der Aufzeichnung. Wenn keine Referenzpunkte eingegeben werden, kann eine Aufzeichnung beliebig gestartet und beendet werden.

Der *Step-Button* ermöglicht es, Referenzen für die einzelnen Schritte des Users zu definieren. Bei jedem Schritt muss der Button betätigt werden. Daraufhin wird mithilfe eines Zeitstempels der Zeitpunkt und somit auch die Anzahl der Schritte erfasst. Diese Option der App wurde für die Arbeit nicht verwendet, da die verwendeten Testgeräte das Drücken des Buttons nicht immer schnell genug registrieren konnten.

Am Ende jeder Aufzeichnung erscheint eine Dialogbox zur Abfrage der Anzahl der getätigten Schritte. Wurden die Schritte mitgezählt, können diese eingegeben werden, ansonsten muss die Dialogbox mit „*cancel*“ beendet werden.

3.2.2 Einführung in Android und die App Programmierung

Android - das offene Betriebssystem

Mit einem Marktanteil von über 70 % [41] ist Android das verbreitetste Betriebssystem für mobile Geräte wie Smartphones und Tablets. Ein Grund dafür ist sicher, dass Android für Entwickler bestimmte Vorteile bietet. Die Apps können in der weitverbreiteten und vielen vertrauten Programmiersprache JAVA geschrieben werden. Die vollständige und umfangreiche Entwicklungsumgebung [3] steht kostenlos und öffentlichen zum Download zur Verfügung.

Eine wichtige Rolle bei der Entwicklung von Android spielte die Open Handset Alliance (OHA), einem Konsortium von Halbleiter- und Mobiltelefonherstellern, Netzbetreibern und Softwareentwicklern und in ähnlichen Branchen tätigen Firmen. Die OHA setzte von Beginn an auf eine auf Offenheit und Flexibilität gründende Zusammenarbeit mit Entwicklern, Herstellern und Betreibern. Daher wurden praktisch alle Teile des Android-Softwarestapels als Open Source veröffentlicht. Dies führte dazu, dass sich eine breite Basis an der Weiterentwicklung von Android beteiligte. Seit der ersten Veröffentlichung im November 2007 ist eine Reihe an Versionen und Varianten von Android erschienen. [49]

Da das System jedem offen steht, können Hersteller angepasste Versionen für deren Geräte anbieten. Dies hat dazu geführt, dass es von einigen Android-Versionen eine Vielzahl an Varianten gibt. Dies sollte bei der Entwicklung einer App stets berücksichtigt werden.

Aufbau und Lebenszyklus einer Android Applikation

Die Android-Plattform kann in mehrere Schichten unterteilt werden. Auf der untersten Ebene befindet sich ein Linux-Kernel, der die Hardwarekomponenten der Geräte anspricht. Darüber befinden sich eine Reihe von JAVA und C Libraries sowie die Kernfunktionen des Android-Systems. Ausgeführt werden die Applikationen von der virtuellen Maschine Dalvik [23], welche den Bytecode interpretiert. Auf diese Schicht baut das Android-Framework auf. Mit dessen Funktionen können die Apps auf das System zugreifen. Über dieses Framework kommunizieren die Apps mit den Hardwarekomponenten. Über diesen Weg werden auch die Sensoren und Module, die in der Arbeit verwendet wurden, angesprochen. [49]

Eine Android-Applikation definiert ihre Benutzeroberflächen über XML-basierte Layout-Dateien. Eine GUI kann jeweils weitere GUIs aufrufen bzw. initialisieren. Diese GUIs werden während der Laufzeit der App vom System als Objektbaum initialisiert und angeordnet.

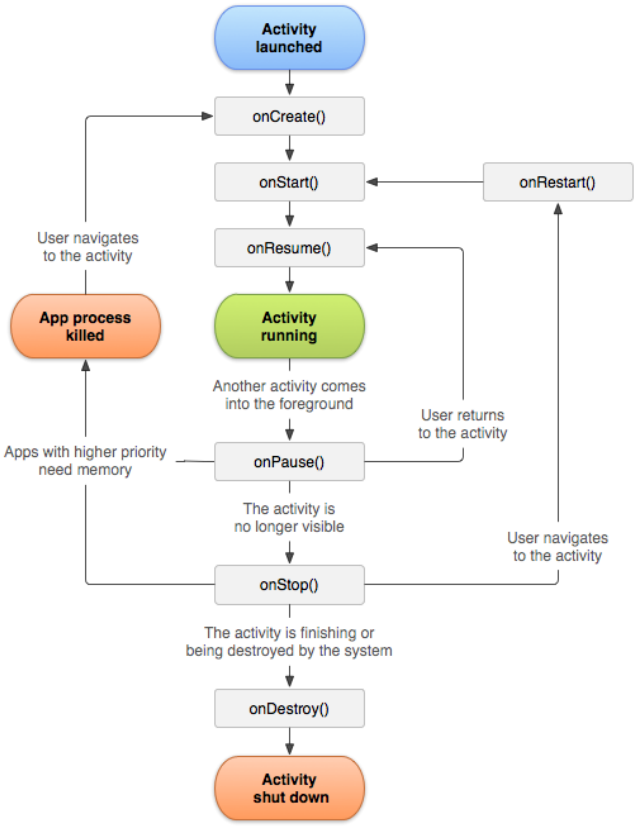
Aus den einzelnen Teil-GUIs kann eine Anwendung so in Funktionsblöcke unterteilt werden. Bei Android ist das Aufteilen einer App in solche aufgabenorientierten Teile bzw. Funktionsblöcke ein fundamentales Architekturmuster. Diese GUIs werden Activities [4] genannt. Die Benutzeroberflächen werden in einer XML-Datei activity_*NAME*.xml definiert. Das Objekt, das die GUI erzeugt und die Methoden implementiert, wird durch die dazugehörige JAVA Klasse definiert. Dieses Objekt erbt die Methoden und Eigenschaften von der Klasse *Activity* [5]. [49]

Activities durchleben einen definierten Lebenszyklus, der in Abbildung 3-5 dargestellt ist. Wenn die Activity gestartet wird, führt diese die Methode *`onCreate()`* aus, deren Code immer überschrieben wird. In ihr werden gewöhnlich die Elemente der Benutzeroberfläche und die Variablen initialisiert. Als nächstes wird die Methode `onStart()` ausgeführt. In ihr werden die Funktion implementiert, welche ausgeführt werden sollen, wenn die App geöffnet wird.
Die Ausführung der Methode `onResume()` erfolgt, bevor der Benutzer auf der GUI zugreifen kann. Sie wird auch aufgerufen, wenn das Objekt aus dem Pause-Modus zurückkehrt.

Nachdem die Methode `onResume()` ausgeführt wurde, ist die Activity zur Interaktion mit dem User bereit. Die Funktionen der App können dann vom User genutzt werden. Dieser kann die Activity auch in den Pause Modus setzen, z. B. durch das Drücken des Settings-Buttons. Bevor die Activity in den Pause-Modus geht, wird die Methode `onPause()` ausgeführt. Wird zur einer anderen App gewechselt, wird `onStop()` durchlaufen.

Je nachdem wird nach dem Zurückkehren wieder `onResume()` oder beim Rückkehren aus einer anderen App `onStart()` ausgeführt. Diese Zusammenhänge sind in Abbildung 3-5 dargestellt.

Abbildung 3-5 Lebenszyklus einer Activity [2]



Ein weiteres Konzept von Android Apps ist die Trennung der Programmlogik von den App-Ressourcen. So sollten z. B. alle Texte in der Datei *strings.xml* definiert werden. Das macht es leichter, Apps für mehrere Sprachen zu adaptieren. Getrennt werden auch das Layout und das Objekt, das diese implementiert.

Eine weitere wichtige Datei ist die AndroidManifest.xml. Jene enthält eine Liste der Komponenten, aus denen das Programm besteht und definiert, welche Berechtigungen die App benötigt, um deren Funktionen zu erfüllen. Darin werden z. B. Berechtigungen definiert, damit Apps auf den GPS-Receiver und WLAN-Modul zugreifen dürfen. Die Zustimmung muss der Benutzer beim Installieren geben.

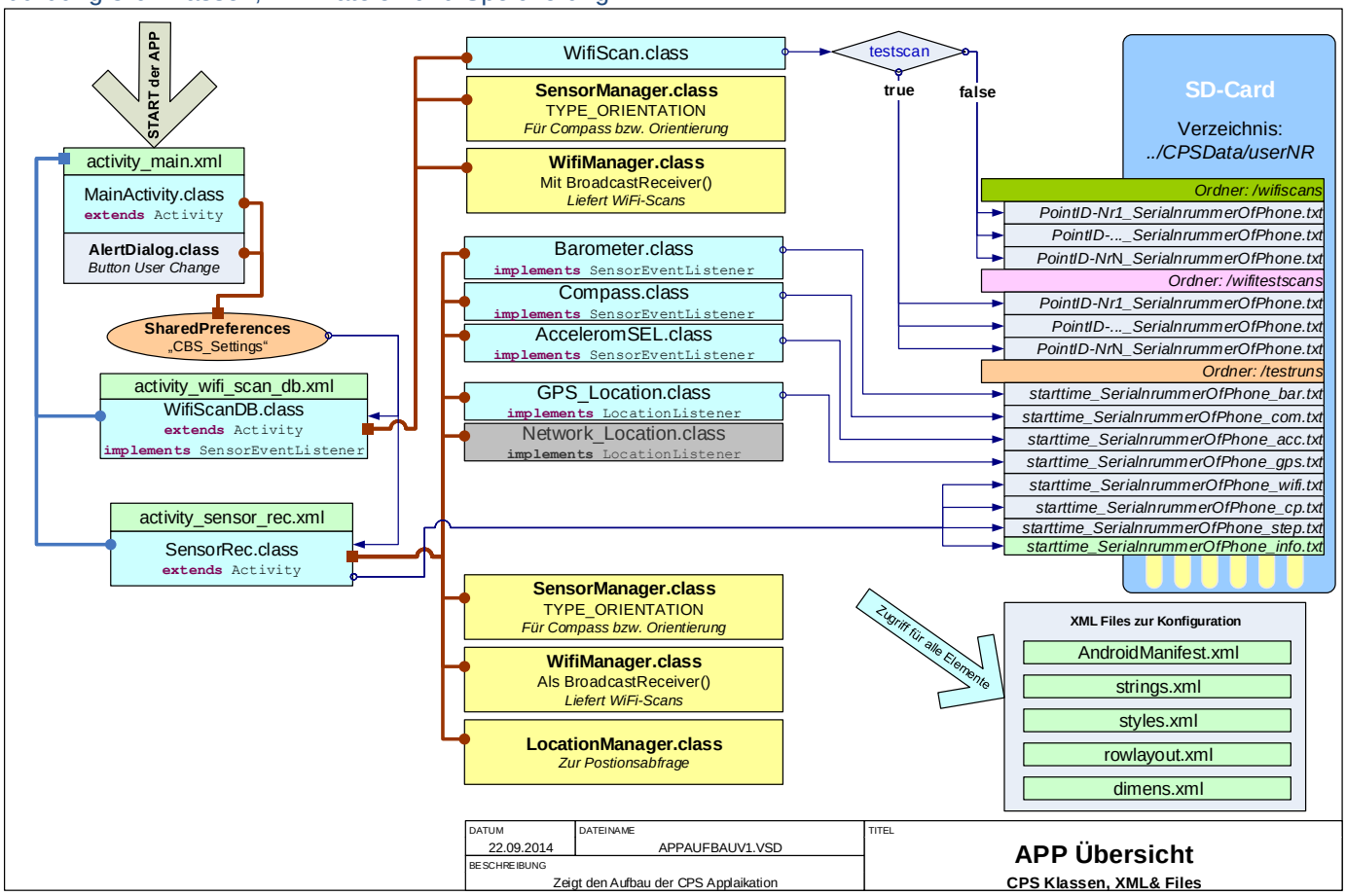
3.2.3 Systemanordnung des CPS-Apps – Klassen und Ressourcen

In diesem Abschnitt werden die wichtigsten Elemente der CPS Apps erläutert und die programmierten Java-Klassen und ihre Zusammenhänge dargestellt. Die App ist optimiert für die Android Version 4.4.2 (API 19), läuft aber auch mit früheren Versionen.

Die weiteren Abschnitte dienen der Dokumentation und geben einen Überblick über den Aufbau der entwickelten App. So kann gegebenenfalls die App leichter um zusätzliche Funktionen erweitert werden. Eine detaillierte und vollständige Dokumentation des Programmes würde den Rahmen dieser Diplomarbeit sprengen. Strukturen und Abläufe werden vereinfacht dargestellt. Abbildung 3-6 skizziert den Aufbau der App, dargestellt sind XML-Files, Java-Klassen und die Speicherorte der Daten. Die Daten werden auf der SD-

Karte im Verzeichnis „*CPSData*“ gespeichert. Die erzeugten Datensätze werden in Kapitel 3.2.4 erläutert.

Abbildung 3-6: Klassen, XM-Dateien und Speicherung



Das Startmenü

Beim Start der App wird das zuvor beschriebene **Startmenü** aus dem Java-Objekt der Klasse *MainActivity* erzeugt, dessen GUI ist durch die Datei *activity_main.xml* definiert. Vom Menü aus wird die Initialisierung der Activities für den Kompass-WLAN-Scanner und den Sensor-WLAN-Recorder gestartet. Das Startmenü verwaltet mithilfe einer AlertDialog-Box den Wechsel zwischen Datenbank- bzw. User-ID. Die aktuelle UserID und die dazugehörigen Pfade werden in dem SharedPreferences [16] Objekt *CPS_Settings* gespeichert. Auf *CPS_Settings* können alle Objekte des Apps zugreifen, um die Einstellungen zur UserID abfragen zu können.

Kompass-WLAN-Scanner

Der Kompass-WLAN-Scanner wird aus der **Klasse WifiScanDB** erzeugt. Diese greift auf die Datei *activity_wifi_scan_db.xml* zur Generierung der GUI zu. Die Klasse implementiert das Android Inferface *SensorEventListener* [13] und erbt dessen Methoden, um die gemessenen Werte eines Sensors erhalten zu können. Verwendet wird der Orientierungssensor, der anhand eines Objekts der Klasse *SensorManager* [15] initialisiert wird.

Um nach den APs scannen zu können, wird eine Instanz der Klasse *WifiManager* [17] erzeugt. Diese Android-Klasse erlaubt es, das WLAN-Modul anzusprechen und WLAN-Scans durchzuführen. Die Scandauer ist von Gerät zu Gerät verschieden. Um den Zeitpunkt des Endes eines WLAN-Scans zu erkennen, wird ein Objekt der Klasse *BroadcastReceiver* [7]

erzeugt. Mit dem *BroadcastReceiver* wird erkannt, wann der WLAN-Scan abgeschlossen ist. Erst dann können neue RSS-Werte abgefragt werden.

Der Datensatz für einen WLAN-Scan wird mit einem Objekt der Klasse *WiFiScan* generiert und gespeichert.

Sensor-WLAN-Recorder

Der Sensor-WLAN-Recorder wird durch die **Klasse SensorRec** und Layout-Datei *activity_sensor_rec.xml* definiert. Das Objekt erzeugt alle nötigen Sensor-Objekte und verwendet die Klassen *WiFiManger* und *BroadcastReceiver,* um parallel zur Aufzeichnung der Sensordaten nach RSS-Werten der APs zu scannen und diese abzuspeichern.

Der Sensor-WLAN-Recorder zeichnet die GPS-Koordinaten, Barometer-, Beschleunigungssensor- und Orientierungssensordaten sowie RSS-Werte über den Zeitraum des Testlaufs auf. Gespeichert werden diese Daten für jeden Testlauf in jeweils neue Dateien, wie in Abbildung 3-6 dargestellt.

Für das Erfassen und Abspeichern der Sensordaten wurde jeweils eine eigene Klasse erstellt. Die Klassen *Barometer, Compass und AcceleromSEL* implementieren das Interface *SensorEventListener*. Dadurch erben die Objekte jene Methoden, die es ermöglichen, neu erfasste Werte der Sensoren zu registrieren.

Die Klasse G*PS_Location* implementiert einen LocationListener, der es möglich macht, die Änderungen der GPS-Koordinaten zu erfassen.

Die Sensor-Objekte laufen als Threads. Erfasst ein Sensor neue Daten, werden diese in einem Textfile gespeichert. Weiteres zu den Datensätzen ist dem nächsten Abschnitt 3.2.4 zu entnehmen.

3.2.4 Struktur der Datensätze

Die Datensätze werden in Textfiles auf der SD-Karte im Verzeichnis *CPSDATA/UserID* in der Folge definierter Unterverzeichnisse gespeichert(?). Die WLAN-Scan-Datensätze der Kalibrierungsphase werden im Unterverzeichnis *wifiscans* gespeichert. Jene Datensätze für die Testphase werden unter *wifitestscans* abgelegt. Die Datensätze der Testläufe werden im Unterverzeichnis *testruns* abgelegt.

Für alle erzeugten Datensätze wird in den Dateinamen die Seriennummer des Gerätes eingebunden. Das macht es möglich, parallel mehrere Geräte mit derselben UserID zu verwenden und die Datenkonsistenz zu erhalten, auch wenn die Daten auf dem synchronisierten Verzeichnis zusammengeführt werden.

Als Zeitstempel werden alle Datensätze mit der Systemzeit in den Nanosekunden (System.nanoTime() [65]) ihrer Entstehung abgespeichert. Dieses Format wird verwendet, da die Daten der Sensoren mit der Zeit in Nanosekunden versehen sind.

Kompass-WLAN-Scanner

Der Dateiname eines WLAN-Scans setzt sich aus der PointID und der Seriennummer des Gerätes zusammen. So kann an einem Punkt mit verschiedenen Handys mit der gleichen ProjekID gemessen werden. Dabei kommt es zu keiner Inkonsistenzen der Daten, wenn diese auf dem Dropbox-Ordner zusammengefügt werden.

Bei den WLAN-Scan-Datensätzen wird für jede Position bzw. PunktID eine Datei erzeugt. Ein Datensatz kann aus mehreren Zeilen bestehen, da jede Zeile einen gemessenen AP repräsentiert. Das hat den Vorteil, dass in allen Zeilen die gleiche Anzahl an Werten steht. Eine solche Datei kann leichter in MATLAB eingelesen werden. Über die Zeitstempel des WLANS-Scans können die Zeilen zu einem kompletten Datensatz zusammengeführt werden.

Gespeichert werden die WLAN-Scans auf einem Gerät für eine PointID(?), immer in der gleichen Datei. Neue Werte werden an das Ende der Datei angehängt. Beim Synchronisieren werden die alten Dateien durch die erweiterten ersetzt.

Daten der APs werden in mehreren Zeilen gespeichert. Eine Zeile enthält den Zeitstempel, die PointID, die MAC-Adressen (BSSID), den RSS-Wert, die RichtungsID, den Azimut-Wert, den Netznamen (SSID), den Modellnamen des Handys, Kommentar und ab Android Version 17 die Access Point *scantime*. Diese AP *scantime* enthält den Zeitpunkt, an dem der RSS-Wert des Access Points vom WLAN-Modul gemessen wurde. Da aber für die Testgeräte nur Betriebssysteme bis zur Version 16 verfügbar waren, konnte diese Zeit nicht verwendet werden. In der Arbeit wurde mit der Scan-Startzeit als Zeitstempel gearbeitet. Für zukünftige Untersuchungen steht diese Funktion dennoch zu Verfügung.

Sensor-WLAN-Recorder

Die Datensätze der Sensoren und WLAN-Scans eines Testlaufes werden in jeweils einzelnen Textfiles gespeichert. Alle Dateien eines Testlaufes enthalten zusätzlich im Dateinamen den Startzeitpunkt in Form von der Systemzeit in Millisekunden. So sind die Dateien eines Testlaufes mit Seriennummer und Startzeitpunkt versehen, was Datenkonsistenz auch mit mehreren Geräten beim Zusammenführen sicherstellt.

Ein vollständiger Aufzeichnungsdatensatz besteht aus sieben Textfiles mit den Endungen *info.txt, wifi.txt, cp.txt, acc.txt, com.txt, bar.txt* und *gps.txt*. Die jeweiligen Textfiles werden nur dann erzeugt, wenn die entsprechenden Daten verfügbar waren.

In der *Info.txt* werden Testlaufname, Modellname, SDK, Startdatum, Startzeit, Enddatum, Endzeit und Schrittanzahl (wenn eingeben sonst -1) gespeichert.

Die Datei mit der Endung *wifi.txt* enthält die WLAN-Scans, die während der Aufzeichnung ermittelt wurden. Ein Datensatz wird in mehreren Zeilen mit Zeitstempel, SSID, BSSID und RSS-Wert gespeichert. Die Datensätze sind so aufgebaut wie die des Kompass-WLAN-Scanners.

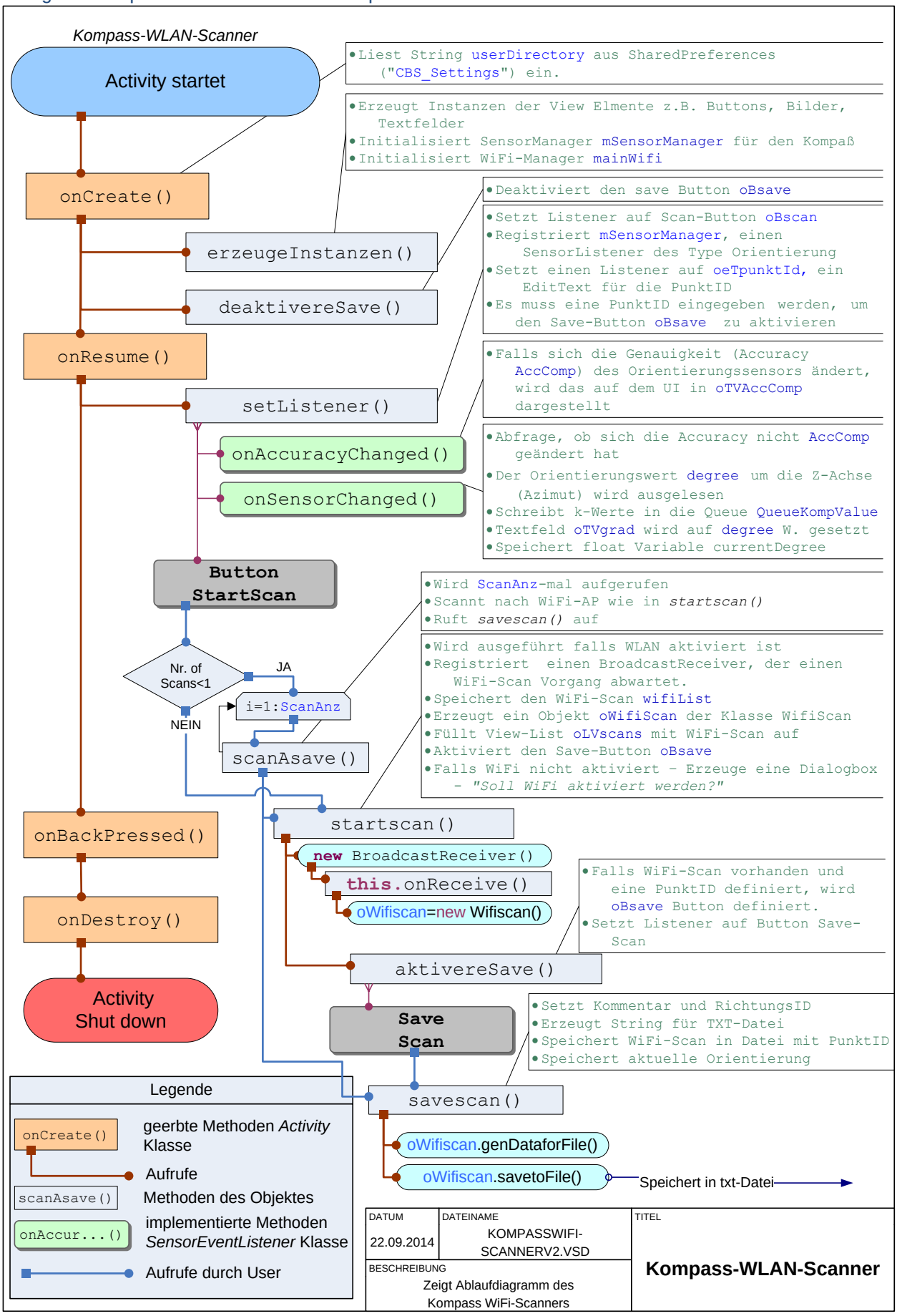
Das mit *cp.txt* endende Textfile beschreibt den Zeitpunkt, an dem ein Referenzpunkt passiert wurde. Beschrieben wird dies mit Zeitstempel und der PointID.

In der Datei mit der Kennung *com.txt* werden die Werte des Orientierungssensors gespeichert. Ein Datensatz besteht aus Zeitstempel, Azimut-Wert (digitale Kompass), Pitch- (Vorwärtsneigung) und Roll-Wert (Seitwärtsneigung) und dem Genauigkeits-Wert mit steigender Genauigkeit von 0-3.

Die Daten des Accelerometers werden im Textfile mit der Endung *acc.txt* gespeichert. Ein Datensatz besteht aus einer Zeile, in der Zeitstempel, die Beschleunigungswerte in x, y- und z-Achse sowie der Accuracy-Wert stehen.

3.2.5 Programmabläufe und Methoden des CPS-App

Abbildung 3-7 Kompass-WLAN-Scanner Ablaufplan und Methoden der Klasse *WiFiScanDB*



Ein Datensatz des Barometers, der in der Datei mit der Kennung *bar.txt* gespeichert ist, besteht aus Zeitstempel, dem Druck in Millibar und dem Genauigkeits-Wert mit steigender Genauigkeit von 0-3.

Die GPS-Koordinaten werden in der Datei mit der Endung *gps.txt* gespeichert. Der Datensatz besteht aus dem Zeitstempel und der Zeit des LocationManager in Millisekunden, dem Breitengrad in Dezimalgrad (lat), dem Längengrad in Dezimalgrad (lon) und dem geschätzten Genauigkeits-Wert in Metern.

Klasse *WiFiscanDB* (Kompass-WLAN-Scanner)

In Abbildung 3-7 ist der Ablauf und die Struktur der Methoden der Klasse *WiFiScanDB* in einem schematischen Ablaufplan dargestellt. Dieses Schema soll die wichtigsten Methoden und die verwendeten Objekte sowie Strukturen erläutern. In Abbildung 3-3 ist die dazu gehörende grafische Benutzeroberfläche dargestellt. Die Methoden `onCreate()` und `onResume()` werden von der Klasse *Activity* geerbt und überschrieben. In der Methode `onCreate()`werden die Elemente der GUI, die in *activity_wifi_scan_db.xml* definiert sind, erzeugt. Diese Elemente sind im Android Packes View [6] enthalten. Diese stellt die Klassen für die Elemente für eine Benutzeroberfläche, also die GUI, zu Verfügung. In `onCreate()`werden außerdem die Objekte der Klasse *WifiManager* und *SensorManager* erzeugt. Diese liefern WLAN-Scans und die Daten des Orientierungssensors.

In der Methode `onCreate()`, die beim Erzeugen der *Activity* ausgeführt wird, wird der Button "Save" deaktiviert. Dieser wird erst aktiviert, wenn ein WLAN-Scan vorhanden ist und die PointID eingegeben wurde.

Die Methode `onResume()` setzt *Event Listeners [66]* auf die Buttons der GUI. Diese "überwachen" die Buttons und führen eine Methode aus, sobald der Button gedrückt wurde. Außerdem wird der Orientierungssensor mit dem *SensorManager* [15] registriert.

Da das Objekt das Interface *SensorListener [13]* implementiert, wird die Methode `onSensorChanged()` ausgeführt, wenn der Sensor neue Werte liefert. In dieser Methode sind die Funktionen des digitalen Kompasses implementiert. Die Sensordaten des Azimuts werden in einer Queue (Warteschlange) gespeichert. Der aktuellste Wert wird grafisch und numerisch auf der GUI dargestellt.

Erst durch das Drücken des *Start-Scan-Buttons* wird eine neue Aktion durchgeführt. Abhängig von der gewählten Anzahl an WLAN-Scans werden die zwei Fälle dahingehend unterschieden, ob ein oder mehrere WLAN-Scans durgeführt werden sollen.

Wenn keine Veränderungen bei der Anzahl der zu tätigenden WLAN-Scans vorgenommen wurden, ist die Variable `ScanAnz` gleich eins. In diesem Fall wird die Funktion `startscan()` aufgerufen.

Die Funktion startet einen WLAN-Scan und initiiert ein Objekt der Klasse *BroadcastReceiver*. Ist der gestartete WLAN-Scan abgeschlossen, wird die Methode `onReceive()`des *Broadcast-Receiver* Objekt ausgeführt. In ihr wird die Methode `getScanResults()` ausgeführt, darin wird wiederum der aktuelle WLAN-Scan als *ScanResult* [12] abgefragt. Dieser enthält die RSS-Werte der gefunden APs und Informationen wie SSID, BSSI usw., wie in [12] beschrieben.

In `onReceive()` wird das Objekt `oWifiscan` der Klasse *WifiScan* erzeugt. Bei der Initialisierung wird dem Objekt der aktuelle Azimut-Wert, das *ScanResult,* die PointID, die RichtungsID, Startzeit, Endzeit und das aktuelle Userverzeichnis übergeben. Dieses Objekt erzeugt ein Listen-Objekt, welches dem Layout-Objekt *ListView,* der Liste der gefunden APs in Abbildung 3-3*,* übergeben werden kann.

Am Ende der Methode wird der *Broadcast-Receiver* abgemeldet. Ansonsten würde diese Methode immer dann ausgeführt werden, wenn im Hintergrund ein WLAN-Scan durchgeführt wird. Das kann passieren, wenn das System oder eine andere App im Hintergrund einen WLAN-Scan initialisiert hat. Dieser WLAN-Scan würde dann von der App übernommen werden. Da ein WLAN-Scan nicht unmittelbar gespeichert wird, kann das dazu führen, dass der WLAN-Scan nicht am gewollten Ort durchgeführt wurde.

Nachdem der komplette WLAN-Scan Vorgang abgeschlossen und verarbeitet wurde, wird der "Scan-Button" aktiviert, falls bzw. sobald eine PointID eingeben wurde. Gespeichert wird der WLAN-Scan erst durch das Betätigen des "Save-Button".

Ist die `ScanAnz` größer als eins, werden mehrere WLAN-Scans hintereinander durchgeführt. In diesem Fall wird die Methode `scanAsave()` entsprechend des Wertens von `ScanAnz` wiederholt durchgeführt. Diese Methode führt WLAN-Scans mit `startscan()` durch, die unmittelbar mit `savescan()` gespeichert werden.

Die Methode `savescan()` wird entweder durch das Drücken des Button *Save-Scan* oder durch die Methode `scanAsave()` ausgeführt. In `savescan()` werden die Methoden von `oWifiscan` benutzt, um die Daten aus WLAN-Scan, Kompass und Metainformationen in der Datei zu speichern. Wie und wo die Daten gespeichert werden, wurde in Abschnitt 3.2.4 erklärt.

Klasse SensorRec (Sensor-WLAN-Recorder)

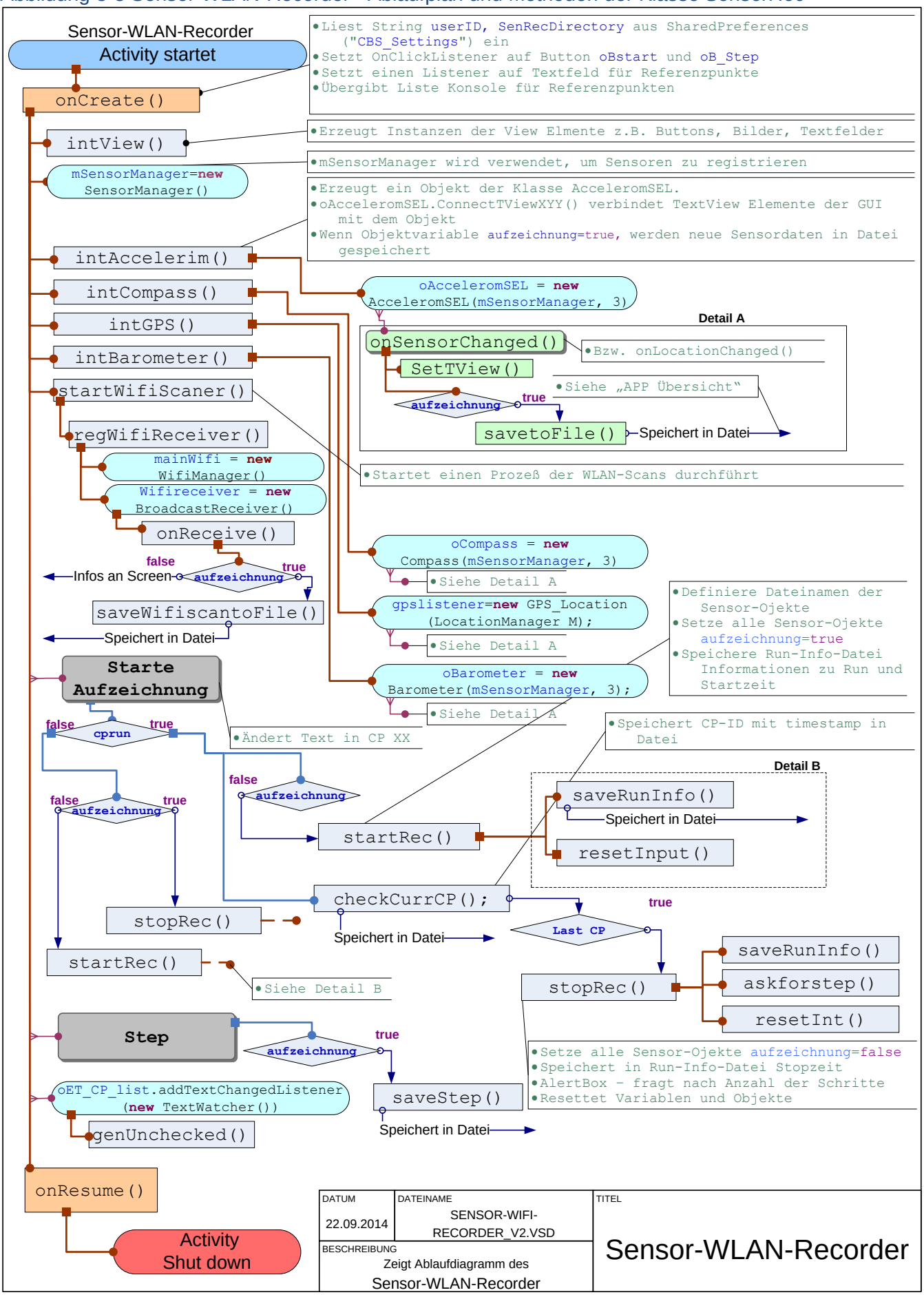
Einen Überblick über die wichtigsten Methoden und den Programmablauf der Klasse *SensorRec* gibt die Abbildung 3-8. In Abbildung 3-4 ist die dazugehörige Benutzeroberfläche zu sehen.

Die Methode `onCreate()` wird überschrieben. Darin werden alle benötigten Objekte erzeugt und Listener auf die Buttons und die Textfelder gesetzt. Es werden die Elemente der GUI initialisiert und das Objekt `mSensorManager` der Klasse *SensorManager* erzeugt. Mit dem Sensor Manager werden alle Sensoren registriert und verwaltet.

Wenn das Objekt gestartet wurde, befindet es sich noch nicht im Aufzeichnungsmodus, jedoch liefern alle Sensoren und Module permanent Daten, die auf der GUI dargestellt werden. Für Accelerometer, Orientierung-Sensor und Barometer wurden eigene Klassen implementiert. Diese Klassen verwenden das Interface *SensorEventListener*. Liefert ein Sensor einen neuen Wert, wird die Methode `onSensorChanged()` ausgeführt. Mit den aktuellen Werten werden die entsprechenden Elemente der GUI aktualisiert. Wenn sich das Objekt im Aufzeichnungsmodus befindet, werden die Datensätze der Sensoren als String erzeugt und mit der Methode `savetoFile()` in den jeweiligen Dateien gespeichert.

Zum Aufzeichnen der GPS-Koordinaten wurde die Klasse *GPS_Location* programmiert. Wie bei den Objekten der Sensoren übernimmt dieses die Darstellung der Daten in der GUI und deren Speicherung. Initialisiert wird das Objekt `gpsListener` mit einem

Abbildung 3-8 Sensor-WLAN-Recorder - Ablaufplan und Methoden der Klasse *SensorRec*



LoactionManger [10]. Der `gpsListener` benutzt das Interface L*ocationListener* [9]. Wird eine Änderung der GPS-Koordinaten registriert, wird die Methode `onLocationChanged()` ausgeführt. Darin werden die Daten auf der GUI aktualisiert und die Methode `savetoFile()` aufgerufen, falls sich das Objekt im Aufzeichnungsmodus befindet.

In der Klasse *SensorRec* ist der WLAN-Scanner implementiert. Dieser funktioniert ähnlich wie in der Klasse *WiFiScanDB*m, also dem Kompass-WLAN-Scanner. Der Unterschied besteht darin, dass die Speicherung der WLAN-Scans in der Klasse *SensorRec* direkt implementiert ist und nur Zeitstempel, SSID, BSSID und RSS erfasst werden, wenn der

Aufzeichnungsmodus aktiviert ist. Der WLAN-Scanner läuft permanent im Hintergrund und gibt die Anzahl der gefundenen APs auf der GUI aus.

Durch das Betätigen des *Starte-Aufzeichnung-Button* `oBstart` werden alle Objekte mit der Methode `startRec()`in den Aufzeichnungsmodus gesetzt. Sensor- und GPS-Daten, RSS-Werte und Testlauf-Informationen werden auf der SD-Karte als Textdatei abgespeichert. Die Datensätze und Speicherpfade wurden in 3.2.4 erläutert.

Zusätzlich zu den Sensor-Daten können Referenzpunkte gesetzt werden. Diese müssen zuvor in dem Textfeld *„Referenzpunkte Reihenfolge“* eingegeben werden. Auf das Textfeld `oET_CP_list` wird dazu ein Listener gesetzt. Wird eine Reihenfolge eingegeben, wird das Programm über die Variable `cprun` in den Referenzpunkte-Modus gesetzt. In diesem Modus wird der Text des `oBstart` auf den nächsten zu erreichenden Referenzpunkt geändert.

Dies wird von der Methode `genUnchecked()` durchgeführt. Mit dem Betätigen des `oBstart`-Button wird der Referenzpunkt bestätigt. Dabei werden Zeitpunkt und Referenzpunkt gespeichert. Der Ablauf wurde bereits bei den Erläuterungen zur Funktionsweise des Sensor-WLAN-Recorder erklärt.

Außerdem ist es möglich, währenden das System im Aufzeichnungsmodus ist, *Step-Button* (siehe Abbildung 3-4) zu drücken und mit der Methode `saveStep()` wird die Systemzeit in Nanosekunden in der Dateien *_step.txt* gespeichert.

3.3 MATLAB-Framework

Die mit dem CPS-App erzeugten WLAN-Scan- und Testlauf-Datensätze können mit den in diesem Abschnitt beschriebenen Funktionen in MATLAB importiert werden. Dieses Framework an Funktionen (MFW) macht es möglich, mit den aufgezeichneten Datensätzen unterschiedliche Szenarien mit verschiedenen Methoden bzw. Algorithmen zu untersuchen.

Im ersten Teil dieses Abschnittes wird beschrieben, wie die Datensätze im MATLAB-Framework dargestellt und aufgebaut sind. Im nächsten Teil wird beschrieben, wie die WLAN-Scan-Datensätze des Kompass-WLAN-Scanners importiert werden. Im Anschluss folgt die Erläuterung zum Import der Testlauf-Datensätze in das Framework. Wie die ermittelten Positionen grafisch auf einer Karte dargestellt werden können, ist im letzten Teil erklärt.

3.3.1 Hauptdatensätze

Die in 3.2.4 beschrieben Datensätze werden im MATLAB Framework als *Structures* [58] zur Verfügung gestellt. *Structures* sind Datentypen in MATLAB, die es erlauben, Daten verschiedener Typen in Gruppen zusammenzufassen. Es werden Datenfelder verwendet, die *Fields* genannt werden. Diese können beliebige Bezeichnungen erhalten, die es erleichtern, die Struktur der Daten zu beschreiben. Die verwendeten Datensätze werden in diesem Kapitel dargelegt.

In **Fehler! Verweisquelle konnte nicht gefunden werden.** wird skizziert, wie die Daten des ompass-WLAN-Scanners in Matlab importiert werden. Die Hauptdatensätze `WIFISCANS`, `WIFIPOINTS` und `WIFIDIRECTION` sind in der Grafik grün markiert und werden im Anschluss erläutert. Wie die Testlauf-Datensätze in die Structure Variable `TESTRUN` geladen werden, wird im Abschnitt 3.3.3 und in Abbildung 3-13 dargelegt.

Structure Variable `WIFISCANS`

In der Structure-Variable `WIFISCANS` werden WLAN-Scans mit allen Daten gespeichert. In Tabelle 3-1 ist der Aufbau der Structure erläutert. Die RSS-Werte der APs werden in den Vektoren *Scan1* und *Scan2* zusammengefasst, diese entsprechen den RSSI-Vektoren (siehe 4.3.1). Ein Beispiel für einen Datensatz eines WLAN-Scans ist in Abbildung 3-9 grafisch dargestellt. Zu sehen sind die gemessenen RSS-Werte der gefundenen APs, der Azimut-Wert des Orientierungssensors und die zum WLAN-Scan gehörenden Metadaten wie *pointID* und *directionID* und Kommentar (*comment*).

Ein WLAN-Scan wird durch einen ganzzahligen Vektor in MATLAB repräsentiert. Gespeichert wird der RSSI-Vektor in zwei verschiedenen Vektoren, *Scan1* und *Scan2*. Beide Vektoren enthalten dieselben Daten. Diese Unterscheidung erfolgt lediglich, um spätere Berechnungen vereinfacht durchführen zu können. Der Unterschied besteht darin, dass in Vektor *Scan1* (F. 4-3) nicht erreichte AP mit einem Minimalwert (-101dBm) repräsentiert und in Vektor *Scan2* (F. 4-5) als NaN („*Not a Number*“) definiert werden. Gespeichert werden in einem WLAN-Scan die empfangen Signalstärken der gefundenen APs (Access Points). Die Dimension aller RSSI-Vektoren ist gleich n, das ist die Anzahl aller gefunden APs.

Abbildung 3-9 Grafische Darstellung eines Datensatzes der Structure-Variable WIFISCANS

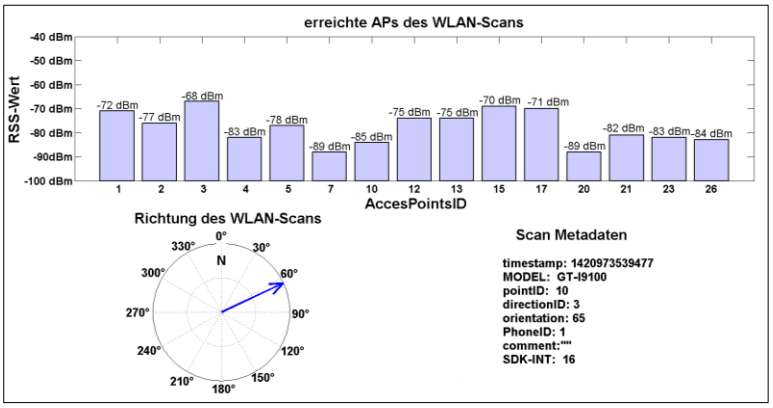


Tabelle 3-1 Aufbau der Variabel WIFISCANS bzw. eines WLAN-Datensatzes

WIFISCANS		<1xW struct>	Jeder Datensatz entspricht einem vollständigen WLAN-Scan und einhält RSSI-Vektoren und alle Metainformationen.
	timestamp	<double>	Zeitstempel: Startzeit d. WLAN -Scans in Nanosekunden - System.*nanoTime*()
	MODEL	<char>	Modellbezeichnung des Gerätes bzw. des Handys
	pointID	<double>	PointID – Die ID ermöglicht eine eindeutige Zuordnung zu einer Position.
	dirction_ID	<double>	Vom Benutzer eingegebene ID für die Richtungskat. [1, 2, 3, 4] sind möglich.
	orientation	<double>	Kompass des Handys. Azimut-Wert wird vom Orientierungssensor ermittelt.
	PhoneID	<double>	ID des verwendeten Telefons wird anhand der Modellbezeichnung definiert.
	comment	<char>	Möglicher Kommentar des Benutzers.
	SDK_INT	<double>	API Version.
	Scan1	<1xN double>	**RSSI-Vektor1**(1:N): RSS-Werte in dBm der APs - nicht erreichte AP:-101dbm.
	Scan2	<1xN double>	**RSSI- Vektor 2**(1:N): RSS-Werte in dBm der APs - nicht erreichte AP: NaN

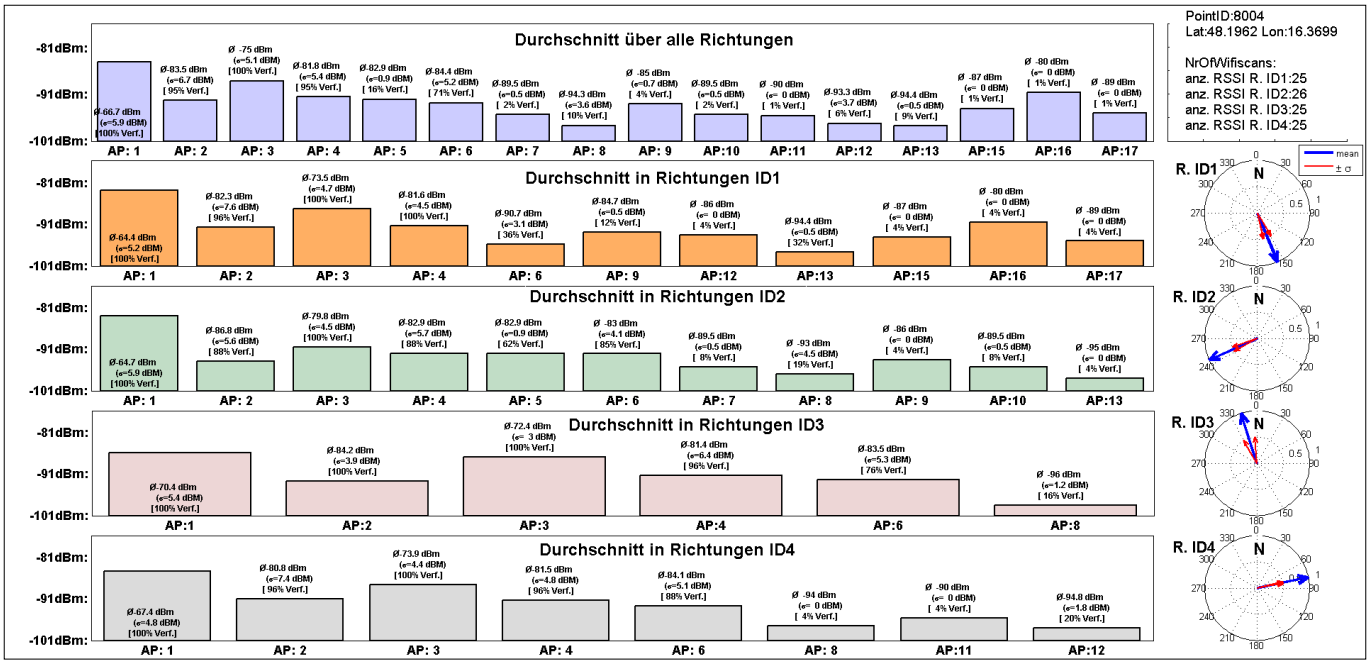
N… Anz. aller AP in DB; W… Anzahl aller WLAN-Scans in der DB;

Structure Variable WIFIPOINTS

In der Structure WIFIPOINTS werden alle WLAN-Scan-Datensätze (WIFISCANS) anhand ihrer *PointID* zusammengefasst. In den Matrizen LevelMat1 und LevelMat2 sind die RSSI-Vektoren Scan1 und Scan2 zeilenweise untereinander gehängt. Mit diesen Matrizen wurde das arithmetische Mittel, der Median, die Standardabweichung und die Varianz der RSS-Werte berechnet. Weiters werden auch die ermittelten Richtungen während des WLAN-Scans gespeichert. Die Datensätze der WLAN-Scans können nach ihrer *RichungsID* bzw. Richtungsklasse eingeteilt werden. Für die Arbeit wurden 4 Klassen verwendet. Die Daten sind in den Feldern *D1*, *D2*, *D3* und *D4* als Substructures gespeichert. Diese Substructures sind nach demselben Schema aufgebaut wie die Structure WIFIPOINTS. Die Substructures enthalten die gleichen Felder, jedoch nur die WLAN-Scans mit derselben Richtungskategorie bzw. *RichtungsIDs*.

Alle Felder der Structure WIFIPOINTS sind in Tabelle 3-2 erläutert. In Abbildung 3-10 ist ein Datensatz grafisch dargestellt.

Abbildung 3-10 Grafische Darstellung eines Datensatzes der Structure-Variable `WIFIPOINTS`



Gezeigt werden in der Grafik die Median-Werte der RSS-Werte. Das erste Balkendiagramm zeigt die Medianwerte über alle Richtungsklassen hinweg berechnet. Die vier weiteren Diagramme zeigen die Medianwerte pro Richtungsklasse. Dargestellt sind ebenfalls Informationen zum Punkt und der getätigten Anzahl an WLAN-Scans. Die mittleren Azimutwerte und ihre Varianzen pro *RichtungsID* sind neben den Balkendiagrammen zu sehen. Der Grafik ist zu entnehmen, dass nicht alle APs in allen Richtungsklassen ähnliche RSS-Werte haben bzw. einige APs nur in bestimmten Richtungen erreichbar waren.

Tabelle 3-2 Structure –Aufbau der Variable `WIFIPOINTS`

WIFIPOINTS		<1xP struct>	Jeder Datensatz enthält alle WLAN-Scans an einem Punkt. Aus dem RSSI-Vektoren werden akinetisches Mittel, Median, Standardabweichung, Varianz und Verfügbarkeit der APs in alle Richtungen einzeln ermittelt.
	pointID	<double>	PointID – Dient zur Identifiziert einer Position über eine ID.
	LevelMat1	<WPxN double>	Alle RSSI-Vektoren an einem Punkt mit Scan1. Eine Zeile entspricht einem RSSI-Vektor.
	LevelMat2	<WPxN double>	Alle RSSI-Vektoren einem Punkt mit Scan2. Eine Zeile entspricht einem RSSI-Vektor.
	mean1	<1xN double>	Mittelwert-Vektor berechnet durch mean(LevelMat1).
	mean2	<1xN double>	Mittelwert-Vektor berechnet durch nanmean(LevelMat2)
	median1	<1xN double>	Median-Vektor berechnet durch median (LevelMat1)
	median2	<1xN double>	Median-Vektor berechnet durch nanmedian (LevelMat2)
	availability	<1xN double>	Verfügbarkeit der AP an dem Punkt in %
	std1	<1xN double>	Vektor mit Standardabweichung der RSSI1s mit Scan1.
	var1	<1xN double>	Vektor mit Varianz der RSSI1s mit Scan1.
	std2	<1xN double>	Vektor mit Standardabweichung der RSSI2s mit Scan2.
	var2	<1xN double>	Vektor mit Varianz der RSSI2s mit Scan2.
	orientationsD	<WPx1 double>	Ermittelte Azimut Orientierung während des WiFi-Scans
	D1	<struct>	WIFIPOINT in Richtungs-Kategorie 1 – Enthält Felder LevelMat1 … Var2 in Richtung 1
	D2	<struct>	WIFIPOINT in Richtungs-Kategorie 2 – Enthält Felder LevelMat1 … Var2 in Richtung 2
	D3	<struct>	WIFIPOINT in Richtungs-Kategorie 3 – Enthält Felder LevelMat1 … Var2 in Richtung 3
	D4	<struct>	WIFIPOINT in Richtungs-Kategorie 4 – Enthält Felder LevelMat1 … Var2 in Richtung 4

N… Anz. aller AP in DB; W… Anzahl aller WiFi-Scans in der DB; P… Anzahl alle PointID in der DB;

WP… Anzahl der WiFi-Scan an dem Punkt x

Structure Variable `WIFIDIRECTION`

Enthält die Daten der Substructures pro *RichtungsID* bzw. Richtungsklasse. Zum Erstellen wurden die Daten der Felder *D1*, *D2*, *D3* und *D4* der Structure `WIFIPOINTS` zusammengeführt. Diese Variable dient dazu, Fingerprinting in je Richtungskategorie zu testen. Die Idee wird in 5.1.2 betrachtet und wird hier nicht näher ausgeführt. Der Aufbau der Structure `WIFIPOINTS` entspricht der Structure `WIFIPOINTS` pro Richtungsklasse.

Structure Variable `WIFITESTSCANS`

Enthält die RSSI-Vektoren der Testphase und die Metadaten der WLAN-Scans. Der Aufbau der Structure ist ident zur Structure `WIFISCANS,` die in

Tabelle 3-1 beschrieben wurde.

Structure Variable `TESTRUNS`

Die Testlauf-Datensätze sind in der Structure-Variable `TESTRUNS` gespeichert. Die Abbildung 3-13 zeigt, wie die vom WLAN-Sensor-Recorder erzeugten Datensätze in MATLAB importiert werden. Der Aufbau der Structure ist in Tabelle 3-3 erläutert.

Die Daten der Sensoren sind in den jeweiligen Feldern als Substructures gespeichert. Jede Substructure enthält die Daten der in 3.2.4 beschriebenen Datensätze. Alle Datensätze wurden mit einem Zeitstempel (*timestamp*) versehen. Mit diesen Zeitstempeln können die Datensätze miteinander in Verbindung gebracht werden.

In Abbildung 3-11 ist ein Datensatz grafisch dargestellt. In der oberen Grafik werden die ermittelten APs während eines Testlaufs angezeigt. Die Größe der Balken zeigt anteilsmäßig die Stärke der Signale der APs. Je stärker ein Signal bzw. je größer ein RSS-Wert, desto mehr Anteil hat der entsprechenden Teil des APs. Gekennzeichnet sind die Balken mit den IDs der APs. Diese Grafik stellt dar, wie die einzelnen APs im Verlauf des Testlaufes erscheinen und wieder außer Reichweite gelangen. Zu sehen ist ebenfalls, wie die RSS-Werte der einzelnen APs zu- und abnehmen.

Dargestellt sind die *Daten des Accelerometers* in den 3-Achsen, wie in 2.3.1 beschrieben. Die Daten der z-Achse sind grün, die der x-Achse blau und die der y-Achse rot dargestellt. Die aufgezeichneten Daten des *digitalen Kompasses* bzw. des Azimut-Wertes vom Orientierungssensor sind als blaue Kurve darunter abgebildet.

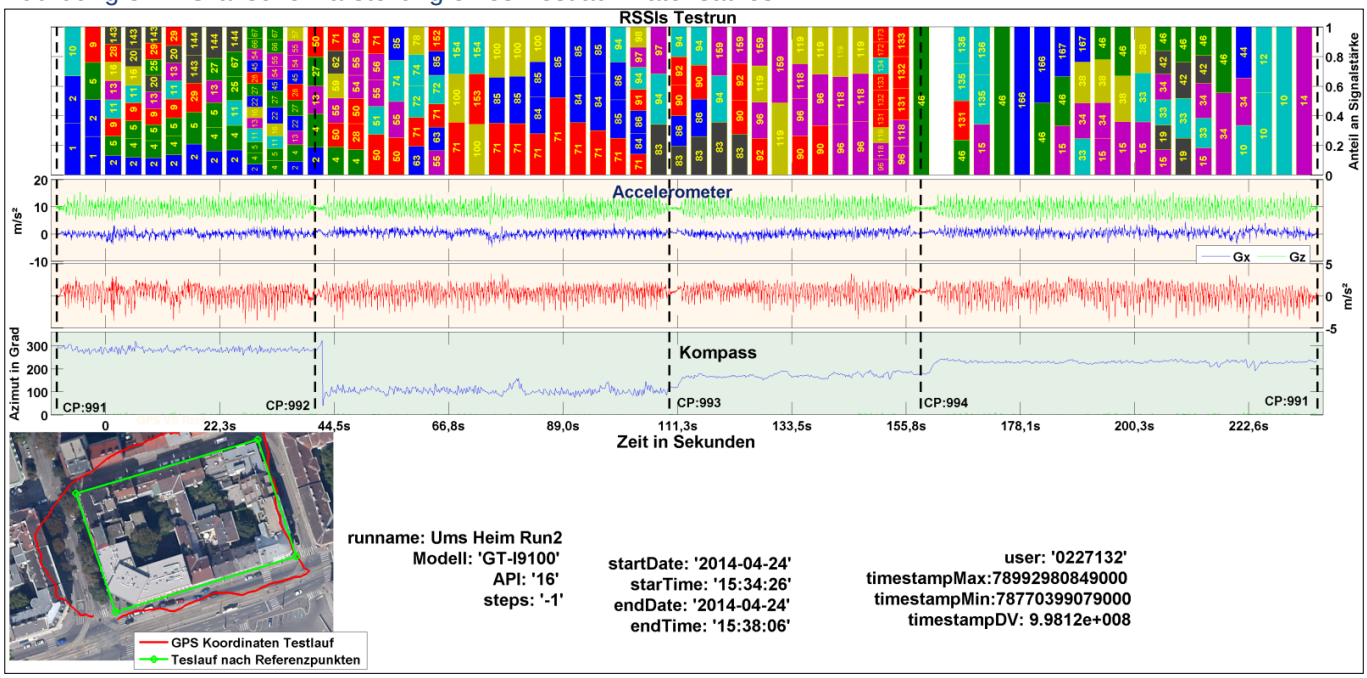
Die GPS-Daten und der Referenzweg anhand der Referenzpunkte können in der Karte links unten betrachtet werden. Die Metadaten eines Testlaufes sind daneben angeführt.

Tabelle 3-3 Aufbau der Variabel TESTRUNS bzw. eines Testlauf-Datensatzes

TESTRUNS		<1xT struct>	Ein Datensatz eines Testlaufes. *timestamp:* Zeitstempel; xxxN Anzahl der Datensätze ; *acc* (Genauigkeitswert);
	acc	<accNx1 struct>	Daten des Beschleunigungssensors - *timestamp*; *Gx, Gy, Gz* (Daten in 3-Achsen); *acc*
	com	<comNx1 struct>	Daten des Orientierungssensors - *timestamp*; *x, y, z* (3 Winkel siehe 2.3.3); *acc*
	cp	<cpNx1 struct>	Referenzpunkte – *timestamp; cpID* (PointID)
	gps	<gpsNx1 struct>	GPS-Koordinaten(?) – *timestamp*; *timestampGPS*; *lat*; *lon*; *acc*
	wifi	<wifiNx1 struct>	WLAN-Scan-Daten des Testlaufes *timestamp*; *SSID*; *BSSID*; *level*; *AP_ID*;
	step	< stepNx1 double>	Zeitstempel des Step-Buttons
	runname	<string>	Bezeichnung des Testlauf
	Modell	<string>	Handymodell des Testlaufes
	API	<string>	Androidversion des Testlaufes
	startDate	<string>	Startdatum JJJ-MM-TT des Testlaufes
	starTime	<string>	Startzeit hh:mm:ss des Testlaufes
	endDate	<string>	Enddatum JJJ-MM-TT des Testlaufes
	endTime	<string>	Endzeit hh:mm:ss des Testlaufes
	steps	<string>	gezählte Schrittanzahl; Falls keine Schritte gezählt wurden, wird -1 verwendet
	user	<string>	Eingeben *UserID*
	timestampMax	<string>	Zeitstempel mit höchstem Wert aller Daten
	timestampMin	<string>	Zeitstempel mit geringstem Wert aller Daten
	timestampDV	<string>	Geringste Zeitdifferenz bei Sensordaten
	wifiscans	< wifiscansNx1 struct>	RSSI-Vektoren des Testlaufes; *timestamp*; *Scan1*; *Scan2*

T… Anz. aller Testläufe in DB; xxxN… Anzahl der Sensordaten des Sensors xxx; wifiscansN… Anzahl alle WLAN-Scan je Testlauf;

Abbildung 3-11 Grafische Darstellung eines Testlauf-Datensatzes



3.3.2 Import der Daten des Kompass-WLAN-Scanners

Wie der Import und die Analyse der Daten des Kompass-WLAN-Scanners in MATLAB durchgeführt wird, ist in **Fehler! Verweisquelle konnte nicht gefunden werden.** skizziert. ie Funktionen sind in [37-MFW\functions] zu finden.

Wie in Abschnitt 3.2.4 beschrieben, werden die Datensätze des Kompass-WLAN-Scanners in den User-Verzeichnissen gespeichert. Diese befinden sich auf der SD-Karte des Gerätes im Verzeichnis *CPSData*. Die WLAN-Scans der Kalibrierungsphase sind in den

User-Verzeichnissen im Unterverzeichnis *wifiscans* gespeichert. Die für die Simulation der Positionierungsphase benutzen WLAN-Scans werden im Unterverzeichnis *wifitestscans* abgelegt.

Mit den Apps Dropbox und Dropbox Sync werden die Daten auf ein DropBox-Konto geladen. Der Rechner, auf dem das MALTAB-Framework (MFW) entwickelt wurde, war ebenfalls mit diesem DropBox-Konto verbunden. Dadurch konnte das MFW direkt auf diese Daten zugreifen, sobald diese synchronisiert wurden.

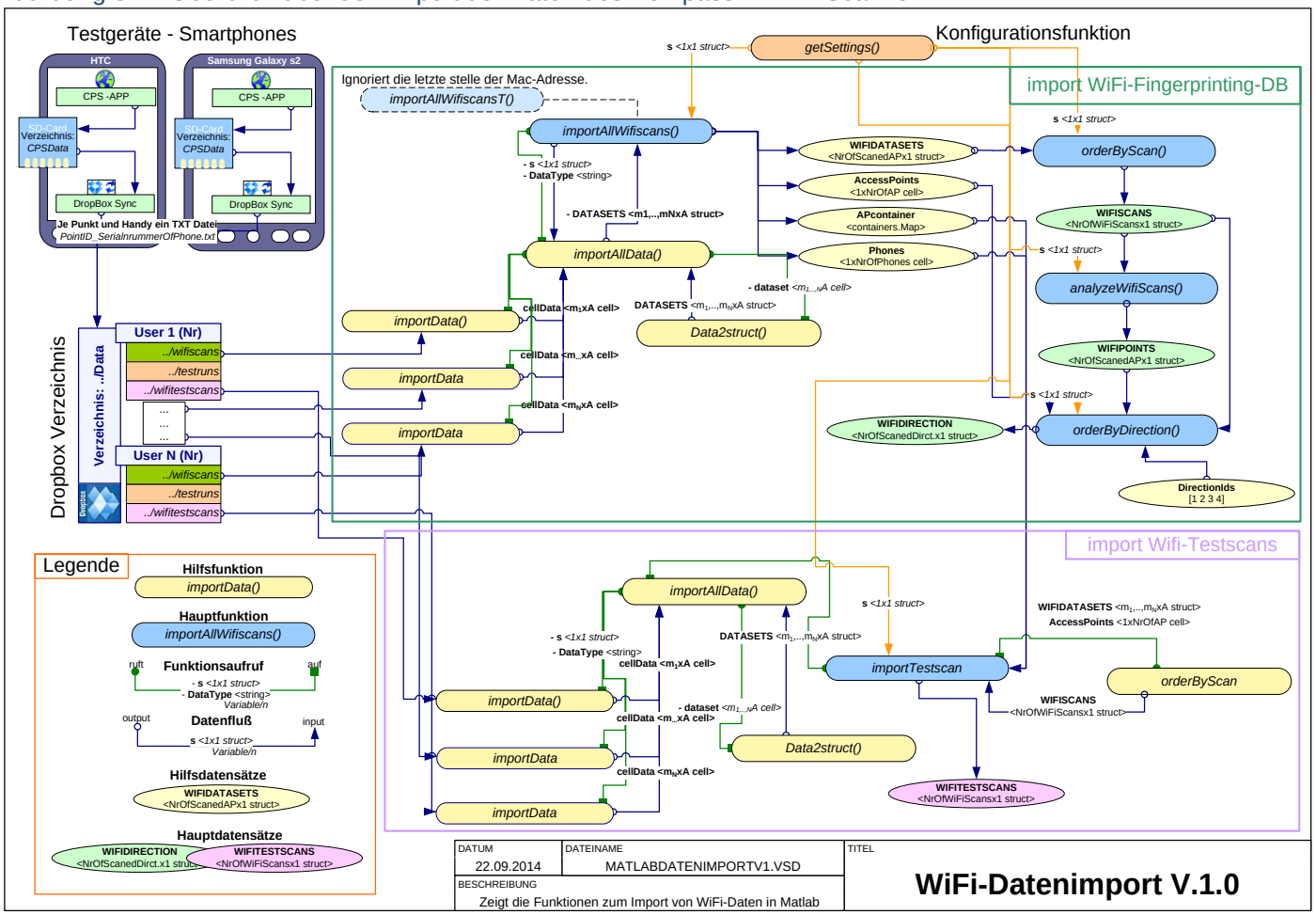
Für die Konfiguration des MFW ist die Funktion `getSettings()` zuständig. Mit ihr wird die Konfigurations-Variable `s` erzeugt, die alle Einstellungen speichert. Gespeichert werden z.B. der Pfad zu den Daten und Karten, der Minimalwert für nicht erreichte APs, Arten der Sensoren und Aufbau der Daten und die zu verwendenden User-Verzeichnisse.

Der Funktion `importAllWifiscans(s)` wird die Konfigurations-Variable übergeben. Diese Funktion importiert alle Files aller erfassten PointIDs, die im Unterverzeichnis *wifiscans* gespeichert sind. Diese Dateien enthalten die WLAN-Scan Datensätze. Es werden alle Daten aller UserIDs und PointIDs im `WIFIDATASET` zusammengefasst, die in der Konfigurations-Variable `s` bestimmt wurden. Die Funktion importData() öffnet die einzelnen Dateien und verwendet die Matlab-Funktion textscan(), um die Daten zu laden. Diese Funktion wird für alle Datensätze verwendet und über die Variable `s` konfiguriert. Das Format für den Import der verschieden Datensätze ist dort definiert. Die importierten Daten werden als CellArray an die Funktion `ImportAllData()` übergeben. Diese fügt alle Daten aller UserID zu einem Datensatz zusammen und wandelt diese in eine Structure mit der Funktion `Data2struct()` um.

In `importAllWifiscans()` werden, nachdem die Variable `WIFIDATASET` importiert wurde, den APs anhand der MAC-Adresse eindeutige IDs zugewiesen. Um eine bessere Laufzeit zu erreichen, werden containers.Map-Objekte [59] verwendet. Diese erlauben es, für bereits vergebene MAC-Adressen die IDs schnell wiederzufinden. Alle MAC-Adressen sind im CellArray `AccesPoints` gespeichert, die Indizes der Adressen entsprechen den AP-IDs. Für die verwendeten Geräte bzw. Handys werden auf die gleiche Weise IDs vergeben. Die IDs der APs und Handys werden jedem Datensatz der Structure `WIFIDATASET` hinzugefügt. Der Aufbau der Datensätze entspricht jener der WLAN-Scan-Datensätze, wie sie in 3.2.4 beschrieben wurden.

Die Funktion `orderByScan()` führt die WLAN-Scan Datensätze in der Structure `WIFIDATASET` zur Structure Variable `WIFISCANS` zusammen und gibt sie aus. Dabei werden die RSS-Werte zu RSSI-Vektoren zusammengeführt. Alle erfassten RSSI-Vektoren haben die gleiche Dimension, diese ergibt sich aus der Anzahl der gefundenen APs. Die Struktur der RSSI-Vektoren wird genauer in 4.3.1 ausgeführt.

Abbildung 3-12 Überblick über den Import der Daten des Kompass-WLAN-Scanner



In `analyzeWiFiScans()` werden die WLAN-Scans in der Structure `WIFISCANS` den *PointIDs* zugeordnet. Die RSSI-Vektoren mit den gleichen *PointIDs* werden aneinandergehängt und in den Zeilen der Matrizen *LevelMat1* und *LevelMat1* gespeichert. Für jede Position werden draus arithmetische Mittel, Median, Varianz und Standardabweichung der RSS-Werte der APs berechnet.

Da es die zwei Darstellungsformen *LevelMat1* und *LevelMat1* für die erfassten RSSI-Vektoren gibt, werden die Berechnungen auf jede Form angewandt. Weiter wird die Verfügbarkeit der APs an einem Punkt berechnet. Dieselben Berechnungen werden für alle vier Richtungskategorien durchgeführt. Ausgegeben wird dies in der **Structure** Variable WIFIPOINTS.

Die Funktion `orderByDiretion()` führt die Datensätze aus der Structure `WIFIPOINTS` je nach Richtungskategorien zusammen. So können RSSI-Vektoren mit gleicher RichtungsID miteinander verglichen werden. Diese Funktion erzeugt die Structure Variable `WIFIDIRECTION`.

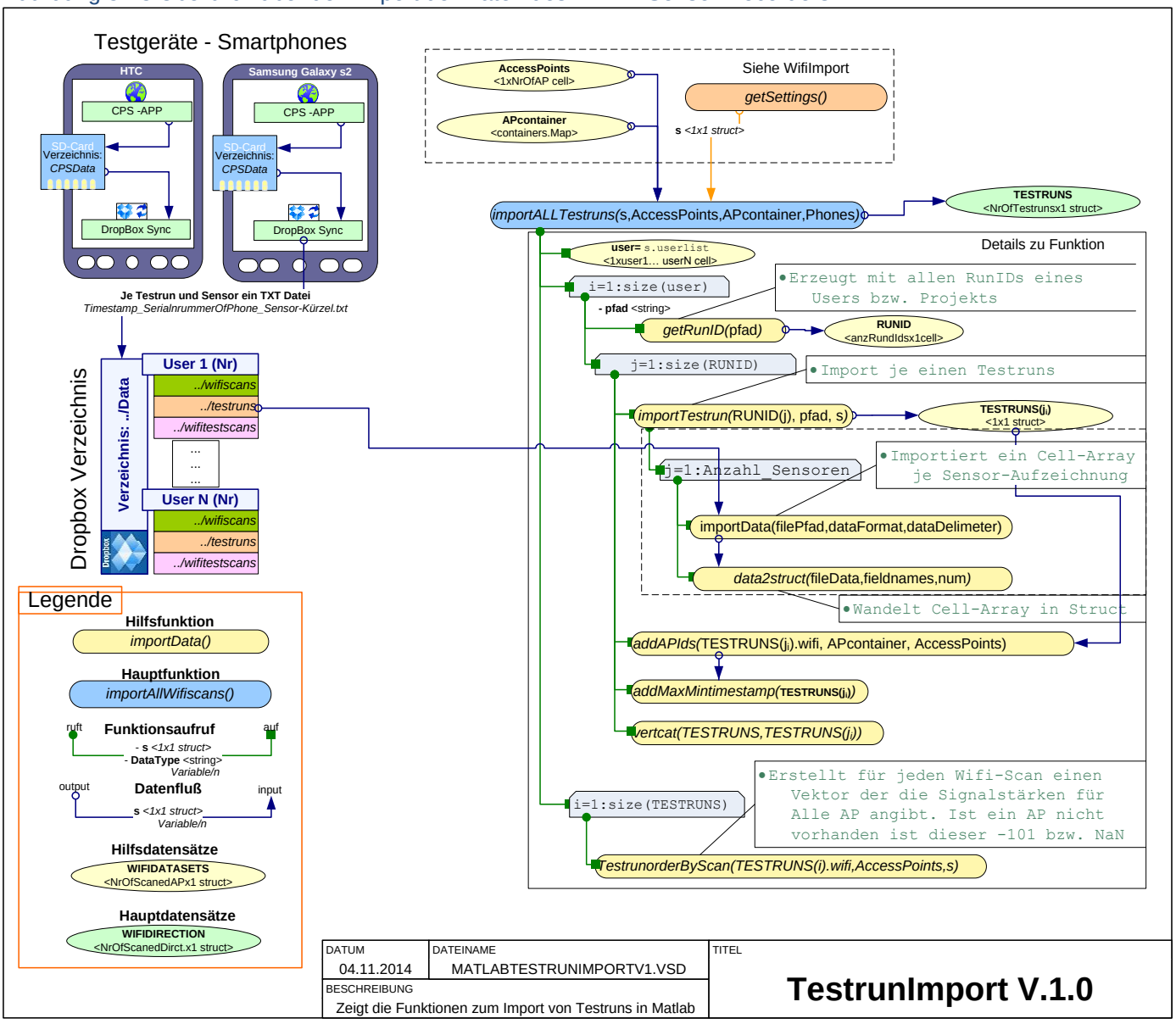
Die Funktion `importTestscans()` lädt die Daten der Structure Variable `WIFITESTSCANS` und verwendet die Funktionen `ImportAllData()` und `Data2struct()`, um die Daten der WLAN-Scans der Testphase zu laden. Wie die Funktion `orderByScan()` ordnet `importTestscans()` RSS-Werte der gescannten APs zu in einem RSSI-Vektor und versieht jeden Datensatz mit den entsprechend Metadaten.

Der Funktion `importTestscans()` werden auch das *CellArray* `AccesPoints` und die *Container.Map* `APcontainer` übergeben. Diese werden verwendet, um bekannten APs die bereits gewählten IDs zuzuweisen. Wenn nicht erfasste APs in den Daten gefunden werden, bekommen diese neue IDs zugewiesen und werden in `AccesPoints` und `APcontainer` angehängt und gespeichert. Damit haben die RSSI-Vektoren der Structure Variable `WIFITESTSCANS` und der Structure Variable `WIFISCANS` den gleichen Aufbau und verwenden dieselben APs-IDs.

3.3.3 Import der Daten des WLAN-Sensor-Recorders

Der Ablauf, nach dem die Structure Variable `TESTRUNS` mit der Funktion `importALLTestruns()` erzeugt wird, ist in Abbildung 3-13 skizziert. Die Funktion verwendet die Konfigurationsvariable `s`. Diese definiert auch, welche User-Verzeichnisse (`s.userlist`) importiert werden sollen.

Abbildung 3-13 Überblick über den Import der Daten des WLAN-Sensor-Recorders



Der Funktion werden das *CellArray* [51] `AccesPoints` und die *Container.Map* `APcontainer` übergeben. Diese werden verwendet, um den APs der erfassten WLAN-

Scans während des Testlaufes die gleichen AP-IDs wie in `WIFISCANS` zuzuweisen. Wenn unbekannte APs in den Testläufen erfasst wurden, bekommen diese neue IDs zugewiesen und werden zu den Variablen `AccesPoints` und `APcontainer` hinzugefügt.

Die Funktion `importALLTestruns()` wird so lange durchlaufen, bis alle User-Verzeichnisse in `s.userlist` importiert wurden. Die Testläufe werden nacheinander angehängt.

Um ein Userverzeichnis zur importieren, wird erst die Funktion `getRunID()` aufgerufen. Diese ermittelt die Anzahl der Testläufe im Verzeichnis und deren `RUNIDs`. Die `RUNIDs` setzt sich aus dem Zeitstempel und der Seriennummer zusammen, wie in Abschnitt 3.2.4 erklärt wurde.

Anhand der Variable `RUNID` werden die Testlauf-Datensätze mit der Funktion `importTestrun()` geladen. Diese verwendet die Methoden `importData()` und die Konfigurationsvariable `s` um die aufgezeichneten Datensätze zu laden. Der Aufbau eines Datensatzes ist im Feld `s.DatensatzName.dataFormat` definiert. So können beliebige Datensätze, die in der Konfigurationsvariable `s` definiert wurden, in eine Structure geladen werden. Das macht das MATLAB-Framework schnell und einfach erweiterbar.

Die Funktion `addMaXMintimestamp()` ermittelt den kleinsten und größten Zeitstempel-Wert aller Datensätze. Dies sind die Zeitpunkte mit dem ersten bzw. letzten Datensatz und erleichtern das Zusammenführen der Daten.

Wenn alle `TESTRUNS` importiert wurden, werden die RSS-Werte der WLAN-Scans zu Vektoren zusammengefasst. Das wird von der Funktion `TestrunodrderByScan()` erledigt. In ihr werden auch die RSSI-Vektoren für die jeweiligen Testläufe erstellt.

3.3.4 Einpassung und Einzeichnung der Koordinaten

Das MATLAB Framework bietet die Möglichkeit, Positionen bzw. Koordinaten auf einer Karte darzustellen. In Android liefert das GPS-Modul die Geokoordinaten in Dezimalgrad, diese können auch von Google Maps verarbeitet und angezeigt werden. Um diese GPS-Koordinaten in einem von MATLAB geladenem Bild anzeigen zu können, müssen diese in x- und y-Koordinaten des Bildes transformiert werden. Durch das Verbinden einzelner Positionen anhand ihrer Koordinaten können diese zu einer Strecke vereinigt werden.

Prinzipiell muss beim Kartennetzentwurf [83] die gekrümmte, dreidimensionale Oberfläche der Erde auf eine flache, zweidimensionale Karte projiziert werden. In Google Maps wird die Mercator-Projektion [82] verwendet. Sie ist eine Form der Zylinderprojektion. Bei dieser Projektion wird in Richtung der Zylinderachse passend verzerrt, um eine winkeltreue Abbildung der Erdoberfläche zu erreichen. Durch die Winkeltreue bleiben geometrische Formen im Kleinen unverzerrt.

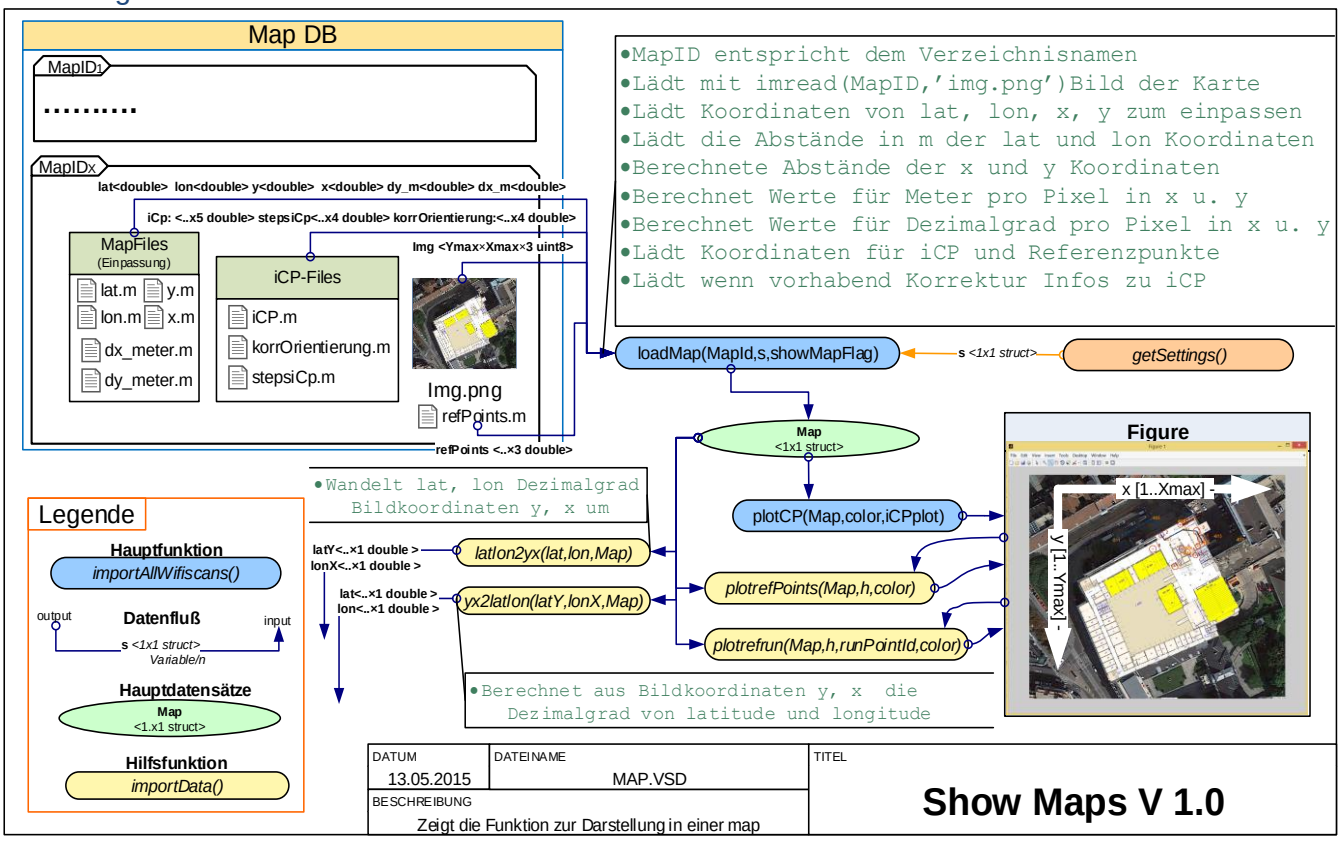
Für die Arbeit wurden kleine Kartenausschnitte im Bereich von wenigen hundert Metern aus Google Maps verwendet. Da diese Ausschnitte winkeltreu sind, können beliebige Kartenkoordinaten mit den Eckpunkten und deren Geokoordinaten durch lineare Interpolationen bestimmt werden. Hierzu wurden Eckpunkte gewählt, diese auf den Karten markiert und die Geokoordinaten mit Google Maps ausgelesen. So ist es möglich, die Geokoordinaten mit den in MATLAB verwendeten x- und y-Koordinaten des Bildes in Verbindung zu bringen.

Die Funktion `loadMap()`, deren Zusammenhänge in Abbildung 3-14 dargestellt sind, lädt eine Karte in das MATLAB-Framework. Ein kompletter Datensatz einer Karte befindet sich in einem Verzeichnis, das mit Namen (*MapID*) der Karte bezeichnet wurde. Der Funktion `loadMap()` wird die *MapID* übergeben, um die Structure Variable `Map` zu erhalten, deren Aufbau in Tabelle 3-4 beschrieben ist.

Ein Karten-Datensatz besteht aus dem Bild der Karte (Datei *Img.png)* und den *MapFiles,* welche die Informationen für die Einpassung enthalten. Die Dateien *lat.m* und *lon.m* enthalten die Koordinaten in Dezimalgrad der gewählten Eckpunkte. Diese wurden in Google Maps markiert und die Geokoordinaten ausgelesen. Die Dateien *y.m* und *x.m* enthalten die x- und y-Koordinaten der Eckpunkte für das Koordinatensystem des Kartenbildes in MATLAB. Die Achsen dieses Bild-Koordinatensystems sind in der Abbildung 3-14 rechts unten dargestellt. Bei den aus Google Maps ausgeschnittenen Karten ist die Oberkante nach Norden ausgerichtet, sodass die y-Komponente des Bildes dem Breitengrad und x-Komponente dem Längengrad entspricht.

Da Positionsänderungen in Schritten bzw. in Metern ausgedrückt werden sollen, wurde ein Umrechnungswert für die Meter pro Pixel berechnet. Dieser Wert wird aus den Distanzen zwischen den Eckpunkten im Geo- und Bild-Koordinatensystem ermittelt. Die Distanzen in Metern im Geo-Koordinatensystem wurden mithilfe des in [48] verfügbaren Tools abgefragt. Diese sind in den Dateien *dx_meter.m* und *dy_meter* gespeichert. Die Distanzen im Bild-Koordinatensystem werden in MATLAB berechnet.

Abbildung 3-14 Funktionschema - Laden der Karten in MATLAB



Mit den Funktionen `latlon2yx()` und `yx2latlon()` können die Koordinaten der zwei verwendeten Koordinatensysteme in das jeweils andere System transformiert werden. Die

Funktionen benutzen lineare Interpolation für die Transformation der Koordinaten. Daher kann die Funktion nur auf Karten angewandt werden, die winkeltreu sind und deren Oberkanten nach Norden zeigen.

Die Funktion `plotCP()` zeichnet die iCPs anhand der Variable `Map` Structure ein. Die Metadaten zu den iCPs werden bereits mit der Funktion `loadMap()` geladen. Diese sind in der Datei *iCP.m* im Verzeichnis des Karten-Datensatzes gespeichert.

Mit der Funktion `plotrefPoirnts()` werden die in `Map` gespeicherten Referenzpunkte in die Karte eingezeichnet. Mit der Funktion `plotrefRuns()` kann der Referenzweg eines Testlaufes anhand der Referenzpunkte eingezeichnet werden.

Tabelle 3-4 Aufbau der Structure Variable Map

MAP	<struct>	Map mit Bild und Anpassungsinfos
img	<Y×X×3uint8>	Karte als Bild
lat	<1x4 double>	Geographische Breite in Dezimalgrad der vier Eck-Koordinaten
lon	<1x4 double>	Geographische Länge in Dezimalgrad der vier Eck-Koordinaten
x	<1x4 double>	x- Koordinaten der Eckpunkte im Bild-Koordinatensystem
y	<1x4 double>	y- Koordinaten der Eckpunkte im Bild-Koordinatensystem
dx_m	<double>	Distanz zwischen den Einpassungspixeln in X-Richtung in m.
dy_m	<double>	Distanz zwischen den Einpassungspixeln in Y-Richtung in m.
dx_pix	<double>	Distanz zwischen den Einpassungspixeln in X-Richtung in Pixel
dy_pix	<double>	Distanz zwischen den Einpassungspixeln in Y-Richtung in Pixel
dx_lon	<double>	Distanz zwischen den Einpassungspixeln in X-Richtung in Dezimalgrad
dy_lat	<double>	Distanz zwischen den Einpassungspixeln in Y-Richtung in Dezimalgrad
mPpix_x	<double>	Meter/Pixel in X-Richtung
mPpix_y	<double>	Meter/Pixel in Y-Richtung
mPpix	<double>	Meter/Pixel - Mittelwert
lonPpix_dx	<double>	Longitude-Dezimalgrad /Pixel in X-Richtung
latPpix_dy	<double>	Latitude- Dezimalgrad /Pixel in Y-Richtung
iCp	<..x4 double>	Metadata der iCP – Tabelle mit Koordinaten und Daten
refPoints	<..x3 double>	Tabelle mit Koordinaten der Referenzpunkte
pixStepsize	<double>	Schrittgröße in Pixel. Wird von INS-Algorithmus verwendet
korrOrient	<..x4 double>	Werte für die Korrektur für Azimut-Werte an ausgewählten iCP
stepsiCp	<..x4 double>	Werte für die Korrektur für Schrittgröße an ausgewählten iCP

3.4 Zusammenfassung

Das hier vorgestellte Datenerfassungs- und Analysesystem (DAAS) besteht aus einer Android-Applikation (CPS-App) zur Datenerfassung mit Smartphones und dem MATLAB-Framework (MFW) zur Analyse dieser Daten.

Mit dem CPS-App [37] ist es möglich, WLAN- Scans zu erfassen und diese mit Positionen in Verbindung zu bringen. Diese WLAN-Scans werden auch mit dem Azimut-Wert, dem digitalen Kompass des Handys, versehen und können einer Richtungskategorie zugeordnet werden. Die App kann zwischen Datensätzen für die Kalibrierungs- und Testphase unterscheiden.

Diese Datensätze können in das MFW [37, MFW] importiert und in verschiedenen Structure Variablen zusammengefasst werden. Berechnet werden im MFW das arithmetische Mittel, der Median, die Standardabweichung, die Varianz und die Verfügbarkeit der Signalstärken der Access Points (APs) an den gewählten Positionen.

Das CPS-App kann, während mit dem Handy eine Testrecke abgegangen wird, Testläufe aufzeichnen. Dabei werden die Sensor- und GPS-Daten mit WLAN-Scans festgehalten.

Definiert kann die Teststrecke mit Referenzpunkten werden, deren erreichen mit dem App bestätigt werden kann.

Wie die WLAN-Scans können auch die Testläufe in MFW importiert werden. Das erlaubt es, die ermittelten Daten für die verschiedenen Analysen und Simulationen zu verwenden. Im nächsten Kapitel wurde das System verwendet, um verschiedene Ansätze und Algorithmen zu simulieren und zu untersuchen.

Das MFW kann ermittelte Positionen und Strecken auf Kartenausschnitten darstellen.

4 Analyse von WLAN-Fingerprinting-Ansätzen

Für das Konzept der intelligenten Check Points, das in **Kapitel 5** und **Kapitel 6** weiter diskutiert wird, ist es nötig, dass diese so präzise wie möglich erkannt werden. Aus diesem Grund bestand ein Teil der Arbeit darin, einen zuverlässigen WLAN-Fingerprinting-Algorithmus zu entwickeln. Dieses Kapitel beschreibt verschiedene getestete Varianten des WLAN-Fingerprinting. Untersucht werden Methoden zur Mittelung von RSS-Werten, Auswahlstrategien für die beste Position und eine Gewichtungsstrategie für bestimmte APs.

Zu diesem Zweck wurde das CPS-App (siehe 3.2) verwendet, um WLAN-Datensätze zu erfassen. Erstellt wurden zwei Datenbanken, eine Fingerprinting-Datenbank (FPDB) für die Kalibrierung des Systems und eine Test-Datenbank (TEST-DB) als Datengrundlage für die Positionierungsphase. Mit dem in MALTAB programmierten Framework (MFW) (siehe 3.3), der FPDB und der TEST-DB war es möglich, die Positionierung mit WLAN-Fingerprinting zu simulieren. Die Ergebnisse der untersuchten Fingerprinting-Varianten werden hier gegenübergestellt.

4.1 Einleitung und Überblick

In Abschnitt 2.2 wurden die Grundalgen des WLAN-Fingerprintings bereits erläutert. Beim WLAN-Fingerprinting werden die Signalstärken der erreichbaren Access Points (APs) gemessen. Die Kombinationen von Signalstärken (RSS - Received Signal Strength) werden einer Position zugeordnet. Die ermittelten RSS-Werte dienen als Indikatoren, um die sogenannten RSSI-Werte („Received Signal Strength Indicator“) zu bestimmen. In dieser Arbeit bestehen diese aus den empfangen Signalstärken bzw. aus definierten Werten, falls APs nicht erreicht wurden. So können jeder Position in der Datenbank RSSI-Vektoren der gleichen Dimension zugeordnet werden. Die RSSI-Vektoren sind in 4.3.1 in *(F. 4-3)* und *(F. 4-5)* beschrieben und definiert. Die Datenerfassung in wird in 4.2 erläutert.

Ein einzelner RSSI-Vektor ist für eine Position nicht repräsentativ, da die RSS-Werte Schwankungen [61] unterzogen sind. Daher werden in diesem Kapitel Methoden zur Wahl von gemittelten RSSI-Vektoren untersucht und verglichen. Dafür wurden die erfassten WLAN-Scans verwendet und verschiedene Methoden zur Mittelung von RSSI-Vektoren gegenüber gestellt. Verglichen wurden Methoden, die den Median [36] und das arithmetische Mittel [27] nutzen. Wie diese berechnet wurden, ist in 4.3.3 erläutert. Untersucht wurde außerdem, wie die variierende Verfügbarkeit von APs berücksichtigt werden kann. Die Ergebnisse sind in Abschnitt 4.5.1 dargelegt.

Neben dem Betreiben des Fingerprintings mit einer Datenbank aus gemittelten Vektoren ist es möglich, alle erfassten RSSI-Vektoren der Kalibrierungsphase als Fingerprinting-Datenbank zu verwenden. Untersucht wurden zwei Ansätze. Bei beiden wird der zu betrachtende RSSI-Vektor mit allen Vektoren in der Fingerprinting-DB bezüglich des euklidischen Abstandes verglichen. Ausgewählt werden eine bestimmte Anzahl von am besten übereinstimmenden RSSI-Vektoren, anhand derer dann eine Position ermittelt wird. Beim einen Ansatz wird jene Position gewählt, welche am häufigsten durch deren RSSI-Vektoren repräsentiert wurde. Beim zweiten Ansatz werden den möglichen Positionen jeweils Wahrscheinlichkeiten zugewiesen, die wahrscheinlichste Position wird dann gewählt. Beschrieben sind diese Algorithmen in 4.4.3, die erzielten Ergebnisse sind in 4.5.3 erläutert.

Gegenübergestellt wurde auch, wie sich die Erkennungsraten ändern, wenn für jedes Smartphone eine eigene oder eine gemeinsame Fingerprinting-Datenbank verwenden wird.

4.2 Datenerfassung

Zur Erfassung der Daten standen zwei Smartphones, ein Galaxy S2 und ein HTC EVO 3D zur Verfügung. Die technischen Details sind im **Anhang A** in 8.2 erläutert. Für die Erfassung der WLAN-Scan-Datensätze wurde der Kompass-WLAN-Scanner des CPS-Apps verwendet, der in 3.2 beschrieben wurde. Unterschieden wurden zwei Arten von Datensätzen, jene der Kalibrierungsphase und jene der Testphase. Die entsprechende Option bietet das CPS-App.

Für die zwei gewählten Testgebiete, von denen eines sich größtenteils innerhalb eines Gebäudes befindet (EI-Testgebiet) und das andere im urbanen Stadtgebiet (Heim-Testgebiet), wurden jeweils vier Richtungskategorien definiert. Diese sind von 1 bis 4 nummeriert, die *RichtungsIDs* sind in Abbildung 4-1 und Abbildung 4-4 dargestellt. Die Richtungen orientieren sich an den Hauptbewegungsrichtungen der Umgebungen, also z. B. die Ausrichtung der Gänge oder Straßen und stehen im 90°-Winkel zueinander.

In den Testgebieten wurden bestimmte Positionen bzw. Punkte definiert, die für die Untersuchungen verwendet wurden. An diesen Punkten wurden WLAN-Scans jeweils in eine der vier definierten Richtungen mehrmals durchgeführt. Das Handy wurde dabei waagerecht in die gewählte Richtung gehalten, wie in Abbildung 6-1 gezeigt ist.

Die WLAN-Scans wurden zu verschieden Zeitpunkten erfasst, um eine möglichst breite Datenbasis zu erhalten. Dadurch soll die mögliche Bandbreite der RSS-Werte, der APs, an den untersuchten Positionen festgehalten werden. Weiter wurde die Erfassung der Datensätze für die Kalibrierungs- und Testphase zu verschieden Zeitpunkten durchgeführt. Dies sollte verhindern, dass die erfassten RSSI-Vektoren nur aufgrund der zeitlich nahen Messung geringe euklidische Abstände aufweisen, was zu verfälschten Ergebnissen führen würde. Die zeitliche Trennung von Kalibrierungsphase und Testphase soll das verhindern und eine wirklichkeitsnahe Testsituation widerspiegeln.

Für das EI-Testgebiet wurden 49 Punkte untersucht. An diesen Positionen wurden insgesamt 5726 WLAN-Scans durchgeführt, um die Datensätze zur Erstellung einer Fingerprinting-Datenbank zu erhalten. Für die Test-DB wurden 796 Datensätze an ausgewählten Positionen erfasst.

Im Heim-Testgebiet wurden 23 Positionen ausgewählt, an diesen wurden 920 WLAN-Scans für die Fingerprinting-DB festgehalten. Zur Analyse wurden 568 Scans durchgeführt und als Test-DB verwendet.

Die Daten sind unter [37, Testdaten] abrufbar.

4.2.1 Testgebiete

Das **EI-Testgebiet** befindet sich im neuen Elektrotechnik-Institutsgebäude der Technischen Universität Wien, in der Gußhausstraße 27-29. Es ist beschränkt auf den Haupteingangsbereich, die Stiege 1 und den 3. Stock, im Bereich der Forschungsgruppe für Ingenieurgeodäsie. Die Koordinaten der verwendeten Punkte sind im **Anhang A** in 8.3 definiert.

Das EI-Testgebiet besteht aus drei Teilgebieten. In Abbildung 4-1 ist der Eingangsbereich zum Stiegenhaus-1 dargestellt. Der Bereich des Stiegenhauses im 3. Stock ist in Abbildung

4-2 abgebildet. Das Teilgebiet in Abbildung 4-3 beinhaltet die Türen zu dem Büro von Prof. Retscher (Raumnummer: CA 03 33) und den benachbarten Räumen. Die Gebäudepläne [42] sind auf der Webseite der Universität verfügbar.

Abbildung 4-1 Eingangsbereich EI-Testgebiet

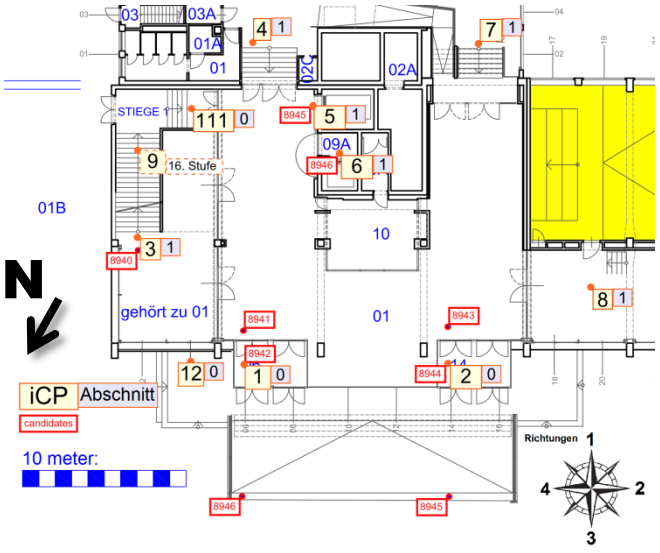


Abbildung 4-2 Stiegenhaus EI-Testgebiet

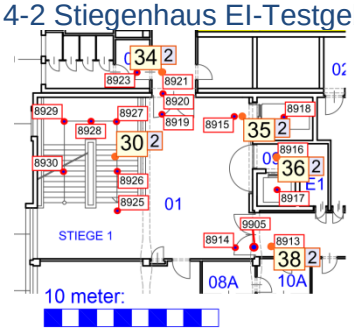
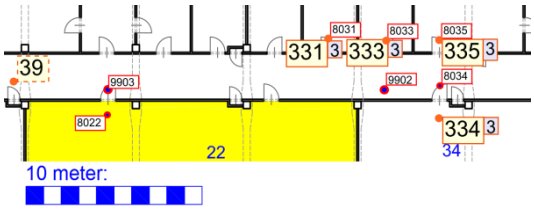
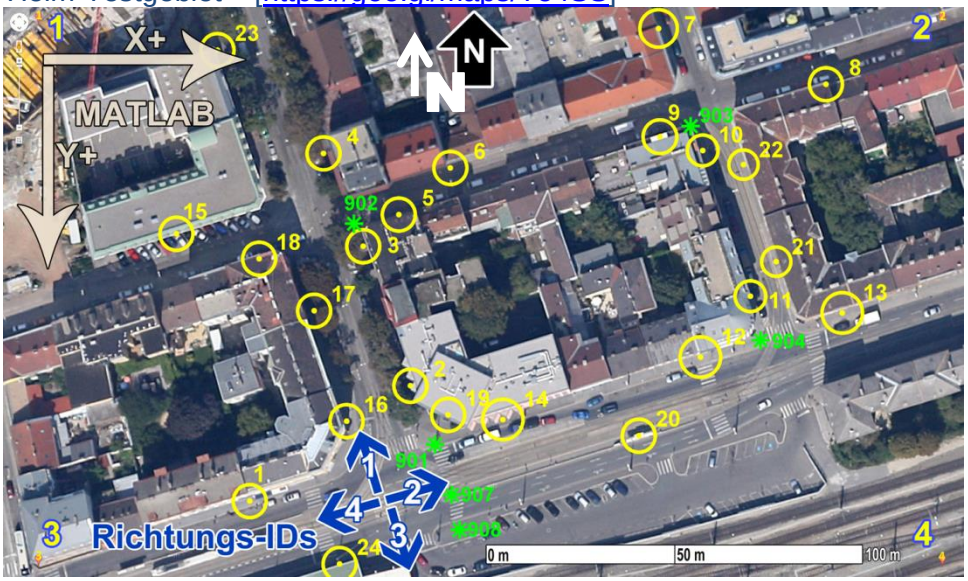


Abbildung 4-3 Stiegenhaus Gang B EI-Testgebiet



Das **Heim-Testgebiet** ist um den Häuserblock des Studentenwohnheims in der Eichenstraße 46 definiert. Vermessen wurden 23 Punkte, die in Abbildung 4-4 dargestellt sind. Die Koordinaten sind im **Anhang A** in 8.3.1 zu finden.

Abbildung 4-4 Heim-Testgebiet – [https://goo.gl/maps/Te4SS]



4.3 Darstellung und Mittelung der RSSI-Vektoren in MATLAB

Wie in 3.3.2 beschrieben, können mithilfe des MFW die Datensätze der durchgeführten WLAN-Scans importiert werden. Die ermittelten Signalstärken werden als RSSI-Vektoren in einem Array gespeichert. Der Aufbau der RSSI-Vektoren ist in dem Abschnitt 4.3.1 beschrieben. Für eine Position werden alle ermittelten RSSI-Vektoren in einer Matrix zusammengefasst; dies ist in 4.3.2 erklärt. Mit diesen Matrizen können die Mittelwerte, wie in 4.5.1 beschrieben, mit den in MATLAB verfügbaren Funktionen berechnet werden. Diese

gemittelten RSSI-Vektoren können zu einer FPDB zusammengefasst werden, wie in Abschnitt in 4.3.4 gezeigt wird.

4.3.1 Definition der RSSI-Vektoren

Ein WLAN-Scan-Datensatz beinhaltet die empfangenen Signalstärken (RSS-Werte) der erreichten APs. Diese Werte dienen als Indikatoren für das WLAN-Fingerprinting und werden hier als RSSI-Werte („Received Signal Strength Indicator“) bezeichnet.

Dargestellt werden diese Werte in zwei RSSI-Vektoren, $\overline{Scan1}$ und $\overline{Scan2}$, die in den Formeln *(F. 4-1)* bis *(F. 4-6)* mathematisch definiert sind.

$$n \dots Anzahl\ aller\ AP\ im\ Testbereich \qquad (F.\ 4\text{-}1)$$

$$\overline{AccessPoints} = [AP_1, AP_2 \dots, AP_n] \quad AP_x \in \{\mathbb{Z}_{<0} \cap \{1, 2, \dots, n\}\} \qquad (F.\ 4\text{-}2)$$

$$\overline{Scan1} = [S1_{AP1}, S1_{AP2}, \dots, S1_{APn}] \quad S1_{APx} \in \{\mathbb{Z}_{<0} \cap \{-101, \dots -1\}\} \qquad (F.\ 4\text{-}3)$$

$$S1_{APx} = \begin{cases} RSS - Wert\ des\ APx\ in\ dBm & wenn\ APx\ erreichbar \\ -101\ dbm & wenn\ APx\ nicht\ erreichbar \end{cases} \qquad (F.\ 4\text{-}4)$$

$$\overline{Scan2} = [S2_{AP1}, S2_{AP2}, \dots, S2_{APn}] \quad S2_{APx} \in \{\mathbb{Z}_{<0} \cap \{-101, \dots -1\} \cup NaN\} \qquad (F.\ 4\text{-}5)$$

$$S2_{APx} = \begin{cases} RSS - Wert\ des\ APx\ in\ dBm & wenn\ APx\ erreichbar \\ NaN & wenn\ APx\ nicht\ erreichbar \end{cases} \qquad (F.\ 4\text{-}6)$$

Alle Vektoren haben dieselbe Dimension n. diese ergibt sich aus der Anzahl der ermittelten APs der Kalibrierungsphase.

Eindeutig sind APs durch ihre ausgesendete BSSID (Basic Service Set Identification) bestimmt. Sie entspricht entweder der MAC-Adresse der APs oder wurde als Zufallszahl erzeugt. [80] Diese BSSIDs sind in der Variable `AccessPoints` gespeichert, die in 3.3.1 beschrieben wurde. In ihr werden den APs ganzzahlige IDs zugewiesen. Diese IDs sind in MATLAB leichter handzuhaben und definieren die Stellen x, an denen die RSSI-Werte der APs im Vektor stehen. Diese Variable wird hier durch den Vektor $\overline{AccessPoints}$ repräsentiert.

Die Vektoren $\overline{Scan1}$ und $\overline{Scan2}$ enthalten dieselben RSS-Werte der erreichten APs. Unterschieden wird die Darstellung der nicht erreichten APs. Bei den RSSI-Werten $S1_{APx}$ im Vektor $\overline{Scan1}$ werden nicht erreichte APs mit einem Minimalwert von -101dBm dargestellt. Im Vektor $\overline{Scan2}$ werden bei den RSSI-Werten $S2_{APx}$ im WLAN-Scan nicht vorkommende APs mit NaN („*Not a Number*“ [55]) repräsentiert. Diese Unterscheidung erfolgt, um spätere Berechnungen in MATLAB einfacher durchführen zu können.

Die RSSI-Vektoren $\overline{Scan1}$ und $\overline{Scan2}$ für die Kalibrierungsphase sind im MFW Structure Variable `WIFISCANS` gespeichert.

Die WLAN-Scans aus der Testphase werden mit der Funktion `testWLANTESTSCANS()` in die Structure Variable `WIFITESTSCANS` geladen.

4.3.2 RSSI-Vektoren-Matrix für eine Position

WLAN-Scans wurden an einer Position mehrfach mit verschiedenen Geräten durchgeführt. Bei jedem Durchgang wurden mehrere WLAN-Scans in jeder Richtungskategorie erfasst. Jede Position wird dadurch aus mehreren RSSI-Vektoren repräsentiert. Aus diesen RSSI-Vektoren können RSSI-Mittelwerte-Vektoren gebildet werden. Zur Berechnung der Mittelwerte wurden die erfassten Vektoren einer Position in einer Matrix $\overline{\overline{\mathrm{LevMat}_{\mathrm{PosID}}}}$ zusammengefasst. Der allgemeine Aufbau der Matrix ist in *(F. 4-8)* dargestellt.

M_{PosID}...*Anzahl durchgeführter WLAN-Scans an der Position* mit der ID PosID (F. 4-7)

$$\overline{\overline{LevMat_{PosID}}} = \begin{bmatrix} Scan_1 \\ Scan_m \\ \vdots \\ Scan_{M_{PosID}} \end{bmatrix} = \begin{bmatrix} S_{1,AP1} & \cdots & S_{1,APn} \\ \vdots & \ddots & \vdots \\ S_{M,APn} & \cdots & S_{M,APn} \end{bmatrix} \quad \text{(F. 4-8)}$$

Im MFW werden die ermittelten RSSI-Vektoren für die Position $PosID$ in einer Matrix mit der Dimension $M_{PosID} \times n$ dargestellt, wobei M_{PosID} die Anzahl der durchgeführten WLAN-Scans an der Position beschreibt. Die Matrix enthält in jeder Zeile einen RSSI-Vektor $\mathrm{Scan_m}$, der für einen beliebigen WLAN-Scan an der $PosID$ steht. Der RSSI-Vektor kann die Form von $\overline{Scan1}$ oder $\overline{Scan2}$ haben. Daraus ergeben sich, entsprechend den in 4.3.1 definierten RSSI-Vektoren, zwei Darstellungen für die Matrizen $\overline{\overline{\mathrm{LevMat1_{PosID}}}}$ und $\overline{\overline{\mathrm{LevMat2_{PosID}}}}$.

4.3.3 Mittelwerte

Die Berechnung eines Mittelwerts kann auf unterschiedliche Weise durchgeführt werden. In dieser Arbeit werden das arithmetische Mittel (eng. mean) und der Median verwendet.

Für die Mittelwertberechnungen werden die MATLAB-Funktionen `nanmean()` [52] und `nanmedian()` [53] auf die Matrizen $\overline{\overline{\mathrm{LevMatPos1}}}$ und $\overline{\overline{\mathrm{LevMatPos2}}}$ angewandt. Diese Funktionen ignorieren NaN-Werte bei der Berechnung.

Bei der Matrix $\overline{\overline{\mathrm{LevMat1}}}$ werden nicht erreichte AP mit einem Minimalwert (-101dBm) dargestellt. Dies kann auch als eine Gewichtung verstanden werden. Wird ein AP nur in wenigen Scans gefunden, wird sich der Mittelwert an den Minimalwert annähern. Bei der Berechnung der Mittelwerte mit $\overline{\overline{\mathrm{LevMat2}}}$ werden nicht erreichte APs durch Verwenden des NaN-Wertes ignoriert. Es gibt dann keine Gewichtung, es werden nur die RSS-Werte verwendet, die gemessen wurden.

Ein Beispiel: angenommen, es werden an einer Position 10 WLAN-Scans durchgeführt und nur bei einem Scan wird ein bestimmter AP mit -91 dBm gefunden. Dann würde das arithmetische Mittel für diesen AP mit der Funktion `nanmean()` und $\overline{\overline{\mathrm{LevMat1}}}$ -100 dBm ergeben. Wendet man nanmean() auf $\overline{\overline{\mathrm{LevMatPos2}}}$ an, erhält man für denselben AP -91 dBm.

Da auf zwei Matrizen zwei Mittelwertmethoden angewandt wurden, ergaben sich vier RSSI-Mittelwert-Vektoren für eine Position. Im MFW sind diese in den Feldern *mean1*, *mean2* und *median1*, *median2* in der Structure `Variable WIFIPOINTS` gespeichert.

4.3.4 Fingerprinting-Datenbanken FPDB

Die beschrieben RSSI-Vektoren können zu einer Fingerprinting-Datenbank (FPDB), hier auch allgemein als Matrix $\overline{\overline{\mathrm{FPDB}}}$ (F. 4-14) bezeichnet, zusammengeführt werden.

$$\mathrm{DS}\ldots \text{Anzahl der Datensätze in } \overline{\overline{FPDB}} \qquad \text{(F. 4-9)}$$

$$PosID \ldots ID\ die\ einer\ Position\ zugewiesen\ ist \qquad \text{(F. 4-10)}$$

$$\overline{PointIDs} = \begin{bmatrix} 1{:}\ PosID \\ \vdots \\ DS{:}\ PosID \end{bmatrix} \ldots \textit{weist jedem RSSI-Vektor eine Position zu} \qquad \text{(F. 4-11)}$$

$$S_{PosID,AP_x} \ldots \text{RSSI-Wert für } AP_x \text{ an Position } PosID \qquad \text{(F. 4-12)}$$

$$\overline{RSSI_{PosID}} = \left[S_{PosID,AP_1}, \ldots, S_{PosID,AP_n}\right] \ldots \text{gemittelter RSSI-Vektor für Position i} \qquad \text{(F. 4-13)}$$

$$\overline{\overline{FPDB}} = \begin{bmatrix} RSSI_{1:PosID} \\ \vdots \\ RSSI_{DS:PosID} \end{bmatrix} = \begin{bmatrix} S_{1:PosID,AP_1}, \ldots, S_{1:PosID,AP_n} \\ \vdots \\ S_{DS:PosID,AP_1}, \ldots, S_{DS:PosID,AP_n} \end{bmatrix} \qquad \text{(F. 4-14)}$$

In der Fingerprinting-Datenbank kann jede Positionen PosID durch je einen RSSI-Mittelwert-Vektor $\overline{\mathrm{RSSI_{PosID}}}$ repräsentiert sein. Die Datenbank kann aber auch mehrere RSSI-Vektoren pro Position aufweisen. Die Zuweisung eines RSSI-Vektors $\overline{\mathrm{RSSI_{PosID}}}$ zu dessen Position erfolgt in MFW über den Vektor $\overline{\mathrm{PointIDs}}$. Wie in *(F. 4-11)* dargestellt, enthält dieser Vektor die PosIDs der RSSI-Vektoren.

Damit beschreibt ein Vektor $\overline{\mathrm{RSSI_{i:PosID}}}$ einen erfassten RSSI-Vektor bzw. berechneten RSSI-Mittelwert-Vektor für eine Position PosID, wobei der RSSI-Wert $\mathrm{S_{i:PosID,\,AP_x}}$ des RSSI-Vektors-i, der Signalstärke des $\mathrm{AP_x}$ an der Position PosID darstellt.

Aus den zuvor in 4.3.3 genannten möglichen Mittelwert-Vektoren ergeben sich vier mögliche Fingerprinting-Datenbanken: $\left(\overline{\overline{\mathrm{FPDB_{mean1}}}}, \overline{\overline{\mathrm{FPDB_{mean2}}}}, \overline{\overline{\mathrm{FPDB_{median1}}}}, \overline{\overline{\mathrm{FPDB_{median2}}}}\right)$. Die Darstellung als Matrix ist für MATLAB gut geeignet. Die Berechnung der euklidischen Abstände eines zu testenden RSSI-Vektors zu allen Vektoren bzw. Zeilen der Matrix lässt sich durch die Ausführung eines Befehls ermitteln. Die Berechnung der euklidischen Abstände ist im nächsten Abschnitt erläutert.

4.4 Fingerprinting in MATLAB-Berechnungsmethoden

Für die Datenauswertung wurde eine Reihe von MATLAB-Funktionen implementiert, welche von der Funktion `testWLANTESTSCANS()` `[37, MFW\functions\wifi]` aufgerufen werden. Mithilfe von Parametern können verschiedene Berechnungsmethoden und Kombinationen von Datensätzen definiert werden. Der Funktion können die Datensätze einer FPDB und TEST-DB zur Simulation der Positionierung übergeben werden. Die möglichen Berechnungsmethoden werden in diesem Abschnitt beschrieben.

In 4.4.1 wird beschrieben, wie die euklidischen Abstände eines RSSI-Vektors zu allen Vektoren in einer Datenbank berechnet werden. Im Kapitel 4.4.2 wird ein Ansatz vorgestellt, bei dem bestimmte *RSSI-Werte* gewichtet werden. Im letzten Abschnitt in 4.4.3 werden zwei Algorithmen vorgestellt, die alle erfassten RSSI-Vektoren für die Positionierung verwenden.

4.4.1 Der Euklidische Abstand zwischen den RSSI-Vektoren

Bei dem in 2.2.1 beschriebenen Nearest Neighbor (NN) Ansatz wird der euklidische Abstand zwischen zwei Vektoren genutzt, um zu quantifizieren, wie „ähnlich“ sich diese zwei n-dimensionalen Vektoren sind. Je geringer dieser Abstandswert ist, desto „näher“ liegen die zu vergleichenden Vektoren beisammen.

Beim NN-Algorithmus wird der zur Positionierung herangezogene Vektor mit allen Vektoren in einer Fingerprinting-Datenbank verglichen. Alle RSSI-Vektoren der Datenbank sind einer Position zugewiesen. Der RSSI-Vektor mit dem geringsten euklidischen Abstand in der Datenbank definiert die Position, welche bei der Positionierung ausgewählt wird. Es wird also der nächste Nachbar bzw. der „ähnlichste“ RSSI-Vektor und damit dessen Positionen gewählt.

Im MFW wurde die Berechnung der euklidischen Abstände in der Funktion `disTestscan2() [37, MFW\functions\wifi]` implementiert. Der Funktion wird eine Matrix $\overline{\overline{\mathrm{FPDB}}}$ und ein RSSI-Vektor $\overline{testscan}$ (F. 4-15) übergeben. Die Funktion gibt den Vektor $\overline{\mathrm{EukDıstanz}}$ *(F. 4-16)* zurück, dieser enthält die euklidischen Abstände eukdis zu allen RSSI-Vektoren in der Datenbank. Die Berechnung erfolgt im Grunde durch eine Befehlszeile in MATLAB, $\overline{\mathrm{EukDıstanz}}$=`sqrt(nansum(bsxfun(@minus,` $\overline{\overline{\mathrm{FPDB}}}$`,` $\overline{\mathrm{testscan}}$`).^2,2))`.

$$\overline{testscan} = [Sm_{AP1}, \dots, Sm_{APn}] \text{...ein RSSI-Vektor für die Positionierung} \qquad \text{(F. 4-15)}$$

$$\overline{EukDistanz} = \begin{bmatrix} \sqrt{\left(Sm_{AP1} - S_{1:PosID,AP_1}\right)^2 + \dots + \left(Sm_{APn} - S_{1:PosID,AP_n}\right)^2} \\ \vdots \\ \sqrt{\left(Sm_{AP1} - S_{DS:PosID,AP_1}\right)^2 + \dots + \left(Sm_{APn} - S_{DS:PosID,AP_n}\right)^2} \end{bmatrix} = \begin{bmatrix} eukdis_{1:\,PosID} \\ \vdots \\ eukdis_{DS:PosID} \end{bmatrix} \qquad \text{(F. 4-16)}$$

Der Vektor $\overline{testscan}$ hat die Struktur der in 4.3.1 beschrieben RSSI-Vektoren, wobei $\mathrm{Sm_{APx}}$ entweder den gemessen RSS-Wert des APx darstellt oder einen definierten Wert (-101dBm oder NaN), falls der APx bei einem WLAN-Scan nicht erreichbar war. Im MFW sind diese für die Positionierung erfassten RSSI-Vektoren in der Structure Variable `WIFITESTSCANS` gespeichert.

Die Matrix $\overline{\overline{\mathrm{FPDB}}}$ kann, wie bereits erläutert, für jede Position einen gemittelten, mehrere oder alle erfassten RSSI-Vektoren beinhalten. Die Zuordnung der Positionen zu den RSSI-Vektoren erfolgt immer über den Vektor $\overline{\mathrm{PoıntIDs}}$. Beim NN-Algorithmus wird jene Position gewählt, deren RSSI-Vektor den geringsten euklidischen Abstand zum Vektor $\overline{testscan}$ aufweist. Daher gibt die Funktion `disTestscan2()` neben dem Vektor $\overline{\mathrm{EukDıstanz}}$ auch den geringsten Abstandswert und die Nummer i des Datensatzes zurück. Mit der Datensatznummer wird die PosID mithilfe von $\overline{\mathrm{PoıntIDs(ı)}}$ gefunden.

4.4.2 Gewichtung der Summanden des Euklidischen Abstandes

Der Funktion `disTestscan2()` kann außerdem ein Vektor oder eine Matrix $\overline{weighted_{P_i}}$ *(F. 4-17)* übergeben werden. Der Wert w_{AP_x} bestimmt, mit welchem Faktor die quartierte RSSI-Wert-Differenzen des AP_x gewichtet wird. Die Formel (F. 4-18) beschreibt, wie der gewichtet Abstandswert $eukdis_{Weig,i:PosID}$ für einen RSSI-Vektor i berechnet wird.

$$\overline{weighted} = [w_{AP_1}, \dots, w_{AP_n}] \qquad \text{(F. 4-17)}$$

$$eukdis_{Weig,i:PosID} = \sqrt{\left(Sm_{AP1} - S_{i:PosID,AP_1}\right)^2 \cdot w_{AP_1} + \dots + \left(Sm_{APn} - S_{i:PosID,AP_n}\right)^2 \cdot w_{AP_n}} \qquad \text{(F. 4-18)}$$

Wird der Funktion `disTestscan2()` ein Gewichtungsvektor $\overline{weighted}$ übergeben, so muss dieser die gleiche Dimension wie die RSSI-Vektoren haben. Es wird dann für alle Positionen der gleiche Gewichtungsvektor verwendet.

Wird eine Matrix übergeben, muss diese dieselbe Dimension wie die Matrix $\overline{\overline{FPDB}}$ haben. Jede Zeile bzw. jeder Vektor $\overline{weighted_i}$ *(F. 4-19)* der übergebenen Gewichtungsmatrix wird für die entsprechende Position i verwendet. So kann für jede Position der Abstandwert $eukdis_{Weig,i:PosID}$ von den APs verschieden stark beeinflusst werden.

$$\overline{weighted_i} = [w_{i:AP_1}, \dots, w_{i:AP_n}] \qquad \text{(F. 4-19)}$$

$$eukdis_{Weig,i:PosID} = \sqrt{\left(Sm_{AP1} - S_{i:PosID,AP_1}\right)^2 \cdot w_{i:AP_1} + \dots + \left(Sm_{APn} - S_{i:PosID,AP_n}\right)^2 \cdot w_{i:AP_n}} \qquad \text{(F. 4-20)}$$

Mit dieser Option der Funktion kann z. B. untersucht werden, ob zur Unterscheidung von bestimmten Positionen gewisse APs relevanter sind als andere. Für die Arbeit wurde jedoch nur die Option eines Gewichtungsvektors $\overline{weighted}$ untersucht.

4.4.3 WLAN-Fingerprinting mit allen RSSI-Vektoren in der Datenbank

Alternativ zum Fingerprinting mittels RSSI-Mittelwert-Vektoren wurden zwei Algorithmen untersucht, bei denen die euklidischen Abstände zu allen erfassten RSSI-Vektoren verwendet werden. Bei diesen Algorithmen werden als Fingerprinting-Datenbank alle erfassten RSSI-Vektoren in der Matrix $\overline{\overline{FPDB}}$ gespeichert. Dadurch sind RSSI-Vektoren mehrfach für eine Position in der Datenbank vorhanden. Welcher RSSI-Vektor zu welcher Position gehört, ist durch den Vektor $\overline{PointIDs}$ *(F. 4-11)* definiert.

Die Algorithmen benutzen die Funktion `disTestscan2()`, dieser werden die Matrix $\overline{\overline{FDDB}}$ und der $\overline{testscan}$ Vektor übergeben. Die Algorithmen funktionieren nach folgendem Schema:

- Von der Funktion `disTestscan2()` wird der Vektor $\overline{EukDistanz}$ berechnet.
- Mithilfe der MATLAB-Funktion `sort()` [57] wird der Vektor $\overline{EukDistanz}$ aufsteigend sortiert.
- Die Funktion übergibt den Vektor $\overline{EukDistanz_{sort}}$ (F. 4-21) mit aufsteigend sortierten Abstandswerten $eukdis_{i:PosID}$ und den Vektor, der die Sortierung beschreibt bzw. abbildet. Die Anzahl der Datensätze in der Fingerprinting-DB wird durch die Variable DS beschrieben.

$$\overline{EukDistanz_{sort}} = \begin{bmatrix} eukdis_{1:\,PosID} \\ eukdis_{2:\,PosID} \\ \vdots \\ eukdis_{DS:\,PosID} \end{bmatrix} \quad \text{(F. 4-21)}$$

$$wobei\ gilt, dass\ eukdis_{1:PosID} \leq eukdis_{2:PosID} \leq \cdots \leq eukdis_{DS:PosID} \quad \text{(F. 4-22)}$$

- Anhand der Sortierung wird der Vektor $\overline{\text{PointIDs}_{\text{sort}}}$ mitsortiert, damit die euklidischen Abstandswerte den RSSI-Vektoren zurechenbar bleiben.

$$\overline{PointIDs_{sort}} = \begin{bmatrix} 1: PosID \\ \vdots \\ DS: PosID \end{bmatrix} \quad \text{(F. 4-23)}$$

- Im Vektor $\overline{\text{PointIDs}_{\text{sort}}}$ sind an den ersten k Stellen jene PositionIDs zu finden, deren RSSI-Vektoren die k nächsten Nachbarn zum getesteten Vektor darstellen. Die Position wird je nach Algorithmus aus den Vektoren $\overline{\text{PointIDs}_{\text{sort}}}$ bzw. $\overline{\text{EukDistanz}_{\text{sort}}}$ gewählt. Wie die Wahl der Position getroffen wird, ist im Anschluss erklärt.

MFV-Algorithmus Most frequent values

Bei diesem Ansatz werden die k vordersten Elementen aus dem Vektor $\overline{\text{PointIDs}_{\text{sort}}}$ gewählt. Diese Auswahl ist in *(F. 4-24)* als Vektor $\overline{\text{PointIDs}_{\text{sort}_k}}$ dargestellt.

$$\overline{PointIDs_{sort_k}} = \begin{bmatrix} 1: PosID \\ \vdots \\ k: PosID \end{bmatrix} \quad \text{(F. 4-24)}$$

Gewählt wird die Position, die im Vektor $\overline{\text{PointIDs}_{\text{sort}_k}}$ am öftesten vertreten ist. Wenn Positionen gleich oft vorkommen, wird jene gewählt, die zuerst in den k-Elementen steht, also jene, die den RSSI-Vektor mit der geringsten euklidischen Distanz zum $\overline{testscan}$ Vektor aufweist. In MATLAB wird dazu die Funktion `mode` $(\overline{\text{PointIDs}_{\text{sort}_k}})$ [54] (Most frequent values in array) verwendet.

Likelihood-Algorithmus - Wahl nach einer Wahrscheinlichkeit

Beim MFV-Algorithmus ist die Wahl der Positionen nur von der Anzahl der Referenzen im Vektor $\overline{\text{PointIDs}_{\text{sort}_k}}$ abhängig. Bei dem Likelihood-Algorithmus werden zusätzlich die Distanzwerte in $\overline{\text{EukDistanz}_{\text{sort}_k}}$ berücksichtigt.

$$\overline{EukDistanz_{sort_k}} = \begin{bmatrix} eukdis_{1:\,PosID} \\ eukdis_{2:\,PosID} \\ \vdots \\ eukdis_{k:\,PosID} \end{bmatrix} \quad \text{(F. 4-25)}$$

$$wobei\ gilt, dass\ eukdis_{1:\,PosID} \leq eukdis_{2:\,PosID} \leq \cdots \leq eukdis_{k:\,PosID} \quad \text{(F. 4-26)}$$

Da die Wahrscheinlichkeit höher ist, je kleiner ein Abstandswert in $\overline{EukDistanz_{sort_k}}$ ist, werden diese Werte invertiert verwendet, wobei D_k *(F. 4-27)* die Summe der invertieren Abstandswerte k ist.

$$D_k = \sum_{i=1}^{k} \frac{1}{eukdis_{i:\,PosID}} \qquad \text{(F. 4-27)}$$

Beim Likelihood-Algorithmus werden die Werte in $\overline{EukDistanz_{sort_k}}$ invertiert und durch D_k dividiert. Daraus folgt der Vektor $\overline{P_{all}}$, der in *(F. 4-28)* definiert ist. Der Vektor $\overline{P_{all}}$ enthält somit Wahrscheinlichkeitswerte für die RSSI-Vektoren, die eine Aussage abgeben, wie diese einem RSSI-Vektor in FPDB zuordenbar sind. Die Summe von den Werten in $\overline{P_{all}}$ ergibt 1. Es wird also vorausgesetzt, dass zumindest ein RSSI-Vektor in der Datenbank die passende Position repräsentiert.

$$\overline{P_{all}} = \begin{bmatrix} \frac{eukdis_{1:\,PosID}{}^{-1}}{D_k} \\ \vdots \\ \frac{eukdis_{k:\,PosID}{}^{-1}}{D_k} \end{bmatrix} = \begin{bmatrix} p_1 \\ \vdots \\ p_k \end{bmatrix} \quad es\ gilt \sum_{i=1}^{k} p_i = 1 \qquad \text{(F. 4-28)}$$

Da Positionen in der $\overline{\overline{FPDB}}$ durch RSSI-Vektoren mehrfach vertreten sein können, müssen die einzelnen Wahrscheinlichkeitswerte in $\overline{P_{all}}$, die zur selben Position gehören, zusammengefasst und aufsummiert werden. Die Wahrscheinlichkeit $P_{PointID}$ für die Position $PointID$ wird daher nach der Formel (F. 4-29) berechnet.

$$P_{PointID} = \sum_{i=1}^{k} \overline{P_{all}(i)} = \sum_{i=1}^{k} p_i \quad wenn\ \overline{PointIDs_{sort_k}}(i) = PointID \qquad \text{(F. 4-29)}$$

Für jede Position, die in dem Vektor $\overline{PointIDs_{sort_k}}$ vorkommt, wird somit ein Wahrscheinlichkeitswert berechnet. Dieser ist abhängig von der Anzahl der Treffer und deren Abstandswerte. Je mehr Treffer es für eine Position gibt und je geringer die euklidischen Abstandswerte sind, desto höher wird der Wahrscheinlichkeitswert für diese Position. Gewählt wird jene Position mit dem höchsten Wahrscheinlichkeitswert. Um keine verzerrten Wahrscheinlichkeiten zu erhalten, sollten alle Positionen mit derselben Anzahl an RSSI-Vektoren in der FPDB vertreten sein.

Der Likelihood-Algorithmus wurde mit der Funktion `likelihood2()` [37, `MFW\functions\wifi`] mit den erfassten Daten simuliert und analysiert. Die Ergebnisse des MFV- und des Likelihood-Algorithmus werden im Abschnitt 4.5.3 diskutiert und gegenübergestellt.

4.5 Analyse Test WLAN-Fingerprinting

In diesem Abschnitt werden die in Kapitel 4.4 erläuterten Berechnungsmethoden und Algorithmen mit den erfassten Datensätzen wie in 4.2 beschrieben getestet und analysiert. Als Beurteilungsmerkmal wird die Erkennungsrate (ER), wie in *(F. 4-30)* definiert, verwendet.

Die ER sagt aus, wie viel Prozent der getesteten RSSI-Vektoren aus der verwendeten TEST-DB der korrekten Position zugeordnet wurden.

$$\text{Erkennungsrate(ER)} = \frac{\text{Anzahl korrekt zugewiesener RSSI-Vektoren}}{\text{Anzahl untersuchter RSSI-Vektoren}} \cdot 100\% \qquad \text{(F. 4-30)}$$

Für die Test-DB gibt es, wie für die RSSI-Vektoren, die in 4.3.1 definiert sind, zwei Darstellungsformen. In **Test-DB1** sind nicht erreichte APs mit -101dBm und in **Test-DB2** mit NaN-Werten dargestellt.

Die RSSI-Mittelwert-Vektoren (siehe in 4.3.4) der Kalibrierungsphase wurden in MATLAB, je nach Berechnungsmethode, zu einer Fingerprinting-Datenbank zusammengefasst. Aus den zwei Berechnungsmethoden (mean-DB und median-DB) und den zwei Darstellungsformen für nicht erreichte APs (-101dBm und NaN) ergeben sich vier Varianten für die Fingerprinting-Datenbank.

Aus den vier möglichen gemittelten Fingerprinting-Datenbanken und den zwei Darstellungsarten für die Test-DB ergeben sich für die Analyse acht mögliche Aufrufvarianten der Funktion `disTestscan2()`. In den Tabellen, welche die Erkennungsrate präsentieren (z. B. Tabelle 4-1), sind diese Kombinationen jeweils in den acht Spalten abgebildet.

Außerdem wurden verschiedene Szenarien betrachtet, indem eine bestimmte Auswahl aus den Datensätzen getroffen wurde. Für die Analyse wurden die Datensätze für jedes Testgerät einmal separat und einmal kombiniert untersucht. In den Tabellen, welche die Erkennungsraten enthalten, sind die Szenarien in den Zeilen abzulesen.

Bei der Analyse wurden die erfassten RSSI-Vektoren der Positionierungsphase mit der Test-DB1 und Test-DB2 an die Methode `disTestscan2()` übergeben. Die Datenbanken haben aufgrund ihres Aufbaus unterschiedlichen Einfluss auf die Berechnung der euklidischen Abstände. Wie in 4.4.1 beschrieben, verwendet die Methode `disTestscan2()` die MATLAB-Funktion `nansum()`, bei welcher alle NaN-Werte ignoriert werden. Daraus ergibt sich, dass, wenn die Test-DB2 übergeben wird, nur die RSSI-Werte der gefundenen APs des Testscans berücksichtigt werden. APs, die gewöhnlich an einer Position vorkommen, aber bei einem zu testenden WLAN-Scan nicht gefunden wurden, werden ignoriert. Wird Test-DB1 mit dem Minimalwert (-101dBm) übergeben, haben diese nicht gemessenen APs Einfluss auf die Berechnung, falls diese in der Fingerprinting-Datenbank nicht mit NaN dargestellt sind.

Ein Beispiel: Der Mittelwert eines AP sei an einer bestimmten Position -75 dBm. Wird z. B. bei einem Testscan an dieser Position für diesen AP kein RSS-Wert erfasst, würde bei der Übergabe von Test-DB1 nach Formel *(F. 4-16)* ein zusätzlicher Summand $(\sqrt{\ldots + (101-75)^2 + \cdots})$ unter der Wurzel entstehen. Mit der Test-DB2 hätte der AP keinen Einfluss auf den Abstandswert, da dieser NaN ist und somit ignoriert wird.

Die Ergebnisse der Tests mit den zwei Test-Datenbanken und den vier gemittelten Fingerprinting-Datenbanken sind in 4.5.1 beschrieben. Im Abschnitt 4.5.2 wird gezeigt, wie mit parallel vorhandenen WLAN-Netzen, bei denen für ein AP mehrere RSS-Werte gemessen werden, umgegangen wurde. In Kapitel 4.5.3 werden die erzielten Ergebnisse mit dem MFV-Algorithmus und Likelihood-Algorithmus präsentiert. Die Resultate der Analyse des Benutzens eines Gewichtsvektors für die euklidische Abstandsberechnung (4.4.2) werden in 4.5.4 erläutert.

4.5.1 Mittelung aller ermittelten RSS-Werte

In der ersten Testreihe wurden alle erfassten WLAN-Scans der Kalibrierungsphase zur Erstellung der gemittelten RSSI-Vektoren verwendet. Dabei wurden die WLAN-Scans, die für eine Position in alle vier Richtungen erfasst wurden, zu einem RSSI-Mittelwert-Vektor zusammengefasst. Die sich daraus ergebenden Erkennungsraten mit dem NN-Algorithmus sind in den Spalten der Tabelle 4-1 und Tabelle 4-2 dargestellt. Tabelle 4-1 zeigt die Erkennungsraten für das EI-Testgebiet, die Tabelle 4-2 die für das Heim-Testgebiet.

Die Spalten stellen die acht Kombinationsvarianten der Fingerprinting-Datenbanken und Test-Datenbanken dar, wie eingangs in 4.5 erläutert. In den Zeilen der Tabellen sind die Szenarien dargestellt. So werden in der ersten Zeile (*S2-DB*) nur die RSSI-Vektoren des Samsung Galaxy S2 als Fingerprinting- und Test-Datenbank herangezogen. In der zweiten Zeile (*HTC-DB*) sind die erzielten ER mit den Daten des HTC EVO 3D abgebildet. Die dritte Zeile (*kombinierte DB*) stellt die Ergebnisse dar, wenn alle WLAN-Scans kombiniert werden.

Die mittleren Erkennungsraten errechnen sich aus den Erkennungsraten der zuvor untersuchten Szenarien. Die jeweils höchste Erkennungsrate eines Szenarios ist durch eine dickere Schrift und eine dunklere Hintergrundfarbe gekennzeichnet. Der letzten Spalte sind die mittleren Erkennungsraten eines jeden Szenarios zu entnehmen.

Tabelle 4-1 Erkennungsraten EI-Testgebiet bei Mittelung der RSSI-Werte in alle Richtungen

Erkennungsrate für alle untersuchten Positionen									
RSSI-Vektor mit allen ermittelten RSS-Werten	**RSSI-Mittelwerten-Vektor wurden aus allen vier Richtungen erstellt**								
	Test-DB1 (-101dBm)				**Test-DB2 (NaN)**				**mittlere ER. d. Sze.**
	mean-DB		**median-DB**		**mean-DB**		**median DB**		
	-101dbm	*NaN*	*-101dBm*	*NaN*	*-101dbm*	*NaN*	*-101dbm*	*NaN*	
S2-DB	66,2%	20,4%	46,9%	20,7%	38,3%	***72,5%***	35,3%	72,0%	46,54%
HTC-DB	56,9%	34,8%	49,4%	33,6%	50,6%	67,9%	44,4%	***69,4%***	50,88%
kombinierte-DB	42,6%	17,0%	30,2%	16,1%	41,8%	***55,5%***	34,4%	53,8%	36,43%
mittlere Erkennungsrate	**55,2%**	**24,1%**	**42,1%**	**23,4%**	**43,6%**	**65,3%**	**38,0%**	**65,1%**	44,60%

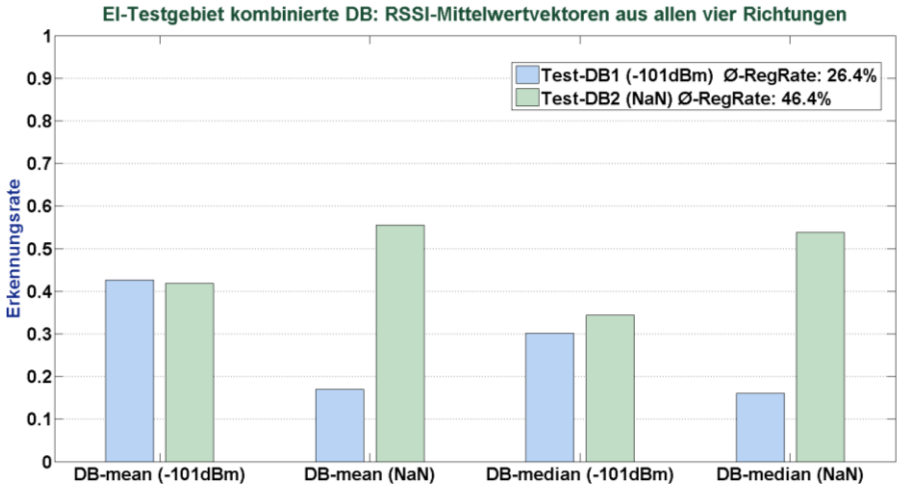
In Tabelle 4-1 ist zu erkennen, dass die höchsten Erkennungsraten durch die Kombinationen der Test-Datenbank Test-DB2 (NaN) und den Fingerprinting-Datenbanken, welche die NaN-Werte enthalten, erzielt wurden.

Die höchste mittlere Erkennungsrate von 65,3% ergab sich, wenn Test-DB2 und mean-DB (NaN) mit der Methode `disTestscan2()` kombiniert wurden. Bei dieser Variante wurden die RSSI-Mittelwert-Vektoren mittels arithmetischen Mittels berechnet, wobei nur die erfassten RSS-Werte der APs für diese Kalkulation verwendet wurden. Bei der Simulation der Positionierungsphase wurden nur jene APs für die Berechnung verwendet, welche bei den Test-WLAN-Scans erfasst wurden. Der Unterschied zwischen den Erkennungsraten bei der Anwendung von *median-DB(NaN)* und *mean-DB(NaN)* ist gering, im Mittel unterscheiden sich diese um 0,2 Prozentpunkte (PP).

Die Erkennungsraten bei der Nutzung der Datenbanken der einzelnen Handys sind höher als die der kombinierten Daten. Betrachtet man die Variante mit der höchsten mittleren Erkennungsrate, bei der die *median-DB(NaN)* und die *Test-DB2(NaN)* kombiniert wurden, ist zu sehen, dass die Erkennungsraten der Datenbanken der Handys im Schnitt ca. eine um 15 PP höhere ERn erzielen als die *kombinierten-DB.* Während bei den Szenarien „*S2-DB“* 72,5% und „*HTC-DB“* 67,9% der Positionen richtig zugeordnet wurden, konnte bei dem Szenario „*kombinierte-DB“* 55,5% richtig erkannt werden. In Abbildung 4-5 sind die

Erkennungsraten bei Verwendung der *„kombinierten-DB“* dargestellt, also jener Testdatenbank, bei der alle erfassten WLAN-Scans der Positionierungsphase verwendet werden.

Abbildung 4-5 Erkennungsraten des Szenarios der kombinierten DB im EI-Testgebiet



Die Abbildung stellt die Werte der Zeile (*kombinierte DB*) in der Tabelle 4-1 grafisch als Balkendiagramm dar. Das Balkendiagramm stellt gegenüber, wie sich die ERn ändern, wenn für das Fingerprinting Test-DB1 und Test-DB2 verwendet werden, sowie welche ERn sich bei der Verwendung des arithmetischen Mittels (DB-mean) und des Medians (DB-median) als Basis für die RSSI-Mittelwert-Vektoren ergeben. Die Ergebnisse, die mit Test-DB1 die Minimalwerte für nicht erreichte APs verwendet, sind blau dargestellt. Jene ERn, die mit der Test-DB2, die für nicht erreichte APs den NaN-Wert verwendet wird, erzielt wurden, sind grün dargestellt. Die ersten vier Balken stellen die ERn, die mit der DB-mean erzielt wurden, dar, Die letzten vier zeigen die ERn, wenn die DB-median benutzt werden.

Die Grafik zeigt, dass sich die höchste ER ergab, wenn Test-DB2 mit mean-DB(NaN) kombiniert wurde. Das ist jene Variante, bei der nicht erreichte APs völlig ignoriert werden. Mit der Test-DB1 werden nur dann ähnliche Raten erreicht, wenn diese mit einer Fingerprinting-DB, die die Minimalwerte (-101dBm) enthält, kombiniert wird. Im Durschnitt betrugen die ER mit Test-DB1 26,4% und mit Test-DB2 46,4%.

Betrachtet man die Mittleren ERn über alle Szenarien hinweg, konnte das zweitbeste Ergebnis mit der Kombination aus der mean-DB (-101dBm) und der Test-DB1 (-101dBm) erzielt werden. Hier liegt die mittlere Erkennungsrate bei 55,2 %. Bei dieser Variante werden für die Berechnung der RSSI-Mittelwert-Vektoren das arithmetische Mittel und Minimalwerte benutzt. Wie in der Einleitung beschrieben, haben bei der Verwendung der Test-DB1 nicht erreichte APs in der Positionierungsphase Einfluss auf die Berechnung des euklidischen Abstandes.

Der Tabelle 4-1 ist zu entnehmen, dass, wenn eine Datenbank, die mit Minimalwerten erstellt wurde, mit einer DB kombiniert wird, die NaN-Werte verwendet, dies tendenziell zu schlechteren Ergebnissen führte.

Tabelle 4-2 spiegelt die bereits zu Tabelle 4-1 diskutierten Kombinationen der Datenbanken und Szenarien für das Heim-Testgebiet wieder. Die Erkennungsraten sind im Schnitt deutlich höher. Ein Grund dafür ist, dass das Testgebiet größer ist und die untersuchten Punkte weiter voneinander entfernt sind.

Einige der untersuchten Punkte sind aber auch nicht weit voneinander entfernt. Es sind jene, die jeweils um die Ecken der Häuser angeordnet sind. Abbildung 4-7 zeigt das Heim-Testgebiet, hier sind beispielsweise die Positionen 3 und 5 oder 9 und 10 in einem Abstand von ca. 14 m und 10 m zu finden. In wieweit es möglich ist, diese zu unterscheiden, war eine der Fragestellungen der Testreihe.

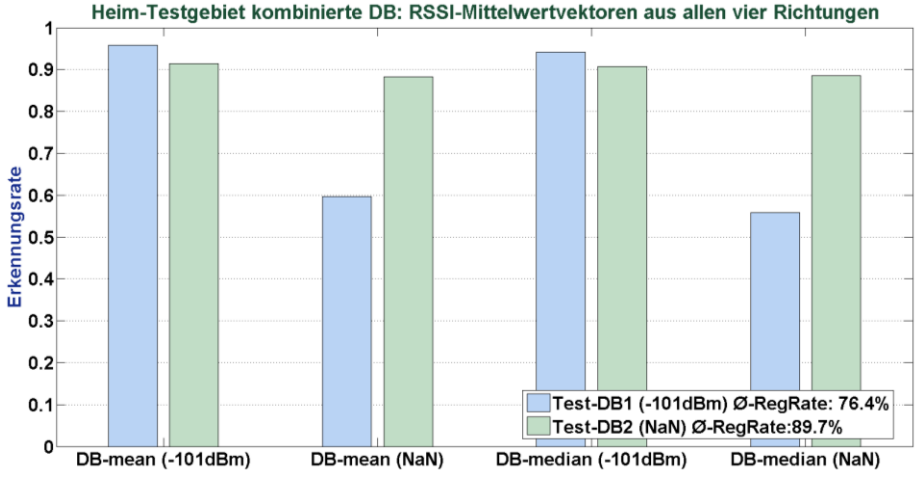
Tabelle 4-2 Heim-Testgebiet Mittelung in alle Richtungen

Erkennungsrate für alle untersuchten Positionen									
RSSI-Vektor mit allen ermittelten RSS-Werten	Mittelung in alle Richtungen								
	Test-DB1(-101dBm)				Test-DB2(NaN)				mittlere ER. Sze.
	mean-DB		median-DB		mean-DB		median-DB		
	-101dbm	*NaN*	*-101dbm*	*NaN*	*-101dbm*	*NaN*	*-101dbm*	*NaN*	
S2-DB	***94,1%***	67,5%	90,2%	66,8%	91,6%	92,0%	90,2%	92,3%	85,59%
HTC-DB	95,7%	86,2%	92,6%	85,5%	92,9%	***96,1%***	90,4%	***96,1%***	91,94%
kombinierte-DB	***95,8%***	***59,7%***	***94,2%***	***55,8%***	***91,4%***	***88,2%***	***90,7%***	***88,6%***	83,05%
mittlere Erkennungsrate V.	***95,2%***	***71,1%***	***92,3%***	***69,4%***	***92,0%***	***92,1%***	***90,4%***	***92,3%***	86,85%

Die höchste mittlere Erkennungsrate wurde im Heim-Testgebiet mit der Variante mean-DB(-101dBm) und Test-DB1 erzielt. Das ist jene Variante, bei der nicht erreichte APs mit -101 dBm dargestellt werden. Die Mittelung der RSSI-Vektoren mit -101 dBm als Minimalwert kann, wie schon erwähnt, als eine Gewichtung betrachtet werden. Diese Gewichtung eliminiert instabile APs, was sich im öffentlichen Raum wohl vorteilhaft auswirkt. Es kann ja davon ausgegangen werden, dass es sich bei den meisten APs um Private Netze handelt, deren Verfügbarkeit nicht immer gewährleistet ist. Nichtsdestotrotz sind die Unterschiede bei Verwendung der Test-DB2 (NaN) gering. Mit den RSSI-Vektoren des HTC wurden bei der Nutzung der Test-DB2 sogar um 0,4 % Prozentpunkte bessere Ergebnisse erzielt.

In Abbildung 4-6 ist das Szenario mit der „*kombinierte-DB“,* bei dem alle RSSI-Vektoren verwendet werden, als Balkengrafik dargestellt. Das Diagramm stellt die Werte der Zeile „*kombinierte-DB“* der Tabelle 4-2 grafisch dar. Die Grafik zeigt, wie Abbildung 4-5, eine Gegenüberstellung der erzielenden ERn der möglichen Datenbank-Kombinationen.

Abbildung 4-6 Erkennungsraten des Szenarios der kombinierten DB im Heim-Testgebiet



Die höchste mittlere Erkennungsrate beim Szenario mit der „*kombinierte-DB“* wurde im Heim-Testgebiet mit der Variante mean-DB(-101dBm) und Test-DB1 erzielt. Aus der Grafik und aus Tabelle 4-2 zeigt sich, dass die Kombination einer DB mit Minimalwerten und einer DB mit NaN-Werten zu schlechteren Ergebnissen führte. Besonders deutlich wird dies, wenn die

Test-DB1 für die Positionierung verwendet wird. Der umgekehrte Fall, bei dem Test-DB2(NaN) verwendet wird, wo also nur erfasste APs bei der Positionierung berücksichtigt werden, führte kaum zu schlechteren Ergebnissen. Beim Szenario mit der „*kombinierte-DB*“ betrug die mittlere Erkennungsrate daher 76,4 %, wenn Test-DB1, und 89,7 %, wenn Test-DB2 verwendet wurde.

Tabelle 4-3 zeigt im Detail die Erkennungsraten jener Positionen im Heim-Testgebiet, die nicht zu 100 % korrekt zugewiesen werden konnten. Es handelt sich um die Variante Test-DB1/meanDB(-101dBm) für das Szenario „*kombinierte-DB*“. Die Spalte „*verwechselt*“ beschreibt, wie oft und zu welcher Position eine PointID falsch zugewiesen wurde. Die nicht angeführten ERn der restlichen 16 untersuchten Punkte beträgt 100 %. Betrachtet man die Abbildung 4-7, ist zu erkennen, dass nahe, aber um Gebäudeecken angeordnete Positionen nicht verwechselt werden, da diese nicht in Tabelle 4-3 angeführt wurden.

Beispiele für diese sind die Punkte 3 und 5 oder 9 und 10. Deren Erkennungsrate liegt bei 100 %, daher sind diese Positionen auch in der Tabelle 4-3 nicht angeführt.

Tabelle 4-3 Auszug ER Details kombinierte DB

ER <100% der PointIDs – beste Variante			
PointID	ER	korrekt/Gesamt	verwechselte **IDs**
12	94.4%	34/36	2x**20**
13	79.2%	19/24	5x**12**
16	91.7%	11/12	1x**2**
17	91.7%	22/24	1x**16**; 1x**3**
21	95.8%	23/24	1x**11**
22	95.8%	23/24	1x**21**
24	91.7%	22/24	2x**1**

Auszug aus der Variante
Test-DB1/meanDB(-101dBm)
für die kombinierte-DB für Heim-Testgebiet

Abbildung 4-7 Heim-Testgebiet Ausschnitt

Verwechslungen gibt es bei den PointIDs, die zwar weiter auseinanderliegen, jedoch eine direkte Sichtverbindung haben. Bestes Beispiel ist die Position 12, diese wird zweimal mit der ca. 30 m entfernten Position 20 verwechselt. Außerdem wird Position 12 fünfmal falsch anstatt der 39 m entfernten Position 13 erkannt. Hingegen wurden die Positionen 3 und 5, sowie 9 und 10, die in etwa 10 m bzw. 14 m voneinander entfernt sind, nicht verwechselt.

Dieses Ergebnis bestätigt die Annahme, dass durch Beachtung von baulichen Strukturen bessere Ergebnisse bei den Erkennungsraten (ERn) erzielt werden können.

Zusammenfassung und Konklusion zu RSSI-Mittelwert-Vektoren

In diesem Abschnitt wurde das WLAN-Fingerprinting simuliert, in dem verschiedene Datenbanken aus RSSI-Mittelwert-Vektoren erstellt wurden und mit zwei Test-Datenbanken mittels des NN-Algorithmus analysiert wurden. Untersucht wurden zwei Berechnungsmethoden für den Mittelwert, das arithmetische Mittel (Mean) und der Median. Außerdem wurde analysiert, wie mit nicht erreichten APs umgegangen werden kann, in dem ein Minimalwert verwendet wird oder die APs mit NaN-Wert ignoriert werden.

Zwischen dem arithmetischen Mittel und dem Median konnte kein signifikanter Unterschied festgestellt werden. Vergleicht man die besten gemittelten ERn des arithmetischen Mittels und Medians, konnten mit dem Mean etwas höhere Erkennungsraten erzielt werden. Im EI-Testgebiet liegt der Unterschied bei 0,2 PP und im Heim-Testgebiet bei 2,9 PP.

Die Verwendung von Minimalwerten bei nicht erreichten APs wirkte sich im EI-Testgebiet negativ auf die ER aus, im Heim-Testgebiet konnten die ERn erhöht werden. Eine eindeutige Aussage, welche Strategie zu besseren Ergebnissen führt, kann daher nicht gemacht werden. Ein Hinweis auf diese Differenz könnte an den Unterschieden in den Testgebieten liegen, diese unterscheiden sich vor allem durch die Anzahl und der Verfügbarkeit der APs. Wie schon erläutert, werden bei der Verwendung eines Minimalwertes kaum Verfügbare APs eliminiert.

Dass beim WLAN-Fingerprinting die Verwendung der Minimalwerte beim Heim-Testgebiet zu höheren ER führt, spricht dafür, dass diese Strategie sinnvoll sein kann. Denn bei diesem Testgebiet kommen weniger und unzuverlässigere APs vor, die durch diese Strategie weniger stark gewichtet werden.

Als weniger sinnvoll erwies sich die Kombination von den zwei Darstellungsformen der Datenbanken, also wenn z. B. die Fingerprinting Datenbank mit Minimalwerten berechnet wird und nur gefundene APs (NaN-DB) zur Positionierung herangezogen werden. Die Mischung von Minimalwert- und NaN-Wert-Datenkanten führte immer zu schlechteren Erkennungsraten.

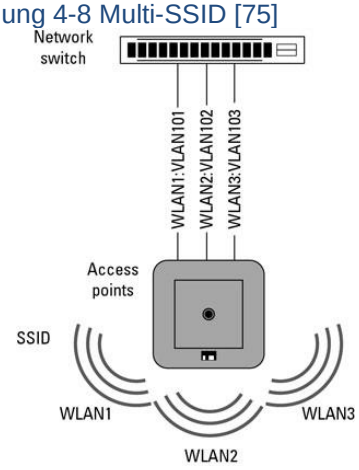
4.5.2 Selektion von RSSI-Werten bei multiplen WLAN-Netzen

Im EI-Testgebiet wurde eine weitere Testreihe durchgeführt, wo bei mehrfachen RSS-Werten eines APs jeweils ein Wert ausgewählt wurde. Der Grund für diese multiplen RSS-Werte ist, dass im UNI-Gebäude parallel mehrere WLAN-Netze [69] angeboten werden. Bei Multi-SSID [75] bietet ein physikalisch einmal vorhandener APs multiple WLAN-Netze mit verschieden MAC-Adressen an, wie in Abbildung 4-8 skizziert ist. Tabelle 4-4 zeigt als Beispiel den Auszug eines WLAN-Scans, der vier WLAN-Netze eines physikalisch einzelnen APs darstellt. Alle vier RSS-Werte sind, abgesehen vom letzten Wert, gleich, und dieser unterscheidet sich auch nur um 1 dBm.

Tabelle 4-4 Auszug WLAN-Scan PointID 8004

SSID	BSSID	level
eduroam	0c:68:03:4c:a5:8f	-66 dBm
wlanipsec	0c:68:03:4c:a5:8e	-66 dBm
tunetguest	0c:68:03:4c:a5:8d	-66 dBm
tunet	0c:68:03:4c:a5:8c	-65 dBm

Abbildung 4-8 Multi-SSID [75]



Da sich die Mac-Adressen eines AP bei diesen multiplen WLAN-Netzen nur um die letzte Stelle unterscheiden, wie in Tabelle 4-4 zu sehen ist, wurde die letzte Stelle beim Import der

RSS-Werte ignoriert. So wurde von den multiplen RSS-Werten des APs nur ein Wert verwendet.

Der Funktion `importAllWifiscans()` kann daher der Parameter *lend* übergeben werden. Dieser definiert die Länge der Zeichen der Mac-Adresse, welche verwendet werden soll. Soll die ganze Mac-Adresse verwendet werden, müssen alle 17 Zeichen beachtet werden. Bei dieser Testreihe wurde der Parameter *lend* auf 16 gesetzt, dadurch wird die letzte Stelle der Mac-Adresse ignoriert. Beim Zusammenfügen des `WIFIDATASETS` zum `WIFISCANS` Datensatz mit der Funktion `orderByScan()` werden die RSS-Werte mit der gleichen `APId` überschrieben. Im obigen Beispiel, in Tabelle 4-4, wurden die 4 WLAN-Netze als ein AP gewertet, gespeichert wurde der RSS-Wert in `WIFISCANS`, der als letzter erfasst wurde, also *tunet* mit -65 dBm.

Ziel dieser Testreihe war es, herauszufinden, ob sich die Selektion der multiplen RSS-Werte eines AP auf die Erkennungsraten auswirkt. Die mehrfachen RSS-Werte eines physikalisch einmal vorhandenen AP wirken sich wie eine Gewichtung für diesen AP aus, da die multiplen RSS-Werte mehrfach genutzt werden. Sind weiter APs vorhanden, die keine multiplen WLAN-Netze zur Verfügung stellen, kann dies das WLAN-Fingerprinting verzerren. Da diese multiplen WLAN-Netze primär im UNI Gebäude vorhanden sind, wurde die hier betrachtete Analyse auch nur im EI-Testgebiet durchgeführt. Die Testreihe wurde für alle WLAN-Scans und die der einzelnen Handys durchgeführt.

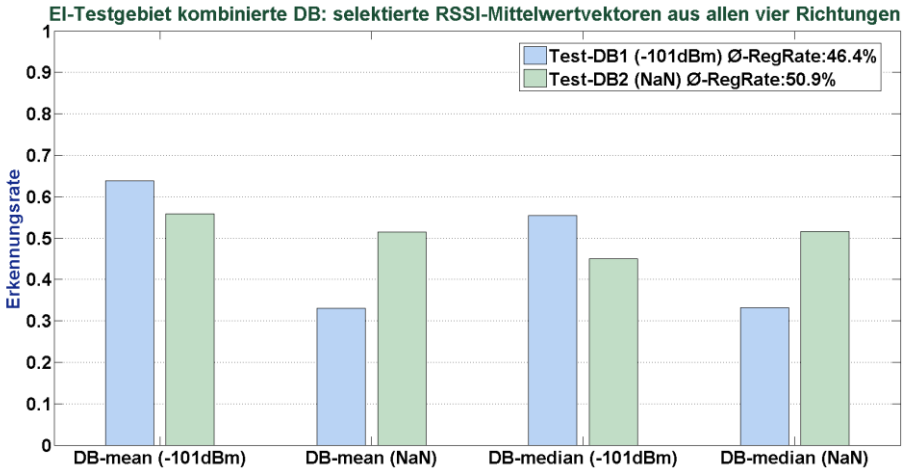
In Tabelle 4-5 sind die Ergebnisse für die Testreihe im EI-Testgebiet dargestellt. Verwendet wurden die gemittelten RSS-Werte aller Richtungen, jedoch wurde für jeden physikalisch einmal vorhandenen AP ein RSS-Wert pro WLAN-Scan ausgewählt.

Tabelle 4-5 EI-Testgebiet mit gemittelten RSSI-Vektoren und selektierten RSS-Werten

Erkennungsrate für alle untersuchten Positionen									
RSSI-Vektor mit selektierten RSS-Werten	**Mittelung in alle Richtungen**								
	Test-DB1(-101dBm)				**Test-DB2(NaN)**				**mittlere ER. Sze.**
	mean-DB		**median DB**		**mean DB**		**median DB**		
	-101dbm	*NaN*	*-101dbm*	*NaN*	*-101dbm*	*NaN*	*-101dbm*	*NaN*	
S2-DB	61,0%	24,2%	48,9%	23,7%	52,1%	***62,7%***	39,3%	61,0%	46,61%
HTC-DB	68,7%	53,6%	66,2%	53,9%	56,4%	64,9%	55,4%	***68,9%***	61,00%
kombinierte-DB	***63,8%***	33,0%	55,4%	33,2%	55,8%	51,4%	45,0%	51,6%	48,65%
mittlere Erkennungsrate	***64,5%***	37,0%	56,8%	36,9%	54,8%	59,7%	46,6%	60,5%	52,10%

Tabelle 4-5 ist mit Tabelle 4-1 vergleichbar, z. B. können die höchsten Erkennungsraten der Szenarien *„kombinierten DB“* verglichen werden. So ist die höchste ER mit den selektierten RSS-Werten um 8,3 PP höher, als wenn alle RSS-Werte benutzt werden. Wird nur ein RSS-Wert pro physikalischen AP verwendet, ergeben die Datenbanken, die den Minimalwert verwenden, höhere Erkennungsraten. Da diese Methode auch beim Heim-Testgebiet, wo es keine mehrfachen RSS-Werte gab, zu besseren Erkennungsraten führte, spricht dies dafür, dass es Sinn macht, Minimalwerte zu verwenden. Abbildung 4-9 zeigt die Erkennungsraten des Szenarios *„kombinierten DB“* mit selektierten RSS-Werten, sie ist zu vergleichen mit der Abbildung 4-5 und stellt die Werte in Tabelle 4-5 dar.

Abbildung 4-9 ERn des Szenarios der selektierten kombinierte DB im EI-Testgebiet



Vergleicht man die mittleren ER über alle Werte der zwei Tabellen, ist zu erkennen, dass diese durch die Selektion der RSS-Werte um ca. 8 PP erhöht wurde. Werden die Handys jedoch einzeln betrachtet, führt die Selektion von RSS-Werten zu geringen höchsten Erkennungsraten. Im Mittel sind die ER der Szenarien „*S2-DB*“ und „*HTC-DB*“ aber nicht gesunken. Die mittlere ER des Szenarios mit der *S2-DB* ist in etwa gleich geblieben, die mit der *HTC-DB* ist um ca. 10 PP erhöht worden. Dies liegt daran, dass bestimmte Kombinationen von Fingerprinting-Datenbanken und Test-Datenbanken, die zuvor zu geringen Erkennungsraten führten, durch die Selektion von RSS-Werten deutlich höhere Raten erreichten. In gewisser Weise zeigt diese Testreihe ein entzerrtes Bild von Tabelle 4-1, da jeder physikalisch einmal vorhandene AP nur einmal bei der Berechnung berücksichtigt wurde. Zugleich zeigt es auch, dass die Gewichtung von bestimmten APs durchaus sinnvoll sein kann. Dies soll in Abschnitt 4.5.4 systematisch genutzt werden. Es kann also angenommen werden, dass eine Reduzierung mehrfacher RSS-Werte physikalisch einmal vorhandener APs sinnvoll ist.

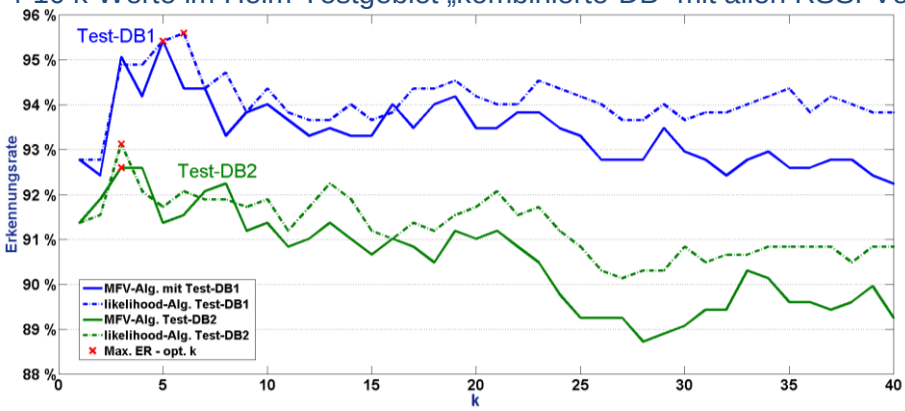
4.5.3 MFV-Algorithmus & Likelihood-Algorithmus

Die in 4.4.3 beschriebenen Algorithmen wurden in der folgenden Testreihe untersucht. Wie bereits beschrieben, werden bei den Algorithmen RSSI-Vektoren mit den geringsten euklidischen Abständen zum Test-Vektor verwendet, um eine Position zu bestimmen.

Vorgestellt wurden zwei Algorithmen, die sich in ihrer Auswahlmethode unterscheiden. Der MFV-Algorithmus wählt aus k *„am besten passenden“* Positionen jene aus, die am öftesten errechnet wurden. Der Likelihood-Algorithmus berechnet für jede infrage kommende Position einen Wahrscheinlichkeitswert und wählt jene mit dem höchsten Wert aus.

Da es zwei Algorithmen und zwei Darstellungsformen der RSSI-Vektoren gibt, ergeben sich vier kombinierbare Varianten. Für jede Variante und jedes Szenario wurde nach dem k Wert gesucht, der die höchste ER ergibt. Dazu wurden verschiedene Werte für k empirisch getestet. In Abbildung 4-10 sind die ermittelten Erkennungsraten für k von 1 bis 40 für das Heim-Testgebiet mit dem Szenario „*kombinierten-DB*“ dargestellt.

Abbildung 4-10 k-Werte im Heim-Testgebiet „kombinierte-DB“ mit allen RSSI-Vektoren



Eine sinnvolle Wahl für k ist abhängig von der Anzahl der RSSI-Vektoren, die eine Position repräsentieren. Im Heim-Testgebiet wurden 40 (2xDurchgänge, 4xRichtungen, 5xWLAN-Scans) RSSI-Vektoren für eine Position in der Kalibrierungsphase erfasst. Wie der Abbildung 4-10 zu entnehmen ist, werden die höchsten Erkennungsraten für k gleich 4 bis 6 erreicht. Daher wäre k=5 eine gute Wahl.

Wenn nicht alle Positionen durch die gleiche Anzahl von RSSI-Vektoren repräsentiert werden, kann es zur Situation kommen, dass eine falsche Position gewählt wird, weil diese öfter in der Datenbank vertreten ist. Daher sollte k anhand der Position mit den wenigsten RSSI-Vektoren definiert werden.

Ergebnisse im Detail

Die erzielten Erkennungsraten sind in Tabelle 4-6 und Tabelle 4-7 für die beiden Testgebiete aufgelistet. In den Tabellen sind jene Erkennungsraten zu finden, die bei einem jeweils optimierten k erreicht wurden.

Tabelle 4-6 EI-Testgebiet - MFV-Algorithmus & Likelihood-Algorithmus

RSSI-Vektor mit…		Alle RSSI-Vektoren in Datenbank				
		MFV-Algorithmus		Likelihood-Algorithmus		mittlere ER. Sze.
		DB1(-101dBm)	DB2(NaN)	DB1(-101dBm)	DB2(NaN)	
… allen ermittelten RSS-Werten	**S2-DB**	54,2%	49,4%	***54,7%***	49,4%	51,93%
	HTC-DB	47,4%	***54,6%***	46,9%	54,1%	50,75%
	kombinierte- DB	***53,5%***	52,0%	***53,6%***	52,0%	52,78%
	mittlere Erkennungsrate	**51,7%**	**52,0%**	**51,7%**	**51,8%**	51,80%
… selektierten RSS-Werten	**S2-DB**	61,0%	61,5%	61,0%	***61,7%***	61,30%
	HTC-DB	***72,7%***	58,4%	***72,7%***	59,1%	65,73%
	kombinierte- DB	62,6%	58,0%	***63,1%***	58,0%	60,43%
	mittlere Erkennungsrate	**65,4%**	**59,3%**	***65,6%***	**59,6%**	62,48%

Vergleicht man die Erkennungsraten der zwei Algorithmen, sind keine signifikanten Unterschiede zu erkennen. Der Likelihood-Algorithmus führt zu etwas besseren Erkennungsraten, die sich jedoch vom MFV-Algorithmus kaum unterscheiden. Betrachtet man Abbildung 4-10, ist zu erkennen, dass der Likelihood-Algorithmus auch bei höheren k-Werten zu besseren Erkennungsraten führt. Der Likelihood-Algorithmus ist also etwas stabiler, was die Wahl von k betrifft.

Vergleicht man die besten Erkennungsraten der Tabelle 4-6, wo alle RSS-Werte verwendet wurden, mit der Tabelle 4-1 (mit den gemittelten RSSI-Vektoren), ist zu erkennen, dass dort

die ERn höher waren, als hier mit dem MFV-Algorithmus und Likelihood-Algorithmus. Die ER des Szenarios „*S2-DB*“ war um 17,8 PP, die des „*HTC-DB*“ um 14,8 PP höher. Mit der *kombinierten-DB* konnte eine um 1,9 PP höhere ER erzielt werden.

Der untere Teil der Tabelle 4-6 kann mit der Tabelle 4-5 verglichen werden. Es ist zu erkennen, dass sich die besten ERn nur geringfügig unterscheiden. Bei der Verwendung der *HTC-DB* konnte hier eine um 3,8 PP höhere Erkennungsrate erreicht werden. Die Varianten mit der S2-DB und der kombinierten-DB sind um ca. 1 PP geringer als mit den gemittelten RSSI-Vektoren.

Tabelle 4-7 Heim-Testgebiet - MFV-Algorithmus & Likelihood-Algorithmus

RSSI-Vektor mit allen ermittelten RSS-Werten	Alle RSSI-Vektoren in Datenbank				
	MFV-Algorithmus		Likelihood-Algorithmus		mittlere ER. Sze.
	DB1(-101dBm)	DB2(NaN)	DB1(-101dBm)	DB2(NaN)	
S2-DB	***93,0%***	91,3%	92,7%	***93,0%***	92,50%
HTC-DB	95,4%	95,4%	***95,7%***	95,4%	95,48%
kombinierte-DB	95,4%	92,6%	***95,6%***	93,1%	94,18%
mittlere Erkennungsrate	**94,6%**	**93,1%**	***94,7%***	**93,8%**	94,05%

Vergleicht man die Ergebnisse des Heim-Testgebietes aus Tabelle 4-7 und Tabelle 4-2 miteinander, sieht man, dass die höchsten Erkennungsraten sich maximal um 1,1 PP unterscheiden. Der Ansatz, bei dem alle RSSI-Vektoren verwendet werden, führt dabei zu etwas geringeren Erkennungsraten. Die ERn der Szenarien „*kombinierten-DB*“ unterscheiden sich nur um 0,2 PP.

Zusammenfassend kann gesagt werden, dass mit dem untersuchten MFV-Algorithmus & Likelihood-Algorithmus keine besseren Erkennungsraten erzielt werden konnten. Die angegebenen ERn sind nur mit dem optimalen k Wert erreicht worden. Dieser ist aber bei einer realen Positionierung nicht immer bekannt, wobei sich k=5 als recht guter Wert für k gezeigt hat.

Neben den schlechteren Erkennungsraten spricht die Laufzeit gegen den Ansatz, alle RSSI-Vektoren für das WLAN-Fingerprinting zu verwenden. Zum Vergleich: wenn zu allen erfassten RSSI-Vektoren der 49 gewählten Punkte der euklidische Abstand berechnet werden soll, bedeutet dies, dass 5726 (kompletter RSSI-Vektoren Datensatz der Kalibrierungsphase) Berechnungen durchgeführt werden müssen. Wird die Datenbank mit den RSSI-Mittelwert-Vektoren verwendet, muss pro Position nur eine Berechnung, also 49, durchgeführt werden.

4.5.4 Gewichtung

Die in 4.4.2 beschriebene Gewichtungsoption der Funktion `disTestscan2()` ermöglicht es, einen gewichteten Abstandswert zu berechnen. Wie dieser gewichtete Abstandswert berechnet wird, ist in Formel *(F. 4-18)* definiert.

Die Idee zur Gewichtung begründet sich darauf, dass für gewisse Positionen die RSS-Werte bestimmter APs relevanter sind als andere. Ein Beispiel, zwei benachbarte Räume unterscheiden sich dadurch, dass nur in einem Raum ein APs X gerade noch erreichbar ist. Da sich sonst die APs und deren RSS-Werte sehr ähnlich sein würden, sollten die RSS-Werte des APs X stärker berücksichtigt werden, um die Räume auseinanderhalten zu können.

Daher können der Funktion `disTestscan2()` ein Gewichtungsvektor oder eine Gewichtungsmatrix übergeben werden. Dadurch werden bei der Berechnung des euklidischen Abstandes bestimmte Summanden anders gewichtet. Für die hier beschriebenen Tests wurde jedoch nur ein Gewichtungsvektor für alle Positionen verwendet. Gefunden wurde der Vektor mittels einer einfachen Optimierungsstrategie, die nach dem Schema einer „Nearest Neighbor Search" [78] agierte.

Die Optimierungsstrategie

Zu Beginn des Algorithmus werden alle Gewichtungsfaktoren w_{APi}=1 gesetzt. Anschließend wird der 1. Wert gewählt und die Gewichtung von w_{AP1}um den Faktor 1 erhöht. Kommt es zu einer Verbesserung, wird der aktuelle Gewichtungsvektor beibehalten und w_{AP1} *wieder um den Faktor 1 erhöht*. Dies wird so lange gemacht, bis keine Verbesserung mehr erzielt wird.

Anschließend wird mit dem nächsten Gewichtungsfaktor fortgefahren und der oben beschriebene Vorgang wiederholt. So werden alle Gewichtungsfaktoren von w_{AP1} *bis* w_{APn} *durchlaufen.* Kommt der Algorithmus zum letzten Gewichtungsfaktor, beginnt der Algorithmus wieder beim ersten Faktor. Abgebrochen wird, wenn es in einem Durchgang keine Verbesserungen mehr gibt oder die definierte Zeit abgelaufen ist. In der Arbeit wurde 5 Min. nach einem besseren Ergebnis gesucht.

Bei dieser Untersuchung zur Gewichtung der RSS-Werte der APs fand nur ein sehr einfaches Optimierungsverfahren Anwendung. Für weitere Untersuchungen könnte ein komplexeres Verfahren genutzt werden. Einen Überblick über heuristische Optimierungsverfahren wird in [69] gegeben.

Erzielte Ergebnisse

Tabelle 4-8 und Tabelle 4-9 zeigen für alle untersuchten Szenarien die besten ERn. Dabei werden auch die ERn verwendet, die im nächsten **Kapitel 5** durch die Miteinbeziehung der Orientierungsinformationen erreicht wurden. Neben den besten ERn sind die verbesserten ERn durch die Benutzung des gefundenen Gewichtungsvektors angegeben. Die Tabelle 4-8 zeigt die Ergebnisse für das EI-Testgebiet. Betrachtet wurden die Datenbanken, die Szenarien der einzelnen Handys und das Szenario *„kombinierte-DB"* jeweils mit und ohne selektierten RSS-Werten. Die Tabelle 4-9 zeigt die Ergebnisse des Heim-Testgebietes für die drei Datenbanken.

Tabelle 4-8 EI-Testgebiet beste ERn - gewichtet

RSSI-Vektor mit..		*Beste ER*	*beste Methode AP gewichtet*	*Verb. Abs.*
... allen ermittelten RSS-Werten	**S2-DB**	72,5%	75,6%	3,1 %
	HTC-DB	69,4%	75,7%	6,3 %
	kombinierte-DB	56,9%	62,7%	5,8 %
	mittlere Erkennungsrate	***66,3%***	***71,3%***	***5,0 %***
... einem RSS-Werten pro phys. AP	**S2-DB**	65,2%	69,3%	4,1 %
	HTC-SB	75,2%	81,7%	6,5 %
	kombinierte- DB	66,8%	70,4%	3,6 %
	mittlere Erkennungsrate	***69,1%***	***73,8%***	***4,1 %***

Tabelle 4-9 Heim-Testgebiet beste ERn - gewichtet

RSSI-Vektor mit allen ermittelten RSS-Werten	*Beste ER*	*beste Methode AP gewichtet*	
S2-DB	94,1%	99,3%	5,2 %
HTC-DB	96,5%	97,2%	0,7 %
kombinierte-DB	95,8%	95,8%	0,0 %
mittlere Erkennungsrate	***95,4%***	***97,4%***	***2,0 %***

Durch die Verwendung eines Gewichtungsvektors konnten die besten Erkennungsraten im EI-Testgebiet im Mittel um ca. 5 PP verbessert werden. Die erzielten Verbesserungen sind in der Tabelle 4-8 dargestellt. Im Heim-Testgebiet konnten im Mittel ca. 2 PP höhere

Erkennungsraten erzielt werden. Die detaillierten Ergebnisse sind in der Tabelle 4-9 zu sehen.

Durch die Gewichtung konnten bessere Ergebnisse erzielt werden. Es zeigt sich, dass diese Strategie grundsätzlich funktioniert. Aufgrund des sehr einfachen Optimierungsverfahrens kann nicht davon ausgegangen werden, dass die besten Gewichtungsvektoren gefunden wurden. Daher sind noch höher ERn durchaus im Bereich des Möglichen. Dies würde jedoch weitere Untersuchungen erfordern.

4.6 Zusammenfassung

Dieses Kapitel setzt sich mit der Simulation und Analyse verschiedener Varianten des WLAN-Fingerprintings auseinander. Dazu wurden WLAN-Scans in zwei Testgebieten mit dem CPS-App erfasst und in MATLAB importiert. Mit diesem Framework an Funktionen wurden die beschriebenen Untersuchungen durchgeführt. Mit dem entwickelten MFW können in Zukunft noch viele weitere Analysen gemacht werden.

Getestet wurde eine Methode, bei der für jede Position ein Mittelwert-RSSI-Vektor berechnet wurde. Für die Berechnung eines Mittelwertes wurden das arithmetische Mittel und der Median getestet. Zwischen den zwei Methoden konnte kein großer Unterschied festgestellt werden. Das arithmetische Mittel führte jedoch in allen Tests zu etwas besseren Erkennungsraten.

Mit den Algorithmen, die alle erfassten RSSI-Vektoren für das Fingerprinting benutzten, konnten keine besseren Ergebnisse erzielt werden. Da diese rechenintensiver sind als wenn nur ein RSSI-Vektor pro Position berücksichtigt wird, sind diese Algorithmen für herkömmliches WLAN-Fingerprinting nicht besonders geeignet.

Allgemein konnten tendenziell höhere Erkennungsraten erzielt werden, wenn die erfassten WLAN-Scans der Testgeräte separat betrachtet wurden, als wenn diese in einer Fingerprinting-Datenbank kombiniert wurden.

Die Verwendung eines RSS-Minimalwertes führte, wenn APs nicht erreicht werden, im Heim-Testgebiet zu besseren Ergebnissen. Ebenfalls zu etwas besseren Erkennungsraten führte diese Strategie, wenn multiple RSS-Werte von physikalisch einmal vorhandenen APs selektiert wurden.

Durch die Selektion mehrfacher RSS-Werte von einzelnen APs im EI-Testgebiet konnte bei der „kombinierten-DB“ eine deutlich höhere Erkennungsrate erzielt werden.

Im Heim-Testgebiet konnte gezeigt werden, dass durch bauliche Strukturen Positionen so gewählt werden können, dass diese mittels WLAN-Fingerprinting gut unterschieden werden können, auch wenn diese nicht weit voneinander entfernt waren.

Gezeigt konnte auch werden, dass mittels eines Gewichtungsvektors für die einzelnen Summanden der RSS-Werte des euklidischen Abstandes die Erkennungsraten verbessert werden können.

5 WLAN-Fingerprinting mit intelligenten Checkpoints (iCPs)

In der Einleitung in 1.2 wurde bereits das Konzept der iCPs erläutert. Ansätze, wie diese iCPs mittels WLAN-Fingerprinting unterschieden und erkannt werden könnten, werden in diesem Kapitel vorgestellt und diskutiert. In weiterer Folge sollen die iCPs in der Bewegung erkannt werden und der zurückgelegte Weg zwischen den iCPs mit einem INS-Algorithmus erfasst werden. Dies wird in **Kapitel 6** behandelt. Kapitel 5 fasst die Testreihen zusammen, in denen versucht wurde, die iCPs mittels WLAN-Fingerprinting, Handy-Sensoren und anhand einer logischen Abfolge zu erkennen.

Das Konzept der iCPs macht sich die Struktur eines Gebäudes zunutze. Um in einen bestimmten Raum eines Gebäudes zu gelangen, müssen gewisse Punkte passiert werden. Zuerst muss ein Eingang gewählt werden, der meist in eine Aula oder Ähnliches führt. Dann führt der Weg entweder durch diesen Vorraum oder es wird das Stockwerk gewechselt. Um in den nächsten Stock zu gelangen, muss entweder der Lift oder die Treppe benutzt werden, usw. Mit iCPs sollen diese Wegentscheidungen erkannt und die logische Abfolge genutzt werden. Intelligent sind diese Checkpoints auf mehrfache Weise.

Erstens sollten die Positionen so gewählt werden, dass sie gut zu unterscheiden sind. Dazu können die vorhandenen baulichen Gegebenheiten genutzt werden, so dass möglichst kein Sichtkontakt zwischen den Punkten besteht. Wie bereits in Tabelle 4-3 gezeigt wurde, eignen sich zum Beispiel Ecken und ähnliche Strukturen gut, um iCPs mittels Fingerprinting unterscheiden zu können. Für das EI-Testgebiet wurden dazu 17 iCP gewählt; dies ist in 5.2 beschrieben.

Zweitens können die iCPs aufgrund der Gebäudestruktur nur in einer bestimmen Reihenfolge passiert werden. Dieser Zusammenhang kann auf verschiedene Weise beschrieben werden. Die naheliegendste Form ist ein Graph, der die Zusammenhänge vollständig beschreibt. Die Erstellung für ein ganzes Gebäude ist aufwendig. Möglich wäre auch eine Logik, welche die iCPs in Ebenen einteilt, wie dies in Abbildung 1-1 skizziert wurde. In dieser Arbeit wurde eine einfache Logik verwendet, die iCPs in Abschnitte einteilt, bei denen Wegentscheidungen getroffen werden. Die Resultate, die beim WLAN-Fingerprinting durch die Miteinbeziehung eines sequenziellen Ablaufs erzielt wurden, werden in 5.3 präsentiert und diskutiert.

Des Weiteren können Sensordaten des Handys und ortsgebundene Informationen an den Positionen der iCPs genutzt werden. Ein Beispiel ist die Verknüpfung des WLAN-Fingerprintings mit dem Orientierungssensor des Handys. Dadurch ist es möglich, dass WLAN-Fingerprinting orientierungsabhängig zu betreiben. Die Resultate dieser Tests sind in 5.1.2 dargestellt und erläutert. Weitere Möglichkeiten, den Orientierungssensor mit ortsgebundener Information zu kombinieren, werden im **Kapitel 6** dargelegt.

Die verwendeten Testdaten sind in [37, Testdaten\EI-Testgebiet\iCPEbenen] abrufbar.

5.1 WLAN-Fingerprinting mit Berücksichtigung der Richtung des Smartphones

Beim herkömmlichen WLAN-Fingerprinting werden für eine Position WLAN-Scans in vier um 90° versetzte Richtungen erfasst und die RSS-Werte gemittelt [71] . Der Grund ist, dass der menschliche Körper die Signale der APs abschwächt, da dieser zu ca. aus 70 % Wasser besteht. Messungen in [39] zeigten, dass die Signalstärke im Bereich von 10-30 dBm abnehmen kann, wenn der WLAN-Receiver durch einen menschlichen Körper abgeschirmt wurde.

Ausgehend davon, dass beim Navigieren mit einem Handy dieses in der Hand vor dem Körper gehalten wird (Abbildung 6-1), kann die Orientierung des Gerätes benutzt werden, um daraus zu schließen, in welcher Richtung das Handy durch den menschlichen Körper abgeschattet wird. Ist dies bekannt, müsste nicht auf gemittelte Werte der vier Richtungen zurückgegriffen werden. Für das Fingerprinting könnten dann die Werte herangezogen werden, die mit der entsprechend gleichen Haltung erfasst wurden. Wie Informationen des Orientierungssensors mit dem WLAN-Fingerprinting kombiniert werden können, wird hier betrachtet und die erzielten Resultate präsentiert. Die Ergebnisse werden mit denen aus 4.5 verglichen, wo die RSS-Werte aus allen Richtungen gemittelt wurden.

Für die Kalibrierungs- und Positionierungsphase wurden WLAN-Scans für eine Position immer in allen vier definierten Richtungen (siehe 8.3.1 und 8.3.2) erfasst. Mit dem CPS-App konnte vermerkt werden, in welche der vier Richtungen gemessen wurde. Zusätzlich zu dieser Eingabe wurde auch der Azimut-Wert des Orientierungssensors gespeichert, wie in 3.2.4 erläutert. Diese Informationen sind somit mit Daten eines WLAN-Scans verknüpft. Dies wurde zu Analyse des Ansatzes des WLAN-Fingerprintings unter Berücksichtigung der Richtung genutzt.

5.1.1 Erkennung der Richtung

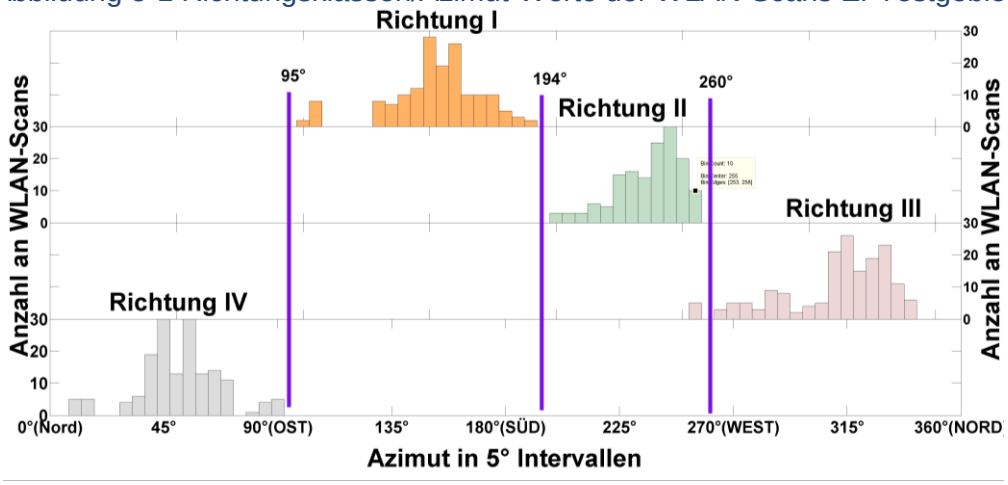
Die WLAN-Scans der Kalibrierungsphase wurden anhand der bei der Erstellung definierten *RichtungsID* klassifiziert. Nach dieser Klassifizierung wurden die Daten der WLAN-Scans in vier Kategorien eingeteilt. So ist es z. B. möglich, RSSI-Mittelwertvektoren für eine Position für eine bestimmte Richtungskategorie zu berechnen. Die Einteilung in diese vier Kategorien geschah bei den Tests durch die entsprechende Eingabe beim Erstellen der Scans. Betrachtet man die Abbildung 5-1, so ist zu erkennen, dass eine automatische Klassifizierung möglich wäre, da sich bestimmte Richtungskategorien in bestimmten Azimuts befinden.

In der Grafik ist die manuelle Klassifizierung der ermittelten Azimut-Werte der WLAN-Scans dargestellt. Die Richtungsklassen I bis IV entsprechen den definierten *RichtungsIDs*. Die Azimut-Werte einer jeden Klasse sind als Histogramm abgebildet. Damit wird der Zusammenhang zwischen der eingeben *RichtungsID* und den ermittelten Azimut-Werten deutlich. Die Azimut-Werte wurden in Intervalle von 5° eingeteilt. Die Balken stellen die Anzahl der Werte dar, die in diesem Bereich erfasst wurden. Obwohl die WLAN-Scans nur in vier definierten Richtungen erfasst wurden, entstehen aufgrund der Abweichungen des Orientierungssensors Azimut-Werte, die einer Normalverteilung gleichen.

Diese Darstellung zeigt, dass aus den Daten der Kalibrierungsphase mögliche Bereiche für die Einteilung zu einer Richtungskategorie entnommen werden könnten. In dem Beispiel wären dies die Bereiche 0°bis 95° für Richtung IV, 95° bis 194° für Richtung I, 194° bis 260° für Richtung II und 260° bis 360°für Richtung III. Die Abbildung 5-1 zeigt, dass eine automatische Klassifizierung der Richtungskategorie anhand der Daten fast ohne Fehler

möglich wäre. Lediglich im Azimut-Wert um 270° gibt es Datensätze, welche sich überschneiden.

Abbildung 5-1 Richtungsklassen/Azimut-Werte der WLAN-Scans EI-Testgebiet



Der Orientierungssensor (siehe 2.3.3) ermittelt den Azimut-Wert anhand des Magnetfeldsensors des Smartphones. Abweichungen können auftreten, wenn der Sensor nicht mehr kalibriert ist oder wenn elektromagnetische Felder den Sensor stören (siehe 6.3) Es muss auch die Deklination [84] zwischen der magnetischen und der geografischen Nordrichtung berücksichtigt werden. Auch Eingabefehler bei der Erstellung der WLAN-Scan-Datensätze sind nicht völlig ausgeschlossen.

5.1.2 Ergebnisse mit Berücksichtig der Richtung

Wie in der Einleitung dargelegt, wurde im Zuge der Arbeit analysiert, ob Fingerprinting zu höheren Erkennungsraten (ERn) führt, wenn die Orientierung des Handys berücksichtigt wird.

Die Analysen wurden im MATALB-Framework mithilfe der Structure Variable `WIFIDIRECTION` durchgeführt. Diese Structure enthält die Daten der Structure Variable `WIFIPOINTS`, jedoch nach den vier Richtungskategorien sortiert.

Bei der Positionierung mittels WLAN-Fingerprinting wurde die *RichtungsID* der Test-Datensätze benutzt, um in der entsprechenden Datenbank der gleichen Richtungskategorie nach der bestpassenden Position zu durchsuchen. Verwendet wurden die gleichen Methoden und Algorithmen, wie in **Kapitel 4** beschrieben wurden. Für diese Testreihe wurden also nur jene RSSI-Vektoren mit gleicher *RichtungsID*, für die Berechnung der euklidischen Abstände herangezogen.

In Tabelle 5-1 sind die Ergebnisse des WLAN-Fingerprints mit gemittelten RSSI-Vektoren und der Berücksichtigung der Richtungskategorie für das EI-Testgebiet angeführt. Im ersten Teil der Tabelle werden die erzielten Erkennungsraten dargestellt, wenn alle RSS-Werte verwendet werden. Der untere Teil der Tabelle zeigt die Ergebnisse, wenn die mehrfachen RSS-Werte eines physikalisch einmal vorhandenen APs selektiert wurden, wie in 4.5.2 erläutert.

Die Ergebnisse sind mit den Tabelle 4-1 und Tabelle 4-5 vergleichbar, da diese die Resultate der gleichen Testreihen zeigen, jedoch ohne Berücksichtigung der Richtung.

Tabelle 5-1 ERn EI-Testgebiet mit gemittelten RSSI-Vektoren in jeweilige Richtungskategorie

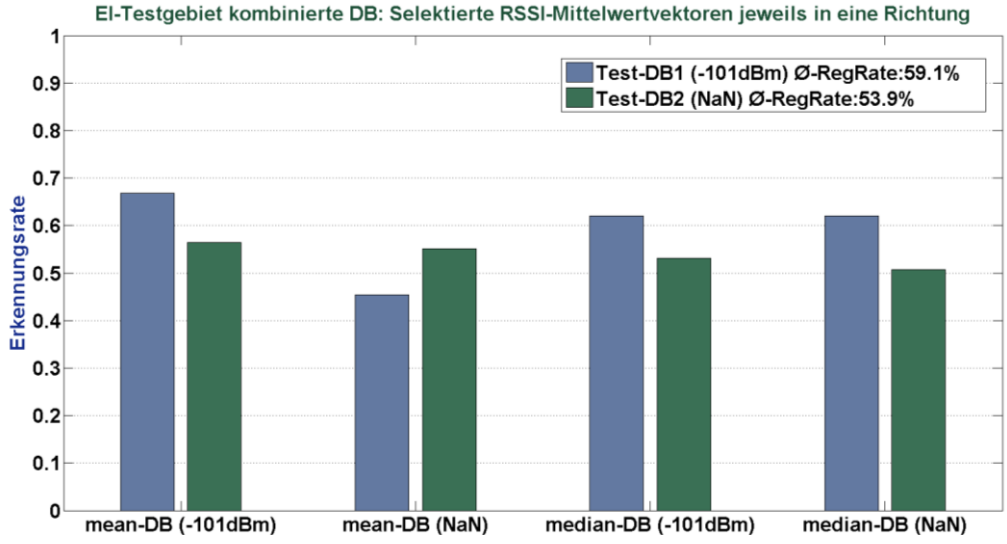
Erkennungsraten für alle untersuchten Positionen im *EI-Testgebiet*										
RSSI-Vektoren	Test-Szenarien	Mittelung jeweils in bestimmte Richtungen								
		Test-DB1(-101dBm)				Test-DB2(NaN)				mittlere ER. d. Sze.
		mean-DB		median-DB		mean-DB		median-DB		
		-101dbm	*NaN*	*-101dbm*	*NaN*	*-101dbm*	*NaN*	*-101dbm*	*NaN*	
multiple RSS-Werte	S2-DB	62,5%	40,8%	46,3%	46,3%	39,5%	***70,5%***	36,0%	69,8%	51,5%
	HTC-DB	52,6%	43,4%	46,4%	46,4%	49,6%	***66,4%***	49,1%	66,2%	52,5%
	kombinierte-DB	46,2%	27,8%	34,9%	34,9%	44,8%	***56,9%***	41,5%	55,8%	42,9%
	mittlere Erkennungsrate	**53,8%**	**37,3%**	**42,5%**	**42,5%**	**44,7%**	***64,6%***	**42,2%**	**63,9%**	**48,9%**
selektierte RSS-Werte	S2-DB	***65,2%***	44,6%	55,9%	55,9%	50,6%	58,7%	46,6%	57,4%	54,4%
	HTC-DB	***75,2%***	58,9%	65,7%	65,7%	63,4%	64,7%	48,4%	63,7%	63,2%
	kombinierte-DB	***66,8%***	45,4%	62,1%	62,1%	56,4%	55,2%	53,1%	50,8%	56,5%
	mittlere Erkennungsrate	***69,1%***	**49,6%**	**61,2%**	**61,2%**	**56,8%**	**59,5%**	**49,4%**	**57,3%**	**58,0%**

Betrachtet man die beste Erkennungsrate des Szenarios „*S2-DB", in welchem* alle RSS-Werte verwendet wurden, ist zu sehen, dass die beste ER mit Berücksichtigung der Richtung um 2 PP niedriger ist. Vergleicht man die mittleren Erkennungsraten dieser Szenarien, schneiden diese mit Berücksichtigung der Richtung um ca. 5 PP besser ab.

Betrachtet man die Szenarien „*HTC-DB*" der Tabellen miteinander, kann man sehen, dass die Berücksichtigung der Richtungskategorie und das Selektieren der mehrfachen RSS-Werte zu einer Erhöhung der besten ERn von 68,9 % auf 75,2 % geführt haben. Die mittlere Erkennungsrate der Szenarien hat sich um 1,6 PP (alle RSS-Werte) und 2,2 PP (selektierte RSS-Werte) erhöht, wenn die Richtung der WLAN-SCANS berücksichtig wurde.

Vergleicht man die Szenarien *„kombinierte-DB*" miteinander, ist zu sehen, dass die mittleren Erkennungsraten der Szenarien mit Berücksichtigung der Richtungskategorie um 6,5 PP (alle RSS-Werte) und 7,9 PP (selektierte RSS-Werte) höher sind. Außerdem sind die besten ERn um 1,4 PP und 3,0 PP höher, wenn die Richtungsinformation miteinbezog wird. Abbildung 5-2 zeigt die Erkennungsraten des Szenarios *„kombinierte-DB*" mit Berücksichtigung der Richtungskategorie. Sie ist mit Abbildung 4-9 vergleichbar, wo die RSSI-Mittelwert-Vektoren über alle Richtungen hinweg gemittelt wurden. Zu sehen ist, dass bei diesem Szenario durch die Berücksichtigung der Richtung die erzielten mittlere ERn mit Test-DB1 und Test-DB2, um 12,7 PP und 3 PP erhört werden konnte.

Abbildung 5-2 EI-Testgebiet Selektierte RSSI-Mittelwert-Vektoren je Richtungskategorie



Die höchsten ERn haben sich nicht geändert, jedoch sind die durchschnittlichen ERn gestiegen. Vor allem die Kombination von Test-DB1 und den Fingerprinting-DB mit NaN-Werten konnte so bessere ERn erzielen.

Ein möglicher Zusammengang besteht darin, dass die Erreichbarkeit von APs, beim Scannen in eine bestimmte Richtung, weniger fluktuiert. Schließlich werden WLAN-Scans immer in der gleichen Richtung durchgeführt und die Abschirmung durch den menschlichen Körper ist immer gleich. Bei gerade noch erreichbaren APs, kann dieses abschirmen mit dem Köper entscheidend sein, ob ein AP noch bei einem WLAN-Scan erfasst wird. Dieser Effekt ist auch in der Abbildung 3-3 zu sehen, dort sind nicht alle APs in allen Richtungen verfügbar.

Daher tritt die Situation, dass ein RSSI-Wert in der Fingerprinting-DB ist, aber nicht bei dem Test-Scans erfasst wird, viel weniger oft auf. Daher funktioniert auch die Kombination Test-DB1 mit den NaN-Fingerprinting-DB besser.

Tabelle 5-2 zeigt die Ergebnisse mit Berücksichtigung der Richtung für das Heim-Testgebiet und ist zu vergleichen mit Tabelle 4-2. Diese zeigt die Varianten und Szenarien mit den gemittelten RSSI-Vektoren ohne Beachtung der Orientierung. Im Durschnitt sind jene ERn mit Berücksichtigung der Richtung um 4,9 PP höher.

Die besten ERn der Szenarien sind gleich oder um maximal 0,6 PP schlechter, wenn nur die RSSI-Vektoren mit der gleichen Richtungskategorie für das WLAN-Fingerprinting verwendet werden. Das bedeutet, dass durch die Berücksichtigung der Richtung nur einzelne RSSI-Vektoren nicht der korrekten Position zugewiesen werden konnten. Grundsätzlich sind die Erkennungsraten beim Heim-Testgebiet bereits hoch, weitere Verbesserungen der ERn sind daher nicht mehr leicht möglich. Denkbar ist natürlich, dass einige wenige RSSI-Vektoren beim Erfassen der WLAN-Scans mit der falschen *RichtungsID* versehen wurden.

Tabelle 5-2 ERn Heim-Testgebiet mit gemittelten RSSI-Vektoren in jeweilige Richtungskategorie

Erkennungsrate für alle untersuchten Positionen im *Heim-Testgebiet*									
RSSI-Vektoren mit allen RSS-Werten	**Test-Szenarien**	**Mittelung jeweils in bestimmte Richtungen**							
		Test-DB1(-101dBm)				**Test-DB2(NaN)**				**mittlere ER. d. Sze.**
		mean DB		**median DB**		**mean DB**		**median DB**		
		-101dbm	*NaN*	*-101dbm*	*NaN*	*-101dbm*	*NaN*	*-101dbm*	*NaN*	
	S2 DB	92,7%	86,7%	90,9%	90,9%	88,1%	93,4%	84,6%	***93,7%***	90,1%
	HTC SB	***96,1%***	92,6%	95,4%	95,4%	92,9%	94,3%	92,9%	94,3%	94,2%
	kombinierte DB	***95,2%***	79,6%	94,5%	94,5%	91,2%	90,5%	89,4%	90,0%	90,6%
	mittlere Erkennungsrate	***94,7%***	**86,3%**	**93,6%**	**93,6%**	**90,7%**	**92,7%**	**89,0%**	**92,7%**	**91,7%**

Aus den Ergebnissen ist zu sehen, dass bei den besten ERn nicht immer oder nur geringe Verbesserungen erreicht werden konnten. Aufgrund zu ähnlicher RSSI-Vektoren verschiedener Positionen oder zu großen Schwankungen der RSS-Werte der Testdaten sind dem WLAN-Fingerprinting Grenzen gesetzt. Sind die RSS-Werte zu verfälscht oder sind die RSSI-Vektoren zweier Positionen zu ähnlich, ist eine korrekte Zuweisung mittels des NN-Algorithmus nicht mehr möglich.

Die mittleren Erkennungsraten der Szenarien konnten durchgehend verbessert werden. Grund dafür ist, dass jene Kombinationen von Datenbanken, die geringe ERn ergaben, durch die Miteinbeziehung der Richtungskategorie zu besseren Ergebnissen führen.

Zusammenfassend kann gesagt werden, dass die Berücksichtigung der Orientierung tendenziell zu besseren Ergebnissen führt, dass aber bei den Varianten, wo die Grenzen der

erreichbaren ERn mit dem NN-Algorithmus bereits erzielt wurden, keine oder nur eine geringfügige Verbesserung möglich war.

Außerdem zeigt sich, dass sich die Selektion von mehrfachen RSS-Werten eines physikalisch einmal vorhandenen AP bewährt. Dies ist beim Fingerprinting-Algorithmus auch so vorgesehen, ansonsten werden bestimmte RSS-Werte mehrfach gezählt, was zu einer zufälligen Gewichtung von bestimmten APs führt.

Tabelle 5-3 und Tabelle 5-4 zeigen die Ergebnisse des MFV-Algorithmus und Likelihood-Algorithmus für das EI-Testgebiet und Heim-Testgebiet mit Rücksichtnahme auf die Orientierung. Diese Tabellen sind mit Tabelle 4-6 und Tabelle 4-7 vergleichbar, welche die gleichen Testreihen widerspiegeln, allerdings ohne Einbeziehung der Richtungsinformation.

Tabelle 5-3 ERn EI-Testgebiet Betrachtung aller RSSI-Vektoren in jeweilige Richtungskategorie

Erkennungsrate mit alle RSSI-Vektoren in Datenbank im *EI-Testgebiet*						
RSSI-Vektoren	**Test-Szenarien**	**RSSI-Vektoren jeweils in bestimmte Richtung**				
		MFV-Algorithmus		**Likelihood-Algorithmus**		**mittlere ER. Sze.**
		DB1(-101dBm)	**DB2(NaN)**	**DB1(-101dBm)**	**DB2(NaN)**	
multiple RSS-Werte	**S2 DB**	55,2%	52,1%	***55,2%***	52,1%	53,7%
	HTC SB	48,4%	52,6%	49,6%	53,1%	50,9%
	kombinierte DB	52,1%	51,3%	49,9%	51,3%	51,2%
	mittlere Erkennungsrate	51,9%	***52,0%***	51,6%	***52,2%***	51,9%
Reduzierte RSS-Werte	**S2-DB**	**62,7%**	57,4%	**62,7%**	58,7%	60,4%
	HTC-SB	73,4%	64,2%	72,7%	63,4%	68,4%
	kombinierte-DB	66,6%	55,2%	66,8%	55,2%	61,0%
	mittlere Erkennungsrate	***67,6%***	58,9%	***67,4%***	59,1%	63,3%

Beim Vergleich der Erkennungsraten zwischen den Tabellen, mit und ohne Berücksichtigung der Richtungskategorie, sind kaum Unterschiede feststellbar. Die Erkennungsraten weichen unter einem Prozentpunkt voneinander ab. Der Grund liegt darin, dass die Richtung bei den Algorithmen bereits indirekt Einfluss nimmt. Denn dadurch, dass alle erfassten RSSI-Vektoren verwendet werden und diese nach den euklidischen Abständen zum Test-Vektor sortierten werden, werden je RSSI-Vektoren, die in der gleich Richtung gemessen wurden, etwas kleinere Abstände ergeben. Somit werden diese RSSI-Vektoren mit gleicher Richtungskategorie eher am Anfang der Sortierung zu finden sein.

Tabelle 5-4 ERn Heim-Testgebiet Betrachtung aller RSSI-Vektoren in jeweilige Richtungskategorie

Erkennungsrate mit alle RSSI-Vektoren in Datenbank im *Heim-Testgebiet*						
RSSI-Vektoren mit allen RSS-Werten	**Test-Szenarien**	**RSSI-Vektoren jeweils in bestimmte Richtung**				
		MFV-Algorithmus		**Likelihood-Algorithmus**		**mittlere ER. Sze.**
		DB1(-101dBm)	**DB2(NaN)**	**DB1(-101dBm)**	**DB2(NaN)**	
	S2-DB	91,6%	90,6%	***92,0%***	90,9%	91,3%
	HTC-SB	***96,5%***	94,0%	***96,5%***	94,3%	95,3%
	kombinierte DB	***94,7%***	91,7%	***94,7%***	91,9%	93,3%
	mittlere Erkennungsrate	94,3%	92,1%	***94,4%***	92,4%	93,3%

Auch wenn bei diesen Algorithmen die ERn durch die Berücksichtigung der Richtungskategorie nicht verbessert werden konnten, hat die Miteinbeziehung dieser Information einen Vorteil. Denn wenn für die Berechnung nur jene RSSI-Vektoren der gleichen Richtung herangezogen werden, kann die Anzahl der durchzuführenden

euklidischen Abstandsberechnungen um ¾ verringert werden, ohne geringere ERn zu erhalten.

5.2 Auswahl der iCPs für Indoor-Navigation EI-Testgebiet

Bei den traditionellen WLAN-Fingerprinting Ansätzen werden Positionen rasterartig definiert und an diesen werden die RSS-Werte ermittelt. In [71] wurde für Teile des EI-Testgebiets (3. Stock, Trakt B) ein Raster von Positionen definiert, an denen RSS-Werte gemessen wurden.

In dieser Arbeit wurden an 73 Positionen die RSS-Werte der APs gemessen. Für größere Räume wurden bis zu fünf Positionen definiert. Für den Ansatz mit iCPs sind weniger Messungen nötig, da nur an bestimmten Positionen, die eine Wegentscheidung darstellen, die RSS-Werte erfasst werden müssen. Für die Tests in der Arbeit wurden zwar nur vier Räume bzw. deren Eingänge verwendet, aber auch wenn für jede Tür zu jedem Raum eine Position definiert worden wäre, sind dies in etwa halb so viele Positionen, die im Gegensatz zur Rasterhaftenanordnung vermessen werden müssten.

Durch die Kombination des WLAN-Fingerprintings mit iCPs und einem INS-Algorithmus, sollte es möglich sein, alle Positionen des Gebäudes abzudecken. Je nachdem, wie zuverlässig die Algorithmen gestaltet werden, könnte auch auf bestimmte iCPs, wie z. B. zu den Eingängen der Räume, teilweise verzichtet werden; z. B. könnte mithilfe eines iCP der gewählte Trakt und mit einem zuverlässigen INS-Algorithmus die Positionen bis zum gewählten Raum erkannt werden.

Für diese Arbeit wurden 17 iCPs aus den untersuchten Positionen in Kapitel 4 gewählt. Diese iCPs decken die Wegentscheidungen vom Eingangsbereich bis zu den vier gewählten Räumen ab. Die iCPs sind in Abbildung 5-3, Abbildung 5-4 und Abbildung 5-5 dargestellt. In Tabelle 5-5 sind die Erkennungsraten der einzelnen iCPs angeführt. Diese ERn wurden im Szenario „*kombinierte-DB*“ erreicht, wenn nur die iCPs mit selektierten RSS-Werten und die Kombination Test-DB1/meanDB(-101dBm), sowie der ermittelte Gewichtungsvektor für das WLAN-Fingerprinting herangezogen wurden.

Abbildung 5-3 Eingangsbereich

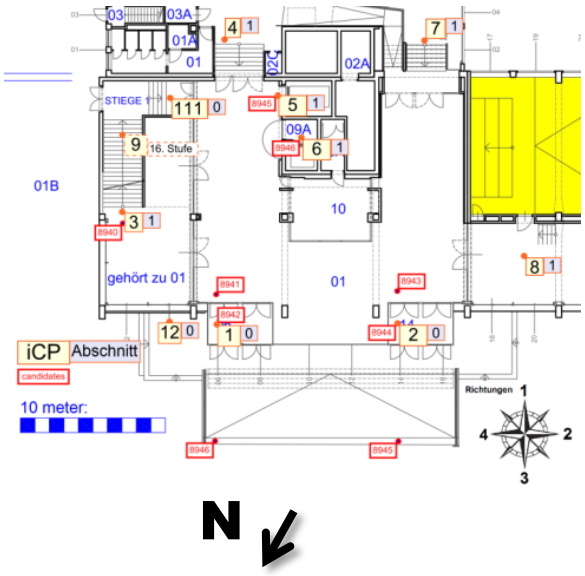


Abbildung 5-4 Stiegenhaus

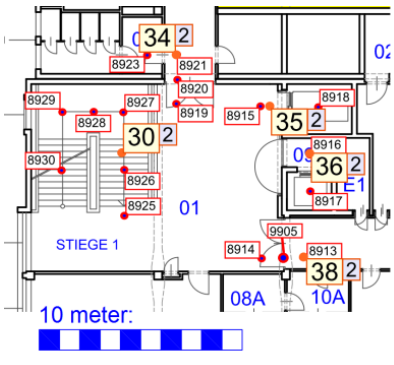


Abbildung 5-5 Gang B

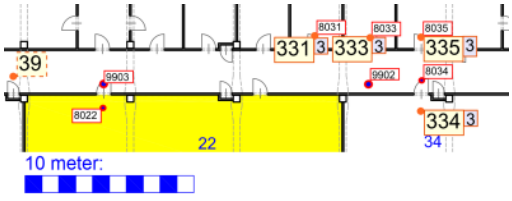


Tabelle 5-5 Details beste ER alle iCPs des Szenario „*kombinierte-DB*“

ER – beste Variante									
Abschn.	*PointID*	*ER*	*korrekt/gesamt*	*falsche **IDs***	*Abschn.*	*PointID*	*ER*	*korrekt/gesamt*	*falsche **IDs***
0	2	93,3%	28/30	2x**1**;	2	35	75,9%	41/54	13x**36**
0	1	97,6%	41/42	1x**3**	2	36	86,5%	141/163	22x35;
0	111	100,0%	25/25	-	2	34	97,1%	67/69	2x**35**
0	12	100,0%	29/29	-	2	38	100,0%	57/57	-
1	5	81,5%	22/27	2x**3**;3x**6**	2	30	100,0%	16/16	-
1	3	75,0%	27/36	9x**5**	3	335	13,3%	2/15	13x**333**
1	4	88,9%	24/27	24x**4**	3	333	57,1%	4/7	1x**331**; 2x**335**
1	6	91,7%	22/24	22x6	3	331	81,5%	22/27	4x**333**;1x**334**
					3	334	100,0%	24/24	-

Test-DB1/meanDB(-101dBm) für die ganze iCP-DB für EI-Testgebiet mit Gewichtung

Für das in Tabelle 5-5 dargestellte Szenario liegt die ERn aller iCPs im Mittel bei 88,1 %. Ohne Gewichtung wäre diese bei 87,1 %, diese ERn sind der Tabelle 5-6 und der Tabelle 5-8 zu entnehmen. Zum Vergleich: die beste ER für die kombinierte DB für alle Positionen lag bei 66,8%; diese ist in der Tabelle 5-1 zu sehen.

Bei den Tests in [71], in welchen eine gerasterte Anordnung von Positionen gewählt wurde, konnten die korrekten Räume mit einer ER von 23,4 % erkannt werden. Die Tests fanden im selben Testgebiet statt und es wurden die gleichen Geräte benutzt.

Die Tabelle 5-5 ist nach den *Abschnitten* (Abschn.) und nach ER sortiert. Der Spalte „*korrekt/gesamt*“ sind die absoluten Zahlen der korrekt zugewiesen bzw. getesteten RSSI-Vektoren zu entnehmen. In der Spalte „*falsche IDs*“ sind jene iCP angeführt, zu deren Positionen ein RSSI-Vektor falsch zugewiesen wurde.

Dem *Abschnitt-0* gehören die Eingänge an; diese iCPs haben hohe Erkennungsraten. Zwei RSSI-Vektoren, die zum iCP-2 gehören, werden falsch dem iCP-1 zugewiesen. Ein Vektor des iCP-1 wird einmal mit dem iCP-3 in *Abschnitt*-1 verwechselt; mit iCP-3 soll später die Wegentscheidung erkannt werden, die Stiege zu benutzen.

In *Abschnitt*-1 hat der iCP-3 die geringste ER. Der iCP wird neunmal mit dem iCP-5, der Position vor dem Lift-2, verwechselt. Die zwei iCPs sind nur durch eine teilweise offene Glaswand abgetrennt. Eine Tabelle, welche beschreibt, wie sich bestimmte Materialien auf die RSS-Werte auswirken, ist in [39 S.14] zu finden.

Im 3. Stock in *Abschnitt*-2 werden die iCP-35 und iCP-36 am häufigsten verwechselt. Die meisten Verwechslungen treten bei den iCPs 331, 333 und 335 in *Abschnitt*-3 auf. Diese befinden sich hinter den Türen zu den vier benachbarten Testräumen. Bei den in Tabelle 5-5 aufgelisteten ER werden noch alle iCP gemeinsam getestet. Im Anschluss wird eine sequenzielle Abarbeitung der Abschnitte simuliert. Die Verbesserungen sind im Detail in Tabelle 5-9 angeführt.

5.3 Sequenzieller Ablauf der Sektoren

Um einen sequenziellen Ablauf des WLAN-Fingerprinting zu simulieren, wurden vier Abschnitte definiert und die Testdaten dementsprechend aufgeteilt analysiert. Wie in **Kapitel 4** wurden die in 4.5 betrachteten Varianten für jeden Abschnitt separat getestet. Die Abschnitte sind dort definiert, wo Wegentscheidungen von den Eingängen zu den vier Testräumen getroffen werden müssen.

Abschnitt-0 beinhaltet die Eingänge, die zum Testgebiet gehören. Ob der Lift, die Treppe oder ein Gang benutzt werden, soll in Abschnitt-1 erkannt werden. Die iCPs des Abschnitt-0 und Abschnitt-1 sind in Abbildung 5-3 dargestellt. Im Stiegenhaus im 3. Stock des Instituts befindet sich der Abschnitt-2. Dieser Abschnitt funktioniert ähnlich wie eine Kreuzung; an einem Punkt wird der Abschnitt betreten, an einem anderen wieder verlassen. Die iCPs dieses Abschnittes sind in Abbildung 5-4 dargestellt. Der letzte Abschnitt-3 umfasst die Eingänge zu den Testräumen. Die Positionen der iCPs sind in Abbildung 5-5 zu sehen.

In Tabelle 5-6 sind die Erkennungsraten erfasst, die beim sequenziellen Fingerprinting mit den gemittelten RSSI-Vektoren und mit Berücksichtigung der Richtungskategorie für die *kombinierte-DB* erzielt wurden. Die Tabelle 5-7 zeigt die Ergebnisse, die mit dem MFV- und Likelihood-Algorithmus erzielt wurden. In der Tabelle sind die ERn mit und ohne Berücksichtigung der Richtung gegenübergestellt.

Das erste Szenario „*ganze iCP-DB*“ verwendet alle iCPs ohne Aufteilung der Daten in Abschnitte. In den weiteren Szenarien sind die kombinierten RSSI-Vektoren beider Handys in Abschnitte aufgeteilt und analysiert worden.

Die mittleren Erkennungsraten beschreiben das arithmetische Mittel aus den Erkennungsraten der Abschnitte. Diese Erkennungsraten können nicht direkt mit den ERn des Szenarios „*ganze iCP-DB*“ verglichen werden. Dort wurden die ERn nach der Formel *(F. 4-30)* berechnet. Laut der Formel wird zur Berechnung der ER die Anzahl der untersuchten RSSI-Vektoren durch die Zahl der richtig zugeordneten Vektoren dividiert. Bei den mittleren Erkennungsraten wird das Mittel aus den ERn der einzelnen Abschnitte berechnet. Da die Anzahl der untersuchten RSSI-Vektoren nicht in allen Anschnitten gleich ist, ergeben sich auch andere Werte für die mittleren Erkennungsraten als für das Szenario „*ganze iCP-DB*“.

Tabelle 5-6 EI-Testgebiet sequenzielles Fingerprinting in jeweilige Richtungskategorie

Erkennungsrate für alle untersuchten iCP im *EI-Testgebiet*										
RSSI-Vektoren...	**Test-Szenarien mit kombinierte-DB**	**Mittelung jeweils in bestimmte Richtungen**								**mittlere ER. Szenario.**
		Test-DB1(-101dBm)				**Test-DB2(NaN)**				
		mean-DB		**median-DB**		**mean-DB**		**median-DB**		
		-101dbm	*NaN*	*-101dbm*	*NaN*	*-101dbm*	*NaN*	*-101dbm*	*NaN*	
...multiple RSS-Werte	**ganze iCP-DB**	***74,3%***	25,1%	60,1%	60,1%	62,2%	70,8%	54,9%	70,7%	59,8%
	Sektion0-DB	***92,9%***	69,8%	67,5%	67,5%	76,2%	89,7%	57,1%	86,5%	75,9%
	Sektion1-DB	58,8%	32,5%	48,2%	48,2%	56,1%	***69,3%***	53,5%	65,8%	54,1%
	Sektion 2-DB	***82,7%***	57,9%	73,3%	73,3%	64,3%	74,9%	60,4%	75,2%	70,3%
	Sektion 3-DB	46,6%	42,5%	34,2%	34,2%	46,6%	***64,4%***	41,1%	***64,4%***	46,8%
	mittlere Erkennungsrate	***70,2%***	**50,7%**	**55,8%**	**55,8%**	**60,8%**	***74,6%***	**53,0%**	**73,0%**	**61,7%**
mit selektierte RSS-Werte	**ganze iCP-DB**	***87,1%***	58,5%	83,6%	83,6%	75,7%	68,5%	69,8%	66,1%	74,1%
	Sektion0-DB	***97,6%***	85,7%	96,0%	96,0%	83,3%	83,3%	61,9%	81,7%	85,7%
	Sektion1-DB	***84,2%***	63,2%	78,9%	78,9%	70,2%	64,0%	58,8%	52,6%	68,9%
	Sektion 2-DB	88,0%	74,1%	89,4%	***89,4%***	78,8%	71,6%	79,7%	69,6%	80,1%
	Sektion 3-DB	72,6%	***75,3%***	74,0%	74,0%	69,9%	69,9%	60,3%	63,0%	69,9%
	mittlere Erkennungsrate	***85,6%***	**74,6%**	**84,6%**	**84,6%**	**75,6%**	**72,2%**	**65,2%**	**66,8%**	76,2%

Tabelle 5-6 stellt ebenfalls die erzielten ERn gegenüber, wenn die multiplen oder die selektierten RSS-Werte der APs verwendet wurden. Bei diesen Ergebnissen, für welche die kombinierte Datenbank der zwei Handys für die Analyse herangezogen wurde, konnte durch die Selektion der RSS-Werten deutlich bessere ERn erzielt werden. Vergleicht man die

mittleren ERn, so konnten diese durch die Selektion der RSS-Werte, um 14,5 PP erhört werden. Im Schnitt konnten auch die besten ERn verbessert werden, und zwar um 11 PP.

Stellt man die beste ER des Szenarios „*ganze iCP-DB*“ (87,1 %) den ERn der einzelnen Abschnitte gegenüber, so ist zu sehen, dass diese nur in Abschnitt-1 und Abschnitt-3 geringer ist, wobei in Abschnitt-1 dieser Unterschied 2,9 PP beträgt. Die höheren ERn ergeben sich daraus, dass bestimmte Positionen aufgrund der Einteilung in Sektoren nicht mehr verwechselt werden können, wie z. B. der iCP-1 und iCP-3, wie in Tabelle 5-5 zu sehen war.

Eine weitere Idee der Einteilung in Abschnitte ist, dass bei diesen das WLAN-Fingerprinting pro Anschnitt optimiert werden kann. Dazu gehört auch, die optimale Variante des WLAN-Fingerprintings zu benutzen. In 5.3.1 werden außerdem die Ergebnisse gezeigt, wenn für jeden Abschnitt ein eigener Gewichtsvektor verwendet wurde.

Tabelle 5-7 stellt die Ergebnisse des MFV- und Likelihood-Algorithmus dar. Gegenübergestellt werden die ERn mit allen und den selektierten RSS-Werten sowie mit und ohne Verwendung der Richtungskategorie. Vergleicht man die Durchschnittswerte der mittleren ERn der selektieren RSS-Werte mit denen, in denen alle RSS-Werte benutzt wurden, sind auch bei diesen Algorithmen deutliche Unterschiede feststellbar. Werden die selektierten RSS-Werte für das WLAN-Fingerprinting benutzt, konnten die ERn um 7,8 PP ohne und 7,1 PP mit Verwendung der Richtungskategorie im Mittel verbessert werden. Im Mittel unterscheiden sich die besten ERn um 14 PP und 14,3 PP ohne bzw. mit Berücksichtigung der Richtungskategorie, wenn die selektierten RSS-Werte benutzt werden.

Zwischen dem MFV- und Likelihood-Algorithmus gibt es keine signifikanten Unterschiede. Mit dem MFV-Alog. werden im Mittel etwas bessere Ergebnisse erzielt, diese sind jedoch unter 1 PP.

Vergleicht man die Ergebnisse von Tabelle 5-7 und Tabelle 5-6, so ist zu sehen, dass in den Abschnitten 2 und 3 mit dem MFV- Algorithmus 4,9 PP und 8,3 PP höhere ERn erzielt werden konnten, als wenn die gemittelten RSSI-Vektoren benutzt wurden. Für diese Abschnitte konnten auch bei der Berücksichtigung der Richtungskategorie um 0,2 PP und 2,8 PP bessere ERn erzielt werden.

Tabelle 5-7 EI-Testgebiet sequenzielles Fingerprinting mit allen erfassten RSSI-Vektoren

Erkennungsraten für alle untersuchten iCPs im *EI-Testgebiet*												
RSSI-Vektoren...	Test-Szenarien kombinierte-DB	RSSI-Vektoren in alle Richtungen				mittlere ER. Sze.	RSSI-Vektoren in bestimmte Richtung				mittlere ER. Sze.
		MFV-Algo.		Likelihood-Algo.			MFV-Algo.		Likelihood-Algo.		
		DB1	DB2	DB1	DB2		DB1	DB2	DB1	DB2	
...multiple RSS-Werte	ganze iCP-DB	74,9%	***76,2%***	74,9%	***76,2%***	75,52%	74,0%	***76,9%***	74,0%	***76,9%***	75,45%
	Sektion0-DB	74,6%	***90,5%***	74,6%	84,1%	80,95%	74,6%	***88,1%***	74,6%	84,1%	80,36%
	Sektion1-DB	***63,2%***	57,9%	62,3%	57,9%	60,31%	***63,2%***	62,3%	***63,2%***	64,9%	63,38%
	Sektion 2-DB	89,4%	***90,3%***	89,4%	***90,3%***	89,83%	86,4%	***89,7%***	86,4%	***89,7%***	88,02%
	Sektion 3-DB	***64,4%***	57,5%	63,0%	56,2%	60,27%	63,0%	53,4%	***64,4%***	49,3%	57,53%
	mittlere Erkennungsrate	**72,89%**	***74,04%***	**72,33%**	**72,11%**	72,84%	**71,8%**	***73,4%***	**72,1%**	**72,0%**	72,32%
mit selektierte RSS-Werte	ganze iCP-DB	***86,2%***	74,6%	85,9%	74,6%	80,28%	***85,7%***	76,5%	***85,7%***	76,5%	81,10%
	Sektion0-DB	***94,4%***	81,7%	***94,4%***	77,8%	87,10%	***92,1%***	79,4%	***92,1%***	75,4%	84,72%
	Sektion1-DB	***82,5%***	74,6%	79,8%	69,3%	76,54%	***80,7%***	62,3%	72,8%	59,6%	68,86%
	Sektion 2-DB	***94,2%***	85,8%	***94,2%***	85,8%	89,97%	***94,4%***	88,3%	***94,4%***	88,3%	91,36%
	Sektion 3-DB	**80,8%**	58,9%	***80,8%***	54,8%	68,84%	***83,6%***	61,6%	82,2%	63,0%	72,60%
	mittlere Erkennungsrate	***88,0%***	**75,3%**	**87,3%**	**71,9%**	80,61%	***87,7%***	**72,9%**	**85,4%**	**71,6%**	79,39%

Abschließend kann gesagt werden, dass die Selektion der mehrfachen RSS-Werte für physikalisch einmal vorhande APs mit den kombinierten Datensätzen zu höheren ERn geführt hat. Diese multiplen RSS-Werte entsprechen einer zufälligen Gewichtung der APs, die mehrere Netzwerke bedienen. Diese Gewichtung kann die Unterscheidung von Positionen mit ähnlichen RSS-Werten erschweren.

Mit dem MFV-Algorithmus wurden etwas bessere ERn bei jenen Positionen erreicht, die nahe beisammen liegen, wie in Abschnitt-3. Dies lässt den Schluss zu, dass Positionen mit sehr ähnlichen RSSI-Vektoren mit dem MFV-Algorithmus etwas besser unterschieden werden können. Dadurch, dass die RSSI-Werte nicht gemittelt werden, können kleinere Unterschiede besser berücksichtigt werden.

Die RSS-Werte der APs sind ebenfalls abhängig von einer Reihe von veränderlichen Umgebungsbedingungen [61] Dadurch sind die RSS-Werte auch abhängig von der Tageszeit bzw. Nutzungs-Situation. Da das EI-Testgebiet Teil des UNI-Geländes ist, ist die Nutzungs-Situation auch abhängig von den Unterrichtszeiten.

Auch wenn dies zu unterschiedlichen RSSI-Vektoren an einer Positionen führt, werden sich diese je nach Situation wiederholen. Mit dem MVF- bzw. Likelihood-Algorithmus kann dies berücksichtigt werden, indem zu verschieden Zeiten bzw. in verschieden Situationen die RSSI-Vektoren ermittelt werden.

Für die in Kapitel 4.5 untersuchten 49 Positionen wurden 5726 RSSI-Vektoren erfasst. Mit dem MVF-Algorithmus musste daher 5726-mal der euklidische Abstand zu diesen n-dimensionalen Vektoren berechnet werden. Hingegen wird bei der Ermittlung der Mittelwertvektoren jeder Position ein RSSI-Mittelwert-Vektor zugewiesen. Daher müssen nur 49 Abstandsberechnungen durchgeführt werden.

Eventuell würde sich anbieten, eine beschränkte Anzahl von RSSI-Vektoren für eine Position, die diese Unterschiede widerspiegeln, in der Datenbank zu speichern, um so für die Berechnung nicht alle ermittelten RSSI-Vektoren verwenden zu müssen. So könnten die Konzepte auch kombiniert werden, indem mehrere gemittelte RSSI-Vektoren, die für verschiedene Nutzungs-Situationen stehen, in der Datenbank gespeichert werden.

Für die 17 iCPs wurden 1514 RSSI-Vektoren ausgewählt. Durch Unterteilung der iCPs in Abschnitte muss ein zu testender RSSI-Vektor nur mit den RSSI-Vektoren der iCPs des aktuellen Abschnitts verglichen werden. Betrachtet man beispielsweise den Abschnitt-0, müssten 314 RSSI-Vektoren betrachtet werden. Wird die Richtungskategorie berücksichtigt, z. B. die RichtungsIDs 1, sind noch 79 RSSI-Vektoren für die Berechnung heranzuziehen. Der Nachteil des Berechnungsaufwandes des MVF- bzw. Likelihood-Algorithmus kann auf diese Weise aufgehoben werden.

Die etwas höheren ERn mit Berücksichtigung der Richtung in Sektor-3 erklären sich vermutlich dadurch, dass die Orientierung des Handys dort einen größeren Einfluss hat: entweder schaut man in den Raum oder in den Gang, wo jeweils verschiedene APs montiert sind. Die Miteinbeziehung der Richtungskategorie reduziert nicht nur die RSSI-Vektoren, sondern kann in derartigen Situationen zu besseren Ergebnissen führen.

5.3.1 Gewichtung pro Abschnitt

In 4.4.2 wurde bereits erläutert, wie die Gewichtung der Summanden bei der euklidischen Distanzberechnung benutzt werden kann, um bestimmte APs verschieden stark zu gewichten. Für diese Gewichtung wurde ein Gewichtungsvektor ermittelt. Wie nach diesem Gewichtungsvektor gesucht wurde, ist in 4.5.4 beschrieben.

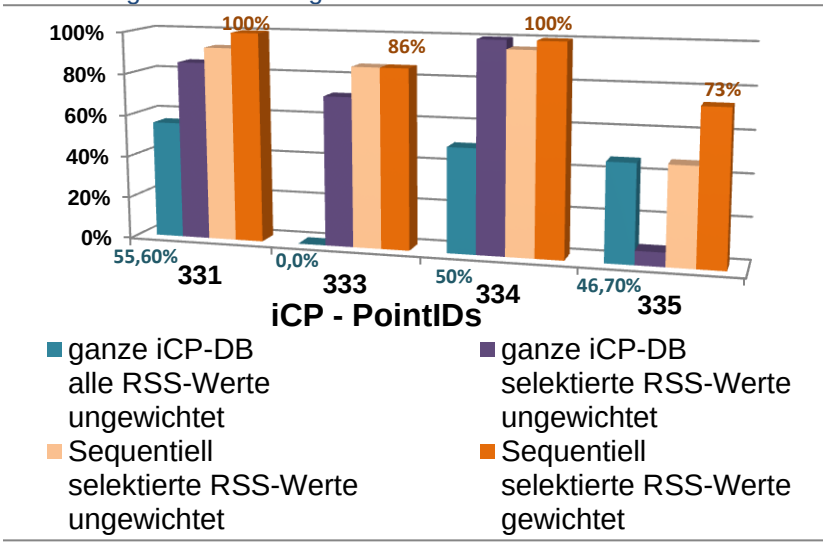
In Tabelle 5-8 sind in der ersten Wertespalte die erzielten besten ERn pro Abschnitt angeführt. Die durch einen Gewichtungsvektor erzielten Verbesserungen der ERn je Abschnitt sind in der zweiten Wertespalte angegeben. Angeführt sind die ERn, wenn alle RSSI-Werte und die selektierten RSSI-Werte benutzt werden. Im Durchschnitt konnten die mittleren ERn um 3,2 PP bzw. um 2,5 PP durch die Gewichtung verbessert werden. Die größte Verbesserung der besten ER konnte in Abschnitt-3 mit 9,6 PP erzielt werden.

In Abbildung 5-6 sind Schritt für Schritt die Verbesserungen der ERn der einzelnen iCPs des Abschnitts-3 mit den bisher diskutierten Methoden dargestellt. Die türkisen Balken zeigen die ERn für die Variante, wenn Test-DB1 und mean-DB(-101dBm) für die Analyse herangezogen werden. Dort werden alle iCPs und alle ermittelten RSS-Werte ohne Gewichtung verwendet. Die violetten Balken stellen die Ergebnisse dieser Variante mit den selektierten RSS-Werten dar. Wie sich die ERn dieser Variante verändern, wenn das Fingerprinting abschnittsweise analysiert wurde, ist in den hellorangen Balken abzulesen.

Tabelle 5-8 ERn je Abschnitt mit Gewichtung

RSSI-Werte	Test-Szenarien	*Beste ER*	*beste Methode AP gewichtet*	*Verb. Abs.*
Vollständigen Verwendung	**ganze iCP-DB**	76,9 %	76,9 %	0,0
	Abschnitt 0-DB	92,9 %	95,2 %	2,3%
	Abschnitt 1-DB	69,3 %	72,8 %	3,5%
	Abschnitt 2-DB	90,3 %	90,3 %	0,0%
	Abschnitt 3-DB	64,4 %	71,2 %	6,8%
	mittlere Erkennungsrate	**79,2 %**	**82,4 %**	3,2%
Selektiert	**ganze iCP-DB**	87,1 %	88,1 %	1,0%
	Abschnitt 0-DB	99,2 %	99,2 %	0,0%
	Abschnitt 1-DB	84,2 %	84,2 %	0,0%
	Abschnitt 2-DB	94,4 %	95,0 %	0,6%
	Abschnitt 3-DB	83,6 %	93,2 %	9,6%
	mittlere Erkennungsrate	**90,4 %**	**92,9 %**	2,5%

Abbildung 5-6 Erkennungsarten der iCPs des Abschnitt 3



Die dunkelorangen Balken zeigen die ERn, wenn diese abschnittsweise analysiert und mittels des Gewichtungsvektors erweitert wurden. Die Abbildung zeigt, wie durch die beschriebenen Maßnahmen die durchschnittliche ER der iCPs des Abschnittes von 38,1 % auf 89,8 % verbessert werden konnte.

In Tabelle 5-9 sind die ERn der einzelnen iCPs dargestellt, wenn diese in Abschnitte aufgeteilt und jeweils ein eigener Gewichtungsvektor verwendet wurde. Aus den 725 von 786 korrekt zugeordneten Test-Vektoren ergibt sich die ER von 92,2%. Sie liegt also um 4,1 PP höher, als wenn die Berechnungen mit iCPs und nur einem Gewichtungsvektor durchgeführt wurden.

Tabelle 5-9 ERn der einzelnen iCP mit beste erzielter gesamt ER

ER – beste Variante pro Sektor gewichtet									
Abschn.	PointID	ER	korrekt/Gesamt	verwechselte **Ids**	Abschn.	PointID	ER	korrekt/Gesamt	verwechselte **Ids**
0	2	93%	28/30	2x**1**;	2	35	87%	47/54	2x**34**;5x**36**
0	1	100%	42/42	-	2	36	94,50%	154/163	8x**35**;1x**38**;
0	11	100%	25/25	-	2	34	97,10%	67/69	2x**35**
0	12	100%	29/29	-	2	38	100%	57/57	-
1	3	75%	54/72	18x**5**	2	30	100%	16/16	-
1	5	85,20%	46/54	2x**3**;6x**6**	3	335	73%	11/15	4x**333**
1	4	88,90%	48/54	6x**3**	3	333	86%	06/07	1x**335**
1	6	91,70%	44/48	4x**5**	3	331	100%	27/27	-
					3	334	100%	24/24	-
gesamte ER		**92,2%**	**725/786**	-					

Test-DB1/meanDB(-101dBm) für die ganze iCP-DB für EI-Testgebiet mit Gewichtung

Durch die Aufteilung in Abschnitte wird zudem das Optimierungsproblem bei der Gewichtung kleiner. Denn dadurch, dass nur jeweils die APs eines Abschnittes betrachtet werden, reduzieren sich der Dimensionen der Gewichtungsvektoren auf die Anzahl der erreichbaren APs der Abschnitte. Die Optimierung muss außerdem nur auf wenige Positionen angewandt werden.

5.4 Zusammenfassung

In diesem Kapitel wurde das Konzept der iCPs weiter untersucht. Es wurde analysiert, mit welchen Erkennungsraten (ERn) die gewählten iCPs erkannt werden. Durch die Auswahl bestimmter wegentscheidender Positionen, die durch baulich vorhandene Strukturen mit WLAN-Fingerprinting unterschieden werden sollen, konnten hohe ERn erzielt werden. Die zu testenden RSSI-Vektoren der gewählten iCPs konnten mit der Kombination Test-DB1/mean-DB(-101dBm) mit einer ER von 87,1% richtig zugeordnet werden.

Das WLAN-Fingerprinting wurde in Kombination mit den Orientierungssensoren der Testgeräte durchgeführt. Dabei wurden die Handys anhand ihrer Richtung erst manuell in Richtungskategorien eingeteilt. Es wurde gezeigt, dass eine automatische Erkennung der vier definierten Richtungskategorien möglich wäre. Untersucht wurde, ob sich die ERn verbessern lassen, wenn das WLAN-Fingerprinting nur mit RSSI-Vektoren der gleichen Richtungskategorie durchgeführt wird. Verbesserungen konnten nur teilweise erreicht werden. Einige der ERn sanken geringfügig durch die Berücksichtigung der Richtungskategorien, wobei dies möglicherweise auch auf Eingabefehler der Richtungskategorien zurückgeführt werden kann. Bei dem MVF- und Likelihood-Algorithmus

konnten mit Berücksichtigung der Richtungskategorien die Anzahl der Berechnungen um ¾ verringert werden, ohne dass sich die ERn verschlechterten.

Weiter wurde untersucht, ob sich die ERn verbessern lassen, wenn zwischen den iCPs die logischen Zusammenhänge der Ablaufreihenfolge beachtet werden. Dafür wurden Abschnitte definiert, die Wegentscheidungen darstellen. Simuliert wurde eine Ablaufreihenfolge, bei der iCPs nach einer logischen Sequenz besucht wurden. Die Datensätze der RSSI-Vektoren der iCPs wurden nach diesen Abschnitten aufgeteilt. Das WLAN-Fingerprinting wurde je Abschnitt simuliert und analysiert. Der Durchschnitt der ERn je Abschnitt lag dann bei 90,4 %. Durch die Verwendung des Gewichtungsvektors pro Abschnitt konnte dieser Wert auf 92,9 % verbessert werden.

In *Abschnitt-3*, der die vier Testräume darstellt, konnten die iCPs im ersten Test nur mit einer durchschnittlichen ER von 38,1 % erkannt werden. Dabei wurde, wie in 4.5.1 beschrieben, die Datenbank-Kombination DB1/mean-DB(-101dBm) verwendet. Durch die Miteinbeziehung der Richtung, der Selektion der mehrfachen RSS-Werte und der Gewichtung von RSS-Werten konnten die iCPs im Mittel mit einer Rate von 89,9% erkannt werden.

Die nächste Herausforderung ist es, die iCPs in der Bewegung zu erkennen - welcher iCP wird zu welchem Zeitpunkt passiert? Dies wird in **Kapitel 6** behandelt.

6 iCP-INS-Algorithmus

Wie bereits in **Kapitel 2** erläutert wurde, kann die Veränderung einer Position durch die Analyse der Daten des Beschleunigungssensors und des digitalen Kompasses berechnet werden. Für ein solches „Inertial Navigation System“ (INS) gibt es verschiedene Ansätze. Der verwendete Ansatz basiert auf der Erkennung von Schritten und deren Richtung, anhand derer die Position verändert wird.

Ein reines INS hat zwei systembedingte Probleme. Erstens muss der Startpunkt dem System bekannt sein. Zweitens wird bei jeder Positionsveränderung ein gewisser Fehler generiert. Daher weichen die ermittelten Positionen immer mehr von den tatsächlichen Positionen ab. Die in der Folge beschriebenen Ansätze versuchen, diese Probleme mithilfe der im vorigen **Kapitel 5** beschriebenen iCPs zu kompensieren.

Dazu wurden entsprechende Algorithmen bzw. Funktionen in MATLAB entwickelt, welche auf die mit dem CPS-App aufgezeichneten Testlauf-Datensätze angewandt wurden. Für die Tests wurden diese Funktionen so gestaltet, dass die Datensätze komplett betrachtet und verarbeitet werden konnten. Damit die Funktionen Echtzeitdaten verarbeiten können, sind Anpassungen, die in den Zusammenfassungen der Unterkapitel diskutiert werden, nötig.

Zum Vergleich der Ansätze wurden Teststrecken definiert. Auf diesen wurden Testläufe aufgezeichnet. Aufgrund der definierten Teststrecke und den erkannten Schritten wurden Referenzschritte ermittelt, die zum Vergleich mit den Ansätzen herangezogen werden können. Dies ist im *ersten Abschnitt dieses Kapitels* genauer erläutert.

Im *nächsten Unterkapitel* werden die entwickelten Schritterkennungsalgorithmen, die durchgeführten Tests und die erzielten Ergebnisse beschrieben.

In *Unterkapitel 6.3* wird auf die Erkennung der Richtung mittels des bereits in 2.3.3 beschriebenen Orientierungssensors eingegangen. Die aufgetretenen Probleme werden ebenfalls in diesem Kapitel dargelegt.

Der *darauffolgende Abschnitt* zeigt, wie aus dem Schritterkennungsalgorithmus und den Daten des Orientierungssensors ein INS-Algorithmus entwickelt wurde. Die erzielten Ergebnisse des INS-Algorithmus werden zusammengefasst dargestellt.

Einen Ansatz zum automatischen Erkennen des Startpunktes wird im Kapitel 6.5.2 vorgestellt und die Ergebnisse zusammengefasst.

In den Unterkapiteln 6.5.3 und 6.5.4 wird ein Ansatz dargelegt, wie die iCPs genutzt werden können, um das Problem des Auftretens von Abweichungen des INS-Algorithmus zu kompensieren. Dabei wird ein einfacher Algorithmus genutzt, um die beste Übereinstimmung der RSSI-Vektoren eines iCPs in einem Testlauf zu erkennen und so auf den Zeitpunkt des Passierens zu schließen. Dieser Zeitpunkt wird vom iCP-INS-Algorithmus benutzt, um die errechnete Position des INS-Algorithmus zu korrigieren.

6.1 Testläufe , Referenzpunkte und –Positionen

Mit dem **Sensor-WLAN-Recorder** des CPS-APP ist es möglich, Testläufe aufzuzeichnen. Bei einem Testlauf wird eine definierte Teststrecke abgegangen; dabei werden GPS-Koordinaten, Sensordaten und WLAN-Scans aufgezeichnet. Die Teststrecken sind im **Anhang A** in 8.4 definiert. Die aufgezeichneten Testläufe sind in 8.5 festgehalten. Die Daten der Testläufe sind unter [37] abrufbar.

Eine Teststrecke kann durch Referenzpunkte und iCPs definiert werden. Durch die direkte Verbindung dieser Punkte wird die Strecke definiert. Bei den Testläufen wurden die Zeitpunkte erfasst, wann ein Referenzpunkt oder iCP erreicht wurde. Dadurch ist es möglich, bestimmten Zeitpunkten bekannte Koordinaten zuzuweisen. Diese Koordinaten können mit ermittelten Koordinaten der untersuchten Algorithmen verglichen werden.

Damit für einen Testlauf nicht nur die Referenzpunkte als Kontrollmöglichkeit zur Verfügung stehen, wurden die Koordinaten zwischen zwei Referenzpunkte mithilfe des entwickelten Schritterkennungs-Algorithmus berechnet. Mit der in MATLAB geschriebenen Funktion `getRefSteps2()` [37, DAAS\functions\combined] wurden aus den Referenzpunkten diese Koordinaten errechnet. Um Referenzschritte zu erhalten, werden bei dieser Funktion die Anzahl der Schritte zwischen zwei Referenzpunkten gezählt, sowie die Zeitpunkte der Referenzschritte ermittelt. Entsprechend der gezählten Schritte wird die Strecke zwischen zwei Punkten aufgeteilt und die Koordinaten berechnet. Durch die bekannten Zeitpunkte der Schritte wird der errechneten Koordinate eine Zeit zugewiesen. Auf diese Weise werden Referenzpositionen bzw. Referenzschritte bereitgestellt, die zum Vergleich der ermittelten Koordinaten der zu analysierenden Algorithmen herangezogen werden. Dies ist möglich, da alle Algorithmen auf der Schritterkennung basieren und über die gleiche Anzahl an Schritten bzw. Positionsveränderungen verfügen.

Zur Vergleichbarkeit der GPS-Aufzeichnung werden mit der eigens erstellten Funktion `getGPSsteps()` [37, DAAS\functions\combined] die GPS-Koordinaten zu den Schrittzeitpunkten ermittelt. Die Funktion liefert daher dieselbe Anzahl an Koordinaten wie gezählte Schritte.

Damit werden die ermittelten Koordinaten der GPS-Aufzeichnung, des INS-Algorithmus und des iCP-INS-Algorithmus miteinander vergleichbar. Zu beachten ist, dass die Referenzschrittanzahl von der tatsächlichen Schrittanzahl abweichen kann. Wie man Tabelle 6-1 jedoch entnehmen kann, arbeitete der im nächsten Kapitel erläuterte `stepdetectionPeak()` zuverlässig unter den getesteten Bedingungen, dementsprechend sind die Abweichungen gering.

Abbildung 6-1 Haltung des Gerätes bei Testläufen.

6.2 Erkennung von Schritten

In Smartphones gehören Beschleunigungssensoren bereits zu jenen, die seit Beginn an eingebaut wurden. Diese Accelerometer (2.3.1) erkennen in 3 Achsen die Beschleunigung des Gerätes. Dadurch lassen sich verschiedene Bewegungsabläufe feststellen. In dieser Arbeit wurden die Daten des Beschleunigungssensors bei den Testläufen aufgezeichnet. Diese Daten wurden verwendet, um die in MATLAB entwickelten Algorithmen zur Schritterkennung zu testen. Dieses Kapitel erläutert die Grundlagen der Schritterkennung mit diesen Daten und beschreibt die zwei entwickelten Algorithmen. Die erzielten Genauigkeiten der erkannten Schritte werden in 6.2.3 gegenübergestellt.

Im Zuge dieser Arbeit wurden zwei Algorithmen zur Erkennung von Schritten im MATLAB-Framework entwickelt und getestet. Diese Algorithmen benutzen die Datensätze des Accelerometers, um Schritte zu erkennen und werden über Parameter wie z. B. Grenzwert und Mindestschrittzeit eingestellt. Zur Analyse dieser Algorithmen wurden 10 Testläufe (siehe 8.5) durchgeführt und die Sensordaten mit dem CPS-App erfasst. Auf Basis dieser Daten wurden optimale Werte für die Parameter ermittelt.

Durch das Durchlaufen der Algorithmen mit verschiedenen möglichen Parametern wurden die optimalen Parameter für die Algorithmen ermittelt. Dafür wurden die Schritterkennungsalgorithmen mit verschiedenen Parametern anhand der aufgezeichneten Testläufe untersucht. Die daraus errechneten Schrittanzahlen wurden mit den Sollschrittanzahlen verglichen. Gewählt wurden jene Parameter, die zu den kleinsten Abweichungen von den Sollschritten führten. Erst wurden Parameter für die In- und Outdoor Teststrecken separat gesucht, anschließend jene, die insgesamt zu den besten Ergebnissen gelangten. Die Sollschritte wurden während des Testlaufes im Kopf mitgezählt und nach der Aufzeichnung über das CPS-App dem Testlauf zugewiesen.

6.2.1 Grundalgen zu Sensordaten und Schritterkennung mit dem Acceleromter

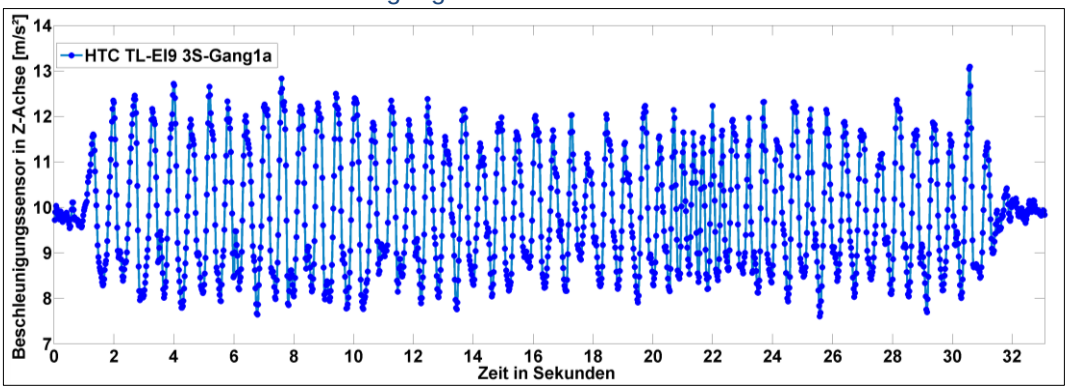
Das in MATLAB programmierte Framework (MFW) speichert die Daten des Beschleunigungssensors in der Structure Variable `TESTRUNS` im Feld „*acc*“. Ein Datensatz des Sensors besteht aus den Beschleunigungswerten in m/s² der drei jeweiligen Achsen des Gerätes und dem Genauigkeitswert, wie in 2.3.3 beschrieben wurde. Wird das Smartphone wie in Abbildung 6-1 dargestellt gehalten, bedeutet eine Vorwärtsbewegung des Handys, dass es in der Richtung der *y-Achse* eine Beschleunigung erfährt. Diese Daten können zur Berechnung der Positionsveränderung (siehe INS 2.1.4) bzw. zur Schritterkennung verwendet werden.

Ein Mensch, der geht, bewegt sich nicht nur in einer Ebene vorwärts, sondern er bewegt sich auch auf und ab. Bei dieser Schrittbewegung erfährt das Handy ebenfalls eine Beschleunigung senkrecht zum Boden. Bei den Tests wurden die Geräte waagerecht gehalten, daher wirkt in der z-Achse die Erdbeschleunigung (9,81 m/s²) permanent auf die Geräte ein. Durch die Auf- und Abbewegung wird im Sensor eine zusätzliche positive bzw. negative Beschleunigung initiiert. Daher addiert bzw. subtrahiert sich diese Beschleunigung zum Erdbeschleunigungswert.

Stellt man diese Beschleunigungswerte für die z-Achse des Testlaufes-EI9 dar und verbindet man die Diskreten Werte miteinander, so entsteht eine Kurve mit Maxima und Minima, die einer „Berg- und Talfahrt“ gleichen. Anhand dieses Bewegungsmusters versuchen Algorithmen Schritte zu erkennen. Für die Schritterkennung sind die Position und Lage des

Telefons entscheidende Faktoren. In [47] werden verschiedene Positionen für das Handy untersucht und verglichen. Für die hier entwickelten und verwendeten Algorithmen wird die Haltung bzw. Position des Handys, wie in Abbildung 6-1 dargestellt, vorausgesetzt.

Abbildung 6-2 TL-EI9 Daten des Beschleunigungssensors in z-Achse

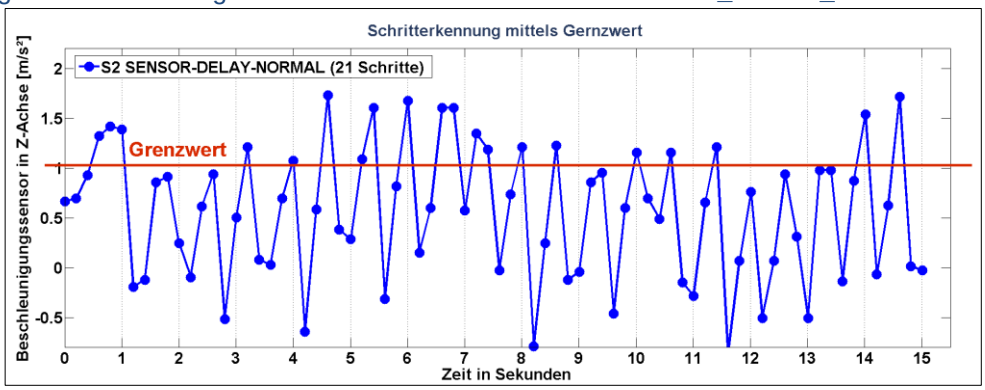


Bei den Sensordaten handelt es sich um Diskrete Werte; diese sind in Abbildung 6-2 und Abbildung 6-3 durch einen Punkt gekennzeichnet. Wie viele Werte ausgegeben werden, ist von der Geschwindigkeit des Sensors des Gerätes abhängig bzw. vom Parameter, der die Abtastrate bestimmt. Android verfügt über vier Stufen zur Definition der Abtastgeschwindigkeit, alternativ kann auch die gewünschte Abtastfrequenz direkt eingestellt werden [15]. Ein Beispiel für die Einstellung *SENSOR_DELAY_NORMAL* für das Galaxy S2 ist in Abbildung 6-3 dargestellt. Bei diesem Sampling-Beispiel ist zu sehen, dass für die meisten Schritte jeweils ein maximaler Wert vom Sensor emittiert wird. Zum Vergleich der TL-EI9, dessen Beschleunigungsdaten in Abbildung 6-2 dargestellt sind, wurde mit der höchstmöglichen Abtastrate aufgezeichnet. An dem Beispiel ist zu sehen, dass dort deutlich mehr Werte um das Maximum herum entstehen. Alle Testläufe, die in dieser Arbeit verwendet wurden, sind mit der höchstmöglichen Abtastrate aufgezeichnet.

6.2.2 Stepdection-Algorithmus mit Grenzwert

Ein einfacher Ansatz, um Schritte zu zählen, ist, ab dem Überschreiten eines bestimmten Grenzwertes einen Schritt zu erkennen. In Abbildung 6-3 ist ein Beispiel für einen Grenzwert dargestellt, der als rote Linie eingezeichnet wurde. Auch wenn in diesem Beispiel mit dem Grenzwert von 1,05 m/s² Schritte teilweise nicht erkannt oder Schritte doppelt gezählt werden, weicht die gezählte Anzahl von Schritten nur um einen Schritt vom Sollwert ab. Der Grund dafür ist, dass sich zu viel und zu wenig gezählte Schritte die Waage halten. In [70] wurde so ein Algorithmus zur Schritterkennung verwendet, der mit einem kalibrierten Grenzwert und einer Abtastfrequenz von ca. 5 Hz gearbeitet hat.

Abbildung 6-3 Aufzeichnung von 21 Schritten mit Abtastrate *SENSOR_DELAY_NORMAL*



Fehler entstehen bei diesem Ansatz, wenn zu viele bzw. zu wenige Werte über dem Grenzwert liegen. Bei einer zu hohen Abtastrate werden zu viele Schritte gezählt, bei einer zu geringen Abtastung werden zu wenige Schritte erkannt. Neben der Wahl der optimalen Abtastrate muss der Grenzwert so zu kalibrieren sein, dass sich im Mittel die zu viel bzw. zu wenig gezählten Schritte aufheben. Der Vorteil des Algorithmus ist, dass er einfach zu realisieren ist und zu guten Ergebnissen führen kann, wenn Abtastrate und Grenzwert gut gewählt sind. Ein Nachteil ist, dass Schritte nicht immer dann erkannt werden, wenn diese durchgeführt wurden.

Bei der Aufzeichnung der `TESTRUNS` mit dem CPS-App wurden die Daten des Beschleunigungssensors mit der höchsten Abtastrate abgefragt. Das ermöglicht es, nach möglichen Parametern unabhängig von der Abtastrate mit dem MFW zu suchen. Der Datensatz, der in Abbildung 6-2 dargestellt ist, wurde mit der maximalen Abtastrate erfasst.

Der Ansatz, der in [70] Verwendung fand, wurde für die Arbeit erweitert bzw. modifiziert. Bei dem implementierten Algorithmus, der im MFW `simplestepdection()` heißt, werden Schritte nur akzeptiert, wenn zwischen diesen eine definierte Zeitspanne liegt. Diese Zeitspanne ersetzt in gewisser Weise die Abtastrate. Dies ist auch eine Erweiterung, da Spitzenwerte, die sehr nahe beisammen liegen, also innerhalb der Zeitspanne sind, nur einmal gezählt werden. Für diesen Algorithmus gibt es zwei Parameter: einen, der den Grenzwert für die Beschleunigungsdaten (*accVstep*) und einen, der die Mindestschrittdauer (*steptimeD*) definiert. Mithilfe von MATALB wurden mögliche Parameter getestet. Die Parameter, die zu den besten Ergebnissen geführt haben, sind in Tabelle 6-1 beschrieben. Die detaillierten Ergebnisse sind im **Anhang B** in 9.1.1 zu finden.

Bei den Tests hat sich gezeigt, dass der Algorithmus auf die *z-Achse* angewandt bessere Ergebnisse erzielt, als wenn die y-Achse verwendet wird, während die Abweichung im Mittel bei den Outdoor-Testläufen unter Verwendung der *y-Achse* 5,7 Schritte betrug. So wichen die Anzahl der Schritte im Mittel nur um 1 Schritt ab, wenn die z-Achse verwendet wurde. Beachtet man die maximale Abweichung, betrug diese bei der Verwendung der *y-Achse* bis zu 10 Schritte. Werden die Daten der *z-Achse* benutzt, beträgt die maximale Abweichung dagegen 4 Schritte.

Für die Optimierung der Parameter wurden die Testläufe für In- und Outdoor zunächst separat betrachtet, um mögliche Unterschiede zwischen dem Gehen in einem Gebäude und auf der Straße zu berücksichtigen. Für die Outdoor-Testläufe H1-H6 zeigten sich als beste Parameter 13,0 m/s² als Grenzwert (*accVstep*) und 0,58 s als Mindestzeitabstand (*steptimeD*)**.** Mit diesen Parametern konnten im Mittel die Anzahl der Schritte auf einen Schritt genau erkannt werden. Bei Indoor-Testläufen erweisen sich 11,5 m/s² und 0,6 s als gute Parameter. Bei diesen Testläufen konnte die Schrittanzahl im Mittel auf 0,5 Schritte genau erkannt werden. Die detaillierten Ergebnisse sind im **Anhang B** in 9.1.1 und in Tabelle 9-2 und Tabelle 9-3 zu finden.

Es wurde auch nach Parametern gesucht, welche die besten Ergebnisse für In- und Outdoor Testläufe ergeben. Dabei konnten im Mittel die Anzahl der Schritte bis auf 1,6 Schritte genau erkannt werden. Die Parameter dafür waren 11,54 m/s² und 0,62 s. Die detaillierten Ergebnisse sind im **Anhang B** in Tabelle 9-4 aufgezeigt!

6.2.3 Stepdection-Algorithmus mittels Maxima-Ermittlung

Betrachtet man in Abbildung 6-2 oder Abbildung 6-3 einen einzelnen Schritt, so besteht dieser aus zumindest einem Maximal- und einem Minimalwert. Zählt man die lokalen Maxima, so erhält man die Anzahl der Schritte.

Der Algorithmus *stepdetectionPeak()* [37, DAAS\functions\stepdection] ermittelt die lokalen Maxima der Daten des Beschleunigungssensors, die zentralen Schritte sind als Pseudocode in Algorithmus 6-1 dargestellt. Für das Finden der lokalen Maxima wurde die Funktion `peakfinder()` [37, DAAS\functions\stepdection] implementiert. Durch Differenzieren der Werte und Anwendung der Signum Funktion werden die lokalen Maxima ermittelt. Der Ablauf ist als Pseudocode in Algorithmus 6-1 beschrieben.

Algorithmus 6-1 stepdectionPeak()

```
[steptimes]= stepdetectionPeak(var testrun, par)
Eingabe: testrun; accVstep; timestamps; steptimeD; sValue
Ausgabe: steptimes
% testrun… Structure die alle Daten eines Testlaufes enthält
% par.accVstep… Grenzwert-Parameter
% par.steptimeD… erlaubte Zeitspanne für Schritte
% par.sValue… Wert für Glättung.
%-----------------------------------------------------------------------------------------------------------------------------------------
% steptimes… Zeitpunkt an dem ein Schritt stattgefunden hat
 1:  Gs= smooth(testrun.acc.Gz, par.sValue); % glätten der Werte des Accelerometers in z-Achse
 2:  function findpeaks()
 3:   Gs1=diff(Gs); % ermittle 1. Ableitung
 4:   sigG1= sign(Gs1); % wende Signum Funktion an
 5:   peaks= find(diff(sigG1)==-2); % Leite sigG1 ab. An den Stellen wo sigG1=-2 ist sind lokale Maxima
 6:
 7:  accVTH=mean(G(peaks))*par.accVstep; % ermittelter Grenzwert
 8:  difftime=[ 0 diff(testrun.acc.timestamp(peaks))];% ermittle Zeit zwischen Maxima
 9:  peaks= peaks(Gs(peaks)> par.accVTH); % eliminiere peaks die kleiner als accVstep sind
10:  steptimes= peaks(difftime > par.steptimeD); % eliminiere peaks die weiniger als steptimeD auseinander liegen
```

Die Maxima werden als Schritt gewertet, wenn diese über dem Grenzwert (*accVTH*) liegen und zwischen den erkannten Schritten mindestens die definierte Zeit (*steptimeD*) vergangen ist. Der Grenzwert-Parameter (*accVstep*) wird in Prozent angegeben, da der Algorithmus den Grenzwert aus den gefundenen Maxima ermittelt. Dafür wird der Mittelwert aus den Maxima gebildet und aus dem definierten Prozentsatz der Grenzwert festlegt. Somit ist der Grenzwert nicht auf einen fixen Betrag festgelegt und wird vom Algorithmus angepasst. Dadurch ist dieser Grenzwert von Sensor und Achse unabhängig. Dieses Schema könnte später genutzt werden, um den Algorithmus für ein Handy und dessen User zu kalibrieren.

Weiter verwendet der Algorithmus die Glättungsfunktion `smooth()` [56] von MATLAB, um die Sensordaten mithilfe eines Tiefpassfilters zu glätten und somit Störungen zu entfernen. Der Wert *sValue* definiert, anhand wie vieler Werte der geglättete Wert errechnet wird.

Dieser Schritterkennungsalgorithmus wich bei der Anzahl der ermittelten Schritte im Mittel für alle Testläufe um 0,5 Schritte ab. Neben der Genauigkeit konnte auch die Zuverlässigkeit verbessert werden. Zu sehen ist das auch daran, dass in einem breiten Wertebereich von Parametern die gleichen Abweichungen erreicht wurden. Als gute Parameter erwiesen sich für *sValue* 5 für *steptimeD* 0,38 s und für *accVstep* 82 %. Die Ergebnisse sind in Tabelle 6-1 zusammengefasst und die Details sind im Anhang B in Abschnitt 9.1.2 zu finden.

6.2.4 Zusammenfassung und Ergebnisse der Schritterkennung

Bei den Tests hat sich die Verwendung der *z-Achse* bei der benutzten Position des Telefons als geeigneter gezeigt. Durch die Verwendung der Daten der *z-Achse* konnten bessere

Ergebnisse erzielt werden. In Tabelle 6-1 sind die mittleren Abweichungen der erkannten Schrittanzahl von der Referenzschrittanzahl zu sehen.

Mit dem `simplestepdection()` Algorithmus, der beim Überschreiten eines Grenzwertes ein Schritt detektiert, konnte im Mittel mit den besten Parametern die Schrittanzahl auf 1,6 Schritte genau erkannt werden. Maximal wich die erkannte Schrittanzahl um 4 Schritte von der gezählten ab. Der `stepdectionPeak()` Algorithmus, der die lokalen Maxima in den Beschleunigungsdaten zum Detektieren von Schritten verwendet, erkannte die Anzahl der Schritte im Mittel auf einen halben Schritt genau. Dabei wich dieser maximal um einen Schritt von den gezählten Schritten ab.

Die Algorithmen wurden offline auf die Daten in den Testläufen angewandt, sind aber so konstruiert, dass eine Adaption auf Echtzeitdaten möglich ist. Beim `simplestepdection()` Algorithmus müsste lediglich der Zeitpunkt des letzten Schrittes gespeichert werden, damit der Mindestzeitabstand seit dem letzten Schritt ermittelt werden kann. Für den `stepdectionPeak()` Algorithmus müssten die Daten des Beschleunigungssensors mindestens für die Zeit eines Schrittes gepuffert werden. Diese Daten könnten wie im Algorithmus 6-1 beschrieben verarbeitet werden. Allerdings müsste der Grenzwert anders definiert werden; entweder durch einen konkreten Wert oder dadurch, dass dieser erst kalibriert wird, indem eine Lernstrecke abgegangen wird.

Weiter ist zu erwähnen, dass zusätzliche Testläufe mit verschiedenen Personen und Geräten nötig wären, um eine allgemeinere Aussagekraft über die Qualität der gewählten Parameter zu bekommen. Für diese Arbeit sind die gefundenen Parameter jedoch ausreichend, um die überlegten Ansätze weiter zu untersuchen.

Bei den Testläufen haben sich leichte Unterschiede zwischen den Schritten im Outdoor- und Indoor-Bereich gezeigt. Schritte im Heim-Testgebiet wurden schneller und mit größeren Schrittweiten ausgeführt. Dies kann man aus den optimalen Parametern ablesen, denn beim Outdoor-Testgebiet sind der optimale Mindestzeitabstand und der Grenzwert höher. Die optimale Schrittweite, siehe Kapitel 6.4, betrug 0,67 m im Outdoor-Bereich und 0,65 m im Indoor-Bereich.

Tabelle 6-1 Übersicht über die Abweichungen in Schritten vom Sollwert

Testläufe zur Schritterkennung mit optimalen Parametern				
Beschreibt, welcher Ansatz und von welcher Achse des Beschleunigunssensor die Daten verwendet wurden.	**Abweichung in Schritten**			**Parameter**
	mittlere	**maximal**	**minimal**	**accVstep:** Grenzwert **steptimeD:** Mindestzeitabstand **smooth:** Glättungsfaktor für smooth()
Schritterkennung OUTDOOR (Heim-Abschnitt-1 auf 57m)				
mit Grenzwert (y-Achse)	5,7	10	2	**accVstep:** 0,50m/s² **steptimeD:** 0,67s
mit Grenzwert (z-Achse)	1,0	2	0	**accVstep:**13,00m/s² **steptimeD:** 0,58s
mit Maxima erkennen (z-Achse)	0,5	1	0	**accVstep:** 82% **steptimeD:** 0,38s **smooth:**5
Schritterkennung INDOOR (Gang1&2 3S 54m und 23m)				
mit Grenzwert in z-Achse	0,5	1	0	**accVstep:** 11,50m/s² **steptimeD:** 0,60s
mit Maxima erkennen (z-Achse)	0,5	1	0	**accVstep:** 80% **steptimeD:** 0,43s **smooth:**5
Schritterkennung IN/OUTDOOR				
mit Grenzwert in z-Achse	1,6	4	0	**accVstep:** 11,54 m/s² **steptimeD:** 0,62 s
mit Maxima erkennen (z-Achse)	0,5	1	0	**accVstep:** 82% **steptimeD:** 0,38 s **smooth:**5

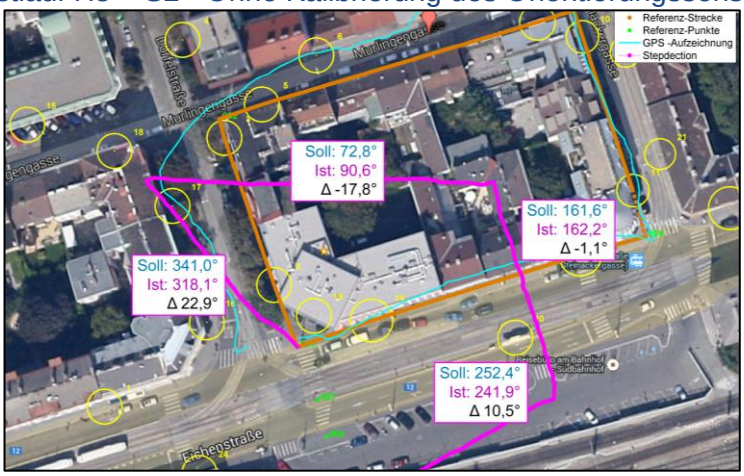
6.3 Erkennung der Richtung

Im Kapitel *„Technische Grundlagen“* wurde in 2.3.3 die Funktionsweise des Orientierungssensors erläutert. Für jene Algorithmen, die in dieser Arbeit benutzt wurden, wird lediglich der Wert der Auslenkung des Azimut-Wertes verwendet. Wird das Gerät wie in Abbildung 6-1 beschrieben gehalten, beschreibt der Orientierungswert in der z-Achse den Winkel zwischen magnetischem Norden und Ausrichtung des Smartphones am Genauesten (siehe 2.3.3). Ist das Handy nach Norden ausgerichtet, erhält man den Wert 0. Eine komplette Rechtsdrehung um die z-Achse durchläuft den Winkel von 0 bis 360°. Wird das Gerät nach Osten ausgerichtet, erhält man einen Winkel von 90°, nach Süden 180° und nach Westen 270° usw. Die Einteilung der Azimut-Werte ist in Abbildung 5-1 ebenso dargestellt.

Die Ermittlung der Richtung ist im Wesentlichen abhängig vom Magnetfeldsensor, der im Gerät verbaut ist. Dieser leitet anhand des Erdmagnetfeldes die Ausrichtung des Handys ab. Damit der Sensor möglichst präzise Daten liefert, muss der Sensor kalibriert sein und darf durch keine anderen magnetischen Felder gestört werden. Für die Kalibrierung muss das Gerät mehrfach um alle Achsen gedreht werden [68], damit das Gerät die Orientierung im Magnetfeld und somit den magnetischen Norden genauer ermitteln kann.

Ein Beispiel eines Testlaufes mit einem schlecht kalibrierten Sensor ist in Abbildung 6-4 dargestellt. Bei diesem Testrun wichen die ermittelten Orientierungen zwischen -17,8° und +22,8° von der Referenzorientierung ab; das ist ein Bereich von über 40°. Um das zu verhindern, musste das Gerät auf einer Ebene mehrmals um die z-Achse gedreht werden.

Abbildung 6-4 Testlauf H8 – S2 - Ohne Kalibrierung des Orientierungssensors



Bei den Testläufen, die im EI-Testgebiet durchgeführt wurden, gibt es Abweichungen zwischen -2,4° und 12,4°, auch bei vorheriger Kalibrierung. In Gebäuden wird das Erdmagnetfeld teilweise abgeschwächt bzw. durch Störquellen verfälscht. Die Abweichungen waren nicht für alle definierten Richtungen gleich groß (siehe 9.2). Neben der allgemeinen Abweichung im Gebäude können auch Störungen lokal auftreten. In Abbildung 6-5 und Abbildung 6-6 sind Ausschnitte aus zwei Testläufen dargestellt. Bei diesen Tests wurde der Gang1 jeweils in eine Richtung abgegangen. In der Nähe der Kabelschächte tritt jeweils dieselbe Ablenkung des Kompasses auf. Diese Abweichung ist mit einer blau gestrichelten Ellipse gekennzeichnet.

Abbildung 6-5 TL-EI9 auf TS-3S-Gang1a

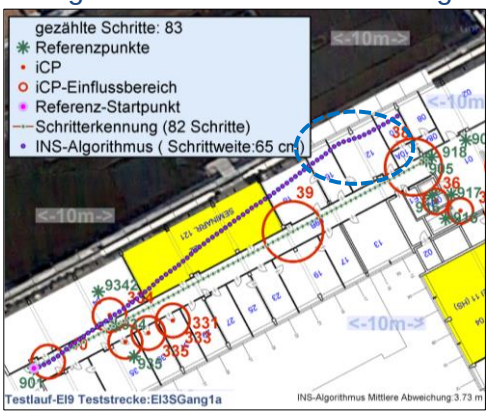
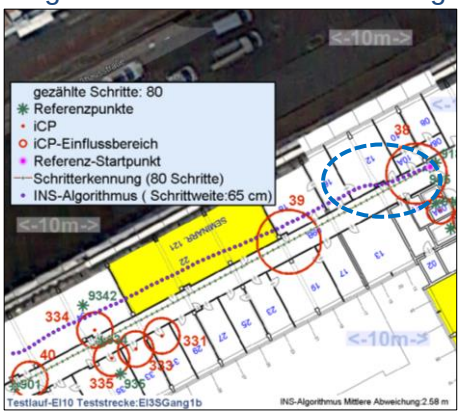


Abbildung 6-6 TL-EI10 auf TS-3S-Gang1b



Es gibt also einige Ursachen, die dazu führen können, dass die gemessene Richtung vom tatsächlichen Wert abweicht. Weiter zu untersuchen wäre, ob die Präzision des Kompasses erhöht werden kann, wenn z. B. für jede Richtungskategorie des Gebäudes ein Kalibrierungswert definiert wird. Vorstellbar wäre auch, dass für bestimmte Positionen im Gebäude Korrekturwerte ermittelt werden. Ein Ansatz, wie die Orientierung positionsabhängig kalibriert werden kann, wird in 6.5.4 diskutiert.

Alle untersuchten Testläufe und die Ergebnisse sind im Anhang in Kapitel 9.2 angeführt.

6.4 INS-Algorithmus – Ermittlung der Positionsveränderung

Durch die Kombination aus Schritterkennungsalgorithmus und Orientierungsdaten wurde ein INS-Algorithmus [37, DAAS\functions\stepdection] zur Ermittlung der Positionsveränderung entwickelt. Der Algorithmus kann also Veränderung der Position anhand der Daten des Beschleunigungs- und Orientierungssensors errechnen. Der in Algorithmus 6-2 als Pseudocode beschriebene INS-Algorithmus wurde auf die Daten der aufgezeichneten Testläufe angewandt. Für jeden Schritt wird ein Differenzschritt bzw. eine Differenzposition ermittelt. Um die absoluten Positionen bestimmen zu können, muss die Startposition bekannt sein. Der entwickelte INS-Algorithmus wurde auf mehrere Testläufe angewandt und der GPS-Aufzeichnung gegenübergestellt. Als Beispiel ist hier die Anwendung des INS-Algorithmus mit dem Testlauf-H7 in Abbildung 6-16 grafisch dargestellt. In Tabelle 6-2 sind die Ergebnisse für diesen Testlauf zusammengefasst.

Abbildung 6-7 INS–Algorithmus Testlauf H7

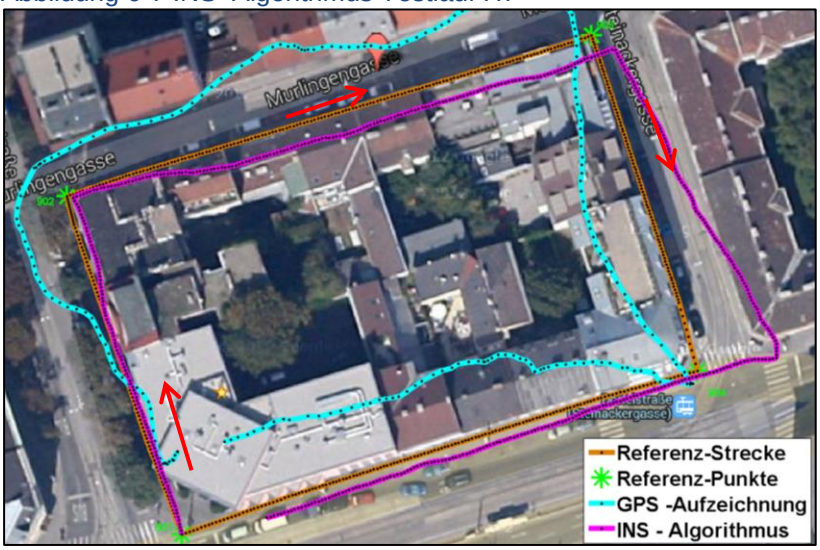


Tabelle 6-2 Testlauf H7

Zusammenfassung der Ergebnisse		
H7 UmHeim HTC	**GPS**	**INS-Algo.**
Abweichung Startposition	14,2 m	n.a.
Abweichung Endposition	17,4 m	17,2 m
Mittlere Abweichung	10,1 m	8,4 m
Maximale Abweichung	19,4 m	17,9 m
Minimale Abweichung	0,4 m	n.a.

Gegenübergestellt werden die errechneten Positionen zum Zeitpunkt der Schritte zu jenen der aufgezeichneten GPS-Positionen. Als Kontrolle dienen die ermittelten Referenzpositionen; wie diese ermittelt wurden, ist in 6.1 beschrieben. Die absoluten Positionen des INS-Algorithmus und der GPS-Aufzeichnung werden mit diesen Referenzschritten verglichen. Alle berechnenden Abweichungen in dieser Arbeit beziehen sich auf diese Referenzpositionen bzw. Referenzschritte.

Wie Tabelle 6-2 zu entnehmen ist, beträgt die mittlere Abweichung der GPS-Positionen 10,1 m, die des INS-Algorithmus 8,4m. Die Startposition des Testlaufs ist nicht vergleichbar, da diese beim INS-Algorithmus definiert ist und nicht ermittelt wurde. Auch die minimale Abweichung ist für den INS-Algorithmus nicht anwendbar (n.a.), da diese an der Startposition null ist.

Von Interesse ist die Abweichung der Endposition. Beim INS-Algorithmus setzen sich Fehler fort und werden aufsummiert. Bei dem Beispiel weicht die errechnete Endposition auf der 306 m langen Teststrecke „*Um Heim*“ um 17,2 m von der Endposition ab. Damit ist sie nur um 0,2 m genauer als die mittels GPS errechnete Position.

Der Algorithmus 6-2 zeigt den Pseudocode des in MALTAB entwickelten INS-Algorithmus. Zuerst werden mit dem in Algorithmus 6-1 beschriebenen Schritterkennungsalgorithmus die Zeitpunkte und die Anzahl der Schritte erkannt. Aus den Daten des Orientierungssensors werden die aufgezeichneten Azimut-Werte zum Zeitpunkt der Schritte erfasst.

Im Laufe der Entwicklung des Algorithmus wurde versucht, die Azimut-Werte, die vom Beginn bis zum Ende eines Schrittes erfasst wurden, zu mitteln, um einen genaueren Richtungswert zu erhalten. (`calStepOrientV2()` [37, DAAS\functions\stepdection]). Bei ersten Untersuchungen konnten jedoch keine Verbesserungen festgestellt werden.

In Zeile 3 werden die Azimut-Werte um einen definierten Korrekturwert angepasst. Diese Option wird vom iCP-INS-Algorithmus-II (siehe 6.5.4) verwendet. Aus den Richtungen und der definierten Schrittgröße werden die Differenzpositionen ermittelt. Mit der vorgegebenen Startposition werden dann die absoluten Positionen ermittelt.

Algorithmus 6-2 INSAlgorithmus()

```
[X,Y,steptimes,Xdiff,Ydiff]]=INSAlgorithmus(var testrun,startPos,par)
Eingabe: testrun, par
Ausgabe: X,Y,steptimes,Xdiff,Ydiff
% testrun… Structure die alle Daten eine Testlaufes enthält
% par.kal… Korrektur-Wert für Orientierung
%par.stepsize… Parameter für die definierte Schrittgröße
%startPos… Structure enthält die Koordinaten der Startposition
%--------------
 1:  steptimes=stepdetectionPeak(testrun, par);% ermittelt die Zeitpunkte an denen ein Schritt stattgefunden hat
 2:  orientation=calStepOrient( testrun, steptimes);% ermittelt den Azimut-Wert während der Schrittzeitpunkte
 3:  orientation=mod(orientation+360+p.kal,360);% korrigiert alle Azimut-Wert um den Korrektur-Wert
 4:
 5:  % errechnet aus Azimut-Wert u. Schrittgröße die Differenz Position
 6:  [Xdiff, Ydiff] = diffPos(orientation,parStepsize);
 7:
 8:  % verschiebt Differenz Schritte zu Startposition
 9:  X=Xdiff+startPos.X;
10:  Y=Ydiff+startPos.Y;
11:
12:  % Fügt die Startposition an
13:  X=[ startPos.X Xdiff];
14:  Y=[ startPos.Y Ydiff];
```

Für die Feinabstimmung der Schrittgröße wurden die Testläufe mit verschiedenen Werten für die Schrittlänge durchlaufen. Die ermittelten Endpositionen wurden mit den Soll-Positionen

verglichen. Jene Schrittgröße mit der geringsten mittleren Abweichung in allen Testläufen wurde in weiterer Folge benutzt. Als bester Werte hat sich eine Schrittlänge von 0,67 m für die Outdoor-Testläufe gezeigt. Bei den Tests im UNI Gebäude erwies sich eine Schrittweite von 0,65 m als bester Wert. Die Ergebnisse dieser Tests sind in Anhang B in Tabelle 9-12 erfasst.

Ergebnisse des INS-Algorithmus

Die Ergebnisse, welche durch die Anwendung des entwickelten INS-Algorithmus auf die ausgewählten Testläufe erzielt wurden, sind in Tabelle 6-3 bis Tabelle 6-6 zusammengefasst. Über alle betrachteten Testläufe wichen die errechneten Positionen im Mittel um 4,0 m ab. Im Vergleich dazu wich im Durschnitt die GPS-Position um 13,9 m von den ermittelten Referenzschritten ab. Mit dem INS-Algorithmus konnte die Endposition im Mittel auf 6,4 m genau erkannt werden. Bei den GPS-Daten wichen die Koordinaten der Endposition um 20,1 m ab. Die maximale Abweichung betrug beim INS-Algorithmus im Durchschnitt 6,5 m. Dies entspricht der Abweichung der Endposition, da die Abweichung meist an dieser maximal wird. Bei der GPS-Position lag die maximale Abweichung im Mittel bei 26,7 m. Alle untersuchten Testläufe sind im Detail in **Anhang B** in Kapitel 9.2 dokumentiert.

Tabelle 6-3 Vergleich GPS zu INS H1-H6

Zusammenfassung der Ergebnisse		
Mittelwerte H1-H6	**GPS**	**INS-Algo.**
Abweichung Startposition	9,7 m	n.a.
Abweichung Endposition	14,5 m	6,2 m
Mittlere Abweichung	10,2 m	3,4 m
Maximale Abweichung	20,0 m	6,5 m
Minimale Abweichung	3,0 m	n.a.

Tabelle 6-4 Ergebnisse INS-Algo. EI9-EI12

Zusammenfassung der Ergebnisse		
Mittelwerte EI9-EI12 GPS war nicht verfügbar		
Mittlere Abweichung Lon.	1,1	m
Mittlere Abweichung Lat.	2,2	m
Mittlere Abweichung	2,5	**m**
Abweichung Lon. Endpunkt	1,8	m
Abweichung Lat. Endpunkt	3,5	m
Abweichung Endpunkt	4,0	**m**
maximale Abweichung	4,5	m

Tabelle 6-5 GPS vs. INS H7,H9,H10

Zusammenfassung der Ergebnisse		
Mittelwerte H7,H9,H10	**GPS**	**INS-Algo.**
Abweichung Startposition	20,9 m	n.a.
Abweichung Endposition	15,3 m	12,3 m
Mittlere Abweichung	11,5 m	6,8 m
Maximale Abweichung	28,2 m	14,4 m
Minimale Abweichung	0,4	n.a.

Tabelle 6-6 Vergleich GPS zu INS EI1-EI5

Zusammenfassung der Ergebnisse		
Mittelwerte E1-EI5	**GPS**	**INS-Algo.**
Abweichung Startposition	22,7 m	n.a.
Abweichung Endposition	29,7 m	5,1 m
Mittlere Abweichung	19,8 m	4,1 m
Maximale Abweichung	33,8 m	7,1
Minimale Abweichung	9,6 m	n.a.

Konklusion und Ausblicke

Wenn man Abbildung 6-7 betrachtet, ist zu erkennen, dass die Abweichung vor allem dadurch verursacht wird, dass die erkannte Orientierung nach der zweiten Abbiegung stark von der tatsächlichen abweicht. Der sich dadurch fortsetzende Fehler führt letztlich zur Abweichung der Endposition und verschlechtert ebenfalls die mittlere Abweichung.

Der INS-Algorithmus hat also die Tendenz, immer mehr von der tatsächlichen Position abzudriften, je länger dieser ausgeführt wird. Wie stark diese Tendenz ist, ist abhängig von dessen Zuverlässigkeit und der Exaktheit der erfassten Sensordaten. Um den INS-Algorithmus permanent nutzen zu können, bedarf es einer stetigen Korrektur der Position nach einer gewissen Zeit. In den weiteren Abschnitten 6.5.3 und 6.5.4 wird ein Ansatz

diskutiert, wie diese Korrekturen durch Kombination mit einer WLAN-Fingerprinting Methode aussehen könnte.

Der hier vorgestellte INS-Algorithmus muss in dieser Form über präzise Koordinaten der Startposition verfügen. Wenn vorhanden sind die GPS-Koordinaten meist nicht präzise genug, um diese als Startposition zu nutzen. Ein Ansatz, wie diese Koordinaten mit einer hohen Genauigkeit berechnet werden können, ist in 6.5.2 erläutert.

6.5 Kombination der Daten von GPS-Receiver, WLAN-Modul, Bewegungs- und Orientierungssensor

Die Methode der Navigation mittels INS-Algorithmus unterliegt zwei systembedingten Problemen. Erstens kann nur von einer bekannten Position ausgegangen werden. Diese muss möglichst genau erfasst sein, da dieser Fehler permanent mitgenommen wird. Das zweite Problem liegt darin, dass die ermittelte Position immer ungenauer wird, je länger mit dem Algorithmus verfahren wird. Der Grund dafür sind Ungenauigkeiten bei der Schritterkennung und der Schrittweite sowie Abweichungen bei der Richtungserkennung. Auch wenn diese Verfahren verbessert werden und präziser arbeiten, werden weiterhin Fehler auftreten, welche sich akkumulieren und früher oder später zu nicht mehr akzeptablen Abweichungen führen.

Im Zuge dieser Arbeit wurde ein Ansatz untersucht, der mittels WLAN-Fingerprinting diese Probleme zu kompensieren versucht. Dazu wurden Positionen ausgewählt, die beim Begehen einer Strecke passiert werden müssen. Diese wurden mittels WLAN-Fingerprinting vermessen und möglichst so gewählt, dass sich diese durch Fingerprinting gut unterscheiden lassen (siehe Kapitel 5.2). Diese intelligenten Check Points (iCP) wurden in Folge verwendet, um einerseits eine Startposition zu finden und anderseits Abweichungen zu korrigieren.

6.5.1 Finden eines iCP in einem Testlauf

Das Konzept der iCPs sieht vor, zu erkennen, wann diese erreicht bzw. ob diese passiert wurden. In **Kapitel 4** und **Kapitel 5** wurden bereits Methoden untersucht, durch WLAN-Fingerprinting festzustellen, an welchem iCP ein WLAN-Scan durchgeführt wurde. In diesem Abschnitt wird ein Ansatz erläutert, bei dem der Zeitpunkt des Passierens eines iCP erkannt wird. Zur Kontrolle für das passieren der iCPs wurde dies bei den Testläufen manuell mit dem CPS-App erfasst. Wie in 6.1 beschrieben ist es so möglich, bestimmten Schritten bzw. Zeitpunkten das Erreichen eines iCP zuzuordnen. Mit dem im Anschluss präsentierten Ansatz wurde das Passieren der iCPs errechnet und mit den Referenzzeitpunkten verglichen. Dieser Ansatz ist für die Anwendung auf aufgezeichnete Testläufe ausgelegt. Um in Echtzeit arbeiten zu können, bedarf es einer Anpassung, die in der Zusammenfassung dieses Abschnittes diskutiert wird.

Um den Zeitpunkt des Passierens eines iCP in einem Testlauf zu erkennen, werden die euklidischen Abstände zwischen den zu einer Zeit t ermittelten RSSI-Vektoren und dem Vektor, der den iCP repräsentiert, berechnet.

Ein RSSI-Vektor des $iCPa$, der aus mehreren WLAN-Scans gemittelt wurde (4.3.3), wird in Folge als $\overline{RSSI_{iCPa}}$ bezeichnet. Dieser RSSI-Mittelwert-Vektor kann wie in *(F. 6-1)* definiert beschrieben werden:

$$\overline{RSSI_{iCPa}} = [\widetilde{Sa_{AP1}}, \widetilde{Sa_{AP2}}, \dots, \widetilde{Sa_{APn}}] \quad \text{(F. 6-1)}$$

$$\widetilde{Sa_{APx}} = \begin{cases} gemittelte\ Signalstärke\ von\ APx\ \ an\ iCPa & wenn\ APx\ an\ iCP\ a\ erreichbar \\ -101\ dBm & wenn\ APx\ an\ iCP\ a\ nicht\ erreibar \end{cases} \quad \text{(F. 6-2)}$$

Die zu einem Zeitpunkt t ermittelten RSS-Werte werden in einem RSSI-Vektor abgespeichert und mit einem Zeitstempel versehen. Diese sind im MFW in der Structure Variable `TESTRUNS` gespeichert. Dieser Vektor zu einer Zeit t ist in *(F. 6-3)* definiert und wird als $\overline{RSSI(t)}$ bezeichnet.

$$\overline{RSSI(t)} = [S(t)_{AP1}, S(t)_{AP2}, \dots, S(t)_{APn}] \quad \text{(F. 6-3)}$$

$$S(t)_{AP1} = \begin{cases} Signalstärke\ von\ APx\ \ zur\ Zeit\ t & wenn\ APx\ \ zur\ Zeit\ t\ erreichbar \\ -101\ dBm & wenn\ APx\ \ zur\ Zeit\ t\ nicht\ erreibar \end{cases} \quad \text{(F. 6-4)}$$

Wie beim herkömmlichen WLAN-Fingerprinting wird ein einzelner RSSI-Vektor ($\overline{RSSI_{iCPa}}$) mit einer Reihe von RSSI-Vektoren ($\overline{RSSI(t)}$) mithilfe des euklidischen Abstandes verglichen. Bei diesem Ansatz ist die Situation allerdings anders, denn die gesuchte Position bzw. der iCP ist bekannt; gesucht wird der RSSI-Vektor im Testlauf, der am besten zum iCP passt. Der Algorithmus geht davon aus, dass jener Vektor $\overline{RSSI(t_{iCPa})}$ mit dem kleinsten euklidischen Abstand $\mathrm{d_{min}}$ dann entstanden ist, als der iCP passiert wurde. Der Zeitpunkt t_{iCPa} ist also der Zeitpunkt, als der iCPa passiert wurde. Es wird also vorausgesetzt dass gilt: $\mathrm{d_{min}} = \mathrm{d}_{t_{iCPa}}$.

In der Funktion `findCPinRun()` [37, DAAS\functions\iCP] wird für jeden RSSI(t) Vektor eines Testlaufes der euklidische Abstand zu dem Vektor $\overline{RSSI_{iCPa}}$ berechnet. Die euklidischen Abstände d_{t_1} bis $d_{t_{end}}$ zum iCPa, des gesamten Testlaufes werden im Vektor $\overline{\mathrm{dis2iCPa}}$ gespeichert. Die Berechnung für $\overline{\mathrm{dis2iCPa}}$ ist in *(F. 6-5)* dargestellt.

$$\overline{dis2iCPa} = \begin{bmatrix} d_{t_1} \\ \vdots \\ d_{t_{end}} \end{bmatrix} = \begin{bmatrix} \sqrt{\left(\widetilde{Sa_{AP1}} - S(t_1)_{AP1}\right)^2} & \dots & \left(\widetilde{Sa_{APn}} - S(t_1)_{APn}\right)^2 \\ \vdots & & \\ \sqrt{\left(\widetilde{Sa_{AP1}} - S(t_{end})_{AP1}\right)^2} & \dots & \left(\widetilde{Sa_{APn}} - S(t_{end})_{APn}\right)^2 \end{bmatrix} \quad \text{(F. 6-5)}$$

t_1 ... Zeitpunkt erster-WLAN-SCAN und t_{end} ... Zeitpunkt letzter-WLAN-SCAN

d_t ...euklidischer Abstand des RSSI-Vektors zur Zeit t zum Mittelwert-Vektor des iCP-a

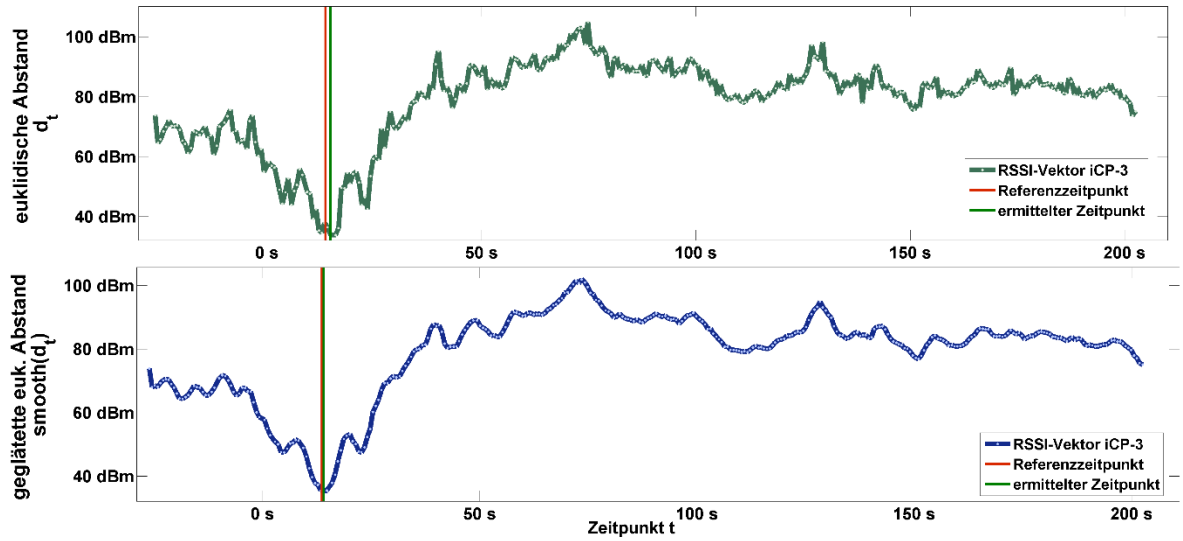
In Abbildung 6-8, in der oberen Darstellung, ist als Beispiel der Vektor $\overline{\mathrm{dis2iCP3}}$ für den Testlauf H7 als Kurve dargestellt. Tatsächlich handelt es sich dabei um diskrete Werte und jeder Wert bezeichnet den euklidischen Abstand des RSSI-Vektors zur Zeit t zum Mittelwert-Vektor des iCP-3.

Mit einer roten Line ist der Referenzzeitpunkt ($t_{iCP3_{Ref}}$) markiert, an dem der iCP-3 passiert wurde. Betrachtet man die Kurve der Abstandswerte, so ist ein Minimum zu erkennen, das

mit dem Zeitpunkt des iCP-3 einhergeht. Wann der iCP-3 passiert wurde, ist in den Daten des Testlaufes mit dem CPS-App manuell erfasst worden. Der Minimalwert $\mathrm{d_{min}}$ in $\overline{dis2iCPa}$ ist mit einem grünen Strich markiert und zeigt den kleinsten berechneten euklidischen Abstand, den der Vektor $\overline{RSSI_{iCP3}}$ der mit den Vektoren $\overline{RSSI(t)}$ bildet. Wie erläutert, wird bei diesem Ansatz angenommen, dass der iCP 3 zum Zeitpunkt t_{iCP3} passiert wurde. Wie der Abbildung zu entnehmen ist, liegen $t_{iCP3_{Ref}}$ und $t_{d_{min}}$ sehr nah beisammen. Zum Zeitpunkt $t_{d_{min}}$ entsprach die Position in etwa der des iCP3.

Für die Arbeit wurde dieser einfache Ansatz verwendet, dabei wird nur das beschriebene Minimum zum Identifizieren des Zeitpunktes des Passierens eines iCP verwendet. Überlegungen für mögliche Varianten werden in der *Zusammenfassung des Abschnittes* gemacht. Untersucht wurde in der Arbeit, ob durch Glätten der ermittelten euklidischen Abstände eine bessere Erkennung des Zeitpunktes $t_{iCPa_{Ref}}$ möglich ist. Dafür wurden die berechneten Werte in $\overline{dis2iCPa}$ mit der MATLAB Funktion `smooth()` geglättet. Dies ist in der unteren Kurve in Abbildung 6-8 dargestellt. In diesem Beispiel kann dadurch der gesuchte Referenzzeitpunkt nahezu exakt ermittelt werden.

Abbildung 6-8 Grafische Darstellung des Vektors $\overline{dis2iCP3}$ in Testlauf H7



Um die Funktion `findCPinRun()` zu analysieren, wurden in einer weiteren Testreihe ausgewählte Testläufe untersucht. Dabei wurden die Zeitpunkte t_{iCPa} für das Passieren der iCPs errechnet und mit den Referenzzeitpunkten $t_{iCPa_{Ref}}$ verglichen. Die Resultate sind in Anhang B im Abschnitt 9.3 aufgelistet. In Tabelle 6-7 und Tabelle 6-8 sind diese zusammengefasst dargestellt, wobei die Abweichung zwischen $t_{iCPa_{Ref}}$ und t_{iCPa} in Schritten angegeben ist. So kann auch die Entfernung (Schrittweite ca. 69 cm) abgeschätzt werden.

Ergebnisse mit der Funktion findCPinRun()

Bei den Outdoor Testläufen konnten im Mittel die iCPs mit dem HTC Testhandy auf 3 Schritte genau erkannt werden. Mit dem S2 Handy war das Erkennen der iCPs deutlich ungenauer. Mit diesem Gerät konnten die iCPs im Mittel nur auf 7,2 Schritte erkannt werden. Der Grund dafür ist, dass ein Scan mit dem Samsung Galaxy S2 im Schnitt 5-mal (ca. 3,5 s) länger dauert als mit dem HTC (ca. 0,7 s).

Für diesen Ansatz ist es aber wesentlich, dass die Scandauer kurz genug ist, damit ungefähr jedem Schritt ein WLAN-Scan bzw. RSSI-Vektor zugewiesen werden kann. Aus den

Ergebnissen zur Ermittlung der besten Parameter für die Schritterkennung (Tabelle 6-1) ist aus dem Grenzwertansatz zu sehen, dass bei den Testläufen zwischen den Schritten ca. 0,62 Sekunden lagen. Daraus kann man ableiten, dass man sich mit dem S2 Handy während eines WLAN-Scans ca. 6 Schritte weit bewegt. Beim HTC wird hingegen etwa bei jedem Schritt ein WLAN-Scan durchgeführt. Aus diesem Grund sind die erzielten Ergebnisse mit dem S2 deutlich schlechter.

Bei der Abbildung 6-8 zeigt die obere Grafik die berechneten Distanzwerte. Diese Kurve beinhaltet jedoch Fluktuationen, weswegen die Werte geglättet wurden. Die untere Kurve zeigt die geglätteten Distanzen. Verwendet wurde dazu die MATLAB Funktion *smooth()* [56]. Dadurch konnten die iCPs im Mittel beim HTC auf einen Schritt und beim S2 auf 5,4 Schritte genau erkannt werden, was eine Verbesserung gegenüber der Methode mit den nicht geglätteten Distanzwerten darstellt.

In Tabelle 6-8 sind die untersuchten Testläufe für das EI-Testgebiet dargestellt. Da mit dem Samsung Galaxy S2 keine guten Ergebnisse aufgrund der längeren Dauer der Scans erzielt wurden, fand für die weiteren Untersuchungen nur mehr das HTC EVO 3D Verwendung. Für die Tests, die größtenteils Indoor durchgeführt wurden, konnten die iCPs im Mittel ohne Glättung der Distanzwerte auf 4,7 Schritte genau erkannt werden. Durch die Glättungsfunktion konnte dieser Mittelwert auf 3,1 Schritte verbessert werden. Betrachtet man die Ergebnisse der einzelnen Testläufe in Tabelle 6-8, so ist zu sehen, dass es teils große Unterschiede bei der Erkennung der iCPs gibt. So weichen die Testläufe TL-EI1 und TL-EI5 im Mittel bei nicht geglätteten Werten um 8,5 Schritte von den Referenzschritten ab. Durch das Glätten der Werte konnte die Abweichung im Mittel auf 5,6 Schritte verbessert werden. Die Ergebnisse sind im Detail in Anhang C in 9.3 zu finden.

Der Unterschied der Genauigkeit zwischen den zwei Testgebieten mit dem HTC Handy steht im Zusammenhang damit, dass im Heim-Testgebiet die Dichte an APs geringer ist und sich so Unterschiede in den Signalstärken der einzelner APs stärker auswirken. Dies wurde bereits diskutiert. Zum anderen konnten bestimmte iCPs nur mit großen Abweichungen erkannt werden. Beispiele für diese iCPs werden im Anschluss diskutiert.

Tabelle 6-7 Zusammenfassung Ergebnisse iCP Erkennung im Heim-Testgebiet

	Erkennung der iCPs - Zusammenfassung Heim-Testgebiet										
	Mittlere Abweichung der Testläufe in Schritten										***gemittelte Schritte***
	Testlauf										
	S2					**HTC**					
	H1	**H2**	**H3**	**H8**	**H10**	**H4**	**H5**	**H6**	**H7**	**H9**	
Ohne Glättung der Werte d_t	13,5	1,5	2,5	6,5	11,8	0,5	6	3,5	3	2,2	**5,1**
gemittelte Schritte - Gerät	***7,2***					***3,0***					--
Mit Glättung der Werte d_t	13,5	1	2,5	6,4	8,8	0	1,5	0,5	1,5	1,7	**3,7**
gemittelte Schritte - Gerät	***6,4***					***1,0***					--

Tabelle 6-8 Zusammenfassung Ergebnisse iCP Erkennung im EI-Testgebiet

	Erkennung der iCPs - Zusammenfassung EI-Testgebiet												
	Mittlere Abweichung der Testläufe in Schritten												*gemittelte Schritte*
	Testlauf												
	EG - Erdgeschoss					3S - 3. Stock							
	EI1	EI2	EI3	EI4	EI5	EI6	EI7	EI8	EI9	EI10	EI11	EI12	
Ohne Glättung der Werte d_t	8	1,3	1	4	9	6,8	0,5	7,5	6,5	9,5	0	5	**4,9**
gemittelte Schritte - Gerät	***4,7***					***5,1***							--
Mit Glättung der Werte d_t	6	0,7	0,5	2,7	5,7	5,5	0,25	7,8	0	9,5	0	4	**3,6**
gemittelte Schritte - Gerät	***3,1***					***3,9***							--

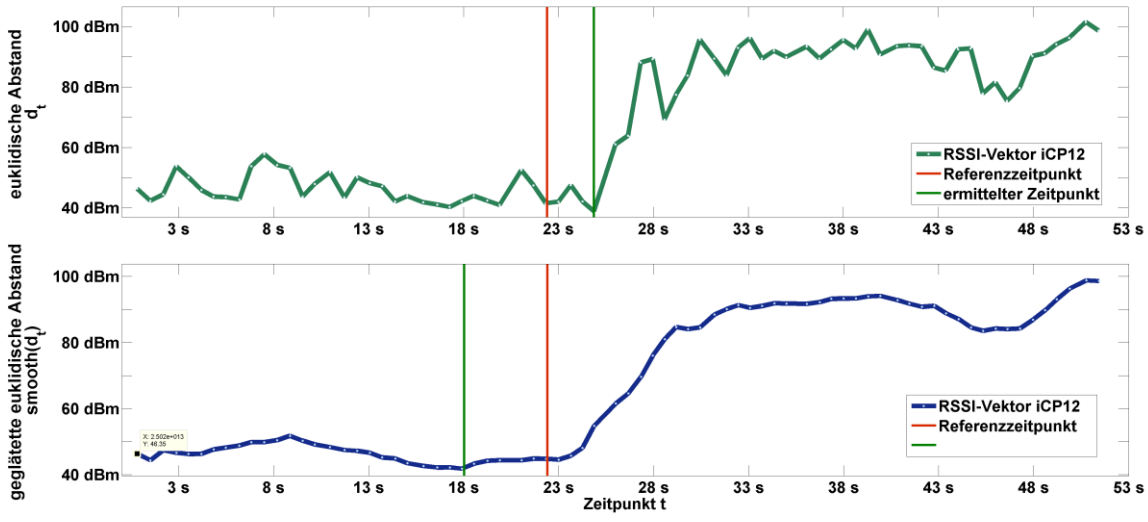
Detaillierte Betrachtung der iCP12, iCP5 und iCP39 des EI-Testgebietes

Grund für die hohen Abweichungen der Testläufe TL-EI1 und TL-EI5 sind die iCP-5 und -12. Der iCP-12 befindet sich vor dem Notausgang des Gebäudes und der iCP 5 vor dem Lift 1. Um diese iCPs zu erreichen, wird ein Gebiet durchschritten, in dem es keine Strukturen oder Objekte gibt, welche die Signale beeinflussen. Die iCP befinden sich zwar vor einer baulichen Struktur, die Signalstärken der APs werden aber erst nach dem Passieren beeinflusst.

Dies zeigt auch die Kurve der euklidischen Abstände des iCP-12 während des TL-EI1; diese sind in Abbildung 6-9 dargestellt. In Abbildung 6-10 sind die hier diskutierten iCPs des EI-Testgebietes im Erdgeschoss (EG) zur Orientierung dargestellt. Bevor das Gebäude betreten wurde, sind die euklidischen Abstände der erfassten RSSI-Vektoren zum Vektor $\overline{\mathrm{RSSI}_{\mathrm{iCP12}}}$ gering und unterscheiden sich kaum. Unmittelbar nachdem die Türe durschritten wurde, ändern sich diese jedoch abrupt um ca. 40 dBm. Die rote Linie in Abbildung 6-9 zeigt den Zeitpunkt, an dem iCP-12 erreicht wurde, kurz vor der Öffnung die Türe.

Bei diesem iCP gibt es kein eindeutiges Minimum, das sich in einem Werte-Tal befindet. Aus diesem Grund wird durch die Glättung der Werte der Zeitpunkt verfälscht. Durch das Glätten der Werte geht der Minimum-Wert bei Sekunde 25 verloren und der iCP wird statt auf Schritt 48 bereits bei Schritt 37 erkannt. Der Referenzschritt ist bei Schritt 47, der iCP-12 wird um 10 Schritte verfehlt, wenn geglättet wird. Die Details zu diesem Testlauf sind im *Anhang C* in Tabelle 9-46 dargestellt.

Abbildung 6-9 Grafische Darstellung des Vektors $\overline{dis2iCP12}$ des Testlauf EI1



Die Kurve für den iCP-5 im Testlauf TL-EI1 ist in Abbildung 6-11 dargestellt. Der iCP-5 befindet sich vor dem Lift 1 in der Aula des Gebäudes. Die Ergebnisse des Testlaufes sind in *Anhang C* in Tabelle 9-50 beschrieben. Für diesen Testlauf wurde der Seiteneingang, an dem sich der iCP-111 befindet, verwendet. Dieser befindet sich im Stiegenhaus zum Untergeschoss des Gebäudes. Der iCP-111 ist etwa 7,5 m vom iCP-5 entfernt. Die RSSI-Vektoren von iCP-5 und -111 werden durch die gleichen APs und ähnlichen RSS-Werte repräsentiert.

Abbildung 6-10 Teil EI-Testgebiet EG

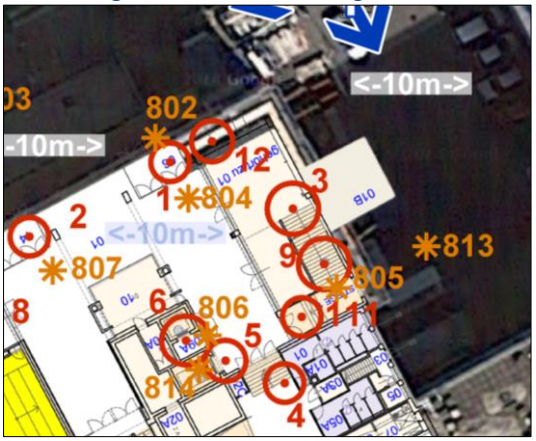
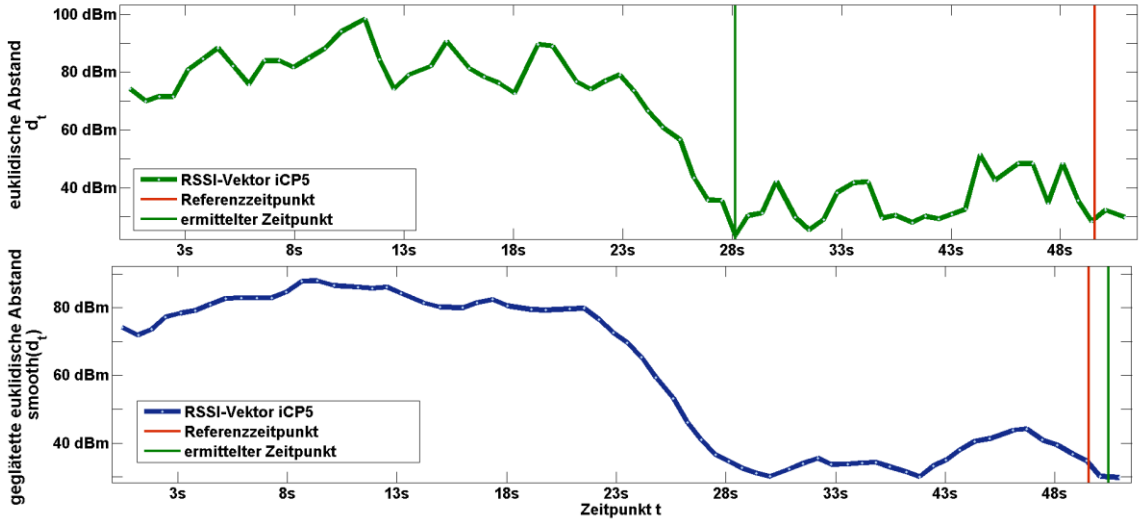


Abbildung 6-11 Grafische Darstellung des Vektors $\overline{dis2iCP5}$ des Testlauf-EI5



Wird die Logik des Gebäudes nicht berücksichtig und werden die Abstandwerte nicht geglättet, so wird der iCP-5 sechs Schritte vor dem iCP-111 erkannt. Der iCP-111 muss aber im Testlauf zuerst passiert werden, da dieser sich hinter der Eingangstür befindet. Betrachtet man die Abbildung 6-11, ist bei Sekunde 28 eine Reduktion des euklidischen Abstandes um ca. 40 dBm zu sehen. Das ist der Zeitpunkt, an dem das Gebäude betreten wurde. Die Abstandswerte-Kurve für den iCP-111 hat eine dieser Abbildung sehr ähnliche Form. In der Kurve mit den geglätteten Werten ist auch gut zu erkennen, dass es mehrere lokale Minima gibt. Durch das Glätten und das Verwenden der Abfolge-Logik wird mit dem iCP-INS-Algorithmus-II (Abschnitt 6.5.4) der iCP-5 korrekt erkannt.

Eine weitere Abstandwerte-Kurve mit zwei Minima ist in Abbildung 6-12 dargestellt. Diese zwei Minima entstehen bei der Untersuchung des iCP-39 in Testlauf TL-EI10. Betrachtet man anhand der Abbildung 6-13 die Positionen der APs und des iCP39, so ist zu sehen, dass die APs und der iCP sich am Gang befinden. Bewegt man sich entlang des Ganges vom Stiegenhaus-01 in Richtung Raum-33, so bewegt man sich von AP33 weg und auf AP34 und AP35 zu. Die Signalstärken der APs nehmen also ab und zu. Da diese Anordnung der APs für jedes Stockwerk besteht, summieren sich diese Effekte für weitere RSS-Werte anderer APs von den weiteren Stockwerken.

Dadurch, dass beim WLAN-Fingerprinting nur absolute Differenzen zu den erwarteten RSS-Werten berechnet werden, besteht kein Unterschied, wenn z. B. der erfasste Wert 10 dBm

über oder unter dem erwarteten Wert liegt. So entsteht hier eine zweite Kombination an RSSI-Werten, die ebenfalls zu einem geringen euklidischen Abstand zum Vektor $\overline{\mathrm{RSSI}_{\mathrm{iCP39}}}$ führt.

Abbildung 6-12 Grafische Darstellung zwei Minima des Vektors $\overline{dis2iCP39}$ des Testlauf-EI10

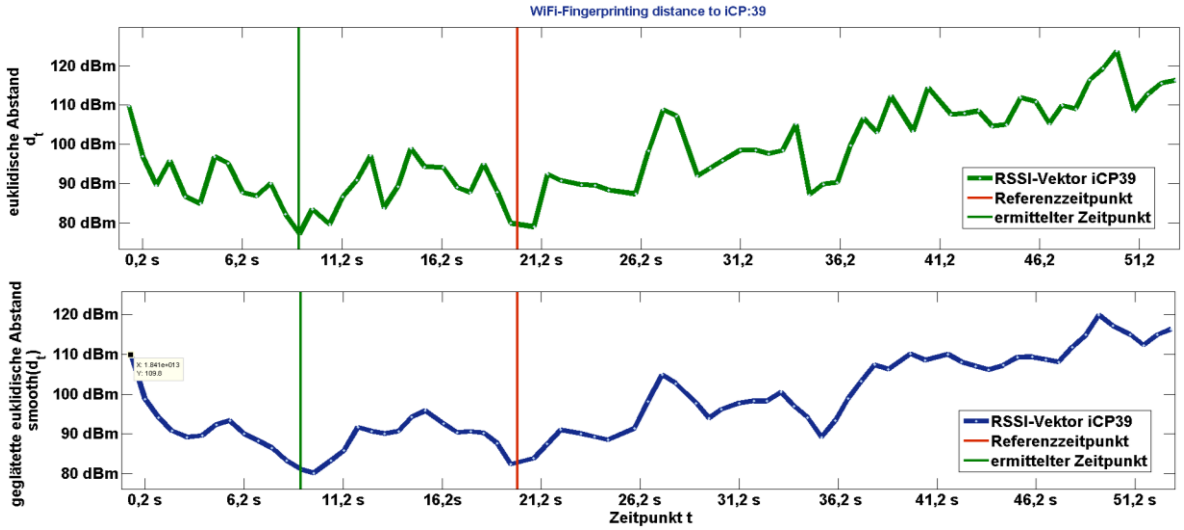
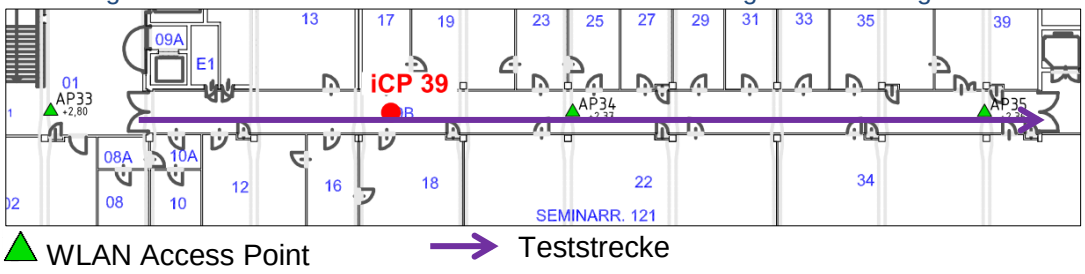


Abbildung 6-13 Position der AccessPoints und des iCP im Gang1 des EI-Testgebietes



Bei dem Beispiel in Abbildung 6-12 wird der iCP-39 bei Schritt 7 erkannt; tatsächlich wird dieser bei Schritt 25 erreicht, der iCP wird also 18 Schritte zu früh erkannt. Die Ergebnisse des Testlaufs EI10 sind im Anhang B in Tabelle 9-55 zu finden. Wird der Gang von der anderen Seite begangen, wird der iCP 39 korrekt erkannt. Dieses Ergebnis ist in Tabelle 9-56 aufgezeigt. Vermieden wird dieses Problem beim iCP-INS-Algorithmus-II (Abschnitt 6.5) durch das Verwenden der logischen Abfolge und der Geltungsbereiche der iCPs. Eine weitere Verbesserung würde vermutlich sein, wenn beim Entwerfen von WLAN-Netzen die APs nicht rasterhaft angeordnet und diese in jedem Stock etwas anders aufgeteilt werden.

Zusammenfassung, Konklusion und Ausblicke zur dynamischen Erkennung der iCPs

Bei der Testreihe zur Untersuchung der dynamischen Erkennung der iCPs konnten die iCPs im Heim-Testgebiet im Mittel auf einen Schritt (mit dem HTC mit Glättung) genau erkannt werden. Indoor konnten die iCPs im EI-Testgebiet durchschnittlich auf drei Schritte (HTC mit Glättung) genau erkannt werden. Gut ließen sich jene iCPs erkennen, wo sich, durch die vorhandenen baulichen Strukturen, die RSS-Werte beim Passieren deutlich veränderten. So ließen sich z. B. jene iCPs bei den Eingängen bzw. im Windfang des Eingangs am besten erkennen.

Die Testreihe beschränkte sich auf die Suche der iCPs in aufgezeichneten Datensätzen. Für eine Anwendung auf Echtzeitdaten wären Modifikationen nötig. Die Funktion `findCPinRun()` kann in aufgezeichneten Datensätzen erkennen, zu welchem Zeitpunkt ein bestimmter iCP passiert wurde. Ihr können aber nur iCPs übergeben werden, die auch

passiert wurden. Um zu erkennen, ob ein iCP gewählt wurde, bedarf es weiterer Einschränkungen. In den nächsten Abschnitten werden dafür Ansätze und Lösungen behandelt.

Ein Problem, das sich bei der Verwendung dieses Ansatzes Outdoor ergibt, ist, dass ein iCP in einem beschränkten Bereich passiert werden muss. Im urbanen Raum kann man jedoch auch auf der Straße gehen bzw. diese an jedem beliebigen Punkt kreuzen. Die iCPs des Heim-Testgebietes könnten auch umgangen werden. In diesem Fall ist der minimale Abstandwert nicht unbedingt zu dem Zeitpunkt entstanden, an dem der iCP passiert wurde. In diesem Fall wäre es eher jener Zeitpunkt, an dem man dem iCP am nächsten war.

In einem Gebäude ist dieses Problem meist nicht gegeben. So kann z.B. ein iCP, der am Gang in der Mitte definiert ist, maximal um die halbe Gangbreite entfernt passiert werden. Diese Abweichung ist akzeptabel, denn die wesentliche Information wird erkannt – nämlich der Umstand, dass der Gang abgegangen wird.

Der Zeitpunkt des Passierens eines iCPs wurde auf den ermittelten minimalen euklidischen Abstandswert im Laufe eines gesamten Testlaufes zurückgeführt. In dieser Form ist das mit Echtzeitdaten nicht möglich, da ja nicht in die Zukunft geschaut werden kann. Die benötigten RSS-Werte können jedoch gepuffert werden. Das könnte eine zeitverzögerte Positionskorrektur möglich machen, wie dies im nächsten Abschnitt für die Rückrechnung der Startposition gemacht wurde.

Um das Erkennen, welcher iCP passiert wurde, zu erweitern, wurden für die iCPs Einflussbereiche eingeführt. Diese können auch dazu dienen, den Ansatz für die Echtzeitnavigation anzuwenden. Die Einflussbereiche sind durch Kreise in den Abbildungen mit den Testläufen dargestellt. In den nächsten Abschnitten werden Algorithmen vorgestellt, welche einen iCP nur dann akzeptieren, wenn der Zeitpunkt des ermittelten minimalen Abstandwertes und die dort berechnete Position in diesem Bereich zusammenfallen.

Die Einflussbereiche könnten auch genutzt werden, um den Algorithmus `findCPinRun()` in Echtzeit nutzen zu können. Da nur die minimalen Abstandswerte zu den RSSI-Vektoren der iCPs innerhalb des Einflussbereiches betrachtet werden, müssten WLAN-Scans nur dann durchgeführt werden, wenn man sich laut der aktuell berechneten Position im Einflussbereich befindet. Wird dies mit einem Grenzwert kombiniert, könnte so ein iCP etwas zeitverzögert, aber in Echtzeit erkannt werden. Das würde zudem Energie sparen, denn permanente Scans verbrauchen viel Strom und entleeren den Akku. Eine einfache Erweiterung wäre daher auch, einen Grenzwert einzuführen.

Betrachtet man die Abbildungen mit den errechneten Abstandswerten zu einem RSSI-Vektor eines iCP, so wären auch Ansätze möglich, die an der Kurvenform das Passieren eines iCP erkennen. Ein Ansatz könnte z. B. darauf aufbauen, den starken Anstieg oder Abfall der Abstandwerte zu erkennen. Dieser Ansatz würde sich besonders für Türen eignen. Mittels Differenzierung der Abstandswerte könnten so Wendepunkte ermittelt und das Passieren erkannt werden. Dies ist jedoch nicht Teil der Arbeit, es sei nur zwecks möglicher weiterführender Untersuchungen erwähnt.

6.5.2 Erkennen des ersten iCP und Rückrechnung der Startposition

Für die Navigation mittels Positionsveränderung (INS) [70] wird eine bekannte Ausgangsposition benötigt. Damit der in Kapitel 6.4 vorgestellte INS-Algorithmus möglichst

präzise Koordinaten liefert, muss die Startposition bekannt sein. In diesem Abschnitt wird ein Ansatz beschrieben, bei dem iCPs verwendet werden, um auf die Startposition rückzuschließen.

Die Grundidee des Ansatzes ist es, das erstmalige Passieren eines iCP zu erfassen und aus den aufgezeichneten Sensordaten die Startposition rückzurechnen. Da die Anzahl der iCPs bei einer völligen Abdeckung z.B. einer Stadt sehr hoch sein kann, werden nur die iCPs betrachtet, die sich in der Nähe befinden. Um eine Position abschätzen zu können, werden beim im Anschluss beschriebenen Algorithmus die GPS-Koordinaten zum Startzeitpunkt des Testlaufes verwendet. Da die k geografisch nähesten iCPs herangezogen werden und das Konzept sich des k-nearest-neighbor-Prinzips (kNN) bedient, wird hier der Begriff bzw. die Arrayvariable `kNNiCP` verwendet. Der Algorithmus 6-3 beschreibt den Ansatz als Pseudocode.

Der hier beschriebene Ansatz wird vom iCP-INS-Algorithmus (Abschnitt 6.5.3) verwendet, um den ersten iCP, der passiert wurde, zu ermitteln. Der für den Innenbereich entwickelte iCP-INS-Algorithmus-II (6.5.4) verwendet die Funktion `Findentrance()`. Diese Funktion arbeitet nach demselben Prinzip wie der Algorithmus 6-3, allerdings um die Bedingung erweitert, dass nur iCPs von Eingängen zur Berechnung der Startposition benutzt werden.

Im TL-H7, in Abbildung 6-14, weicht die errechnete Startposition 3 m von der Referenzposition ab. Zum Vergleich weicht die durch GPS ermittelte Position um 14,2 m ab.

Das Beispiel TL-E2 zeigt den Eingangsbereich bzw. den Außenbereich des Institutsgebäudes. Bei diesem Testlauf beträgt die Abweichung der errechneten Startposition 11,3 m. Dies ist die höchste Abweichung, die bei den untersuchten Testläufen entstanden ist. Auch bei diesem Beispiel (siehe auch: Detaillierte Betrachtung der iCP12, iCP5 und iCP39) sind die kalkulierten Startkoordinaten präziser als die GPS-Koordinaten, welche 37,4 m von der Referenzposition abweichen.

Abbildung 6-14 TL-H7 Heim-Testgebiet

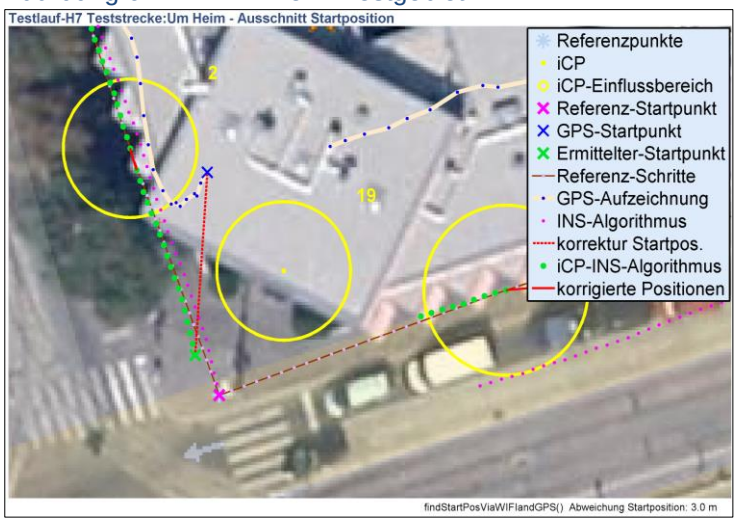
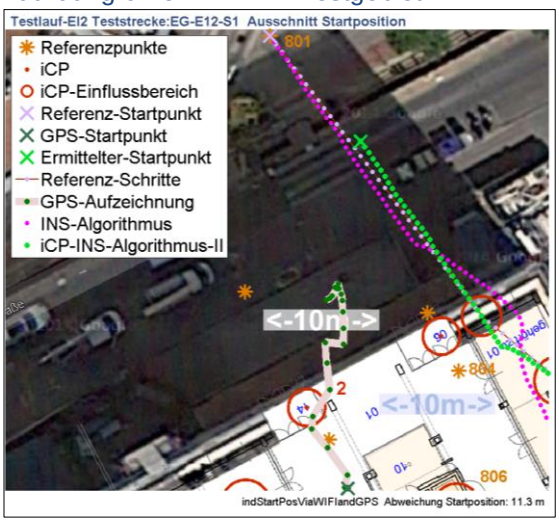


Abbildung 6-15 TL-E2 EI-Testgebiet



Erläuterungen zum Algorithmus - findStartPosViaWIFIandGPS()

Der Algorithmus zur Berechnung der Startposition ist als Pseudocode in Algorithmus 6-3 erläutert. Als erster Schritt ermittelt die Funktion `findKNNiCP()` [37, DAAS\functions\iCP], aus den GPS-Koordinaten zum Startzeitpunkt die k nahesten iCPs. Für das EI-Testgebiet wurden nur iCPs mit dem *Layer=0* betrachtet, das sind diejenigen bei den Eingängen. Auf

alle ermittelten iCPs wird die Funktion *findCPinRun()* angewandt. Betrachtet wird jedoch nicht der ganze Testlauf, sondern eine bestimmte Anzahl (`AnzScans`) der ersten erfassten WLAN-Scans bzw. RSSI-Vektoren. Für die Tests wurde dieser Parameter mit 40 festgelegt, somit muss aber auch ein iCP innerhalb der ersten 40 WLAN-Scans passiert werden. Gewählt wird jener iCP und Zeitpunkt mit dem geringsten Abstandswert zum Vektor $\overline{\mathrm{RSSI}_{\mathrm{iCPa}}}$ und frühesten Zeitpunkt.

Aus den ermittelten Differenzpositionen, die mit dem INS-Algorithmus bestimmt wurden, kann auf die Startposition rückgerechnet werden. Dabei wird von dem Zeitpunkt bzw. Schritt ausgegangen, an dem der iCP passiert wurde. In Abbildung 6-14 und Abbildung 6-15 sind Beispiele dargestellt, wie aus den GPS-Startpositionen und erkannten iCPs mit den erfassten Schritten auf die ermittelte Startposition geschlossen wurde.

In den Abbildungen sind als „Referenz-Startpunkt“ die tatsächlichen Startpunkte markiert. Mit „GPS-Startpunkt“ sind die GPS-Koordinaten zum Startzeitpunkt gekennzeichnet. Der erste iCP, der passiert wurde, ist jener iCP, der zum Rückrechnen auf die ermittelte Startposition, die als “Ermittelter-Startpunkt“ gekennzeichnet ist, benutzt wurde. Rückgerechnet wurde die Startposition mit den errechneten Positionsveränderungen durch den INS-Algorithmus. Genutzt werden die Positionsveränderungen vom Start des Testlaufes bis zum Zeitpunkt des Passierens des ersten iCPs. Dieses Prinzip ist in Algorithmus 6-3 als Pseudocode erläutert.

Algorithmus 6-3 findStartPosViaWIFIandGPS()

[startPos]=findStartPosViaWIFIandGPS (**var** testrun, iCP, par)

Eingabe: iCP, testruns par
Ausgabe: startPos
%testrun… Structure die alle Daten eines Testlaufes enthält
%iCP… Structure die alle Daten zu iCPs enthält
%par.k… Anzahl der nähesten iCPs die betrachtet werden.
%par.AnzScans… Anzahl an WLAN-Scans, die betrachtet werden, um nach dem ersten iCP zu suchen

```
% ermittelte die k nahestehen iCP zur Start GPS-Position
[kNNiCP]=findKNNiCP(testrun.GPS(1),par.k); % im EI-Testgebiet werden nur die Eingangs-iCPs betrachtet
for i=1:size(kNNiCP) % tue dies für alle k nahestehen
   RSSIiCP=iCP(kNNiCP(i)).RSSI; % lese RSSI-Vektor für iCP aus
   % ermittelt minimalen Euklidischen-Abstand und dessen Zeitpunkt des RSSI-Vektor iCP in Testlauf
    [minV(i) stepIDs(i)]=findCPinRun(testrun.RSSIs(1:par.AnzScans), RSSIiCP);
end
% sortiert Minima nach Größe und nach Zeitstemple Aufsteigend
iCPids=orderbyrelevance(minV, stepIDs);
FirstiCP= iCP(iCPids(1)); %wähle den ersten iCP – (der als erster passiert wurde)
CPstep= stepIDs(iCPids(1)); % erfasse Zeitpunkt bzw.- Schritt an dem der iCP passiert wurde

[~,~,steptimes,Xdiff,Ydiff]]=INSAlgorithmus(var testrun,~ ,par) % berechne Differenz-Positionen
% berechne aus iCP Koordinaten und Differenz-Positionen bis zu iCP Schritt die Startposition
startPos.X= FirstiCP.X-sum(Xdiff(1:CPstep));
startPos.Y= FirstiCP.Y-sum(Ydiff(1:CPstep));
```

Algorithmus 6-3 wurde für aufgezeichnete Testlauf-Datensätze erstellt. Um das Prinzip auf Echtzeitdaten anzuwenden, sind weitere Anpassungen nötig. Erstens müssten die Positionsveränderungen solange gepuffert werden, bis ein iCP passiert wurde. Des Weiteren würde es eine Weiterentwicklung des *findCPinRun()* erfordern, um das Erreichen eines iCPs in einer beliebigen Zeitspanne zu erkennen – z. B. durch die Einführung von Grenzwerten für die euklidischen Abstandswerte zu den RSSI-Vektoren der iCPs.

Erzielte Ergebnisse

In Tabelle 6-9 und Tabelle 6-10 sind die Abweichungen der berechneten Startpositionen den GPS-Startpositionen gegenübergestellt. Im Mittel aller Tests konnte die Startposition auf 5,0 m genau bestimmt werden. Wird der Testlauf H8 ausgenommen, bei dem vorher der

Kompass nicht kalibriert wurde, beträgt die mittlere Abweichung von der Referenz-Starposition 3,6 m. Mit GPS wich die Startposition im Mittel um 20,0 m ab.

Tabelle 6-9 Ermittlung Startpos. Heim-Testgebiet

Abweichung Startposition		
Bezeichnung der Testläufe	GPS-Startpos.	iCP-INS-Algo.
Testlauf H7	14,2 m	3,0 m
Testlauf H8	12,6 m	16,1 m
Testlauf H9	11,6 m	0,7 m
Testlauf H10	36,8 m	1,8 m
Mittelwert	***18,8*** m	***5,4*** m
MW ohne TL-H8	***20,9*** m	***1,8*** m

Tabelle 6-10 Ermittlung Startpos. EI-Testgebiet

Abweichung Startposition in Meter		
Bezeichnung der Testläufe	GPS-Startpos.	iCP-INS-Algo. II
Testlauf EI1	45,5 m	3,0 m
Testlauf EI2	37,4 m	11,3
Testlauf EI3	1,4 m	2,0 m
Testlauf EI4	20,4 m	1,5 m
Testlauf EI5	8,9 m	5,5 m
Mittelwert	***22,7 m***	***4,7 m***

Die Ergebnisse der durchgeführten Tests sind im **Anhang C** zu finden.

Im Mittel konnte mit dieser Methode die Startposition eines Testlaufes um 17,4 m genauer ermittelt werden, als mit dem GPS-Receiver. Damit der Ansatz funktioniert, muss ein iCP passiert und zum richtigen Zeitpunkt erkannt werden. Dann kann auf eine frühere Position zurückgerechnet werden. Die relativ große Abweichung bei der errechneten Startposition von TL-EI2 kommt daher zustande, dass der iCP-12 zu früh erkannt wurde, und zwar um 9 Schritte. Dies wurde bereits in Abschnitt 6.5.1 anhand von Abbildung 6-14 diskutiert. Mit iCP-1 und iCP-2, die nach der Eingangstüre im Windfang definiert sind, konnten die Startpositionen mit geringeren Abweichungen erkannt werden.

Zusammenfassung, Konklusion und Ausblicke zur Ermittlung der Startpostion

Der vorgestellte Ansatz, um den ersten iCP in einem durchgeführten Testlauf zu erkennen, bedient sich der in Abschnitt 6.5.1 vorgestellten Funktion `findCPinRun()` und verknüpft diese mit einfachen Bedingungen. Gewählt wird jener iCP, dessen RSSI-Vektor den geringsten Abstand in den Ersten 40 (`AnzScans`) erfassten RSSI-Vektoren des Testlaufes aufweist.

Der Algorithmus kann so nicht auf Echtzeitdaten angewandt werden und versagt, wenn der iCP nicht in der Zeit der ersten erfassten RSSI-Vektoren passiert wird. Die in Kapitel 6.5.1 im Abschnitt Zusammenfassung, Konklusion und Ausblicke diskutierten Weiterentwicklungen für die Funktion `findCPinRun()` würden es möglich machen, auf den Parameter `AnzScans` zu verzichten. So könnte z. B. ein iCP ab einem bestimmten Grenzwert des euklidischen Abstandswertes erkannt werden. Damit wäre auch eine Anwendung auf Echtzeitdaten möglich. Der Algorithmus muss dann solange RSSI-Vektoren und Daten für den INS-Algorithmus puffern, bis der erste iCP passiert wurde. Dann könnte wie beschrieben auf eine Startposition zurückgerechnet werden, wobei die Rückrechnung alter Positionen nicht immer notwendig ist. Für die Verwendung des in 6.5.4 beschriebenen Ansatzes wäre es ausreichend, den Moment des Betretens des Gebäudes und des Einganges zu erkennen, um mittels des iCP-INS-Algorithmus-II die Fortbewegung im Gebäude weiterzuverfolgen.

Soll die Navigation innerhalb eines Gebäudes oder eines Ortes gestartet werden, an dem kein GPS-Signal zur Verfügung steht, könnten alternativ die ungefähre Position über das GSM-Netz des Mobiltelefons abgefragt werden. Möglich wäre es auch, mit den iCPs und dem KNN-Fingerprinting-Algorithmus (Abschnitt 2.2.2) eine Position abzuschätzen. Dann könnte nach dem Passieren des ersten iCP die Startposition ermittelt werden.

6.5.3 iCP-INS-Algorithmus - Korrektur der Position durch iCPs

Wie in Abschnitt 6.4 gezeigt wurde, kann man mit dem Beschleunigungssensor und dem Kompass eines Smartphones auf Änderungen der Position schließen. Der beschriebe INS-Algorithmus benötigt eine Startposition und driftet mit jeder Positionsveränderung mehr von der tatsächlichen Position ab. Wie auf die Startposition geschlossen werden kann, wurde im letzten Abschnitt diskutiert. Ein Ansatz, wie die Abweichungen des INS-Algorithmus ständig korrigiert werden können, wird hier beschrieben.

Die in 6.5.1 beschriebene Funktion `findCPinRun()` bringt durch das Erkennen des Zeitpunktes des Passierens eines iCPs Zeit und Ort in Zusammenhang. Bei dem Ansatz soll diese Information genutzt werden, um eine mit dem INS-Algorithmus berechnete Position, falls nötig, zu korrigieren. Dabei wird der Zeitpunkt t des Passierens eines iCP mit der Funktion `findCPinRun()` festgestellt und die Position zur Zeit t auf die des iCP korrigiert. Es gilt jedoch die Einschränkung, dass zum Zeitpunkt t die mit dem INS-Algorithmus errechnete Position innerhalb des definierten Einflussbereiches des iCP sein muss. Diese Einflussbereiche sind in Abbildung 6-16 als gelbe Kreise gekennzeichnet.

Getestet wurde der Ansatz mit der in MATLAB geschriebenen Funktion `iCPINSAlgorithmus()` [37, DAAS\functions\iCP]. Der Algorithmus 6-4 erläutert diese Funktion als Pseudocode. Die Funktion wurde auf die aufgezeichneten Testläufe angewandt, für die Anwendung auf Echtzeit bedürfte es weiterer Anpassungen.

Der iCP-INS-Algorithmus (Algorithmus 6-4) verwendet den zuvor dargestellten Algorithmus 6-3 zum Ermitteln des Startpunktes. Mit dem in Algorithmus 6-2 gezeigten INS-Algorithmus werden die Differenzpositionen der Schritte errechnet. Aus Startposition und Differenzpositionen lassen sich die absoluten Positionen berechnen. Der Algorithmus durchläuft alle erkannten Schritte bzw. Positionen eines Testlaufes und ermittelt die sich in der Nähe befindenden iCPs.

Weiter erkennt der Algorithmus, wann der Einflussbereich eines iCP erreicht wird. Befindet sich der zuvor ermittelte Schritt, an dem der iCP möglicherweise passiert wurde, im Einflussbereich des iCP, wird dieser Schritt als Korrekturschritt verwendet. Dabei wird die Differenz zwischen errechneter Position und iCP-Position der Differenzposition bei diesem Schritt hinzugefügt. Diese „korrigierten Positionen“ sind in Abbildung 6-16 mit roten Linien markiert. Der Funktion `findCPinRun()` werden nur jene k iCPs (`kNNiCP`) des Testlaufes übergeben, die sich in der Nähe befinden und nicht zuletzt zur Korrektur verwendet wurden. Der iCP-INS-Algorithmus wird beendet, sobald alle Schritte bzw. Positionsveränderungen durchlaufen wurden.

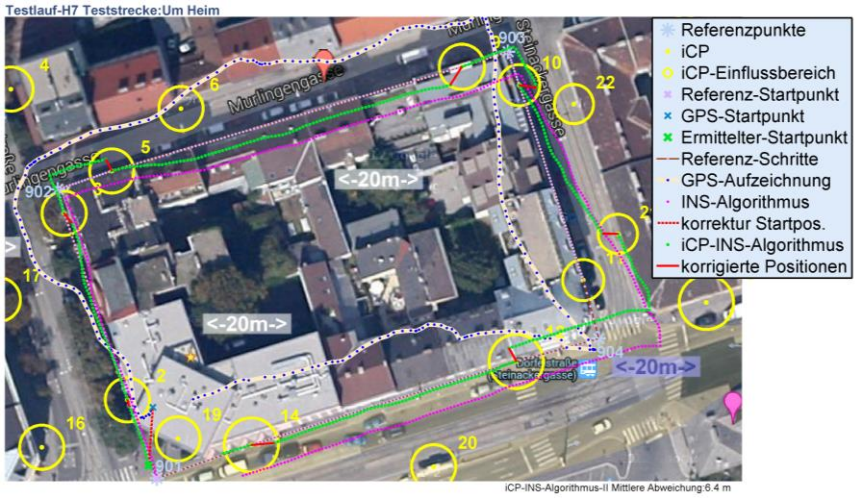
Algorithmus 6-4 iCPINSAlgorithmus ()

```
[X,Y, Xdiff, Ydiff] = iCPINSAlgorithmus(var: testrun, iCP, par)
Eingabe: GPS, RSSIs, scans, k
Ausgabe: X,Y, Xdiff, Ydiff
%testrun… Structure die alle Daten eine Testlaufes enthält
%iCP… Structure die alle Daten zu iCP enthält
%par… Structure mit Parameter
 1:   % findet den ersten iCP der passiert wurde und berechne
 2:   [startPos]=findStartPosViaWIFIandGPS (testrun, iCP, par);
 3:   % errechne mit INS-Algorithmus Differenz Positionen
 4:   [~,~,steptimes,Xdiff,Ydiff]]=INSAlgorithmus(testrun,startPos,par);
 5:   % definiere Startbedingungen
 6:   isPos=startPos;
 7:   step=1;
 8:   laststep=size(laststep);
 9:   wifiscans=testrun.RSSIs;
10:   while step<=laststep % durchlaufe alle Schritte
11:     kNNiCP=findKNNiCP(isPos,k); % erstelle Liste aller k nahestehen iCP
12:     for i<=size(kNNiCP,2)
13:   % Finde  Punkt an dem iCP – passiert wurde
14:       [minV(i) stepIDs(i)]=findCPinRun(testrun.RSSIs(step:laststep), RSSIiCP);
15:       // ermittle newPos aus starPos und Xdiff(1:stepIDs(i)),Ydiff(1:stepIDs(i))
16:   % Ermittle ob zu dem Zeitpunkt bzw. Schritt im Einflussbereich von iCP
17:         [PosInR, distanz]=isPosInR(iCP(kNNiCP(i)), newPos);
18:   % Wenn ermittelter Minimale Abstand von RSSI(t) zu iCP(i).RSSI in Einflussbereich
19:       if PosInR==1
20:         // korrigiere Xdiff(stepIDs(i)),Ydiff(stepIDs(i)) um Differenz zu Pos. von aktuellen ICP
 21         break;
22:       end
22:       step= stepIDs(i));
23:     end
24:     step++;
25:   end
26:   // ermittle neue X,Y aus startPos und Xdiff,Ydiff
```

Die zur Analyse verwendeten Testläufe wurden im Heim-Testgebiet durchgeführt und dienten zur Untersuchung der Anwendung des Konzeptes der iCPs im urbanen Raum. In Abbildung 6-16 ist ein Testlauf mit dem HTC um den Block des Heim-Testgebiets dargestellt. Die Ergebnisse der analysierten Testläufe sind in **Anhang C** 10.2 detailliert dargestellt. In Tabelle 6-11 sind diese Resultate zusammengefasst.

Der in Abbildung 6-16 dargestellte Testlauf H7 stellt die Ergebnisse der GPS-Aufzeichnung dem INS- und iCP-INS-Algorithmus gegenüber. Über die Referenzpunkte und iCPs sind die Teststrecken und Referenzschritte definiert (siehe Kapitel 6.1). Jeder Punkt in den Testläufen repräsentiert einen Schritt, mit dem eine Positionsveränderung einhergeht. Zu sehen ist auch die mithilfe des zuvor beschriebenen Ansatzes errechnete Startposition (grünes Kreuz). Diese weicht 3 m vom Referenz-Start ab, während mit GPS die Abweichung 14,2 m beträgt. Die erfassten Schritte des iCP-INS-Algorithmus sind durch grüne Kreise dargestellt. Die roten Markierungen repräsentieren die Korrekturen durch den iCP-INS- Algorithmus. Zum Vergleich ist der INS-Algorithmus durch violette Punkte dargestellt. Die mittlere Abweichung des INS-Algorithmus liegt bei 8,4 m. Mithilfe des iCP-INS- Algorithmus konnte diese Abweichung auf 6,4 m reduziert werden.

Abbildung 6-16 TL-H7 „Um Heim HTC“



Erzielte Ergebnisse

Tabelle 6-11 fasst die Ergebnisse zusammen, die mit dem iCP-INS-Algorithmus im Heim-Testgebiet erzielt wurden.

Tabelle 6-11 Zusammenfassung Vergleich iCP/INS Algorithmen mit GPS-Aufzeichnung

Vergleich iCP/INS Algorithmen mit GPS-Aufzeichnung											
Bezeichnung der Testläufe	Abweichungen in Meter										
	Mittlere			Endposition			Maximale			Startpos.	
	GPS	INS-Algo.	iCP-INS-Algo.	GPS	INS-Algo.	iCP-INS-Algo.	GPS	INS-Algo..	iCP-INS-Algo.	GPS	iCP-INS-Algo.
Testlauf H7 – HTC	10,1	8,4	6,4	17,4	17,2	14,2	19,4	17,9	17,6	14,2	3,0
Testlauf H8 – S2	9,3	37,2	31,0	2,3	52,2	41,3	22,9	52,2	41,3	12,6	16,1
Testlauf H9 – HTC	11,6	3,7	3,8	18,6	7,2	3,3	25,7	7,3	8,3	11,6	0,7
Testlauf H10 –S2	12,9	8,3	4,4	9,9	12,4	6,9	39,5	18,0	11,2	36,8	1,8
Mittelwert	***11,0***	***14,4***	***11,4***	***12,1***	***22,3***	***16,4***	***26,9***	***23,9***	***19,6***	***18,8***	***5,4***
MW ohne H8	***11,5***	***6,8***	***4,9***	***15,3***	***12,3***	***8,1***	***28,2***	***14,4***	***12,4***	***20,9***	***1,8***

Testlauf H8, beim dem der Kompass nicht kalibriert wurde, konnte mit dem iCP-INS-Algorithmus nicht korrigiert werden. Der Testlauf ist in Abbildung 6-4 dargestellt; durch die große Abweichung des Azimut-Wertes wurden die Einflussbereiche der iCPs nicht erreicht.

Blendet man Testlauf H8 aus, beträgt die mittlere Abweichung des iCP-INS-Algorithmus 4,9 m. Das ist um 1,9 m genauer, als die mit dem INS-Algorithmus erzielten Ergebnisse.

Vergleicht man die ermittelten Strecken mit den GPS-Aufzeichnungen, konnte der iCP-INS-Algorithmus diese im Durchschnitt um 6,6 m genauer erkennen. Die errechneten Startpositionen konnten im Schnitt auf 1,8 m genau erkannt werden. Bei den GPS-Koordinaten wich diese im Mittel um 20,9 m ab.

Aufgrund des bereits in 6.5.1 beschriebenen Problems der längeren Scan-Zeit des S2 wurden mit diesem Handy schlechtere Ergebnisse erzielt. Betrachtet man jedoch die Durchschnittswerte der maximalen Abweichungen, betragen diese bei der GPS-Aufzeichnung 28,2 m und beim iCP-INS-Algorithmus trotzdem nur 12,4 m. Die detaillierten Resultate der Testläufe sind in **Anhang C** in 10.2 protokolliert.

Zusammenfassung, Konklusion und Ausblicke zu iCP-INS-Algorithmus

Mit dem vorgestellten Ansatz war es möglich, die Präzision der errechneten Positionen des INS-Algorithmus im Durchschnitt um 1,9 m zu verbessern. Die Endposition konnte im Mittel um 4,2 m genauer berechnet werden. Die Verbesserungen wurden dadurch erreicht, dass beim Passieren der iCPs die errechneten Positionen immer wieder korrigiert wurden.

Die iCPs des Heim-Testgebietes wurden an Punkten definiert, die einer Wegentscheidung vorausgehen oder folgen und mit hoher Wahrscheinlichkeit passiert werden. Geachtet wurde bei der Auswahl auch darauf, dass diese Punkte sich aufgrund von strukturellen Gegebenheiten mittels WLAN-Fingerprinting gut unterscheiden ließen, wie in den vorigen Kapiteln (siehe z. B Tabelle 4-3 und Tabelle 5-9) dargestellt wurde. Daher wurden die iCPs z. B. an den Ecken der Häuser etwas nach innen versetzt angeordnet.

Weichen die Werte des digitalen Kompasses, wie beim Testlauf H8, zu sehr vom wahren Wert ab, versagt der Ansatz in dieser Form. Auch beim Testlauf H7 wird das Problem sichtbar, denn nachdem der Referenzpunkt 903 passiert wurde, weichen die Azimut-Werte des Orientierungssensors zu sehr vom tatsächlichen Wert ab. So scheint es, als würde die Straße überquert werden, obwohl der Testlauf entlang des Gehsteiges auf der rechten Seite der Straße verlaufen ist. Dadurch, dass nur der minimale Abstandswert und der Einflussbereich beachtet werden, wird der iCP-22 akzeptiert, obwohl dieser nicht passiert wurde. Der iCP-11, der tatsächlich passiert wurde, wird nicht erkannt. Für einen stabileren Algorithmus müsste dieser Ansatz weiter verbessert werden. Auch hier wäre zu untersuchen, ob dies durch die Einführung eines Grenzwertes für den minimalen Abstandswert zu verhindern wäre.

Der erläuterte iCP-INS-Algorithmus ist nicht direkt auf Echtzeitdaten übertragbar. Der Algorithmus müsste Differenzpositionen seit dem Passieren des letzten iCP puffern. Damit lässt sich die neue Position errechnen, wenn der nächste iCP erkannt wurde. Außerdem müssten die RSSI-Vektoren gepuffert werden. Der Zeitpunkt des minimalen Abstandes zum RSSI-Vektor des iCP kann erst beim Verlassen des Einflussbereichs des iCP berechnet werden. Die Position könnte erst etwas zeitverzögert zum Passieren korrigiert werden.

6.5.4 iCP-INS-Algorithmus-II

Der in diesem Abschnitt präsentierte iCP-INS-Algorithmus-II stellt eine Weiterentwicklung des iCP-INS-Algorithmus dar. Ein wesentlicher Unterschied ist, dass dieser Algorithmus die in Kapitel 5.3 vorgestellte logische Abfolge der iCPs berücksichtigt. Außerdem werden die iCPs intelligenter. So wird beim Passieren bestimmter iCPs die Schrittweite auf den Abstand der Stiegen korrigiert und die Azimut-Werte kalibriert. Die Korrektur der Position wird bei diesem Algorithmus auf die vorangegangen Schrittpositionen stetig verteilt, um ein kontinuierliches Abbild der Strecke zu erhalten. Der iCP-INS-Algorithmus-II ist als Pseudocode in Algorithmus 6-5 beschrieben.

In Abbildung 6-17 und Abbildung 6-18 sind die Ergebnisse zweier Testläufe im EI-Testgebiet abgebildet. Testlauf-EI7 befindet sich im 3. Stock und ist die Fortführung von Testlauf-EI3, dessen Endpunkt als Startpunkt für TL-EI7 verwendet wurde.

In Abbildung 6-17 ist der Eingangsbereich des Gebäudes, Lifte und das Stiegenhaus-1 zu sehen. Die Strecke des dargestellten Testlaufes hat ihren Startpunkt außerhalb des Gebäudes. Betreten wird das Haus bei Eingang-1, der Endpunkt des Testlaufes ist vor dem

Lift-2. Neben der Referenzstrecke sind die GPS-Aufzeichnungen und die errechneten Strecken des INS- und iCP-INS-Algorithmus-II eingetragen. Wie in der Abbildung zu sehen ist, sind die Daten der GPS-Aufzeichnung für eine präzise Erfassung der Strecke unbrauchbar. Der INS-Algorithmus benötigt die Koordinaten des Startpunktes und driftet bis zur Endposition um 3,5 m ab. Der iCP-INS-Algorithmus-II erkennt den richtigen Eingang und dass sich die letzte Position des Testlaufes auf der des iCP-6 befindet. Die mit dem in 6.5.2 beschriebenen Ansatz ermittelte Startposition weicht um 1,9 m ab. Die detaillierten Ergebnisse des Testlaufes sind in **Anhang C** in 10.3.3 festgehalten.

Der 3. Stock des EI-Testgebiets ist in Abbildung 6-18 zu sehen, der darin dargestellte Testlauf-EI7 führt den Testlauf-EI3 im 3. Stock fort. Die Strecke verläuft von Lift-2 entlang Gang1 zum Zeichensaal-1 (Raum 334). Ein GPS-Signal war für diesen Testlauf nicht verfügbar. Der INS-Algorithmus weicht von der Referenzstrecke aufgrund des verfälschten Richtungswerts ab, diese Abweichung wurde bereits in 6.3 diskutiert. Mit dem iCP-INS-Algorithmus-II wurden diese Abweichungen korrigiert und der richtige Raum konnte erkannt werden. Die Ergebnisse des Testlaufes sind im Detail in **Anhang C** in 10.3.7 dargelegt.

Abbildung 6-17 TL-EI3

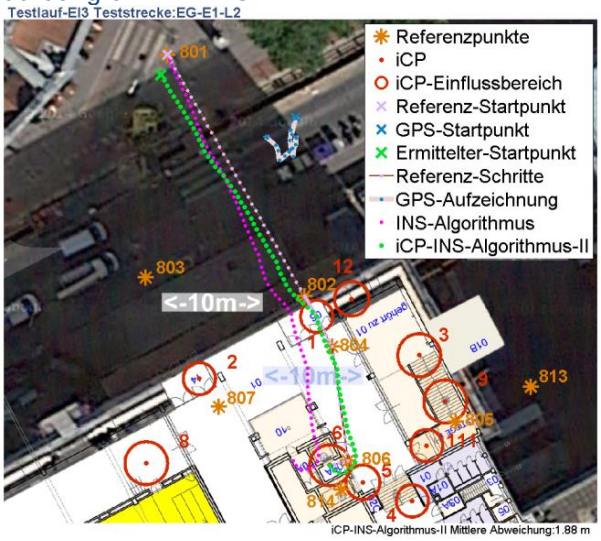
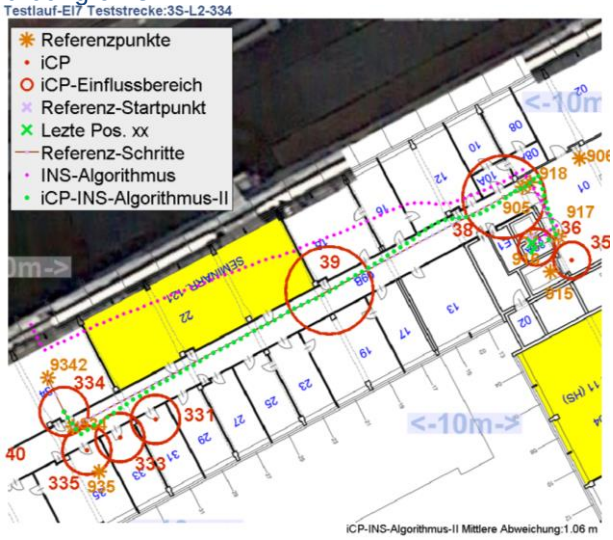


Abbildung 6-18 TL-EI7



Erkennen des Einganges und Berechnung der Startposition

Beim iCP-INS-Algorithmus-II wird der erste iCP, der passiert wird, dazu verwendet, den Startpunkt zu berechnen. Bei dieser Variante werden nur die iCPs verwendet, welche die Eingänge repräsentieren. Aus den GPS-Koordinaten werden die *k* Eingänge ermittelt, die sich in der Nähe befinden. Dies ist in der Funktion `Findentrance()` [37, DAAS\functions\iCP] implementiert, die in Algorithmus 6-5 Anwendung findet. Betrachtet wird eine bestimmte Anzahl von anfänglichen Schritten, wie bei Algorithmus 6-3. Ausgewählt wird jener iCP, dessen RSSI-Vektor den kleinsten euklidischen Abstand zu den ermittelten RSSI-Vektoren bildet. Der entsprechende Wert und Zeitpunkt wird mit der Funktion `findCPinRun()` berechnet. Wie bereits in 6.5.2 gezeigt, konnten im Mittel die Startpositionen so auf 4,7 m genau erkannt werden.

Korrektur der Orientierung

Wie in 6.3 besprochen wurde, kann der erhaltene Richtungswert, der den Azimut darstellt, in Gebäuden aufgrund von elektromagnetischen Feldern stark vom wahren Wert abweichen. Dieser Ansatz geht davon aus, dass aufgrund von baulichen Gegebenheiten manche iCPs

nur in einer bestimmten Richtung passiert werden können und diese Information genutzt werden kann, um einen Korrekturwert zu berechnen. Ein Beispiel sind die iCPs an den Eingängen und auf den Gängen, die normalerweise nach einer bestimmten Richtung passiert werden.

Für die entsprechenden iCPs wurden diese vorgegebenen Richtungen bzw. Sollwerte für die Richtung definiert, sowie eine Anzahl von Schritten, für die diese Richtungen vor und nach den iCPs ebenfalls gelten. Dadurch ist die Anpassung von mehreren Schritten abhängig. In **Anhang A** in Tabelle 8-6 und Tabelle 8-12 sind diese iCPs, die Sollwerte der Richtungen und die Anzahl der Schritte definiert.

Abbildung 6-19 und Abbildung 6-20 stellen die ermittelten Strecken mit (iCP-INS-Algorithmus-II bzw. grüne Strecke) und ohne Korrektur (INS-Algorithmus bzw. violette Strecke) der Richtung gegenüber.

Abbildung 6-19 TL-EI3 Korrektur der Orientierung

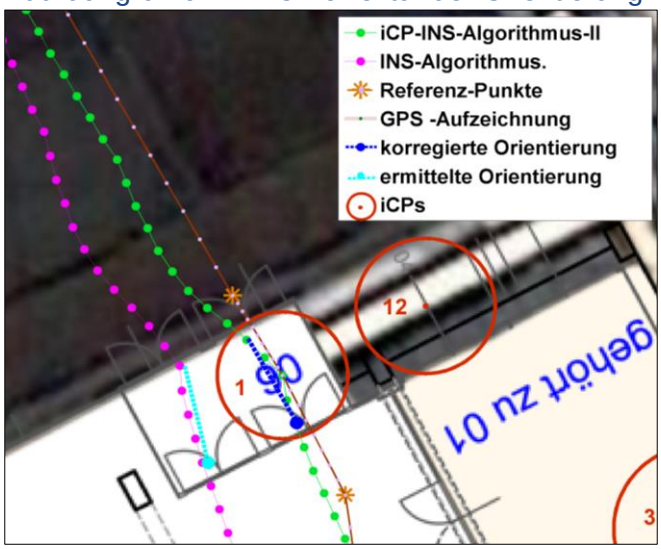
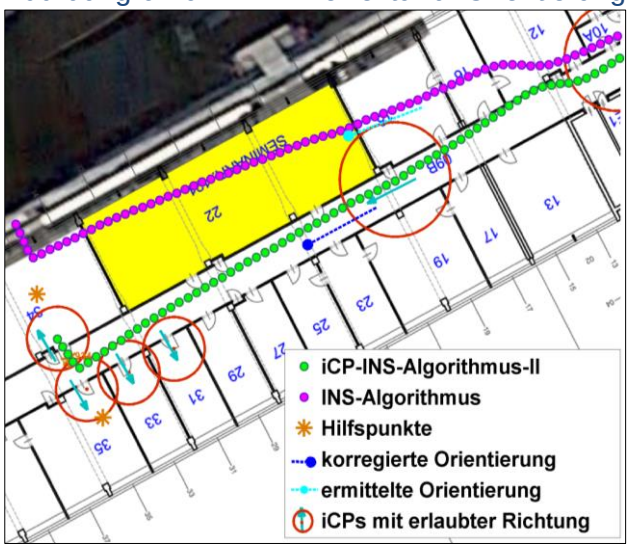


Abbildung 6-20 TL-EI7 Korrektur d. Orientierung



In Abbildung 6-19 sind die ermittelten Richtungen und die entsprechenden Schritte des INS-Algorithmus Türkis dargestellt. Aus den Sollwerten, als blaue Pfeile eingezeichnet, wird für die erfassten Richtungen ein Korrekturwert berechnet. Handelt es sich bei dem iCP um den Eingang, so wird der Korrekturwert für das Zurückrechnen der Startposition verwendet. Der Korrekturwert wird so lange beibehalten, bis ein neuer Korrekturwert bei einem anderen iCP errechnet wird. In Abbildung 6-20 wird der iCP-39 am Gang genutzt, um einen neuen Korrekturwert für die Orientierung zu erhalten. Bei Algorithmus 6-5, der den iCP-INS-Algorithmus-II erläutert, werden in Zeile 8 und 26 der Korrekturwert für die Orientierung berechnet und angewandt.

Anpassung der Schrittweite

Der INS-Algorithmus, der als Basis für diesen Ansatz dient, beruht darauf, die Veränderungen einer Position jeweils mit einem Schritt durchzuführen. Es wurde dabei davon ausgegangen, dass jeder Schritt mit der gleichen Schrittweite ausgeführt wurde. Neben den schon diskutierten etwas unterschiedlichen Schrittweiten für Indoor und Outdoor, kann sich die Schrittweite aufgrund von baulichen Vorgaben, wie etwa Treppen, ändern. Bei Treppen ist die Weite der Schritte durch den Abstand der Treppen definiert. Daher passt der iCP-INS-Algorithmus-II diese Schrittweite dort an.

Vergleicht man in Abbildung 6-21 den INS-Algorithmus mit dem iCP-INS-Algorithmus-II, so ist zu sehen, dass, nachdem der iCP-3 passiert wurde, die Schrittweite auf 0,27 m für 16 Schritte lang angepasst wurde. Dies entspricht dem Treppenabstand und der Anzahl an Stufen bis zum iCP-9. Wird dieser passiert, passt der iCP-9 ebenfalls die Schrittweite für 5 weitere Stufen an. Vergleicht man die Strecke, die mit dem INS-Algorithmus ermittelt wurde, so ist zu sehen, dass die Schrittweite im Bereich der Stiege zu weit wäre. Diese Anpassung wird im Pseudocode im Algorithmus 6-5 in Zeile 30 und 31 vorgenommen.

Abbildung 6-21 TL_EI4 Schrittgröße anpassen

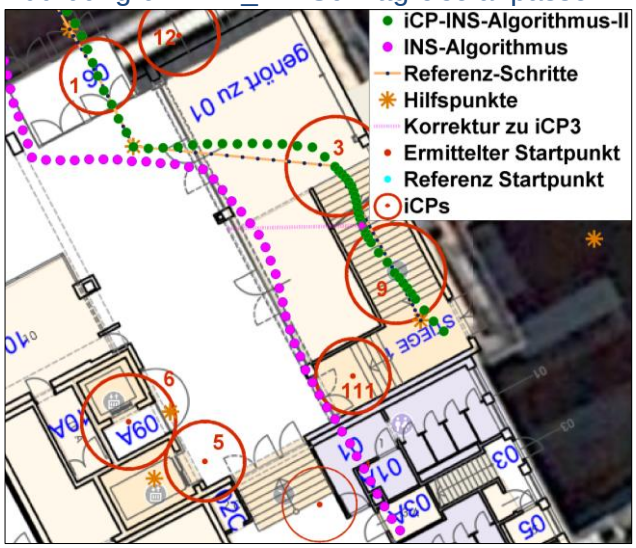
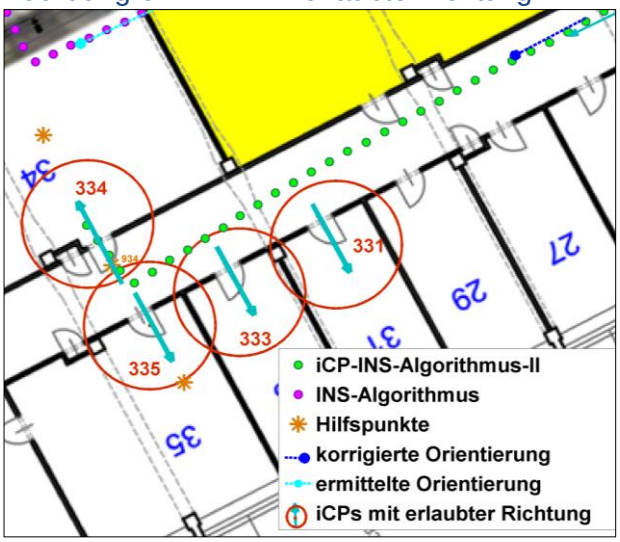


Abbildung 6-22 TL-EI7 erlaubte Richtung



Erlaubte Richtungen

Für die Vermessung der iCPs wurden 4 Richtungen definiert, in denen jeweils RSSI-Vektoren ermittelt wurden. Diese Richtungen sind in **Anhang A** in Abbildung 8-2 und Abbildung 8-3 abgebildet und in Tabelle 8-15 definiert. In 5.1.1 wurde gezeigt, wie mittels Einteilung der Orientierungswerte in vier Bereiche diese Richtungen erkannt werden können. In dem in MATLAB programmierten Ansatz wird auf diese Weise jedem Schritt eine der vier Richtungskategorien zugewiesen. Dies wird von dem iCP-INS-Algorithmus-II verwendet, um weitere Einschränkungen für das Erkennen eines iCP zu machen. Wie in Abbildung 6-22 zu sehen ist, müssen die Türen zu den Büros und zu Zeichensaal-1 nach bestimmten Richtungskategorien durchschritten werden. Das hat den Sinn, dass iCPs nicht erkannt werden, wenn man nur den Gang entlang geht. Daher wird die Einschränkung gemacht, dass die iCPs 335, 333 und 331 nur in Richtung-1 passiert werden können und der iCP-334 nur in Richtung-3. Diese Bedingung wird in Zeile 23 im Pseudocode gestellt. Wenn man sich in dem Raum befindet, müsste natürlich das Gegenteil gelten.

Erzielte Ergebnisse mit dem iCP-INS-Algorithmus-II

Die mit dem iCP-INS-Algorithmus-II erzielten Ergebnisse sind für die Testläufe im EI-Testgebiet in Tabelle 6-12 bis Tabelle 6-13 zusammengefasst.

Die im Eingangsbereich durchgeführten Testläufe sind in Tabelle 6-12 dargestellt. In ihr werden die Resultate der GPS-Aufzeichnung, des INS- und iCP-INS-Algorithmus-II verglichen. Betrachtet werden mittlere und maximale Abweichung sowie die Abweichung der End- und Startposition.

Wie der Tabelle zu entnehmen ist, wichen die mit GPS ermittelten Koordinaten im Mittel um 19,8 m ab. Die mit dem INS-Algorithmus ermittelte Strecke konnte im Durchschnitt auf 4,1 m genau berechnet werden. Mit dem iCP-INS-Algorithmus-II konnte die höchste Genauigkeit erzielt werden, dieser wich im Mittel um 2,3 m von den Referenzpositionen ab.

Die Startposition kann nur mit der ersten GPS-Position des Testlaufes und der des iCP-INS-Algorithmus-II verglichen werden. Beim INS-Algorithmus muss diese bekannt sein, da von ihr aus die Berechnung beginnt. Die Ergebnisse wurden bereits in Abschnitt 6.5.2 diskutiert.

Tabelle 6-12 Zusammenfassung iCP/INS Algorithmen-II mit GPS-Aufz. EI-Testgebiet EG

Vergleich iCP/INS Algorithmen mit GPS-Aufzeichnung											
Bezeichnung der Testläufe	Abweichungen in Meter										
	Mittlere			Endposition			Maximale			Startpos.	
	GPS	INS-Algo.	iCP-INS-Algo. II	GPS	INS-Algo.	iCP-INS-Algo. II	GPS	INS-Algo..	iCP-INS-Algo. II	GPS	iCP-INS-Algo. II
Testlauf EI1	34,5	2,9	2,3	36,6	3,5	2,2	45,7	4,3	4,5	45,5	3,0
Testlauf EI2	20,7	6,0	4,0	25,8	4,0	2,0	37,4	10,0	11,3	37,4	11,3
Testlauf EI3	16,4	3,2	1,9	31,3	3,5	0,0	31,3	6,4	4,8	1,4	2,0
Testlauf EI4	12,8	4,3	1,5	22,6	8,5	1,0	22,6	8,5	4,7	20,4	1,5
Testlauf EI5	14,7	4,2	1,9	32,2	5,9	1,1	32,2	6,4	5,5	8,9	5,5
Mittelwert	***19,8***	***4,1***	***2,3***	***29,7***	***5,1***	***1,3***	***33,8***	***7,1***	***6,2***	***22,7***	***4,7***

Betrachtet man, mit welcher Abweichung die Endposition berechnet wurde, ist zu sehen, dass diese beim iCP-INS-Algorithmus-II im Durchschnitt auf 1,3 m genau erkannt wurde. Die letzte ermittelte GPS-Koordinate des Testlaufes wich hingegen im Mittel um 29,7 m ab. Diese starke Abweichung hängt damit zusammen, dass alle Endpunkte im Gebäude waren und die durch das GPS-Modul ermittelten Koordinaten dort hohe Abweichungen aufweisen. Die Details zu den Resultaten der Testläufe sind in **Anhang C** in Abschnitt 10.3.1 bis 10.3.8 dokumentiert.

Die in Tabelle 6-13 zusammengefassten Testläufe, die im 3. Stock des EI-Testgebietes erfasst wurden, stellen INS- und iCP-INS-Algorithmus-II gegenüber. Bei diesen Testläufen war GPS nicht verfügbar. Mit einer mittleren Abweichung von 1,3 m lieferte der iCP-INS-Algorithmus-II im Durchschnitt um 3,4 m genauere Positionen als der INS-Algorithmus. Um 4,7 m unterscheidet sich im Mittel die Genauigkeit der berechneten Endpositionen. Zu beachten ist, dass beim INS-Algorithmus von der tatsächlichen Startposition ausgegangen wurde. Würde man von den Endpositionen der Testläufe im Eingangsbereich ausgehen, wäre diese Abweichung noch größer.

Tabelle 6-13 Zusammenfassung iCP/INS Algorithmen - EI-Testgebiet 3. Stock

Vergleich iCP/INS Algorithmen						
Bezeichnung der Testläufe	Abweichungen in Meter					
	Mittlere		Endposition		Maximale	
	INS-Algo.	iCP-INS-Algo. II	INS-Algo.	iCP-INS-Algo. II	INS-Algo..	iCP-INS-Algo. II
Testlauf EI6	4,8	1,5	8,7	2,1	9,2	2,7
Testlauf EI7	3,8	1,1	4,6	3,1	6,9	3,7
Testlauf EI8	5,5	1,3	6,5	0,5	8,0	3,4
Mittelwert	***4,7***	***1,3***	***6,6***	***1,9***	***8,0***	***3,3***

Algorithmus 6-5 iCPINSAlgorithmusII()

iCPINSAlgorithmusII(**var:** iCP, testrun, startPos, par,layer)

```
Eingabe: iCP, testrun, startPos, par,layer
Ausgabe: X,Y
%testrun… Structure, die alle Daten eines Testlaufes enthält
%iCP… Structure die alle Daten zu iCP enthält
%par… Structure mit Parameter
%layer… Abschnitt in dem gestartet wird. layer=0 sind die Eingänge
 1:  % Als Startposition werden GPS-Koordinaten bei Start des Testlaufes
 2:  testrun.GPS(1)=startPos;
 3:  % Mit INS-Algorithmus werden Differenz Positionen ermittelt
 4:  [~,~,steptimes,Xdiff,Ydiff]]=INSAlgorithmus(testrun,startPos,par);
 5:  kNNiCP=findKNNiCP(isPos, par.k ,layer); % Ermittle die iCP der Eingänge in der Nähe
 6:  % ermittelte welcher Eingang benutzt wurde durch ermittelt des ersten Minima der iCP
 7:  [entranceiCP, stepEingang,ScanId]=findentrance(testrun,ICP,layer,kNNiCP,par);
 8:  // korrigiere Winkel bei Eingang par.winkelkorr
 9:  // berechne startPos.
10:  step=stepEingang;
11:  while step<=laststep
12:      // layer++ bzw. ermittle nächste mögliche iCP aus Logik
13:      % Untersuche mögliche iCP – Welcher wurde passiert? – Was muss korrigiert werden?
14:      for i<=size(kNNiCPbyrel,2)
15:          % ermittle Minima zu möglichen iCP
16:          [minV(i) stepiCP(i)]=findCPinRun(wifiscans (step: stepout), iCP(kNNiCP(i)).RSSI);
17:          // berechen newPos mit stepiCP(i) und Xdiff, Ydiff
18:          % Abfrage ob newPos im Einfluss Bereich des iCP
19:          [PosInR, distanz]=isPosInR(iCP(kNNiCP(i)).Pos,newPos);
20:          // berechne Richtung zu Schrittzeitpunk newStep
21:          // ermittle erlaubte Richtungen für iCP
22:          % Wenn die neue Position in Einflussbereich (R) und Richtung erlaubt
23:          if PosInR=1 und Richtung erlaubt
24:          // Berechne aus newPos und der dem iCP nähesten Pos. Aus Xdiff,Ydiff eine mittlere Position
25:          // verteile diese Differenz dX und dY auf alle Xdiff,Ydiff bis zu letzten iCP
26:          end
27:
28:          // Wenn o ICP Azimut Korrekturwert hat.
29:            // Falls ja, berechne Differenz der Orientierung und setze p.kal=diffOrientierung
30:          // Wenn ab ob iCP Schrittgrößen Einschränkung
31:            // Falls ja, knorrigere Xdiff, Ydiff mit Schrittgöße und Anzahl vor/nach iCP
32:          step=newStep; % ;
33:      end
34:  step++;
35:  end
```

Betrachtet man alle Testläufe, wich der iCP-INS-Algorithmus-II um 1,9 m und der INS-Algorithmus um 4,3 m von den Positionen der Referenzschritte ab.Die Endposition konnte vom iCP-INS-Algorithmus-II durchschnittlich auf 1,5 m und mit dem INS-Algorithmus auf 5,7 m genau erkannt werden.

Konklusion und Ausblicke

Wie an den Resultaten der Testläufe zu sehen ist, konnten mit dem vorgestellten Ansatz gute Ergebnisse bei der Indoor-Navigation erzielt werden. Bei den Testläufen wurden alle iCPs korrekt erkannt und so war es möglich, für alle Testläufe die korrekte Endposition bzw. den richtigen Raum zu erkennen. Der vorgestellte iCP-INS-Algorithmus-II konnte durch das Korrigieren der Positionen das Abdriften des INS-Algorithmus verhindern.

Der iCP-INS-Algorithmus-II zeigt einen Ansatz, wie Daten von GPS, Sensoren und WLAN-Moduls kombiniert werden können, um diese zusammen zur Navigation zu nutzen. Gezeigt wurde auch, wie ortsgebundene Informationen genutzt werden können, um die Navigation weiter zu verbessern. So wurden die Schrittweiten bei den Treppen angepasst und die Ausrichtung der Eingänge und des Ganges für Richtungskorrekturen berücksichtigt. So konnten Abweichungen in Richtung und Schrittweite ausgeglichen werden.

Für den Ansatz ist es wesentlich, dass erkannt wird, wann welcher iCP passiert wurde. Die Funktion `findCPinRun()` ist eine einfache Methode, um den Zeitpunkt des Passierens zu erkennen. Welcher iCP erkannt wird, ist davon abhängig, dass der richtige Eingang erkannt wird. Anhand der logischen Sequenz, den Einflussbereichen und den Richtungseinschränkungen, wird das passieren eines iCP erkannt.

Wird ein falscher iCP gewählt, würde der Ansatz in dieser Form versagen. Daher wäre eine Weiterentwicklung vorstellbar, die auf der Berechnung der Wahrscheinlichkeit beruht, wann welcher iCP passiert wurde. Dazu müssten alle bereits genutzten Informationen, wie errechnete Positionen, minimaler euklidischer Abstand zu RSSI-Vektor des iCP und logische Abfolge genutzt werden, um einen Wahrscheinlichkeitswert zu errechnen.

Weiter könnte für diesen auch die Richtung beim Passieren der iCPs berücksichtigt werden. Wie schon besprochen, können bestimmte Punkte nur in einer bestimmten Richtung durschritten werden. Eventuell könnten auch Daten des Beschleunigungssensors genutzt werden, um zu erkennen, ob ein Lift oder eine Treppe gewählt wurde. Damit könnten z. B. die iCP-3 und iCP-5 besser unterschieden werden. Einige Handys verfügen über ein eingebautes Barometer, womit der Wechsel in ein Stockwerk erfasst werden könnte. All diese Informationen könnten genutzt werden, um eine Wahrscheinlichkeit zu berechnen, ob und wann ein iCP passiert wurde.

Die in MATLAB erstellte Funktion verwendet die Daten der aufgezeichneten Testläufe und müsste für Echtzeitdaten angepasst werden. Mögliche Anpassungen wurden bereits in den vorigen Abschnitten diskutiert. Grundsätzlich müssen die Daten zwischen dem Passieren von zwei iCPs gepuffert und die Positionen stetig nachträglich angepasst werden.

7 Zusammenfassung und Ausblick

7.1 Zusammenfassung

Die Frage, wie sich die Daten des GPS-Receivers, WLAN-Moduls und der Sensoren eines Smartphones kombinieren lassen, um höhere Genauigkeiten bei der Indoor und Outdoor Positionierung zu erzielen, gab den Anstoß zu dieser Arbeit.

So entwickelte sich die Idee für den in **Kapitel 1** vorgeschlagenen und in der Arbeit untersuchten Ansatz der intelligenten Checkpoints (iCPs). Bei diesem Ansatz sollen die wegentscheidenden Positionen mittels WLAN-Fingerprinting erkannt werden. Die iCPs sollen genutzt werden, um bereits berechnete Positionen zu korrigieren und so eine höhere Genauigkeit zu erreichen. Außerdem sollen der Beschleunigungs- und Orientierungssensor zum Einsatz kommen, um die Positionsveränderung zwischen den iCPs kontinuierlich zu erfassen.

Von den in **Kapitel 2** dargelegten Positionierungstechniken erwies sich ein WLAN-Fingerprinting Ansatz, der Nearest-Neighbor-Algorithmus, für das Erkennen der iCPs am vielversprechendsten. Der NN-Algorithmus eignet sich, um eine zuvor untersuchte Position anhand der gemessenen Signalstärken (RSS) der Access Points (APs) eines WLAN-Scans wiederzuerkennen.

Um von einer bekannten Position aus Veränderungen dieser zu erfassen und so zu navigieren, bietet sich ein auf einem Inertial Navigation System (INS siehe 2.1.4) basierender Ansatz an. Daher war ein weiteres Ziel der Arbeit, zwischen den mittels Fingerprinting erkannten iCPs mit einem INS-Algorithmus zu navigieren.

Damit mögliche Methoden und Ansätze getestet, analysiert und miteinander verglichen werden konnten, bestand ein Teil der Arbeit darin, eine Test- und Simulationsumgebung zu erstellen. Dieses Datenerfassungs- und Analysesystem (DAAS) wurde in **Kapitel 3** beschrieben. Das DAAS besteht aus zwei Teilkomponenten: aus einer eigens programmierten Android-Applikation (App), nämlich der CPS-App (Combined Positioning System siehe 3.2), mit der die Daten erfasst wurden, und dem MATLAB-Framework (MFW siehe 3.3), mit dem die erfassten Daten importiert und analysiert werden konnten. Mit dem DAAS können auch in Zukunft alle für die Positionierung verwendbaren Daten aufgezeichnet und für Simulationen verwendet werden.

Mit der CPS-App können zwei Arten von Datensätzen erfasst werden: WLAN-Scans und Testläufe. Die WLAN-Scans werden auf einer bestimmten Position erstellt. Dabei werden die empfangenen Signalstärken der Access Points (APs) sowie der Azimut-Wert des Orientierungssensors erfasst. Zur Aufzeichnung der Testläufe muss eine Teststrecke abgegangen werden. Die dabei auftretenden Sensordaten, GPS-Koordinaten und WLAN-Scans werden abgespeichert. Bei den Testläufen kann das Erreichen von Referenzpunkten manuell festgehalten werden, um eine Referenzstrecke zu erhalten.

Für die grafische Darstellung von Testläufen ist es möglich, Bilder von Karten in das MFW zu laden und einzupassen. Berechnete Positionen können so auf diesen Karten eingetragen und ganze Testrecken und errechnete Strecken von Testläufen dargestellt werden.

Der Aufbau und die Funktionen des DAAS wurde in **Kapitel 3** beschrieben und die Struktur der Datensätze erläutert. So kann das System in Zukunft für eine Reihe von weiteren Analysen und Simulationen genutzt werden bzw. kann das System einfach erweitert werden.

In **Kapitel 4** ist dokumentiert, wie das DAAS genutzt wurde, um verschiedene Varianten des NN-Algorithmus für WLAN-Fingerprinting zu analysieren. Dabei wurden mithilfe der CPS-Apps auf zwei Testgeräten (HTC EVO 3D und Samsung Galaxy S2) in zwei Testgebieten (EI-Testgebiet und Heim-Testgebiet) Daten erfasst. Getestet und analysiert wurde der NN-Algorithmus im MFW.

Der benutzte NN-Algorithmus (siehe auch 4.4.1) verwendet die Signalstärken der APs als Indikatoren (RSSI), um RSSI-Vektoren (4.3) zu erstellen. Berechnet werden die euklidischen Abstände zwischen den RSSI-Vektoren, um den „nächsten Nachbarn“ bzw. die „ähnlichsten“ RSSI-Vektoren und somit deren Positionen zu ermitteln.

Um die RSSI-Vektoren für eine Position zu definieren, wurden zwei Mittelwert-Berechnungsmethoden (4.3.3), das arithmetische Mittel (mean) und der Median (median) untersucht. Außerdem wurde analysiert, wie mit abwechselnd erreichbaren APs an einer Position umgegangen werden kann, indem ein Minimalwert (-101dBm) oder der NaN-Wert zum Ignorieren dieser verwendet wurde. Untersucht wurden Testreihen zu verschiedenen Bedienungen, mit jeweils drei Szenarien, bei denen die Daten der Handys, also des S2 (*„S2-DB“*) und des HTC (*„HTC-DB“*) erst separat und dann kombiniert (*„kombinierte-DB“*) betrachtet wurden.

Zwischen dem arithmetischen Mittel und dem Median als Mittelwert konnte kein signifikanter Unterschied festgestellt werden. Vergleicht man die besten Erkennungsraten (ERn) der untersuchten Szenarien, konnten mit dem arithmetischen Mittel als Basis für die RSSI-Vektoren etwas höhere Raten erzielt werden. Beim EI-Testgebiet lag der Unterschied bei 0,2 Prozentpunkten (PP) und im Heim-Testgebiet bei 2,9 PP (4.5.1).

Bei der ersten Testreihe (4.5.1) wirkte sich die Verwendung von Minimalwerten bei nicht erreichten APs im EI-Testgebiet geringfügig negativ auf die ER aus, im Heim-Testgebiet konnten die ERn erhöht werden. Eine eindeutige Aussage, welche Strategie zu besseren Ergebnissen führt, konnte daher nicht gemacht werden. Ein Zusammenhang könnte zwischen der Verfügbarkeit von APs stehen, da kaum verfügbare APs durch die Minimalwerte anders gewichtet werden.

In einer weiteren Testreihe (4.5.2) wurden von den APs im EI-Testgebiet, die Multi-SSID bereitstellen, jeweils nur der letzte der mehrfachen RSS-Werte eines APs eines WLAN-Scans für die Berechnung verwendet. So konnte bei den kombinierten Daten der zwei Handys die beste Erkennungsrate um 8,3 PP erhöht werden. Im Mittel über alle betrachteten Szenarien konnten die ERn um 8 PP verbessert werden. Bei dieser Testreihe mit den selektierten RSS-Werten konnte beim Einsatz von Minimalwerten für nicht erreichte APs bei dem Szenario *„kombinierte-DB“* eine um 21,3 PP höhere ER als mit allen erfassten RSS-Werten erzielt werden. Eine Gegenüberstellung ist in Tabelle 4-1 und Tabelle 4-5 zu sehen.

Für die Szenarien, bei welchen die Daten der Handys separat betrachtet wurden, konnten grundsätzlich höhere ERn erreicht werden, als bei einer Kombination dieser. Im EI-Testgebiet lag beispielsweise bei der ersten Testreihe die beste ER für das Szenario *„S2-DB“* bei 72,5 % und für *„HTC-DB“* bei 69,4 %. Mit den kombinierten Daten lag die beste ER bei 55,5 %.

Diese mehrfachen RSS-Werte der Multi-SSID APs kommen einer zufälligen Gewichtung von RSS-Werten gleich. In Kapitel 4.5.4 wurde nach einem Gewichtungsvektor für die einzelnen Summanden der RSS-Werte für die Berechnung des euklidischen Abstandes gesucht, der zu einer Verbesserung der Erkennungsraten führt. Durch die Verwendung eines Gewichtungsvektors konnten die besten Erkennungsraten im EI-Testgebiet im Mittel um ca. 5 PP verbessert werden. Im Heim-Testgebiet konnten im Mittel ca. 2 PP höhere Erkennungsraten erzielt werden.

Durch die Selektion mehrfacher RSS-Werte und den Einsatz eines Gewichtungsvektors und den Minimalwerten für nicht erreichte APs konnten für die kombinierten Daten der Handys die ERn von 55,5 % auf 70,8 % erhöht werden. Im Heim-Testgebiet konnten von Beginn an höhere ERn erzielt werden; hier konnten nur geringfügige Verbesserungen erreicht werden.

In 4.5.3 sind die Ergebnisse zu zwei weiteren Varianten des NN-Algorithmus zu finden, welche in 4.4.3 erläutert wurden. Bei diesen Algorithmen werden alle erfassten RSSI-Vektoren der Kalibrierungsphase als Fingerprinting-Datenbank verwendet, es werden also keine Mittelwerte berechnet. Für die Positionierung wird der zu untersuchende RSSI-Vektor mit allen erfassten Vektoren in der Datenbank verglichen. Ausgewählt wird eine bestimmte Anzahl von k der am besten übereinstimmenden RSSI-Vektoren in der Datenbank. Beim *MFV-Algorithmus* wird jene Position gewählt, welche am häufigsten durch deren RSSI-Vektoren repräsentiert wurde. Beim *Likelihood-Algorithmus* werden den möglichen Positionen jeweils Wahrscheinlichkeiten zugewiesen; die wahrscheinlichste Position wird dann gewählt.

Mit dem untersuchten MFV-Algorithmus sowie dem Likelihood-Algorithmus konnten keine besseren Erkennungsraten erzielt werden. Vergleicht man die Erkennungsraten der zwei Algorithmen, sind keine signifikanten Unterschiede zu erkennen. Der Likelihood-Algorithmus führt zu etwas besseren Erkennungsraten. Die besten ERn sind nur mit dem optimalen k-Wert erreicht worden. Dieser ist aber bei einer realen Positionierung nicht immer bekannt, wobei sich k=5 als geeigneter Wert für k gezeigt hat. Bei der sequenziellen Betrachtung in Kapitel 5.3 konnten die Eingangstüren zu den Büros im EI-Testgebiet mit dem MFV-Algorithmus sowie dem Likelihood-Algorithmus mit höheren ERn erkannt werden.

In **Kapitel 5** sind weitergehende Testreihen zum Konzept der iCPs dargelegt. Untersucht wurde, wie sich die ERn beim WLAN-Fingerprinting ändern, wenn dieses mit Richtungsinformationen verknüpft wird. Für die Tests wurden die erfassten WLAN-Scans in jeweils vier Richtungskategorien manuell eingeteilt und untersucht, ob sich die ERn verbessern lassen, wenn das WLAN-Fingerprinting nur mit RSSI-Vektoren der gleichen Richtungskategorie durchgeführt wird. Außerdem wurde in 5.1.1 anhand der Testdaten gezeigt, dass eine automatische Erkennung der vier definierten Richtungskategorien möglich ist.

Verbesserungen konnten durch Miteinbeziehung der Richtungskategorien nur teilweise erreicht werden. Einige der besten ERn sanken geringfügig durch die Berücksichtigung der Richtungskategorien. Dies kann möglicherweise auch auf Eingabefehler der Richtungskategorien zurückgeführt werden. Bei dem MVF- und Likelihood-Algorithmus konnte unter Berücksichtigung der Richtungskategorien die Anzahl der Berechnungen um 75 % verringert werden, ohne dass sich die ERn verschlechterten.

Die Tests haben gezeigt, dass es möglich ist, Fingerprinting richtungsabhängig zu betreiben, ohne dass sich die besten ERn signifikant verschlechtern. Im Mittel über alle betrachteten

Szenarien konnten die ERn erhöht werden, im EI-Testgebiet um 5,9 PP und im Heim-Testgebiet um 4,8 PP.

Durch die in 5.2 beschriebene Auswahl wegentscheidender Positionen, die durch baulich vorhandene Strukturen beeinflusst wurden, konnten hohe ERn mit WLAN-Fingerprinting erzielt werden. Die zu testenden RSSI-Vektoren der gewählten iCPs konnten mit der kombinierten DB mit einer ER von 87,1 % zugeordnet werden.

Weiters wurde untersucht, ob sich die ERn verbessern lassen, wenn zwischen den iCPs die logischen Zusammenhänge anhand einer Ablauffolge beachtet werden. Dafür wurden im EI-Testgebiet Abschnitte definiert, die Wegentscheidungen darstellen. Simuliert wurde eine Folge, bei der iCPs nach einer logischen Sequenz besucht werden. Die Datensätze der RSSI-Vektoren, der iCPs, wurden nach den Abschnitten aufgeteilt. Das WLAN-Fingerprinting wurde je Abschnitt simuliert und analysiert. Der Durschnitt der ERn je Abschnitt lag bei 90,4 %. Durch die Verwendung eines Gewichtungsvektors pro Abschnitt konnte die ER auf 92,9 % verbessert werden.

Im letzten Abschnitt, der die vier Türen zu den Testräumen darstellt, konnten die iCPs zuerst nur mit einer durchschnittlichen ER von 38,1 % erkannt werden. Durch die Miteinbeziehung der Richtung, der Selektion der mehrfachen RSS-Werte und der Gewichtung von RSSI-Werten, konnten die iCPs im Mittel mit einer Rate von 89,9 % erkannt werden (5.3.1).

Kapitel 6 dokumentiert die Entwicklungsschritte, welche durchgeführt wurden, um einen Algorithmus zu erstellen, der auf der Erkennung von Schritten, Richtung und iCPs basiert. Als Basis dient ein INS-Algorithmus; dieser unterliegt zwei systembedingten Problemen. Erstens kann nur von einer bekannten Position ausgegangen werden. Zweitens wir die ermittelte Position immer ungenauer, je länger mit dem Algorithmus verfahren wird. Grund dafür ist, dass die Sensoren und Algorithmen vom wahren Wert abdriften.

In 6.2 wurden zwei Schritterkennungsalgorithmen betrachtet. Mit dem *simplestepdetection()* Algorithmus, der beim Überschreiten eines Grenzwertes einen Schritt detektiert, konnte im Mittel die Schrittanzahl auf 1,6 Schritte erkannt werden *(*6.2.2). Maximal wich die erkannte Schrittanzahl um 4 Schritte ab. Der *stepdetectionPeak()* Algorithmus, der die lokalen Maxima in den Beschleunigungsdaten zum Detektieren von Schritten verwendet, erkannte die Anzahl der Schritte im Mittel auf einen halben Schritt genau. Dabei wich dieser maximal um einen Schritt ab (6.2.3).

Anhand der mit der CPS-App erfassten Referenzpunkte während der Testläufe können im MFW die Referenzstrecken rekonstruiert werden. Mithilfe des Schritterkennungsalgorithmus wurden auf den Teststrecken Referenzschritte bzw. Positionen berechnet. Diese wurden für die weiteren Algorithmen zur Berechnung der Abweichungen der einzelnen Schritte herangezogen. Um auch berechnete Schrittpositionen mit den GPS-Aufzeichnungen vergleichen zu können, wurden zu den Zeitpunkten der Schritte die GPS-Koordinaten dafür benutzt.

Der INS-Algorithmus wurde durch die Kombination des Schritterkennungsalgorithmus und der Azimut-Werte des Orientierungssensors erstellt. Dieser ist in 6.4 beschrieben und wurde mit einer Reihe von Testläufen untersucht. Über alle betrachteten Testläufe wichen die errechneten Positionen im Mittel um 4,0 m ab. Im Vergleich dazu wich im Durchschnitt die GPS-Position um 13,9 m von den ermittelten Referenzschritten ab.

Um die mit dem INS-Algorithmus errechneten Positionen mithilfe der iCPs korrigieren zu können, müssen iCPs in der Bewegung erkannt werden. Daher wurden die Aufzeichnungen der WLAN-Scans eines Testlaufes benutzt, um den Zeitpunkt des Passierens der iCPs zu erkennen. Ein iCP wird anhand des minimalsten euklidischen Abstandes zu dessen RSSI-Vektor zu den WLAN-Scans des Testlaufes erkannt. Bei der Testreihe in 6.5.1 zur Untersuchung der dynamischen Erkennung der iCPs konnten die iCPs im Heim-Testgebiet mit dem HTC im Mittel auf einen Schritt genau erkannt werden. Indoor im EI-Testgebiet konnten die iCPs durchschnittlich auf drei Schritte genau erkannt werden. Gut ließen sich jene iCPs erkennen, wo sich durch die vorhandenen baulichen Strukturen die RSS-Werte beim Passieren deutlich veränderten. So ließen sich z. B. jene iCPs bei den Eingängen bzw. im Windfang des Eingangs am besten erkennen. Mit dem S2 konnten die iCPs aufgrund der ca. 5-mal (3,5s) längeren Scandauer im Mittel nur auf 7,2 Schritte genau im Heim-Testgebiet erkannt werden.

Da mit dem in 6.5.1 diskutierten Ansatz nur der Zeitpunkt des Passierens eines iCPs erkannt wird, nicht aber welcher iCP gewählt wurde, wurde in 6.5.2 ein Ansatz dargelegt, der den ersten zu passierenden iCP erkennt und anhand dessen die Startposition mithilfe der Daten des INS-Algorithmus zurückrechnet. Im Mittel konnte die Startposition auf 5 m genau erkannt werden, um 17,4 m genauer als mit dem GPS-Receiver. Nimmt man von jedem Testgebiet den Testlauf mit der größten Abweichung aus der Berechnung, liegt die mittlere Abweichung bei 2,8 m.

Um das Passieren der weiteren iCPs zu erkennen, wurden verschiedene weitere Bedingungen bzw. Einschränkungen genutzt. In 6.5.3 wird ein iCP-INS-Algorithmus präsentiert, der anhand von Einflussbereichen und der darin erfassten RSSI-Vektoren die errechneten Positionen des INS-Algorithmus korrigiert. Zur Korrektur wird die Position zum Zeitpunkt des errechneten Passierens eines iCP, wie in 6.5.1 dargelegt, auf dessen bekannte Position gesetzt. Bei richtiger Kalibrierung des digitalen Kompasses konnten die Teststrecken im Heim-Testgebiet im Mittel auf 4,9 m, um 1,9 m genauer als mit INS-Algorithmus und um 8,5 m genauer als mit dem GPS-Receiver, ermitteltet werden.

In 6.5.4 wurde der iCP-INS-Algorithmus-II präsentiert; dieser Algorithmus stellt eine Weiterentwicklung des iCP-INS-Algorithmus dar. Ein wesentlicher Unterschied ist, dass dieser Algorithmus die in Kapitel 5.3 vorgestellte logische Abfolge der iCPs des EI-Testgebietes berücksichtigt. Außerdem werden die iCPs intelligenter. So wird beim Passieren bestimmter iCPs die Schrittweite auf den Abstand der Stiegen korrigiert, ebenso werden Azimut-Werte korrigiert. Die Korrektur der Position wird bei diesem Algorithmus auf die vorangegangenen Schrittpositionen gleichmäßig verteilt, um ein kontinuierliches Abbild der Strecke zu erhalten. Außerdem können iCPs noch mit weiteren Einschränkungen versehen werden. So könnten die iCPs, die zu den Büros führen, nur in der Richtung passiert werden, nach der die Türen ausgerichtet sind. Betrachtet man alle Testläufe, wich der iCP-INS-Algorithmus-II um 1,9 m und der INS-Algorithmus um 4,3 m von den Referenzschritten ab. Die Endposition konnte vom iCP-INS-Algorithmus-II durchschnittlich auf 1,5 m und mit dem INS-Algorithmus auf 5,7 m genau erkannt werden.

7.2 Ausblick

Die in der Arbeit entwickelten und untersuchten Algorithmen verwendeten aufgezeichnete Daten von Testläufen. Den Algorithmen standen also zu jedem Zeitpunkt alle erfassten Daten eines Testlaufs zur Verfügung. Die vorgeschlagenen Ansätze wurden als Offline-Varianten in MATLAB realisiert, es war also möglich in die Zukunft zu sehen, was bei einer realen Positionierung nicht der Fall ist. Relevant ist das z. B. beim Finden des minimalen Abstandswertes zum RSSI-Vektor eines iCPs. Beim Algorithmus 6-4 wurde der komplette Testlauf herangezogen. Wie eine Online-Variante aussehen könnte, wurde bereits in 6.5.1 in der „*Zusammenfassung, Konklusion und Ausblicke zur dynamischen Erkennung der iCPs*“ diskutiert.

Um Online-Varianten der Algorithmen einfacher entwickeln und analysieren zu können, wäre eine Schnittstelle im **MFW** sinnvoll, die jeweils nur einen Datensatz zu einem Zeitpunkt t für die Algorithmen bereitstellt. So könnten Online-Algorithmen bzw. die Positionierung zu einem Zeitpunkt t besser simuliert werden.

Das **CPS-App** könnte mit wenigen Erweiterungen neue Möglichkeiten für die Simulation und Analyse bieten. So könnte beim *Kompass-WLAN-Scanner* zum Azimut-Wert auch die gemessene Magnetfeldstärke erfasst werden. Das würde es ermöglichen, eine Karte der Feldstärken für bestimmte Positionen für ein Gebäude zu erstellen. Diese Karte könnte z. B. zum Kalibrieren der Azimut-Wert an den iCPs genutzt werden.

Für den *Sensor-WLAN-Recorder* könnte eine Java-Schnittelle entwickelt werden, die das Integrieren von weiteren Sensoren erleichtert, da mit neuen Smartphone-Generationen immer wieder neue Sensoren zum Standard werden. So könnte z. B. zusätzlich die Erfassung von Magnetfeldsensor und Gyroskop in das App eingebettet werden, was es möglich machen würde, die Orientierung des Gerätes in MATLAB zu berechnen. Somit könnte ein Algorithmus entwickelt und analysiert werden, der unabhängig von der Ausrichtung des Handys funktioniert.

Das **iCP-Konzept** wurde hier mit deterministischen Algorithmen realisiert. Die Entscheidung, ob ein iCP als passiert erkannt wird, ist also abhängig von bestimmten Bedingungen. Wird ein iCPs falsch oder nicht erkannt, kann das zum Versagen des Ansatzes führen. Daher wäre es vermutlich sinnvoll, einen stochastischen Ansatz mit den hier vorgestellten iCPs zu realisieren. So könnten die verwendeten Informationen, wie euklidischer Abstand zum RSSI-Vektor des iCPs, Richtung des Passierens, die aktuell errechnete Position und die logischen Zusammenhänge der iCPs benutzt werden, um einen Wahrscheinlichkeitswert für das Passieren eines iCPs zu ermitteln. Weiters gibt es noch Sensordaten, die benutzt werden könnten, um bestimmte iCPs zu erkennen. Z. B. könnte versucht werden, mithilfe der Daten des Beschleunigungssensors und des Barometers zu erkennen, ob und zu welchem Zeitpunkt die Stiege oder der Lift benutzt wurden. Bei einem stochastischen Ansatz wäre es sinnvoll, die Genauigkeit der aktuellen berechneten Position ebenfalls in einer Wahrscheinlichkeit auszudrücken.

Eine **Weiterentwicklung der benutzten Algorithmen** würde auch die Zuverlässigkeit des iCP-Konzepts erhöhen. So könnte der INS-Algorithmus durch eine dynamische Schrittweitenerkennung über Integration der Beschleunigungsdaten verbessert werden. Anhand von anderen WLAN-Fingerprinting Algorithmen könnten möglicherweise auch Verbesserungen erzielt werden.

Um das Konzept der iCPs zu realisieren, müssen die iCPs definiert und mit WLAN-Scans erfasst werden. Diese Settings- bzw. Trainingsphase sollte möglichst einfach und unkompliziert sein, damit sie auch von unerfahrenen Usern durchgeführt werden kann. Daher könnte eine App entwickelt werden, die anhand eines Gebäudeplans und den Positionen der APs Vorschläge macht, wo iCPs definiert werden könnten. Der User müsste dann nur vorgeschlagene Positionen begehen und die Messungen könnten automatisch durchgeführt werden.

Vorstellbar wäre auch, dass in der Trainingsphase die iCPs passiert werden müssen und die Daten der Sensoren von der App aufgezeichnet werden. So könnte z. B. ein neurales Netz benutzt werden, um bestimmte Eigenschaften der iCPs zu trainieren. Hier könnten Muster erkannt werden, die z. B. die Richtungs-, Beschleunigungs- und Barometerdaten oder die euklidischen Abstandswerte mit den iCPs in Verbindung bringen. Durch die Analyse der Pläne könnten diese Daten auch mit Stiegen, Liften und Türen in Beziehung gebracht werden.

Die in der Arbeit vorgeschlagenen intelligenten Check Points könnten also in einer vielfältigen Form Anwendung finden, die präsentierten und analysierten Algorithmen stellen einen einfachen Ansatz dar, der weiterentwickelt werden sollte.

8 Anhang A - Code, Testgeräte, Testgebiete und Testläufe

Der Anhang A beschreibt die Ablageorte der erstellten Programme und erfassten Daten. Außerdem werden die Testgeräte, die Testgebiete, die Koordinaten der Referenzpunkte und iCPs beschrieben. Die ausgewählten Teststrecken als auch die durchgeführten Testläufe sind hier ebenfalls dokumentiert.

8.1 Ablageorte Quellcode und CPS-APP, MATLAB-Framework, Karten und Testdaten

Die im Rahmen der Diplomarbeit erstellten Programme und erfassten Daten sind online unter [37] öffentlich verfügbar. In diesem Google-Drive Konto ist im Verzeichnis *CPS-APP/InstallDatei* die Installationsdatei der CPS-App abrufbar. Der Quellcode bzw. das gesamte Eclipse Projekt ist im Verzeichnis *CPS-APP/Eclipse-Projekt* gespeichert.

Das im Zuge der Arbeit erstellte MATLAB-Framework (MFW) ist im Verzeichnis *MFW* zu finden. Es enthält alle für den Import und die Analyse der Daten verwendeten Funktionen. Im Unterverzeichnis *maps* sind die verwendeten Karten und dazugehörigen Konfigurationsdateien gespeichert.

Die mit dem CPS-APP erfassten Daten, die WLAN-Scans und Testläufe, sind im Verzeichnis *Testdaten* gespeichert. Diese Testdaten dienten als Grundlage für alle getätigten Untersuchungen der Arbeit.

8.2 Testgeräte

Für die Arbeit standen zwei Testgeräte zur Verfügung. Das Samsung Galaxy S2 (Kurzbezeichnung: S2) von Samsung und das HTC EVO 3D (HTC). In Tabelle 8-1 sind die Geräte dargestellt und die für die Arbeit relevanten Daten beschrieben.

Tabelle 8-1 Überblick der relevante Daten

Bezeichnung:	Samsung Galaxy S2	HTC EVO 3D
Kurzbezeichnung in Arbeit:	S2	HTC
Modellnummer:	GT-I9100	HTC EVO 3D X515m
Android-Version:	4.1.2 (API 16)	4.0.3 (API 15)
WLAN:	802.11 a/b/g/n	802.11 b/g/n
Sensoren:	Beschleunigungssensor Gyroskop, Kompass	Beschleunigungssensor Gyroskop, Kompass
Datenquelle:	[34]	[33]
Foto	 Foto: [67]	 Foto: [29]

8.3 Testgebiete

In diesem Kapitel sind die zwei Testgebiete definiert. Die dazugehörigen Karten sind in [37] im Verzeichnis *MFW/maps* abrufbar. Definiert sind in diesem Kapitel des Weiteren die iCPs und Referenzpunkte anhand ihrer Koordinaten in Dezimalgrad. Diese wurden, wie in 3.3.4 erläutert, mit dem MFW in die y-, x-Bildkoordinaten in MATLAB transformiert.

Dafür wurden die lokalen y-, x-Bildkoordinaten der Einpassungspunkte, die sich auf die Pixel eines Bildes beziehen, in Matlab ausgelesen. Dafür wurde das Bild mit der Funktion *imshow()* erzeugt und die Koordinaten mit dem Daten Cursor von MATLAB ausgelesen. Die y-, x-Achse und sowie deren Orientierungen sind in den Abbildungen (z. B. Abbildung 8-1) der Karten dargestellt. Der Nullpunkt dieses Koordinatensystems befindet sich am linken oberen Rand des Bildes der Karte.

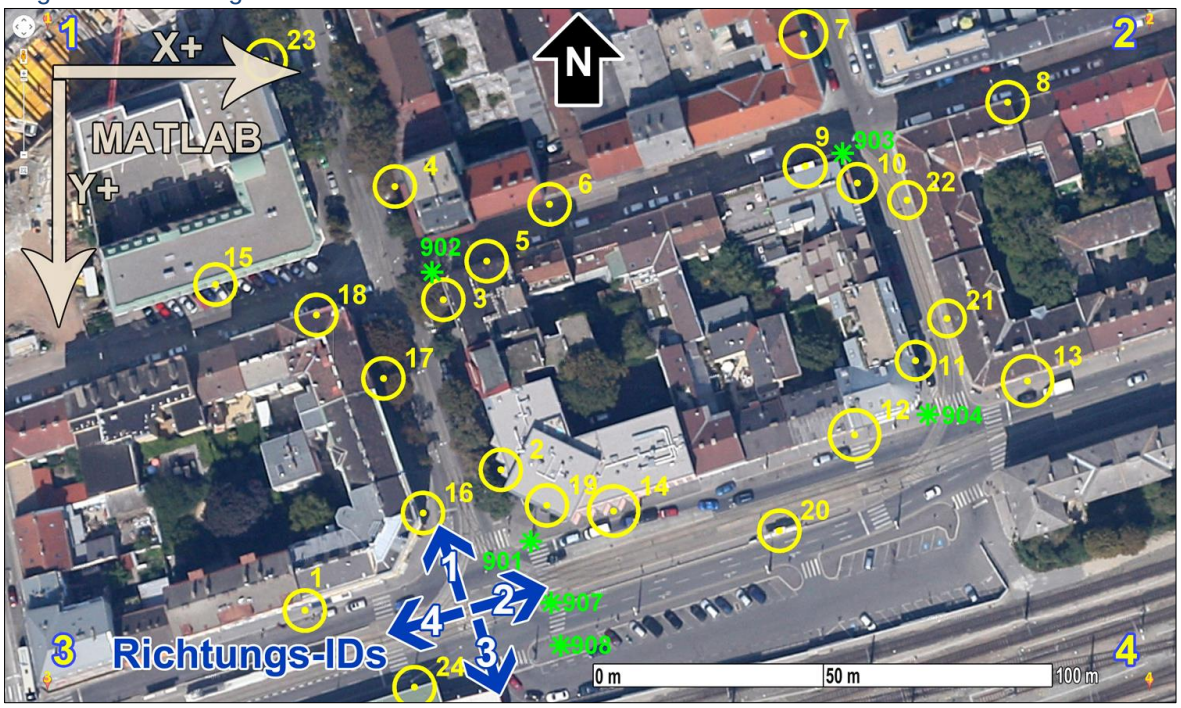
Mit Google-Maps wurden die Koordinaten in Dezimalgrad für die Einpassungspunkte ausgelesen. Mit diesen Geokoordinaten und deren Pendants den y, x- Bildkoordinaten konnten die Koordinaten von beliebigen Positionen zwischen den Einpassungspunkten interpoliert werden. Die verwendeten Koordinaten der Einpassungspunkte sind in Tabelle 8-2, Tabelle 8-5 und Tabelle 8-14 angeführt.

Die Koordinaten der definierten iCPs, sowie die Parameter für die Einflussbereiche, die Korrekturwerte für die Richtung und Schrittweite und die Definitionen für eingeschränkte Passierbarkeit, sind in weiterer Folge angeführt. In den Abbildungen der Testgebiete sind die definierten Richtungskategorien bzw. deren IDs als blaue Pfeile mit weißem Rand dargestellt.

8.3.1 Heim-Testgebiet - Studentenheim Eichenstraße 46 (Outdoor)

Für die reinen Outdoor-Testreihen wurde der Häuserblock um das Studentenheim ÖJAB-Haus Meidling in der Eichenstraße 46, A-1120 Wien verwendet. Die gewählten iCPs und Referenzpunkte sind in diesem Kapitel definiert.

Abbildung 8-1 Heim-Testgebiet Karte: Heim5m



Die Abbildung 8-1 zeigt die Karte „Heim5m“. Diese ist im Verzeichnis *DAAS/maps/Heim5m* in [37] bereitgestellt. Die RichtungsIDs von 1 bis 4 entsprechen den Azimut-Werten 343°,73°,163° und 253°.

Die Koordinaten der **Einpassungspunkte** sind mit den aus Google-Maps bekannten Stecknadelsymbolen an den Ecken des Bildes eingezeichnet und die Koordinaten in Tabelle 8-2 dokumentiert. Die ***Referenzpunkte*** sind in der Abbildung 8-1 grün dargestellt. Sie dienen

dazu, Teststrecken zu definieren. An diesen Punkten kann ein Testlauf gestartet oder die Richtung geändert werden. In der Tabelle 8-4 sind diese Referenzpunkte definiert.

Die Koordinaten der **iCPs** sind in der Tabelle 8-3 dargestellt. Neben den Koordinaten wurden für die iCPs Einflussbereiche definiert. Die Einflussbereiche wurden durch ihren Radius in Metern definiert. Die an den iCPs erfassten WLAN-Scans sind im Verzeichnis *Testdaten/Heim-Testgebiet* in [37] gespeichert.

Tabelle 8-2 Einpassungs-Koord. Heim20m

ID.	Lat.	Lon.	Y	X
1	48.176611	16.334484	55	38
2	48.176611	16.337740	55	1252
3	48.175310	16.334484	782	38
4	48.175310	16.337740	782	1252

Tabelle 8-3 iCheck Points Koord. Heim-Testgebiet

ID.	Lat.	Lon.	r* [m]	ID.	Lat.	Lon.	r* [m]
1	48.175467	16.335245	4,4	13	48.175919	16.337380	5,4
2	48.175744	16.335824	4,4	14	48.175663	16.336157	5,4
3	48.176079	16.335654	4,4	15	48.176109	16.334981	4,4
4	48.176302	16.335511	4,4	16	48.175659	16.335594	4,4
5	48.176155	16.335782	4,4	17	48.175924	16.335477	4,4
6	48.176267	16.335968	4,4	18	48.176049	16.335279	4,4
7	48.176605	16.336726	5	19	48.175674	16.335960	4,4
8	48.176468	16.337321	4,4	20	48.175624	16.336647	4,4
9	48.176342	16.336723	4,5	21	48.176042	16.337142	3,9
10	48.176309	16.336877	4,1	22	48.176275	16.337024	3,9
11	48.175959	16.337049	4,1	23	48.176550	16.335129	4,4
12	48.175813	16.336869	5,4	24	48.175316	16.335569	4,4

*r bezeichnet den Radius des Einflussbereichs

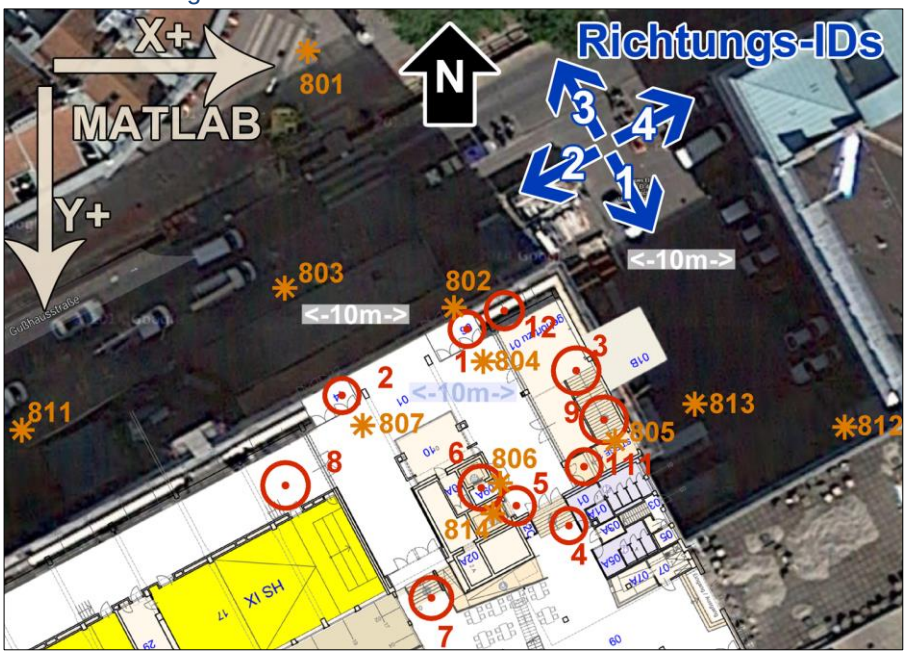
Tabelle 8-4 Referenzprunkte Koord. Heim-T.

ID.	Lat.	Lon.
901	48.175603	16.335903
902	48.176122	16.335637
903	48.176369	16.336847
904	48.175854	16.337091
907	48.175482	16.335973
908	48.175398	16.336001
999	48.176702	16.336664

8.3.2 EI-Testgebiet Bereich Erdgeschoss (EG)

Das Testgebiet befindet sich vor und im Neuen Elektrotechnik Institut der TU Wien in der Gußhausstr. 27-29. Um den Übergang von einem Außenbereich in den Innenbereich eines Gebäudes zu untersuchen, wurde das Testgebiet vor und im Erdgeschoss des Gebäudes des Neuen Elektrotechnik Instituts definiert. Das Gebäude gehört zur TU Wien und in ihm befindet sich auch die Forschungsgruppe für Ingenieurgeodäsie.

Abbildung 8-2 Ausschnitt EI-Testgebiet Karte EImixEG



Die Karte ist in *maps* in [37] im Unterverzeichnis *EImixEG* gespeichert. Die RichtungsIDs von 1 bis 4 entsprechen einem Azimut-Wert von 150°, 240°, 330° und 60°.

Die Koordinaten der **Einpassungspunkte** sind in Tabelle 8-5 dokumentiert. Die ***Referenzpunkte*** sind orange dargestellt und in der Tabelle 8-9 definiert.

Die Koordinaten der **iCPs** sind in der Tabelle 8-8 dargestellt. Bei den iCPs der Indoor-Testreihen wurden der Einflussbereich und die Ebene des iCPs definiert. Die Daten für die Korrektur der Richtung und Schrittgröße sind in der Tabelle 8-6 und Tabelle 8-7 angegeben.

Tabelle 8-5 Einpassungs-Koord. EImixEG

	Lat.	Lon.	Y	X
1	48.196576	16.368607	236	39
2	48.196576	16.370571	236	2968
3	48.195569	16.368607	2489	39
4	48.195569	16.370571	2489	2968

Tabelle 8-6 Korrektur Richtung - EImixEG

iCP	vorher Schritte	nachher Schritte	Orientierung
1	3	1	150.5
2	3	1	150.5
12	0	3	150.5
111	5	0	242.6

Tabelle 8-7 Korrektur Schrittgröße EImixEG

iCP	vorher Schritte	nachher Schritte	Schrittgröße
3	0	16	0,27
9	0	5	0,27

Tabelle 8-8 iCP Koord. EImixEG

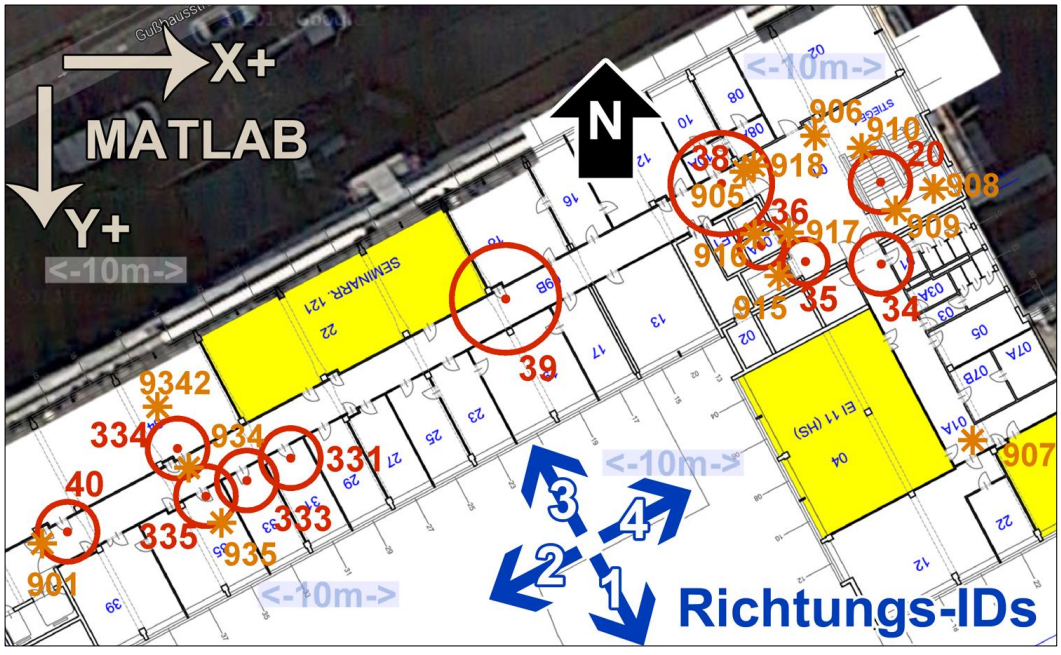
ID.	*Lat.*	Lon.	r*[m]	Ebene
1	48.19642716	16.36976770	1,5	0
2	48.19637487	16.36961951	1,5	0
3	48.19639453	16.36989711	2	1
4	48.19627207	16.36988839	1,5	1
5	48.19628771	16.36982603	1,6	1
6	48.19630201	16.36978446	1,9	1
7	48.19621486	16.36972545	1,8	1
8	48.19630380	16.36955246	2	1
9	48.19635565	16.36992997	2	2
111	48.19631855	16.36990649	1,5	0
12	48.19644191	16.36981195	1,6	0

Tabelle 8-9 Referenzprunkte – EI-T. EG

ID.	*Lat.*	Lon.
801	48.196647000	16.369579000
802	48.196444147	16.369752276
803	48.196459791	16.369551785
804	48.196401686	16.369787143
805	48.196338664	16.369943378
806	48.196305589	16.369807259
807	48.196351179	16.369644319
811	48.196347604	16.369239986
812	48.196349838	16.370215616
813	48.196368164	16.370037924
814	48.196281454	16.369798543

8.3.3 Indoor Testgebiet - neues EI 3. Stock (EImix3S)

Abbildung 8-3 Ausschnitt der Karte EImix3S



Das Indoor Testgebiet befindet sich im 3. Stock des Gebäudes des Elektrotechnik Institutes. In diesem Stock befindet sich die Forschungsgruppe für Ingenieurgeodäsie und das Büro von Prof. Retscher. Die Karte Abbildung 8-3 in wird als EImi3S bezeichnet. Die Koordinaten der **Einpassungspunkte** sind in sind in Tabelle 8-14 dokumentiert. Die ***Referenzpunkte*** sind in Abbildung 8-3 in orange dargestellt. In der Tabelle 8-11 sind diese Referenzpunkte definiert.

Die Koordinaten der **iCPs** sind in der Tabelle 9-10 dargestellt. Die Indoor iCPs sind zusätzlich durch Einflussbereich und deren Abschnitt definiert. Die Daten für die Korrektur der Richtung und Schrittgröße sind in der Tabelle 8-12 und Tabelle 8-13 angegeben. Zusätzlich gibt es noch eine Einschränkung für bestimmte iCPs, nach welcher Richtungskategorie diese passiert werden dürfen. Diese ist in Tabelle 8-15 beschrieben.

Tabelle 8-10 iCP Koord. EImix3S

ID.	*Lat.*	Lon.	r*[m]	Abschnitt
20	48.19633732	16.36989644	2	3
34	48.19628816	16.36989711	2	4
38	48.19633643	16.36975296	3,4	4
39	48.19626716	16.36955849	3,6	5
40	48.19612725	16.36916354	2	5
334	48.19617731	16.36926278	2	6
35	48.196289499	16.369828717	1,5	3
36	48.196299332	16.369789155	1,5	3
333	48.196158093	16.369325144	2	6
335	48.196148707	16.369288265	2	6
331	48.196171508	16.369364706	2	6

Tabelle 8-11 Referenzprunkte Koord. EImix3S

ID.	Lat.	Lon.
901	48.196120101	16.369140747
905	48.196343581	16.369773733
906	48.196365035	16.369837434
907	48.196182229	16.369979587
908	48.196333301	16.369944719
909	48.196320339	16.369910522
910	48.196357884	16.369879677
934	48.196165691	16.369272842
915	48.196280859	16.369805141
916	48.196306036	16.369783121
917	48.196306930	16.369813965
918	48.196347157	16.369782450
9342	48.196202342	16.369244680
935	48.1961326161	16.369302346

Tabelle 8-12- Korrektur Richtung EImix3S

iCP	vorher Schritte	nachher Schritte	Richtung
39	3	3	242

Tabelle 8-13 Korrektur Schrittgröße EImix3S

iCP	vorher Schritte	nachher Schritte	Schrittgröße
20	3	12	0,27

Tabelle 8-14 Einpassungs-Koord. EImix3S

ID.	Lat.	Lon.	Y	X
1	48.196576	16.368607	236	39
2	48.196576	16.370571	236	2968
3	48.195569	16.368607	2489	39
4	48.195569	16.370571	2489	2968

Tabelle 8-15 Eingeschränkte Passage EImix3S

iCP	Richtung Ein	RichtungID
38	2	4
39	2	4
331	1	3
333	1	3
335	1	3
334	3	1

8.4 Definierte Teststrecken

Für die Arbeit wurden Teststrecken in den Testgebieten durch iCPs und Referenzpunkte definiert. Durch die Verbindung dieser Positionen konnten Strecken als Referenz verwendet werden. Eine Teststrecke kann im CPS-APP (Abschnitt 3.2) mit den IDs der iCPs und Referenzpunkte definiert werden. Das Erreichen dieser Punkte wurde während eines Testlaufes mit dem CPS-APP erfasst. In Tabelle 8-16 sind die Teststrecken für das Heim-Testgebiet und in Tabelle 8-17 und Tabelle 8-18 für das EI-Testgebiet definiert.

Tabelle 8-16 Testrecken Heim-Testgebiet

Bezeichnung	Referenzpunkte und iCP	Länge
Heim Abschnitt 1	901;2;3;902	62 m
Um Heim	901;2;3;902;5;9;903;10;11;904;12;14;901	306 m
Bahnhof nach 7	24;908;907;901;2;3;902;5;9;903;7;999	252 m

Tabelle 8-17 Testrecken EImixEG

Bezeichnung	Referenzpunkte und iCP	Länge
EG-E11-L1	812;813;111;5;814	32m
EG-E12-S1	801;12;3;9;805	44m
EG-E1-L2	801;802;1;804;806;6	43m
EG-E1-S1	801;802;1;804;3;9;805	48m
EG-E2-S1	811;803;2;807;804;3;9;805	69m

Tabelle 8-18 Testrecken EImix3S

Bezeichnung	Referenzpunkte und iCP	Länge
3S-L1-335	915;35;918;38;39;934;335;935	58m
3S-L2-334	916;36;917;918;38;39;934;334;9342	54m
3S-S1-Z1	908;909;20;910;905;38;39;934;334	65m
3S-Gang1a	901;39;38;905	54m
3S-Gang1b	905;38;39;901	54m
3S-Gang2a	906;34;907	23m
3S-Gang2b	907;34;906	23m

8.5 Erfasste Testläufe

Mithilfe des auf den Testgeräten installierten CPS-Apps wurden die definierten Teststrecken abgegangen. Das APP zeichnet dabei die WLAN-SCANS, GPS- und die Sensordaten auf und speicherte diese, wie in 3.2.4 erläutert, in TXT-Dateien. Mit dem CPS-APP wurde ebenfalls der Zeitpunkt des Erreichens eines iCPs oder Referenzpunktes erfasst.

Die Testläufe und deren Speicherorte sind für das Heim-Testgebiet in Tabelle 8-19 und für das EI-Testgebiet in Tabelle 8-20 und Tabelle 8-21 aufgeführt.

Tabelle 8-19 Testläufe Heim-Testgebiet

Nr.	Gerät	Zeitstempel_Seriennummer	Teststrecke	Datum	Zeit	Speicherort in [37] *Testdaten/*
H1	S2	1421159221477_00190e766a614e	Heim Abschnitt 1	2015-01-13	15:27	*stepdection/testruns*
H2	S2	1421159671227_00190e766a614e	Heim Abschnitt 1	2015-01-13	15:34	*stepdection/testruns*
H3	S2	1421159822598_00190e766a614e	Heim Abschnitt 1	2015-01-13	15:37	*stepdection/testruns*
H4	HTC	1421160323383_HT17SV201390	Heim Abschnitt 1	2015-01-13	15:45	*stepdection/testruns*
H5	HTC	1421160478300_HT17SV201390	Heim Abschnitt 1	2015-01-13	15:47	*stepdection/testruns*
H6	HTC	1421160960149_HT17SV201390	Heim Abschnitt 1	2015-01-13	15:56	*stepdection/testruns*
H7	HTC	1420990292207_HT17SV201390	Um Heim	2015-01-11	16:31	Heim-Testgebiet/11204691/testruns
H8	S2	1420990799537_00190e766a614e	Um Heim	2015-01-11	16:39	Heim-Testgebiet/11204691/testruns
H9	HTC	1421007758287_HT17SV201390	Bahnhof nach 7	2015-01-11	21:22	Heim-Testgebiet/11204691/testruns
H10	S2	1421009278255_00190e766a614e	Bahnhof nach 7	2015-01-11	21:47	Heim-Testgebiet/11204691/testruns

Tabelle 8-20 Testläufe EI-Testgebiet EG

Nr.	Gerät	Kennung	Teststrecke	Datum	Zeit	Speicherort in [37] *Testdaten/*
TL-EI1	HTC	1424262202596_HT17SV201390	EG11L1	2015-02-18	13:23	EI-Testgebiet/EIEG/testruns
TL-EI2	HTC	1424271774463_HT17SV201390	EG12S1	2015-02-18	16:02	EI-Testgebiet/EIEG/testruns
TL-EI3	HTC	1423759395915_HT17SV201390	EIEG1L2	2015-02-12	17:43	EI-Testgebiet/EIEG/testruns
TL-EI4	HTC	1423753610555_HT17SV201390	EIEG1S1	2015-02-12	16:06	EI-Testgebiet/EIEG/testruns
TL-EI5	HTC	1424259029604_HT17SV201390	EIEG2S1	2015-02-18	12:30	EI-Testgebiet/EIEG/testruns

Tabelle 8-21 Testläufe EI-Testgebiet 3S

Nr.	Gerät	Kennung	Teststrecke	Datum	Zeit	Speicherort in [37] *Testdaten/*
EI6	HTC	1424278303461_HT17SV201390	EI3SL1335	2015-02-18	17:51	EI-Testgebiet/EI3S/testruns
EI7	HTC	1424276626837_HT17SV201390	EI3SL2334	2015-02-18	17:23	EI-Testgebiet/EI3S/testruns
EI8	HTC	1423840372591_HT17SV201390	EI3SZ1	2015-02-13	16:13	EI-Testgebiet/EI3S/testruns
EI9	HTC	1423831639079_HT17SV201390	EI3SGang1a	2015-02-13	13:47	EI-Testgebiet/EI3S/testruns
EI10	HTC	1423831971915_HT17SV201390	EI3SGang1b	2015-02-13	13:52	EI-Testgebiet/EI3S/testruns
EI11	HTC	1423833431217_HT17SV201390	EI3SGang2a	2015-02-13	14:17	EI-Testgebiet/EI3S/testruns
EI12	HTC	1423833535285_HT17SV201390	EI3SGang2b	2015-02-13	14:18	EI-Testgebiet/EI3S/testruns

9 Anhang B – Schritterkennung, INS-Algorithmus, Erkennung iCPs

Dieser Anhang enthält die Ergebnisse der durchgeführten Analysen der entwickelten Algorithmen zur Schritterkennung und zeigt, wie die Parameter für diese optimiert wurden. Außerdem wurde der entwickelte INS-Algorithmus anhand der aufgezeichneten Testläufe (8.5) analysiert. Die Ergebnisse des vorgeschlagenen Algorithmus, um das Passieren von iCPs zu erkennen, sind auch in Anhang B in 9.3 dargelegt. Die entsprechenden MATLAB-Funktionen sind in [37, DAAS\testruns] in den nach den Algorithmen benannten Verzeichnissen zu finden.

9.1 Schrifterkennungsalgorithmen – Optimierung der Parameter

Für die Arbeit wurden zwei einfache Schritterkennungsalgorithmen in MATLAB implementiert. Die Parameter für die Algorithmen *simplestepdection()* (6.2.2) *und stepdetectionPeak() (6.2.3) wurden* optimiert, indem die Algorithmen mit verschiedenen Parametern auf Testläufe mit bekannter Schrittanzahl angewendet und mit der detektierten Schrittanzahl verglichen wurden. Die Testläufe sowie die gezählten Schritte wurden mit der CPS-APP erfasst und sind in Tabelle 9-1 angeführt.

Tabelle 9-1 Testläufe mit gezählten Schritten zur Optimierung der Stepdetection-Alg.

Testlauf	Bereich	Beschreibung	Gerät	gezählte Schritte	Streckenlänge
H1	Outdoor	1.TL Heim Abschnitt 1	S2	88	62m
H2	Outdoor	2.TL Heim Abschnitt 1	S2	91	62m
H3	Outdoor	3.TL Heim Abschnitt 1	S2	90	62m
H4	Outdoor	1.TL Heim Abschnitt 1	HTC	93	62m
H5	Outdoor	2.TL Heim Abschnitt 1	HTC	92	62m
H6	Outdoor	3.TL Heim Abschnitt 1	HTC	91	62m
EI9	Indoor	Gang 1 entlang EI 3.Stock	HTC	83	54m
EI10	Indoor	Gang 1 zurück EI 3.Stock	HTC	80	54m
EI11	Indoor	Gang 2 entlang EI 3.Stock	HTC	35	23m
EI12	Indoor	Gang 2 zurück EI 3.Stock -	HTC	34	23m

9.1.1 Stepdetection-Algorithmus - *simplestepdection()*

Der Algorithmus detektiert einen Schritt, wenn die Daten des Beschleunigungssensors einen definierten Grenzwert (**accVstep)** überschreiten. Damit dieser akzeptiert wird, muss der letzte Schritt mindestens die definierte Zeit (**steptimeD**) zurück liegen.

Tabelle 9-2 zeigt die erzielten mittleren Abweichungen in Schritten für die Outdoor Testläufe H1 bis H6 mit den entsprechenden Parametern. Die Tabelle stellt einen Ausschnitt der tatsächlich getesteten Parameter mit den geringsten Abweichungen dar. Die im Mittel erzielte minimale Abweichung ist blau markiert. Die grüne Markierung zeigt das Minimum, wenn alle Nachbarwerte eines Wertes addiert werden. Die Berechnung dafür entspricht einer diskreten Faltung mit einer 3x3 Matrix, die an jeder Stelle den Wert 1 enthält.

Tabelle 9-2 Durchschnittliche Fehlschritte *simplestepdection()* in Z-Achse H1-H6 Outdoor

		Abweichung in Schritten							
		accVstep (m/s² in Z-Achse Beschleunigungssensor)							
		12,7	**12,8**	**12,9**	**13,0**	**13,1**	**13,2**	**13,3**	**13,4**
steptimeD (in Sekunden)	**0,55 s**	5,7	5,0	4,0	3,7	3,5	2,7	1,8	1,7
	0,56 s	3,7	3,0	2,3	2,0	2,0	2,0	1,7	1,8
	0,57 s	2,7	2,0	1,2	**1,2**	1,0	1,3	1,3	1,5
	0,58 s	1,5	1,2	1,0	**1,0**	1,0	1,8	2,5	3,0
	0,59 s	1,5	1,5	1,8	2,3	2,3	3,2	3,8	4,3
	0,60 s	2,3	2,3	3,2	3,5	3,5	4,2	4,7	5,2
	0,61 s	2,7	3,3	4,0	4,3	4,3	5,0	5,3	5,7

min. Summe aller Nachbarwerte
minimale Fehlschritte

In Tabelle 9-3 sind die mittleren Abweichungen in Schritten für die Indoor Testläufe EI9 bis EI12 mit den entsprechenden Parameter dargestellt. Tabelle 9-4 zeigt Ergebnisse der mittleren Schrittabweichung wenn Indoor und Outdoor Testläufe H1 bis H6 und EI9 bis EI12 zusammen für die Analyse verwendet wurden.

Tabelle 9-3 Durchschnittliche Fehlschritte *simplestepdection()* in Z-Achse EI9-EI12 Indoor

		Abweichung in Schritten							
		accVstep (m/s² in Z-Achse Beschleunigungssensor)							
		11,2	**11,3**	**11,4**	**11,5**	**11,6**	**11,7**	**11,8**	**11,9**
steptimeD (in Sekunden)	**0,55 s**	3,5	3,3	2,5	2,5	1,5	1,5	0,5	1,3
	0,59 s	2,5	2,5	2,0	2,0	1,3	0,5	0,5	1,3
	0,60 s	1,5	1,3	0,8	**0,5**	1,0	1,0	1,5	1,8
	0,61 s	0,8	0,5	**0,8**	1,0	1,3	1,8	3,0	3,5
	0,62 s	1,0	0,8	0,8	0,8	1,3	1,3	2,0	3,0
	0,63 s	1,5	1,5	2,0	2,0	2,8	3,0	3,8	4,5
	0,64 s	2,3	2,3	2,5	2,8	3,5	3,5	4,3	4,8
	0,65 s	2,5	3,0	3,5	3,8	4,5	4,5	5,0	5,5
	0,66 s	3,5	3,8	4,3	4,3	5,3	5,5	6,3	6,8

min. Summe aller Nachbarwerte
minimale Fehlschritte

Tabelle 9-4 Durch. Fehlschritte *simplestepdection()* in Z-Achse H1-EI12 In/Outdoor

		Abweichung in Schritten				
		accVstep (m/s² in Z-Achse Beschleunigungssensor)				
		11,51	**11,52**	**11,53**	**11,54**	**11,55**
steptimeD (in Sekunden)	**0,59 s**	3,7	3,7	3,7	3,7	3,7
	0,60 s	1,9	1,9	1,9	1,9	1,9
	0,61 s	2,1	2,1	2,1	**2,1**	2,1
	0,62 s	1,7	1,7	1,7	**1,6**	1,6
	0,63 s	2,1	2,1	2,1	2,0	2,0
	0,64 s	2,9	2,9	2,9	2,9	2,9

min. Summe aller Nachbarwerte
minimale Fehlschritte

In Tabelle 9-5 sind die Abweichungen in Schritten im Detail dargestellt. Als Parameter sind jeweils diejenigen verwendet worden, die zur minimalen mittleren Abweichung geführt haben.

Tabelle 9-5 Schrittabweichungen- Der ermittelte Parameter für *simplestepdection() z-Achse*

	Abweichung in Schritten									
	Testlauf									
Testgebiete : Parameter, steptimeD; accVstep	**H1**	**H2**	**H3**	**H4**	**H5**	**H6**	**EI9**	**EI10**	**EI11**	**EI12**
Heim-Testgebiet (Outdoor): 0,58s, 13m/s²	-1	0	-1	-2	-1	-1	-	-	-	-
EI-Testgebiet (Indoor): 0,60s; 11,5m/s²	-	-	-	-	-	-	0	1	1	0
Heim/EI-Testgebiet: 0,62s; 11,54m/s²	-1	-1	0	-1	3	2	1	-1	-2	-4

S2
HTC

Tabelle 9-6 zeigt die Analyse für alle 10 Testläufe auf der y-Achse (siehe Abbildung 2-2) des Handys. An diesem Test ist zu sehen, dass bei der Verwendung der Z-Achse des Handys bessere Ergebnisse erzielt werden konnten.

Tabelle 9-6 Durch. Fehlschritte *simplestepdection()* in y-Achse H1-EI12 In/Outdoor

		Abweichung in Schritten							
		accVstep (m/s² in Z-Achse Beschleunigungssensor)							
		0,45	**0,46**	**0,47**	**0,48**	**0,49**	**0,50**	**0,51**	**0,52**
steptimeD (in Sekunden)	**0,64s**	6,83	6,33	6,33	6,33	6,33	6,67	6,67	6,67
	0,65s	6,50	6,50	6,50	6,50	6,50	6,67	6,33	6,50
	0,66s	6,50	6,33	6,17	6,17	6,17	6,33	6,17	6,17
	0,67s	6,00	***5,67***	5,67	5,67	5,67	**5,67**	5,67	6,00
	0,68s	6,50	6,17	6,17	6,17	6,17	6,00	6,00	6,00
	0,69s	7,00	6,83	6,83	6,83	6,83	7,17	7,17	7,33
	0,70s	7,83	7,83	7,83	7,83	7,83	7,83	7,67	7,83

min. Summe aller Nachbarwerte
minimale Fehlschritte

Tabelle 9-7 Durch. Fehlschritte *simplestepdection()* in Y-Achse H1-EI12 In/Outdoor

	Testlauf					
	H1	H2	H3	H4	H5	H6
Abweichung in Schritten	9	10	2	-5	-6	-7

9.1.2 Stepdetection-Algorithmus - *stepdetectionPeak()*

Der Algorithmus verwendet die Funktion *peakfinder()*. Diese findet Maxima in den Sensordaten. Die Sensordaten werden mit der MATLAB-Funktion *smooth()* geglättet. In *stepdetectionPeak()* wird der Grenzwert angepasst definiert. Aus den erkannten Maxima wird der Mittelwert berechnet und davon ausgehend der Grenzwert nach einem gewissen Prozentsatz (**accVstep**) festgelegt. Damit ein Schritt erkannt wird, müssen die Maxima mindestens die Zeit (**steptimeD**) auseinander sein. In Tabelle 9-8 sind die Ergebnisse für die Outdoor Testläufe dargestellt. Zu vergleichen ist diese Tabelle mit Tabelle 9-2, welche die Ergebnisse für den *stepdetection()* Algorithmus darstellt.

Tabelle 9-8 Durchschnittliche Fehlschritte *stepdetectionPeak()* in Z-Achse H1-H6 Outdoor

		Abweichung in Schritten						
		accVstep (% des gemittelten Maximalwert)						
		79%	**80%**	**81%**	**82%**	**83%**	**84%**	**85%**
steptimeD (in Sekunden)	**0,36 s**	1,3	1,0	1,0	0,7	1,0	1,2	1,0
	0,37 s	1,2	1,0	1,0	0,7	1,0	1,2	1,0
	0,38 s	1,0	0,8	0,8	**0,5**	0,8	1,0	0,8
	0,39 s	0,8	0,7	**0,7**	0,7	1,0	1,2	1,0
	0,40 s	1,0	0,8	0,8	0,8	1,2	1,3	1,2
	0,41 s	1,2	1,0	1,0	1,0	1,3	1,5	1,3

min. Summe aller Nachbarwerte
minimale Fehlschritte

Sensordaten sind Geglättet mit der Funktion smooth(Gz,5)

Die Tabelle 9-9 enthält die Ergebnisse für die Indoor Testläufe. Diese Tabelle ist vergleichbar mit Tabelle 9-3 (Ergebnisse mit simple*stepdetection()*). In- und Outdoor Testläufe H1-EI12 wurden bei der Analyse verwendet. Deren Ergebnisse sind in Tabelle 9-10 dargestellt.

Tabelle 9-9 Durchschnittliche Fehlschritte *stepdetectionPeak()*in Z-Achse EI9-EI12 Outdoor

		Abweichung in Schritten					
		accVstep (% des gemittelten Maximalwert)					
		70%	**75%**	**80%**	**85%**	**90%**	**95%**
steptimeD (in Sekunden)	**0,40 s**	**0,5**	**0,5**	**0,5**	**0,5**	0,8	3,5
	0,41 s	**0,5**	**0,5**	**0,5**	**0,5**	0,8	3,5
	0,42 s	**0,5**	**0,5**	**0,5**	**0,5**	0,8	3,5
	0,43 s	**0,5**	**0,5**	**0,5**	**0,5**	0,8	3,5
	0,44 s	**0,5**	**0,5**	**0,5**	**0,5**	0,8	3,5
	0,45 s	1,0	1,0	1,0	1,0	1,0	3,8
	0,46 s	1,3	1,3	1,3	1,3	1,3	4,0

Sensordaten sind Geglättet mit der Funktion smooth(Gz,5)

Tabelle 9-10 Durchschn. Fehlschritte *simplestepdection()* in Z-Achse H1-EI12 In/Outdoor

		Abweichung in Schritten							
		accVstep (% des gemittelten Maximalwert)							
		76%	**77%**	**78%**	**79%**	**80%**	**81%**	**82%**	**83%**
steptimeD (in Sekunden)	**0,36 s**	1,2	1,1	1,0	1,0	0,8	0,8	0,6	0,8
	0,37 s	1,1	1,0	0,9	0,9	0,8	0,8	0,6	0,8
	0,38 s	1,0	0,9	0,8	0,8	0,7	0,7	**0,5**	0,7
	0,39 s	0,9	0,8	0,7	0,7	0,6	**0,6**	0,6	0,8
	0,40 s	1,0	0,9	0,8	0,8	0,7	0,7	0,7	0,9
	0,41 s	1,1	1,0	0,9	0,9	0,8	0,8	0,8	1,0
	0,42 s	1,0	0,9	0,8	0,8	0,7	0,7	0,7	0,9

Sensordaten sind Geglättet mit der Funktion smooth(Gz,5)

Tabelle 9-12 zeigt die Abweichungen im Detail für jene Parameter, welche zu minimalen mittleren Abweichungen führen.

Tabelle 9-11 Abweichungen im Detail - Der ermittelte Parameter für *stepdection2()*

	Abweichung in Schritten									
	Testlauf									
Testgebiete : Parameter, steptimeD; accVstep	**H1**	**H2**	**H3**	**H4**	**H5**	**H6**	**EI9**	**EI10**	**EI11**	**EI12**
Heim-Testgebiet (Outdoor): 0,38; 82%	0	0	0	-1	1	-1	-	-	-	-
EI-Testgebiet (Indoor): 0,42s; 80%	-	-	-	-	-	-	0	1	1	0
Heim/EI-Testgebiet: 0,38s; 82%	-1	0	0	-1	0	0	0	-1	1	-1

S2
HTC

9.2 INS-Algorithmus

Der INS-Algorithmus kombiniert, wie in 6.4 beschrieben, den Schritterkennungsalgorithmus und Daten des Orientierungssensors miteinander. Als Parameter für den INS-Algorithmus kommt noch die Schrittweite hinzu. Aus den Referenzstrecken und den Testläufen mit den gezählten Schritten ergeben sich für die Schrittweite Outdoor Testläufe 0,68 m und für die Indoor Testläufe 0,66 m. Die Schrittweite wurde jeweils auf die geringste Abweichung zu Soll-Endposition hin optimiert. In Tabelle 9-12 sind Ergebnisse mit unterschiedlichen Schrittweiten dargestellt.

Tabelle 9-12 INS-Algorithmus mit verschieden Schrittweiten

		Abweichung von Endposition zur Referenzposition in m											
		Testlauf											
		Outdoor							**Indoor**				
		Testlauf H1	**Testlauf H2**	**Testlauf H3**	**Testlauf H4**	**Testlauf H5**	**Testlauf H6**	mittlere Abw.	**Testlauf EI9**	**Testlauf EI10**	**Testlauf EI11**	**Testlauf EI12**	mittlere Abw.
Schrittweite (in m)	**0,64**	9,1	9,4	8,8	3,0	6,0	5,2	6,9	6,3	3,8	0,7	5,3	4,0
	0,65	8,7	9,1	8,6	2,3	5,8	4,6	6,5	6,4	3,6	5,0	5,3	**3,9**
	0,66	8,4	9,0	8,4	1,9	5,8	4,1	6,2	6,6	3,5	0,6	5,2	4,0
	0,67	8,2	8,9	8,3	1,8	5,9	3,8	**6,1**	6,8	3,7	0,8	5,2	4,1
	0,68	8,1	8,9	8,3	2,2	6,2	3,7	6,2	7,1	4,0	1,1	5,2	4,4
	0,69	8,0	9,0	8,3	2,8	6,5	3,8	6,4					
	0,7	8,1	9,2	8,5	3,6	7,0	4,1	6,8					

S2
HTC

Tabelle 6-3 fasst die Ergebnisse der Testläufe von H1-H6 zusammen und stellt INS-Algorithmus und GPS-Aufzeichnung gegenüber. Die detaillierten Ergebnisse sind im Anschluss aufgeführt. Die Tabelle 6-4 zeigt die Ergebnisse, die mit dem INS-Algorithmus für die Testläufe EI9 bis EI12 im Mittel erzielt werden konnten. Detaillierte Ergebnisse sind im Anschluss angeführt. In den Abschnitten 9.2.1 bis 9.2.6 werden die Ergebnisse des INS-Algorithmus im Detail für die Testläufe H1-H6 im Heim-Testgebiet gezeigt und stellen diesen der jeweiligen GPS-Aufzeichnung gegenüber.

Tabelle 9-13 Vergleich GPS zu INS H1-H6

Zusammenfassung der Ergebnisse		
Mittelwerte H1-H6	**GPS**	**INS-Algo.**
Abweichung Startposition	9,7 m	n.a.
Abweichung Endposition	14,5 m	6,2 m
Mittlere Abweichung	10,2 m	3,4 m
Maximale Abweichung	20,0 m	6,5 m
Minimale Abweichung	3,0 m	n.a.

Tabelle 9-14 Ergebnisse INS-Algo. EI9-EI12

Zusammenfassung der Ergebnisse	
Mittelwerte EI9-EI12	
Mittlere Abweichung Lon.	1,1 m
Mittlere Abweichung Lat.	2,2 m
Mittlere Abweichung	2,5 **m**
Abweichung Lon. Endpunkt	1,8 m
Abweichung Lat. Endpunkt	3,5 m
Abweichung Endpunkt	4,0 **m**
maximale Abweichung	4,5 m

Von Kapitel 9.2.7 bis 9.2.10 sind die detaillierten Ergebnisse der Analyse des INS-Algorithmus mit den Testläufen EI9-EI11 angeführt.

9.2.1 Testlauf H1 im Heim-Testgebiet auf Teststrecke Abschnitt 1 mit S2

Abbildung 9-1 TL-H1 S2 Schritterkennung / INS-Algorithmus

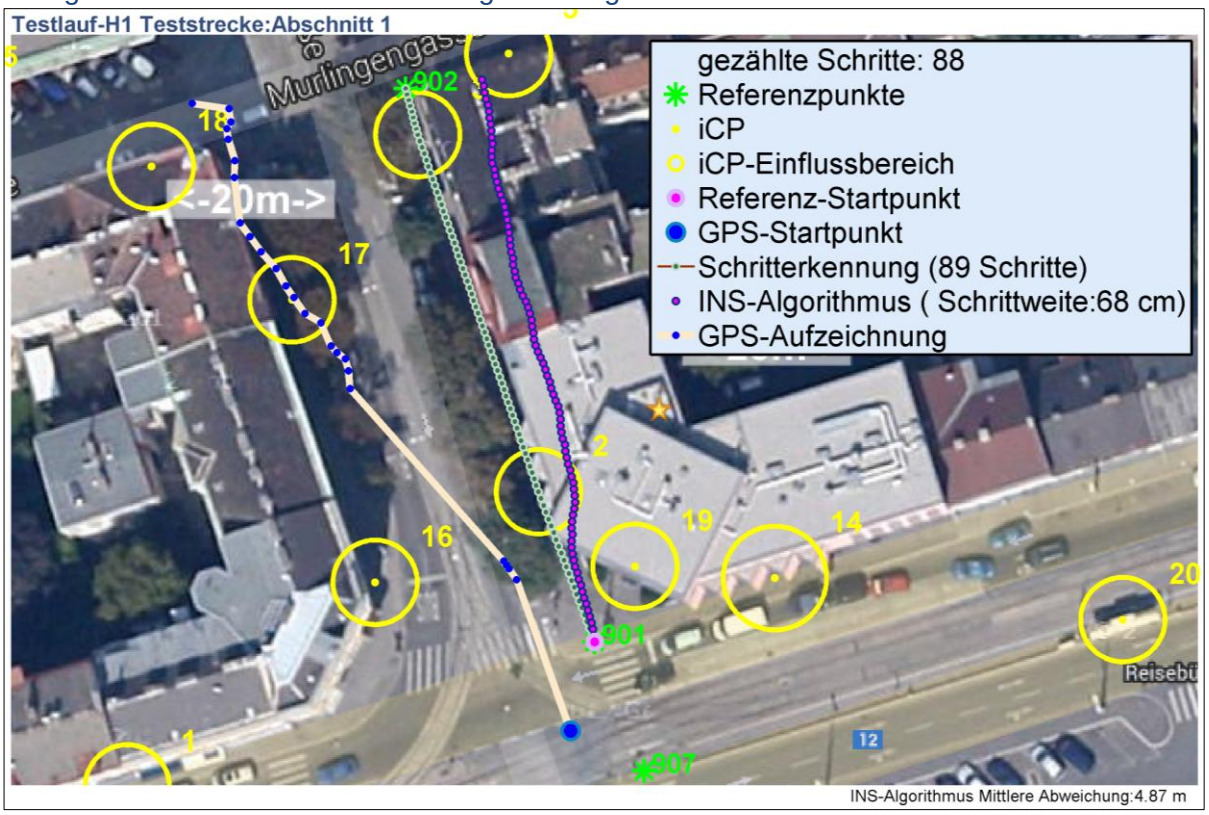


Tabelle 9-15 TL-H1 Zusammenfassung

Zusammenfassung der Ergebnisse		
TL-H1	GPS	INS-Algo.
Abweichung Startposition	9,6 m	n.a.
Abweichung Endposition	22,4 m	8,1 m
Mittlere Abweichung	15,5 m	4,9 m
Maximale Abweichung	23,8 m	8,2 m
Minimale Abweichung	5,7 m	n.a.

Tabelle 9-16 TL-EI1 GPS

GPS Aufzeichnung Ergebnisse		
TL-H1		
Mittlere Abweichung Lon.	12,1	m
Mittlere Abweichung Lat.	5,0	m
Mittlere Abweichung	**15,5**	**m**
Abweichung Lon. Startpunkt	2,5	m
Abweichung Lat. Startpunkt	9,3	m
Abweichung Startpunkt	**9,6**	**m**
Abweichung Lon. Endpunkt	22,3	m
Abweichung Lat. Endpunkt	1,5	m
Abweichung Endpunkt	**22,4**	**m**
maximale Abweichung	23,8	m
minimale Abweichung	5,7	m

Tabelle 9-17 TL-H1 INS-Algorithmus

INS-Algorithmus Ergebnisse		
TL-H1		
Mittlere Abweichung Lon.	4,8	m
Mittlere Abweichung Lat.	0,6	m
Mittlere Abweichung	4,9	m
Abweichung Lon. Endpunkt	8,0	m
Abweichung Lat. Endpunkt	0,9	m
Abweichung Endpunkt	8,1	m
maximale Abweichung	8,2	m

9.2.2 Testlauf H2 im Heim-Testgebiet auf Teststrecke Abschnitt 1 mit S2

Abbildung 9-2 TL-H2 Schritterkennung / INS-Algorithmus

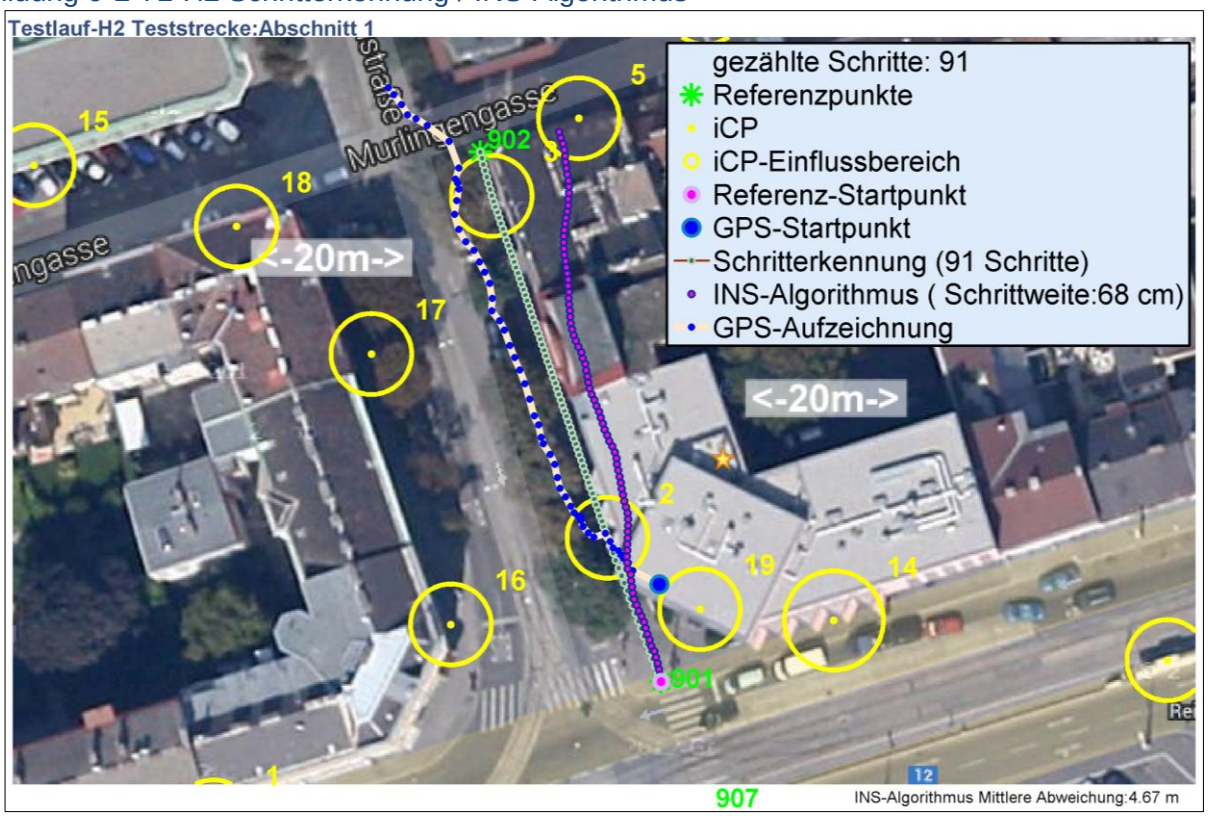


Tabelle 9-18 TL-H1 Zusammenfassung

Zusammenfassung der Ergebnisse		
TL-H2	**GPS**	**INS-Algo.**
Abweichung Startposition	10,6 m	n.a.
Abweichung Endposition	12,2 m	8,9 m
Mittlere Abweichung	4,9 m	4,7 m
Maximale Abweichung	12,2 m	9,0 m
Minimale Abweichung	0,5 m	n.a.

Tabelle 9-19 TL-EI1 GPS

GPS Aufzeichnung Ergebnisse		
TL-H2		
Mittlere Abweichung Lon.	3,0	m
Mittlere Abweichung Lat.	3,2	m
Mittlere Abweichung	**4,9**	**m**
Abweichung Lon. Startpunkt	0,2	m
Abweichung Lat. Startpunkt	10,6	m
Abweichung Startpunkt	**10,6**	**m**
Abweichung Lon. Endpunkt	10,0	m
Abweichung Lat. Endpunkt	7,0	m
Abweichung Endpunkt	**12,2**	**m**
maximale Abweichung	12,2	m
minimale Abweichung	0,5	m

Tabelle 9-20 TL-H1 INS-Algorithmus

INS-Algorithmus Ergebnisse		
TL-H2		
Mittlere Abweichung Lon.	4,3	m
Mittlere Abweichung Lat.	1,3	m
Mittlere Abweichung	4,7	m
Abweichung Lon. Endpunkt	8,6	m
Abweichung Lat. Endpunkt	2,2	m
Abweichung Endpunkt	8,9	m
maximale Abweichung	9,0	m

9.2.3 Testlauf H3 im Heim-Testgebiet auf Teststrecke Abschnitt 1 mit S2

Abbildung 9-3 TL-H3 Schritterkennung / INS-Algorithmus

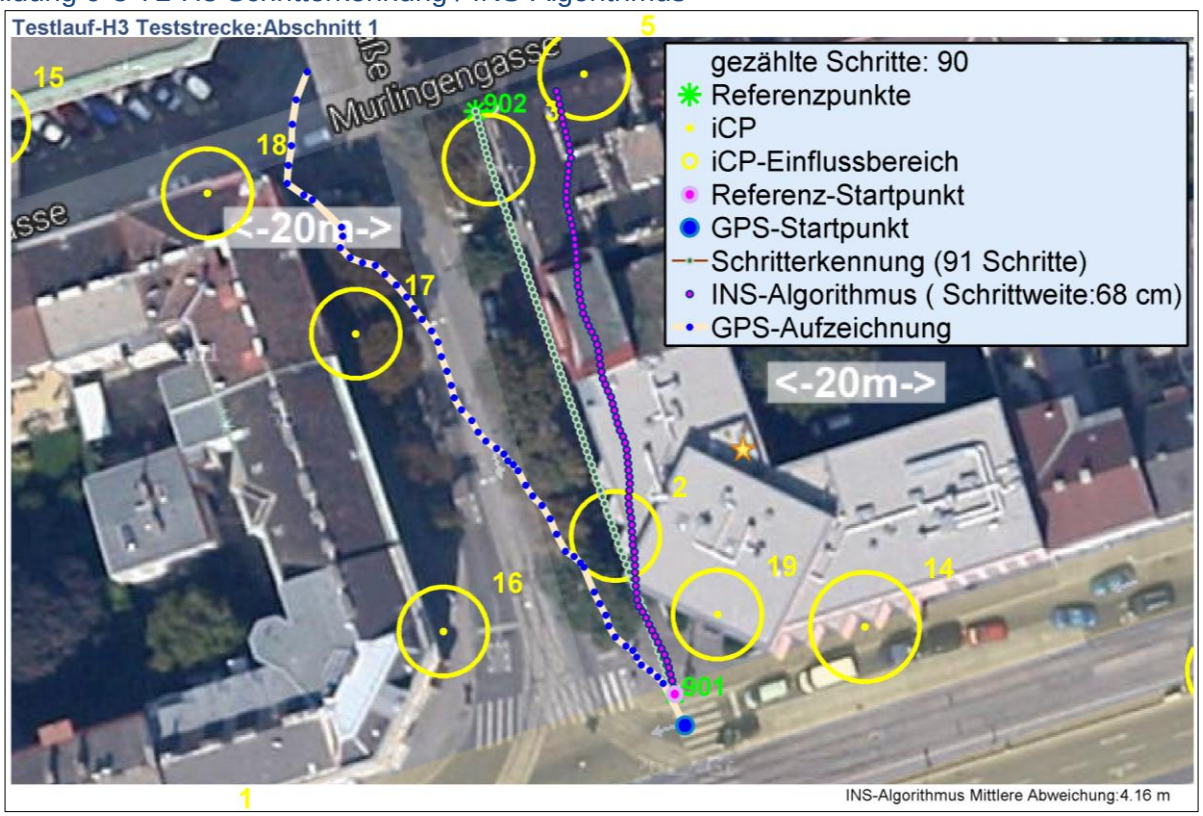


Tabelle 9-21 TL-H3 Zusammenfassung

Zusammenfassung der Ergebnisse		
TL-H3	**GPS**	**INS-Algo.**
Abweichung Startposition	3,2 m	n.a.
Abweichung Endposition	17,1 m	8,3 m
Mittlere Abweichung	9,6 m	4,2 m
Maximale Abweichung	20,0 m	8,3 m
Minimale Abweichung	1,1 m	n.a.

Tabelle 9-22 TL-H3 GPS

GPS Aufzeichnung Ergebnisse		
TL-H3		
Mittlere Abweichung Lon.	7,8	m
Mittlere Abweichung Lat.	4,7	m
Mittlere Abweichung	**9,6**	**m**
Abweichung Lon. Startpunkt	1,0	m
Abweichung Lat. Startpunkt	3,1	m
Abweichung Startpunkt	**3,2**	**m**
Abweichung Lon. Endpunkt	16,7	m
Abweichung Lat. Endpunkt	3,9	m
Abweichung Endpunkt	**17,1**	**m**
maximale Abweichung	20,0	m
minimale Abweichung	1,1	m

Tabelle 9-23 TL-H3 INS-Algorithmus

INS-Algorithmus Ergebnisse		
TL-H3		
Mittlere Abweichung Lon.	4,0	m
Mittlere Abweichung Lat.	0,6	m
Mittlere Abweichung	4,2	m
Abweichung Lon. Endpunkt	8,0	m
Abweichung Lat. Endpunkt	2,0	m
Abweichung Endpunkt	8,3	m
maximale Abweichung	8,3	m

9.2.4 Testlauf H4 im Heim-Testgebiet auf Teststrecke Abschnitt 1 mit HTC

Abbildung 9-4 TL-H4 Schritterkennung / INS-Algorithmus

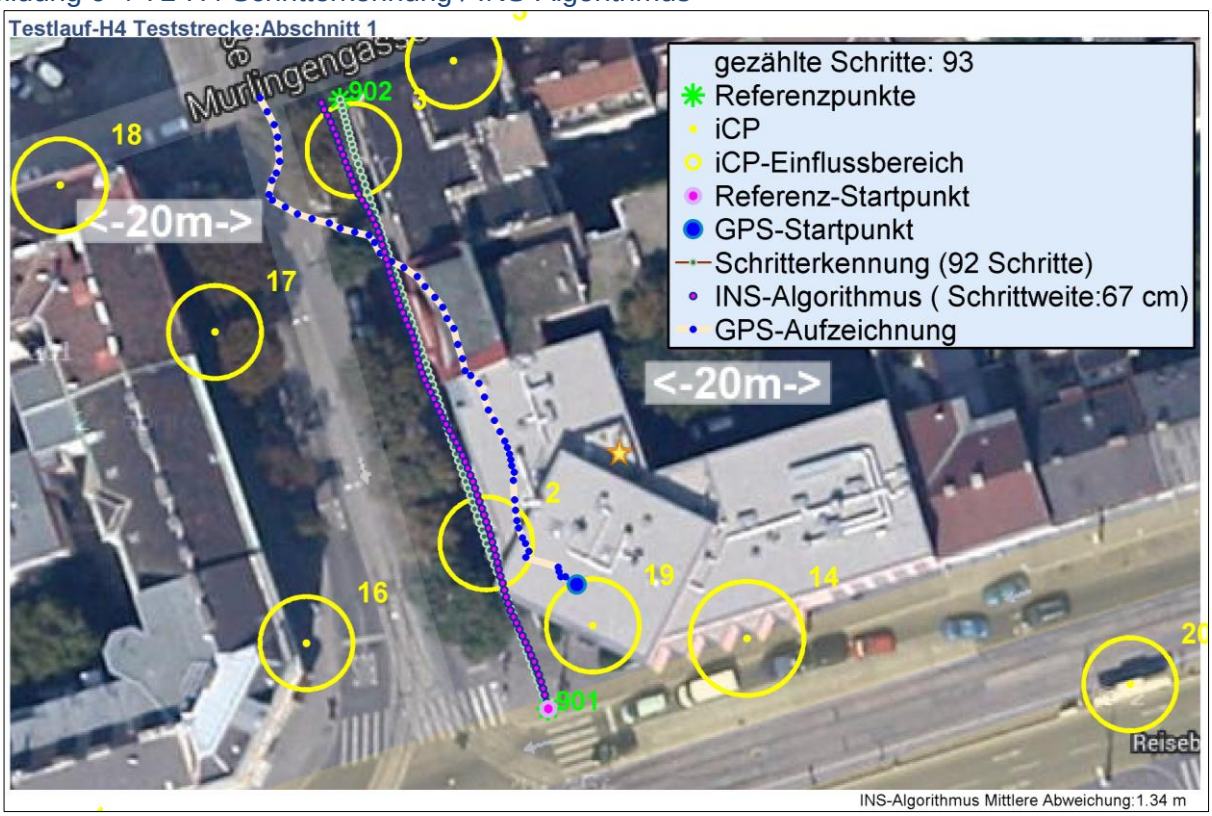


Tabelle 9-24 TL-H4 Zusammenfassung

Zusammenfassung der Ergebnisse		
TL-H4	**GPS**	**INS-Algo.**
Abweichung Startposition	12,1 m	n.a.
Abweichung Endposition	7,6 m	1,8 m
Mittlere Abweichung	5,2 m	1,3 m
Maximale Abweichung	12,1 m	2,4 m
Minimale Abweichung	0,1 m	n.a.

Tabelle 9-25 TL-H4 GPS

GPS Aufzeichnung Ergebnisse	
TL-H4	
Mittlere Abweichung Lon.	3,5 m
Mittlere Abweichung Lat.	3,3 m
Mittlere Abweichung	**5,2 m**
Abweichung Lon. Startpunkt	2,7 m
Abweichung Lat. Startpunkt	11,8 m
Abweichung Startpunkt	**12,1 m**
Abweichung Lon. Endpunkt	7,6 m
Abweichung Lat. Endpunkt	0,2 m
Abweichung Endpunkt	**7,6 m**
maximale Abweichung	12,1 m
minimale Abweichung	0,1 m

Tabelle 9-26 TL-H4 INS-Algorithmus

INS-Algorithmus Ergebnisse	
TL-H4	
Mittlere Abweichung Lon.	0,5 m
Mittlere Abweichung Lat.	1,2 m
Mittlere Abweichung	1,3 m
Abweichung Lon. Endpunkt	1,8 m
Abweichung Lat. Endpunkt	0,3 m
Abweichung Endpunkt	1,8 m
maximale Abweichung	2,4 m

9.2.5 Testlauf H5 im Heim-Testgebiet auf Teststrecke Abschnitt 1 mit HTC

Abbildung 9-5 TL-H5 Schritterkennung / INS-Algorithmus

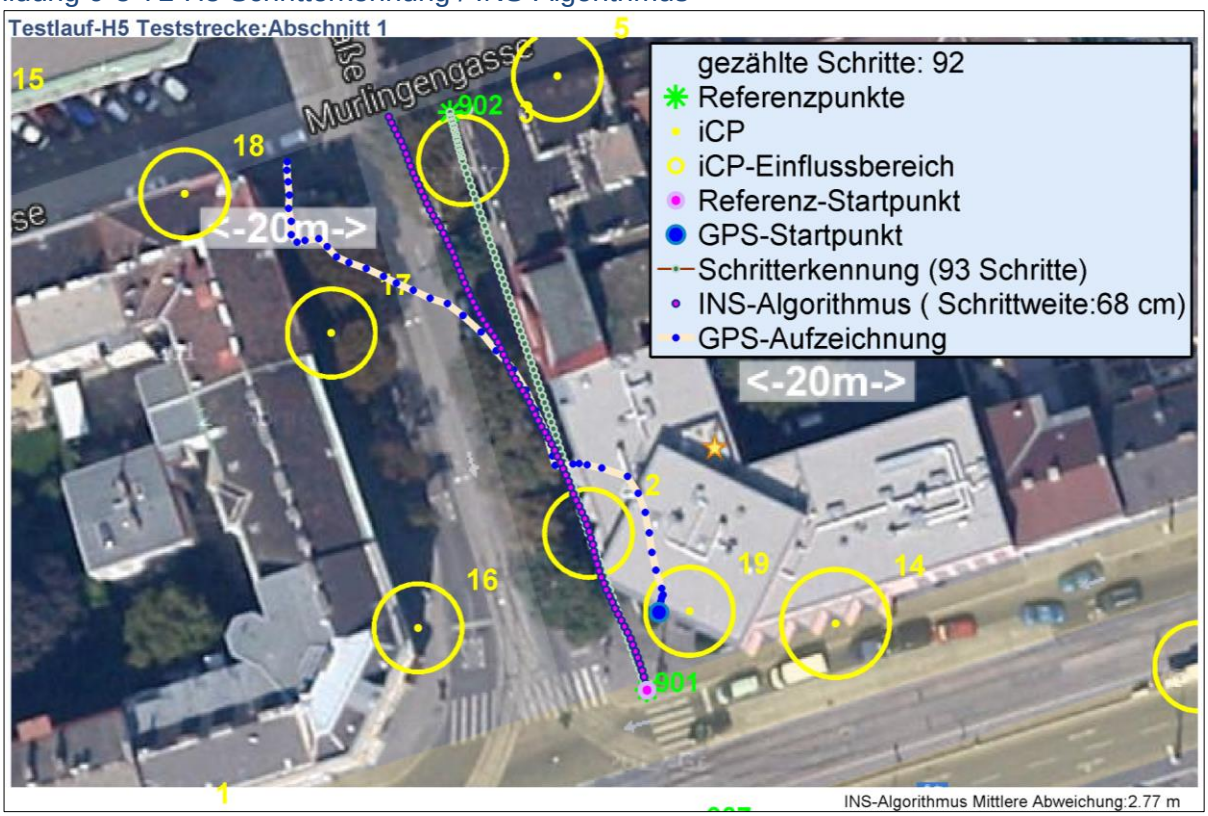


Tabelle 9-27 TL-H5 Zusammenfassung

Zusammenfassung der Ergebnisse		
TL-H5	**GPS**	**INS-Algo.**
Abweichung Startposition	7,9 m	n.a.
Abweichung Endposition	17,1 m	6,2 m
Mittlere Abweichung	10,0 m	2,8 m
Maximale Abweichung	18,9 m	6,2 m
Minimale Abweichung	4,4 m	n.a.

Tabelle 9-28 TL-H5 GPS

GPS Aufzeichnung Ergebnisse		
TL-H5		
Mittlere Abweichung Lon.	7,6	m
Mittlere Abweichung Lat.	5,6	m
Mittlere Abweichung	**10,0**	**m**
Abweichung Lon. Startpunkt	1,2	m
Abweichung Lat. Startpunkt	7,8	m
Abweichung Startpunkt	**7,9**	**m**
Abweichung Lon. Endpunkt	16,3	m
Abweichung Lat. Endpunkt	4,9	m
Abweichung Endpunkt	**17,1**	**m**
maximale Abweichung	18,9	m
minimale Abweichung	4,4	m

Tabelle 9-29 TL-H5 INS-Algorithmus

Zusammenfassung der Ergebnisse		
TL-H5	**GPS**	**INS-Algo.**
Abweichung Startposition	7,9 m	n.a.
Abweichung Endposition	17,1 m	6,2 m
Mittlere Abweichung	10,0 m	2,8 m
Maximale Abweichung	18,9 m	6,2 m
Minimale Abweichung	4,4 m	n.a.

9.2.6 Testlauf H6 im Heim-Testgebiet auf Teststrecke Abschnitt 1 mit HTC

Abbildung 9-6 TL-H6 Schritterkennung / INS-Algorithmus

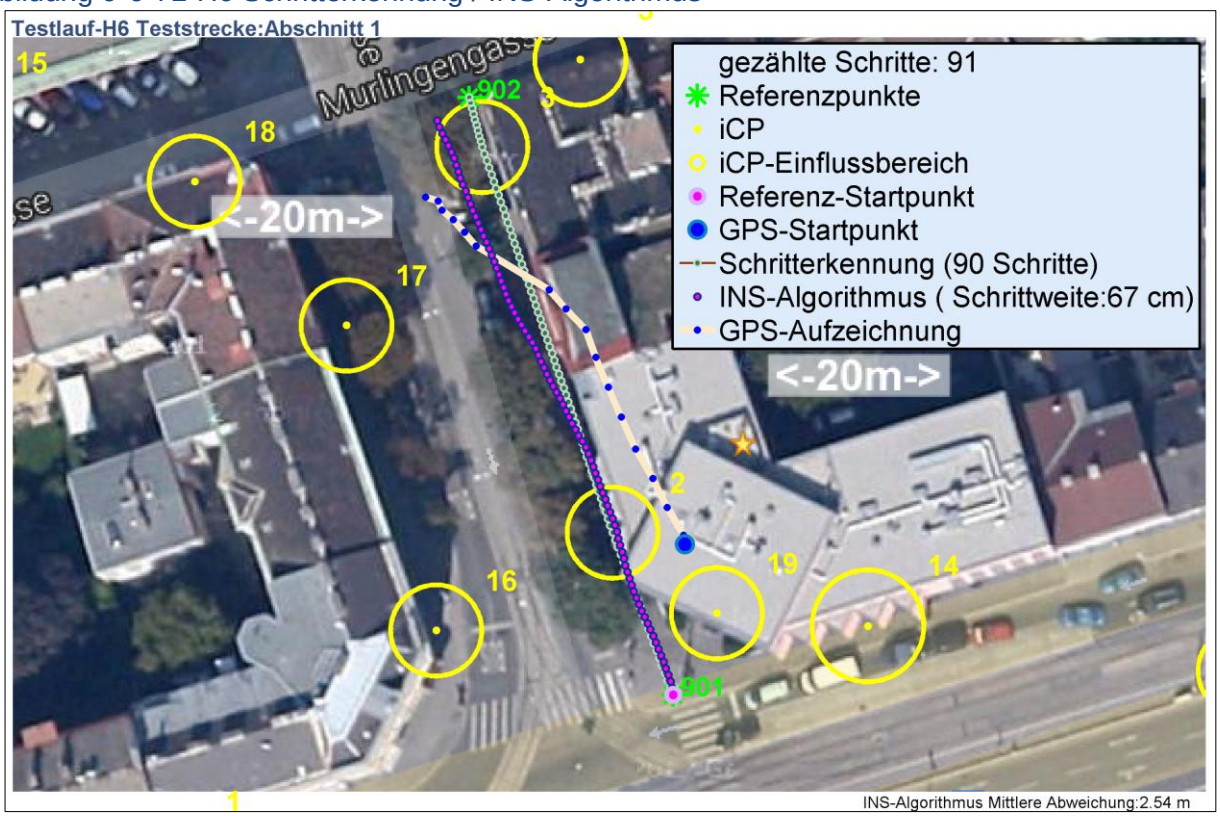


Tabelle 9-30 TL-H4 Zusammenfassung

Zusammenfassung der Ergebnisse		
TL-H6	GPS	INS-Algo.
Abweichung Startposition	14,6 m	n.a.
Abweichung Endposition	10,5 m	3,8 m
Mittlere Abweichung	16,2 m	2,5 m
Maximale Abweichung	32,7 m	4,6 m
Minimale Abweichung	6,2 m	n.a.

Tabelle 9-31 TL-H4 GPS

GPS Aufzeichnung Ergebnisse		
TL-H6		
Mittlere Abweichung Lon.	8,6	m
Mittlere Abweichung Lat.	13,1	m
Mittlere Abweichung	**16,2**	**m**
Abweichung Lon. Startpunkt	1,1	m
Abweichung Lat. Startpunkt	14,6	m
Abweichung Startpunkt	**14,6**	**m**
Abweichung Lon. Endpunkt	4,2	m
Abweichung Lat. Endpunkt	9,6	m
Abweichung Endpunkt	**10,5**	**m**
maximale Abweichung	32,7	m
minimale Abweichung	6,2	m

Tabelle 9-32 TL-H4 INS-Algorithmus

INS-Algorithmus Ergebnisse		
TL-H6		
Mittlere Abweichung Lon.	1,0	m
Mittlere Abweichung Lat.	2,3	m
Mittlere Abweichung	2,5	m
Abweichung Lon. Endpunkt	3,1	m
Abweichung Lat. Endpunkt	2,2	m
Abweichung Endpunkt	3,8	m
maximale Abweichung	4,6	m

9.2.7 Testlauf EI9 im EI-Testgebiet auf Teststrecke 3S-Gang1a mit HTC

Abbildung 9-7 TL-EI9 Schritterkennung / INS-Algorithmus

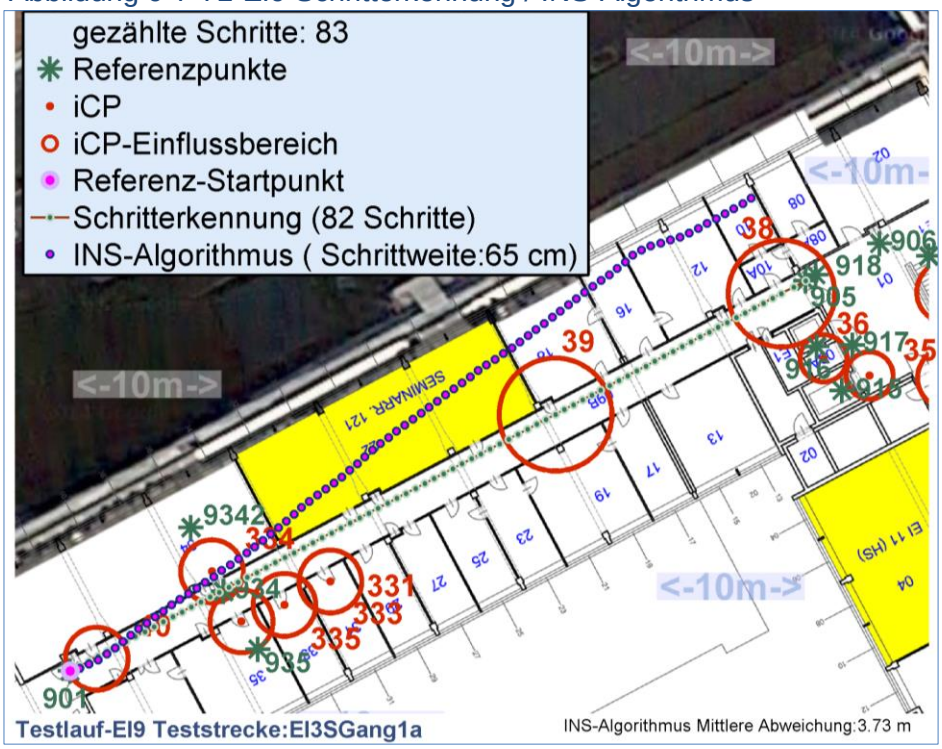


Tabelle 9-33

TL-EI9 INS-Algorithmus

INS-Algorithmus Ergebnisse		
TL-EI9		
Mittlere Abweichung Lon.	2,6	m
Mittlere Abweichung Lat.	2,7	m
Mittlere Abweichung	3,7	m
Abweichung Lon. Endpunkt	3,5	m
Abweichung Lat. Endpunkt	5,3	m
Abweichung Endpunkt	6,3	m
maximale Abweichung	6,8	m

Tabelle 9-34

TL-EI9 Orientierung

Orientierung	
Sollwert	62,3 °
mittlere Istwert	53,9°
Differenz	8,4 °

9.2.8 Testlauf EI10 im EI-Testgebiet auf Teststrecke 3S-Gang1b mit HTC

Abbildung 9-8 TL-EI10 Schritterkennung / INS-Algorithmus

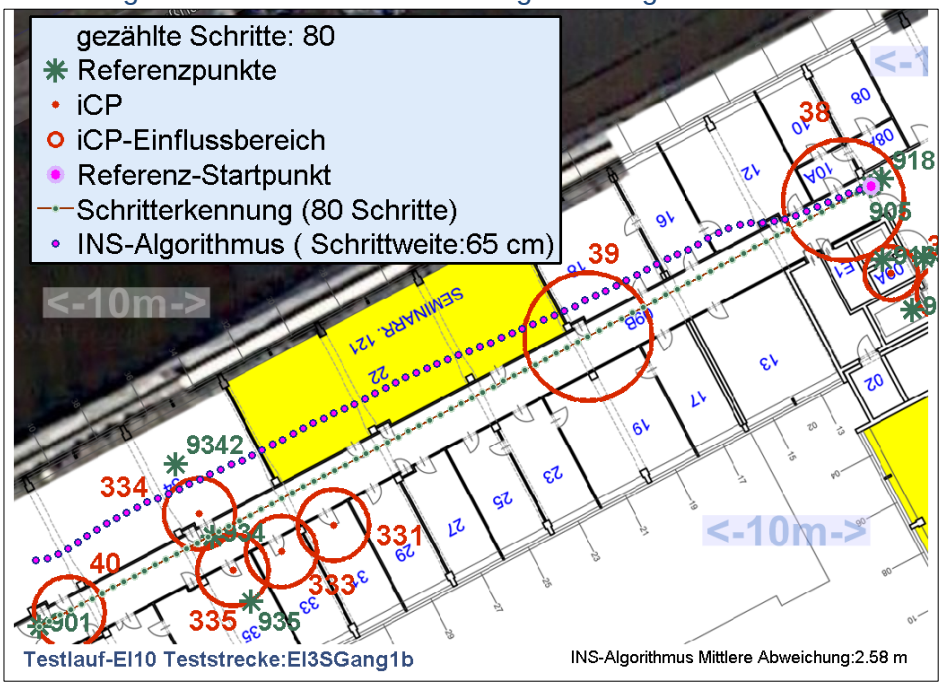


Tabelle 9-35

TL-EI10 INS-Algorithmus

INS-Algorithmus Ergebnisse		
TL-EI10		
Mittlere Abweichung Lon.	0,2	m
Mittlere Abweichung Lat.	2,6	m
Mittlere Abweichung	2,6	m
Abweichung Lon. Endpunkt	0,2	m
Abweichung Lat. Endpunkt	3,7	m
Abweichung Endpunkt	3,7	m
maximale Abweichung	3,9	m

Tabelle 9-36

TL-EI10 Orientierung

Orientierung	
Sollwert	242,1 °
mittlere Istwert	244,5°
Differenz	--2,4 °

9.2.9 Testlauf EI11 im EI-Testgebiet auf Teststrecke 3S-Gang2a mit HTC

Abbildung 9-9 TL-EI11 Schritterkennung / INS-Algorithmus

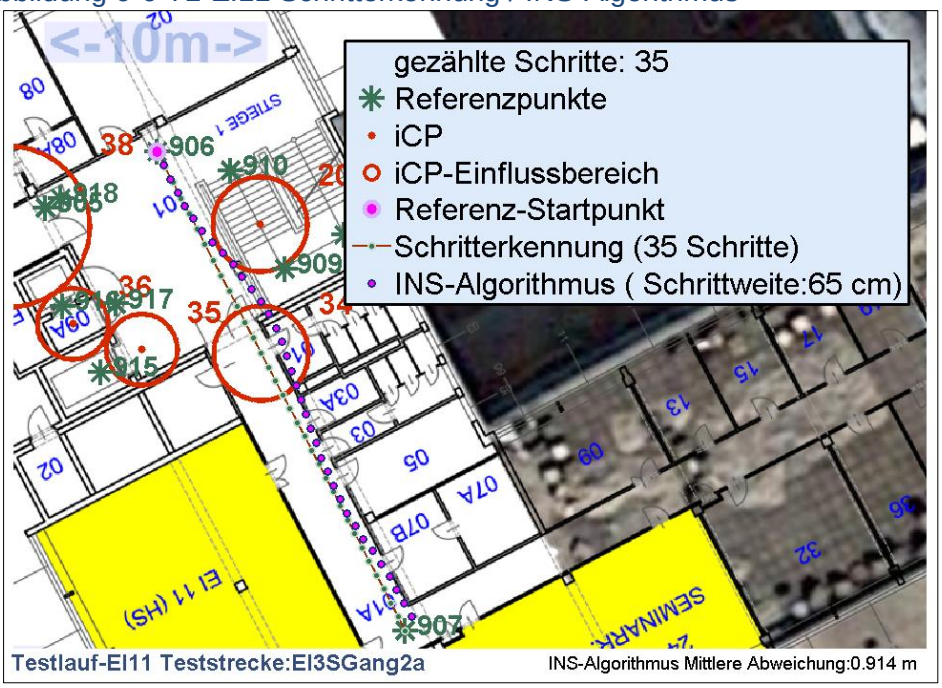


Tabelle 9-37

TL-E11 INS-Algorithmus

INS-Algorithmus Ergebnisse		
TL-EI11		
Mittlere Abweichung Lon.	0,2	m
Mittlere Abweichung Lat.	0,9	m
Mittlere Abweichung	0,9	m
Abweichung Lon. Endpunkt	0,3	m
Abweichung Lat. Endpunkt	0,6	m
Abweichung Endpunkt	0,7	m
maximale Abweichung	1,7	m

Tabelle 9-38

TL-EI11 Orientierung

Orientierung	
Sollwert	152,4 °
mittlere Istwert	151,2 °
Differenz	1,2 °

9.2.10 Testlauf EI11 im EI-Testgebiet auf Teststrecke 3S-Gang1b mit HTC

Abbildung 9-10 TL-EI11 Schritterkennung / INS-Algorithmus

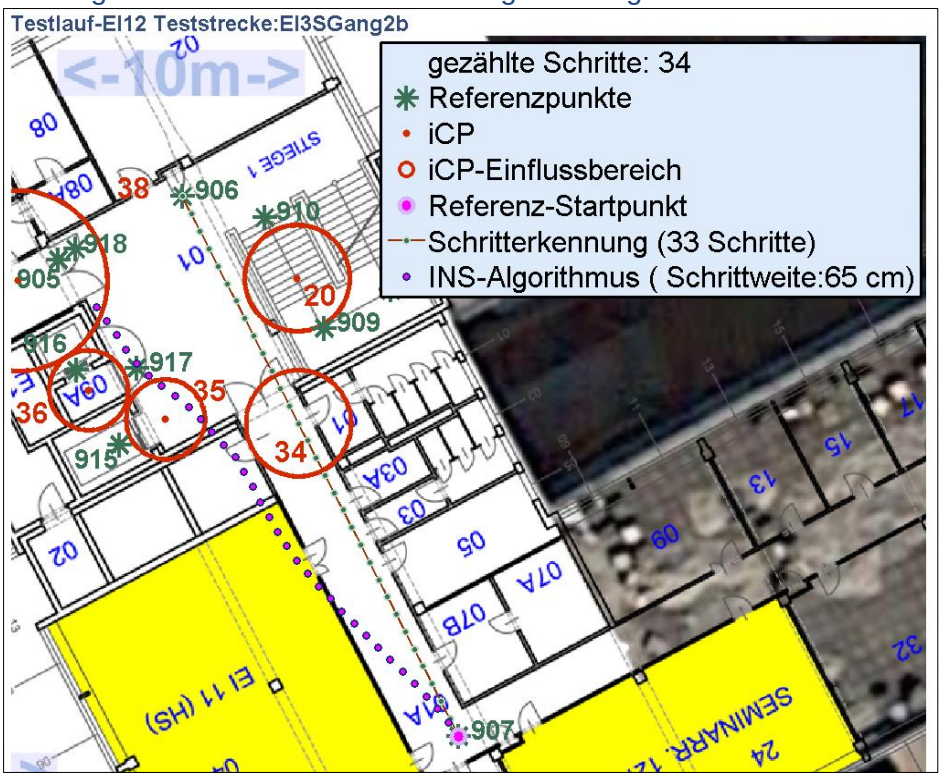


Tabelle 9-39

TL-E12 INS-Algorithmus

INS-Algorithmus Ergebnisse		
TL-EI12		
Mittlere Abweichung Lon.	1,5	m
Mittlere Abweichung Lat.	2,5	m
Mittlere Abweichung	2,9	m
Abweichung Lon. Endpunkt	3,3	m
Abweichung Lat. Endpunkt	4,2	m
Abweichung Endpunkt	5,3	m
maximale Abweichung	5,5	m

Tabelle 9-40

TL-EI11 Orientierung

Orientierung	
Sollwert	332,6 °
mittlere Istwert	320,2 °
Differenz	12,4 °

9.3 Erkennen von iCP in einem Testrun

Dieses Kapitel zeigt die Ergebnisse der Analyse des Algorithmus `findCPinRun()`, der in 6.5.1 beschrieben ist. In Tabelle 9-41 sind die Ergebnisse der Testläufe für das Heim-Testgebiet zusammengefasst. Für das EI-Testgebiet sind die Ergebnisse in Tabelle 9-42 zusammengeführt. Verglichen werden in den Tabellen die Anwendung der Glättungsfunktion auf die ermittelten Abstands-Werte sowie die erzielten Genauigkeiten mit zwei Testgeräten.

Tabelle 9-41 Zusammenfassung Ergebnisse iCP Erkennung im Heim-Testgebiet

	Abweichung in Schritten										*gemittelte Schritte*
	Testlauf										
	S2					**HTC**					
	H1	**H2**	**H3**	**H8**	**H1 0**	**H4**	**H5**	**H6**	**H7**	**H9**	
ohne Glättung	13,5	1,5	2,5	6,5	11,8	0,5	6	3,5	3	2,2	**5,1**
gemittelte Schritte	**7,2**					**3,0**					--
geglättet	13,5	1	2,5	6,4	8,8	0	1,5	0,5	1,5	1,7	**3,7**
gemittelte Schritte	**6,4**					**1,0**					--

Tabelle 9-42 Zusammenfassung Ergebnisse iCP Erkennung im EI-Testgebiet

	Abweichung in Schritten												*gemittelte Schritte*
	Testlauf												
	EG - Erdgeschoss					**3S - 3. Stock**							
	EI1	**EI2**	**EI3**	**EI4**	**EI5**	**EI6**	**EI7**	**EI8**	**EI9**	**EI10**	**EI11**	**E12**	
ohne Glättung	8	1,3	1	4	9	6,8	0,5	7,5	6,5	9,5	0	5	**4,9**
gemittelte Schritte	**4,7**					**5,1**							--
geglättet	6	0,7	0,5	2,7	5,7	5,5	0,25	7,8	0	9,5	0	4	**3,6**
gemittelte Schritte	**3,1**					**3,9**							--

Tabelle 9-43, Tabelle 9-44 und Tabelle 9-45 zeigen die Ergebnisse der Testläufe im Detail.

Tabelle 9-43 Heim-Testgebiet iCP Erkennung H1-H6

Werte in Tabelle sind in Schritten angegeben. *Teststrecke:* ***Abschnitt 1***			**iCP**		mittlere Abweichung von iCP
			2	**3**	
H1		*Referenzschritte*	*35*	*111*	-
Gerät:	S2	ohne Glättung	8	111	13,5
Anz. Scans:	19	keine Verbesserung bei Glättung			13,5
H2		*Referenzschritte*	*26*	*109*	-
Gerät:	S2	ohne Glättung	25	107	1,5
Anz. Scans:	19	geglättet mit sValue=7	26	108	1
H3		*Referenzschritte*	*31*	*107*	-
Gerät:	S2	ohne Glättung	27	106	2,5
Anz. Scans:	18	keine Verbesserung bei Glättung			13,5
H4		*Referenzschritte*	*34*	*109*	-
Gerät:	HTC	ohne Glättung	33	109	0,5
Anz. Scans:	94	geglättet mit sValue=18	33	109	0
H5		*Referenzschritte*	*31*	*106*	-
Gerät:	HTC	ohne Glättung	40	103	6
Anz. Scans:	89	geglättet mit sValue=14	32	104	1,5
H6		*Referenzschritte*	*31*	*107*	-
Gerät:	HTC	ohne Glättung	28	103	3,5
Anz. Scans:	89	geglättet mit sValue=18	32	107	0,5

Tabelle 9-44 Heim-Testgebiet iCP Erkennung H7-H8

Teststrecke: ***UmHeim***			**iCP**							mittlere Abweichung von iCP
			2	3	5	9	10	11	12	14	
H7		*Referenzschritte*	*28*	*87*	*119*	*238*	*267*	*342*	*379*	*471*	-
Gerät:	HTC	ohne Glättung	26	90	119	235	266	336	379	462	3,0
Anz. Scans:	347	geglättet mit sValue=6	27	88	119	237	268	338	381	473	1,5
H8		*Referenzschritte*	*28*	*87*	*119*	*238*	*267*	*342*	*379*	*471*	-
Gerät:	S2	ohne Glättung	31	96	120	234	274	330	368	465	6,5
Anz. Scans:	60	geglättet mit sValue=6	7	88	120	226	265	330	376	457	6,4

Tabelle 9-45 Heim-Testgebiet iCP Erkennung H9-H10

Teststrecke: ***Bahnhof nach 7***			iCP						mittlere Abweichung von iCP
			2	**3**	**5**	**7**	**9**	**24**	
H9		*Referenzschritte*	*117*	*176*	*202*	*383*	*320*	*1*	-
Gerät:	HTC	ohne Glättung	114	177	207	387	320	1	2,2
Anz. Scans:	274	geglättet mit sValue=7	117	177	206	387	321	1	1,7
H10		*Referenzschritte*	*117*	*176*	*202*	*383*	*320*	*1*	-
Gerät:	S2	ohne Glättung	114	179	210	386	307	35	11,8
Anz. Scans:	54	geglättet mit sValue=7	114	171	218	398	323	19	8,8

In den Tabelle 9-46 bis Tabelle 9-50 sind die Details für die Testläufe EI1-EI5 im EI-Testgebiet im Eingangsbereich des Gebäudes angeführt. Die Werte in den Tabellen sind in Schritten angegeben.

Tabelle 9-46 EI-Testgebiet iCP Erkennung EI1 EG-11-L1

Teststrecke: ***EG11L1***			iCP		mittlere Abweichung von iCP
			5	**111**	
EI1		*Referenzschritte*	*56*	*46*	-
Gerät:	HTC	ohne Glättung	41	47	8,0
Anz. Scans:	65	geglättet mit sValue=5	47	43	6,0

Tabelle 9-47 EI-Testgebiet iCP Erkennung EI2 EG-12-S1

Teststrecke: ***EG12S1***			iCP			mittlere Abweichung von iCP
			2	**3**	**9**	
EI2		*Referenzschritte*	*63*	*108*	*127*	-
Gerät:	HTC	ohne Glättung	63	108	127	1,3
Anz. Scans:	120	geglättet mit sValue=7	62	108	126	0,7

Tabelle 9-48 EI-Testgebiet iCP Erkennung EI3 EG-1-L2

Teststrecke: ***EG1L2***			iCP		mittlere Abweichung von iCP
			1	**6**	
EI3		*Referenzschritte*	*49*	*76*	-
Gerät:	HTC	ohne Glättung	47	76	1,0
Anz. Scans:	80	geglättet mit sValue=5	49	76	0,5

Tabelle 9-49 EI-Testgebiet iCP Erkennung EI4 EG-1-S1

Teststrecke: ***EG1S1***			iCP			mittlere Abweichung von iCP
			1	**3**	**9**	
EI4		*Referenzschritte*	*47*	*67*	*86*	-
Gerät:	HTC	ohne Glättung	50	75	85	4,0
Anz. Scans:	88	geglättet mit sValue=8	47	67	86	2,7

Tabelle 9-50 EI-Testgebiet iCP Erkennung EI5 EG-2-S1

Teststrecke: ***EG2S1***			iCP			mittlere Abweichung von iCP
			3	**9**	**12**	
EI5		*Referenzschritte*	*63*	*77*	*47*	
Gerät:	HTC	ohne Glättung	61	53	48	9,0
Anz. Scans:	75	geglättet mit sValue=7	57	78	37	5,7

Die Tabelle 9-51 bis Tabelle 9-57 zeigen die erzielten Ergebnisse für die Testläufe im EI-Testgebiet im 3. Stock des Gebäudes. Die Werte in den Tabelle sind in Schritten angegeben.

Tabelle 9-51 EI-Testgebiet iCP Erkennung EI6 3S-L1-335

Teststrecke: ***3SL1335***			iCP				mittlere Abweichung von iCP
			35	**38**	**39**	**335**	
EI6		*Referenzschritte*	*4*	*21*	*49*	*91*	-
Gerät:	HTC	ohne Glättung	2	29	32	91	6,8
Anz. Scans:	84	geglättet mit sValue=4	2	23	33	93	5,5

Tabelle 9-52 EI-Testgebiet iCP Erkennung EI7 3S-L2-334

Teststrecke: ***3SL2334***			iCP				mittlere Abweichung von iCP
			36	**38**	**39**	**334**	
EI7		*Referenzschritte*	*1*	*7*	*26*	*72*	-
Gerät:	HTC	ohne Glättung	1	7	28	72	0,5
Anz. Scans:	107	geglättet mit sValue=7	1	8	26	72	0,25

Tabelle 9-53 EI-Testgebiet iCP Erkennung EI8 3S-S1-Z1

Teststrecke: ***3SZ1***			iCP				mittlere Abweichung von iCP
			20	**38**	**39**	**334**	
EI8		*Referenzschritte*	*8*	*31*	*60*	*103*	-
Gerät:	HTC	ohne Glättung	1	36	42	103	7,5
Anz. Scans:	92	geglättet mit sValue=6 (16 scans)	1	36	41	103	7,8

Tabelle 9-54 EI-Testgebiet iCP Erkennung EI9 3S-Gang1a

Teststrecke: ***EI3SGang1a***			iCP		mittlere Abweichung von iCP
			38	**39**	
EI9		*Referenzschritte*	*79*	*57*	-
Gerät:	HTC	ohne Glättung	78	69	6,5
Anz. Scans:	77	geglättet mit sValue=6	79	57	0,0

Tabelle 9-55 EI-Testgebiet iCP Erkennung EI10 3S-Gang1b

Teststrecke: ***EI3SGang1b***			iCP		mittlere Abweichung von iCP
			38	**39**	
EI10		*Referenzschritte*	*3*	*25*	-
Gerät:	HTC	ohne Glättung	4	7	9,5
Anz. Scans:	72	geglättet mit sValue=6	4	7	9,5

Tabelle 9-56 EI-Testgebiet iCP Erkennung EI11 3S-Gang2a

Teststrecke: ***3SGang2a***			iCP	mittlere Abweichung von iCP
			34	
EI11		*Referenzschritte*	*17*	-
Gerät:	HTC	ohne Glättung	17	0,0
Anz. Scans:	37	geglättet mit sValue=2	17	0,0

Tabelle 9-57 EI-Testgebiet iCP Erkennung EI11 3S-Gang2b

Teststrecke: ***3SGang2b***			iCP	mittlere Abweichung von iCP
			34	
EI12		*Referenzschritte*	*24*	-
Gerät:	HTC	ohne Glättung	19	5,0
Anz. Scans:	34	geglättet mit sValue=6	20	4,0

10 Anhang C – Vergleiche GPS, INS- und iCP-INS-Algorithmus

In Anhang C sind die Ergebnisse des vorgeschlagen iCP-INS-Algorithmus von Kapitel 6 angeführt. Testläufe wurden dabei im Heim-Testgebiet und im EI-Testgebiet im Eingangsbereich und dem 3. Stock des Gebäudes durchgeführt. Die MATLAB-Files zur Analyse sind in [37, DAAS\testruns] in den nach den Testgebieten bzw. -bereichen benannten Ordner abrufbar.

10.1 Überblick über die Ergebnisse

In Tabelle 6-11 sind die erzielten Ergebnisse mit dem INS- und iCP-INS-Algorithmus für die Testläufe H7-H10 im Heim-Testgebiet dargestellt. Die Ergebnisse werden den GPS-Aufzeichnungen gegenübergestellt. Verglichen werden die mittlere und maximale Abweichung und die Abweichung der Endposition. Der Vergleich der ermittelten Startpositionen, ist für den INS-Algorithmus nicht anwendbar, da dieser vom Referenz-Startpunkt ausgeht und somit ein Vergleich nicht sinnvoll wäre.

Bei der Berechnung der Mittelwerte wurde der Testlauf H8 ausgenommen, da bei diesem der Kompass versagt hat. Grund des Versagens war, dass dieser vorher nicht kalibriert wurde wie in 6.3 erläutert und daher das Ergebnis verzerrt. Die Abweichungen beziehen sich auf die Referenz-Positionen bzw. Referenz-Schritte, was in 6.1 erläutert wird.

In Abschnitt 8.5 sind die Details zu Testläufen H7-H8 im Heim-Testgebiert beschrieben. In Abbildung 10-1 bis Abbildung 10-4 sind INS- und iCP-INS-Algorithmus und GPS-Aufzeichnung für diese Testläufe grafisch dargestellt.

In den Tabelle 10-1 bis Tabelle 10-16 sind die errechneten Abweichungen von den Referenzschritten im Detail angeführt.

Die in Tabelle 6-12 dargestellten Ergebnisse zeigen die erzielten Resultate mit dem INS- und iCP-INS-Algorithmus-II für das EI-Testgebiet im Eingangsbereich des Gebäudes und stellt diese den GPS-Aufzeichnungen der Testläufe gegenüber.

Tabelle 6-13 fasst die Analysen mit den Testläufen EI-Testgebiet im 3. Stock zusammen. Diese Testläufe simulieren die Vorsetzung der Testläufe aus dem Eingangsbereich, indem die Endpositionen übernommen und als Startpositionen angewendet wurden. Im Gebäude waren keine GPS-Signale verfügbar, was den Vergleich unmöglich macht.

In Abschnitt 10.3 sind die Details zu den Testläufen EI1-EI8 im EI-Testgebiert erläutert.

Die Testläufe EI1-EI5 für den Eingangsbereich sind in Abbildung 10-5 bis Abbildung 10-9 grafisch dargestellt. Sie zeigen die Ergebnisse für INS- und iCP-INS-Algorithmus-II und die GPS-Aufzeichnung. Die Details zu den errechneten Abweichungen von den Referenzschritten sind in Tabelle 10-17 bis Tabelle 10-36 angeführt.

Die Testläufe EI6-EI8 im 3. Stock des Gebäudes und die Ergebnisse sind in den Abbildung 10-10 bis Abbildung 10-12 grafisch dargestellt. Sie zeigen die Ergebnisse für INS- und iCP-INS-Algorithmus-II. Die Details zu den errechneten Abweichungen von den Referenzschritten sind in den Tabelle 10-37 bis Tabelle 10-45 angeführt.

10.2 iCP-INS-Algorithmus Testläufe Heim-Testgebiet

10.2.1 Testlauf H7 im Heim-Testgebiet auf Teststrecke Um-Heim mit HTC

Abbildung 10-1 Testlauf H7 GPS, INS- Algorithmus, iCP-INS-Algorithmus

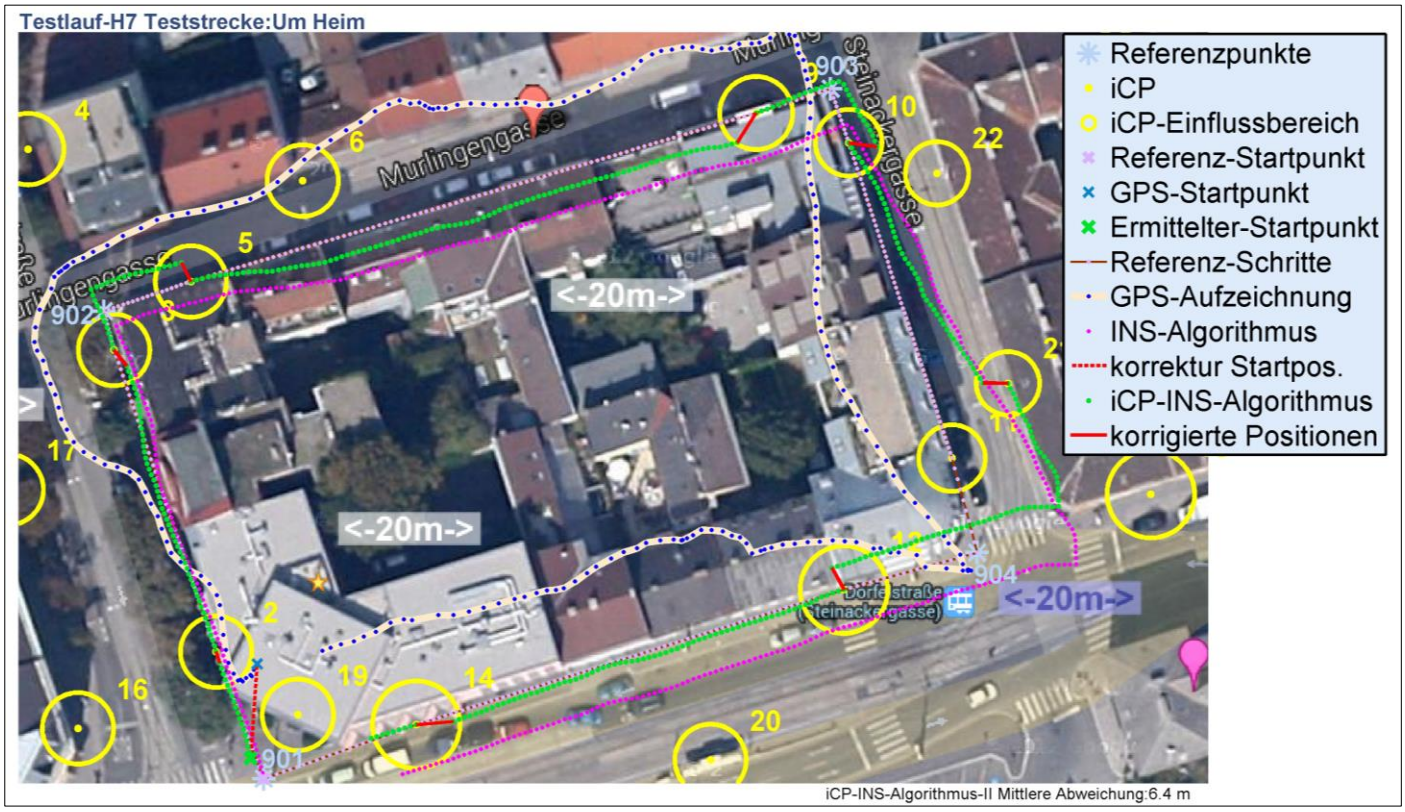


Tabelle 10-1 Zusammenfassung TL-H7

Zusammenfassung der Ergebnisse			
H7 UmHeim HTC	**GPS**	**INS-Algo.**	**iCP-INS-Alog.**
Abweichung Startposition	14,2 m	n.a.	3,0 m
Abweichung Endposition	17,4 m	17,2 m	14,2 m
Mittlere Abweichung	10,1 m	8,4 m	6,4 m
Maximale Abweichung	19,4 m	17,9 m	17,6 m
Minimale Abweichung	0,4 m	n.a.	0,4 m

Abweichung in Schritten	**findstep()**	**INS-Algo.**
mittlere Abweichung erkannter iCP	1,5	34,4

Tabelle 10-2 INS-Algo. TL-H7

INS-Algorithmus Ergebnisse		
H7 UmHeim HTC		
Mittlere Abweichung Lon.	7,0	m
Mittlere Abweichung Lat.	2,9	m
Mittlere Abweichung	8,4	m
Abweichung Lon. Endpunkt	17,2	m
Abweichung Lat. Endpunkt	0,6	m
Abweichung Endpunkt	17,2	m
maximale Abweichung	17,9	m

Tabelle 10-3 iCP-INS- Algo. TL-H7

iCP-INS-Algorithmus Ergebnisse		
H7 Um Heim HTC		
Mittlere Abweichung Lon.	5,4	m
Mittlere Abweichung Lat.	2,7	m
Mittlere Abweichung	**6,4**	**m**
Abweichung Lon. Startpunkt	1,6	m
Abweichung Lat. Startpunkt	2,5	m
Abweichung Startpunkt	**3,0**	**m**
Abweichung Lon. Endpunkt	13,3	m
Abweichung Lat. Endpunkt	5,0	m
Abweichung Endpunkt	**14,2**	**m**
maximale Abweichung	17,6	m
minimale Abweichung	0,0	m
iCP - Abweichung	10-17-21-43-43-NaN-43-64	Schritte
Mittlere Abweichung iCP	34,4	Schritte

Tabelle 10-4 GPS TL-H7

GPS Aufzeichnung Ergebnisse		
H7 UmHeim HTC		
Mittlere Abweichung Lon.	5,8	m
Mittlere Abweichung Lat.	7,4	m
Mittlere Abweichung	**10,1**	**m**
Abweichung Lon. Startpunkt	0,8	m
Abweichung Lat. Startpunkt	14,2	m
Abweichung Startpunkt	**14,2**	**m**
Abweichung Lon. Endpunkt	7,3	m
Abweichung Lat. Endpunkt	15,8	m
Abweichung Endpunkt	**17,4**	**m**
maximale Abweichung	19,4	m
minimale Abweichung	0,4	m

10.2.2 Testlauf H8 im Heim-Testgebiet auf Teststrecke Um-Heim mit S2

Abbildung 10-2 Testlauf H8 GPS, INS- Algorithmus, iCP-INS-Algorithmus

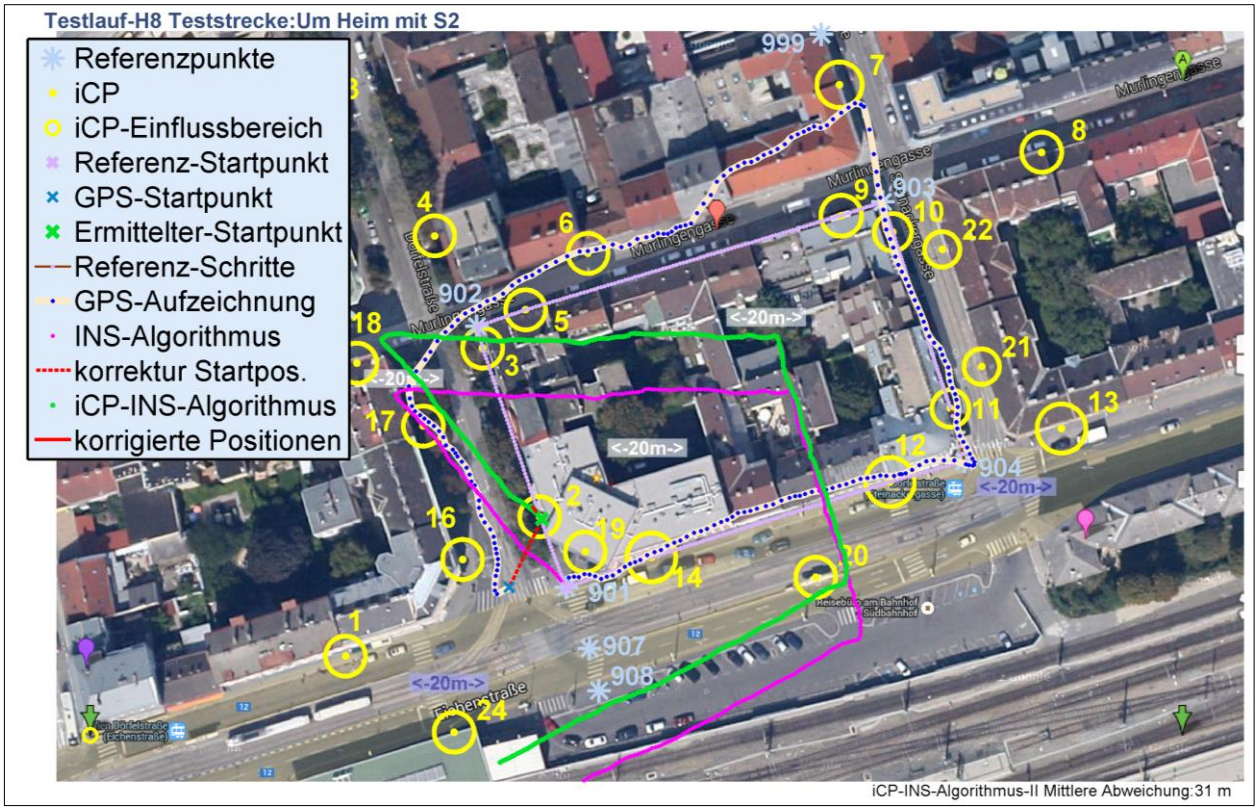


Tabelle 10-5 Zusammenfassung TL-H8

Zusammenfassung der Ergebnisse			
H8 UmHeim S2	**GPS**	**INS-Algo.**	**iCP-INS-Alog.**
Abweichung Startposition	12,6 m	n.a.	16,1 m
Abweichung Endposition	2,3 m	52,2 m	41,3 m
Mittlere Abweichung	9,3 m	37,2 m	31,0 m
Maximale Abweichung	22,9 m	52,2 m	41,3 m
Minimale Abweichung	0,5 m	n.a.	22,0 m

Abweichung in Schritten	**findstep()**	**INS-Algo.**
mittlere Abweichung erkannter iCP	6,375	22,0

Tabelle 10-6 INS-Algo. TL-H8

INS-Algorithmus Ergebnisse		
H8 UmHeim S2		
Mittlere Abweichung Lon.	17,0	m
Mittlere Abweichung Lat.	32,5	m
Mittlere Abweichung	37,2	m
Abweichung Lon. Endpunkt	11,5	m
Abweichung Lat. Endpunkt	51,0	m
Abweichung Endpunkt	52,2	m
maximale Abweichung	52,2	m

Tabelle 10-7 iCP-INS- Algo. TL-H8

iCP-INS-Algorithmus Ergebnisse		
H8 UmHeim S2		
Mittlere Abweichung Lon.	20,1	m
Mittlere Abweichung Lat.	22,0	m
Mittlere Abweichung	**31,0**	**m**
Abweichung Lon. Startpunkt	5,4	m
Abweichung Lat. Startpunkt	15,2	m
Abweichung Startpunkt	**16,1**	**m**
Abweichung Lon. Endpunkt	14,6	m
Abweichung Lat. Endpunkt	38,6	m
Abweichung Endpunkt	**41,3**	**m**
maximale Abweichung	41,3	m
minimale Abweichung	10,7	m
iCP - Abweichung	22	Schritte
Mittlere Abweichung iCP	22,0	Schritte

Tabelle 10-8 GPS TL-H8

GPS Aufzeichnung Ergebnisse		
H8 UmHeim S2		
Mittlere Abweichung Lon.	5,4	m
Mittlere Abweichung Lat.	6,8	m
Mittlere Abweichung	**9,3**	**m**
Abweichung Lon. Startpunkt	12,6	m
Abweichung Lat. Startpunkt	0,0	m
Abweichung Startpunkt	**12,6**	**m**
Abweichung Lon. Endpunkt	0,6	m
Abweichung Lat. Endpunkt	2,3	m
Abweichung Endpunkt	**2,3**	**m**
maximale Abweichung	22,9	m
minimale Abweichung	0,5	m

10.2.3 Testlauf H9 im Heim-Testgebiet auf Teststrecke Bahnhof-nach-7 mit HTC

Abbildung 10-3 Testlauf H9 GPS, INS- Algorithmus, iCP-INS-Algorithmus

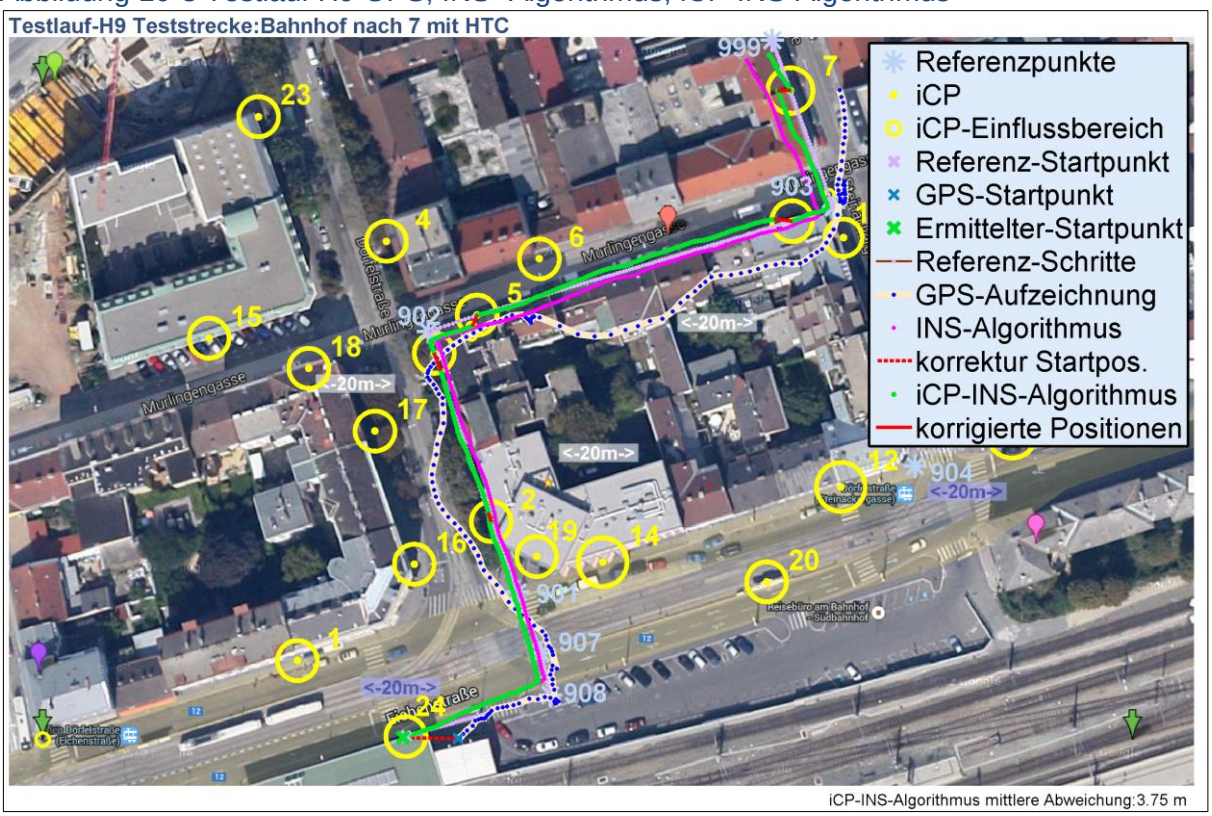


Tabelle 10-9 Zusammenfassung TL-H9

Zusammenfassung der Ergebnisse			
H9 Bahnhofnach7 HTC	GPS	INS-Algo.	iCP-INS-Alog.
Abweichung Startposition	11,6 m	n.a.	0,7 m
Abweichung Endposition	18,6 m	7,2 m	3,3 m
Mittlere Abweichung	11,6 m	3,7 m	3,8 m
Maximale Abweichung	25,7 m	7,3 m	8,3 m
Minimale Abweichung	0,1 m	n.a.	28,7 m

Abweichung in Schritten	findstep()	INS-Algo.
mittlere Abweichung erkannter iCP	2,7	28,7

Tabelle 10-10 INS-Algo. TL-H9

INS-Algorithmus Ergebnisse		
H9 Bahnhofnach7 HTC		
Mittlere Abweichung Lon.	2,3	m
Mittlere Abweichung Lat.	2,6	m
Mittlere Abweichung	3,7	m
Abweichung Lon. Endpunkt	5,8	m
Abweichung Lat. Endpunkt	4,3	m
Abweichung Endpunkt	7,2	m
maximale Abweichung	7,3	m

Tabelle 10-11 iCP-INS- Algo. TL-H9

iCP-INS-Algorithmus Ergebnisse		
H9 Bahnhofnach7 HTC		
Mittlere Abweichung Lon.	2,6	m
Mittlere Abweichung Lat.	2,0	m
Mittlere Abweichung	**3,8**	**m**
Abweichung Lon. Startpunkt	0,7	m
Abweichung Lat. Startpunkt	0,2	m
Abweichung Startpunkt	**0,7**	**m**
Abweichung Lon. Endpunkt	0,7	m
Abweichung Lat. Endpunkt	3,2	m
Abweichung Endpunkt	**3,3**	**m**
maximale Abweichung	8,3	m
minimale Abweichung	0,3	m
iCP - Abweichung	1-14-22-29-48-58	Schritte
Mittlere Abweichung iCP	28,7	Schritte

Tabelle 10-12 GPS TL-H9

GPS Aufzeichnung Ergebnisse		
H9 Bahnhofnach7 HTC		
Mittlere Abweichung Lon.	5,7	m
Mittlere Abweichung Lat.	9,2	m
Mittlere Abweichung	**11,6**	**m**
Abweichung Lon. Startpunkt	11,6	m
Abweichung Lat. Startpunkt	0,5	m
Abweichung Startpunkt	**11,6**	**m**
Abweichung Lon. Endpunkt	14,9	m
Abweichung Lat. Endpunkt	11,1	m
Abweichung Endpunkt	**18,6**	**m**
maximale Abweichung	25,7	m
minimale Abweichung	0,1	m

10.2.4 Testlauf H10 im Heim-Testgebiet auf Teststrecke Bahnhof-nach-7 mit S2

Abbildung 10-4 Testlauf H10 GPS, INS- Algorithmus, iCP-INS-Algorithmus

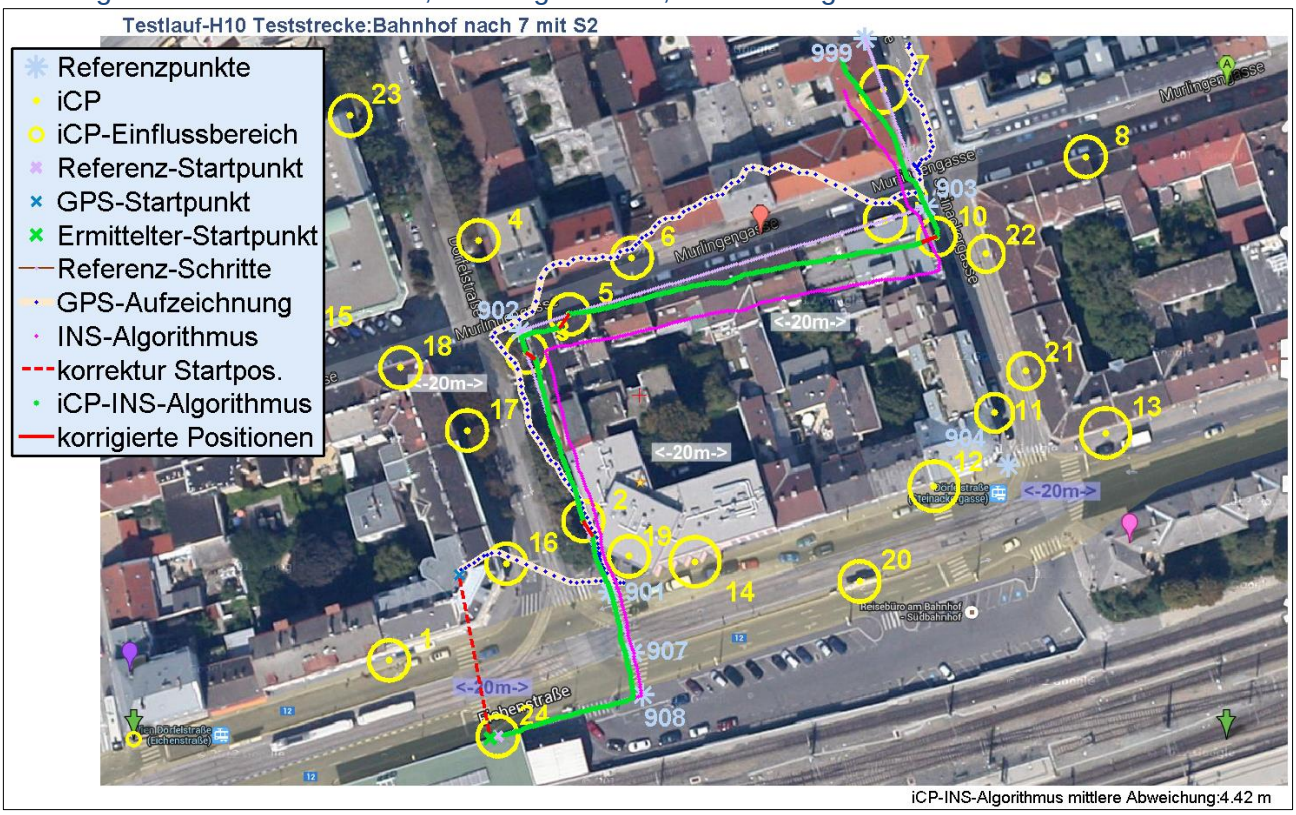


Tabelle 10-13 Zusammenfassung TL-H10

Zusammenfassung der Ergebnisse			
H10 Bahnhofnach7 S2	GPS	INS-Algo.	iCP-INS-Alog.
Abweichung Startposition	36,8 m	n.a.	1,8 m
Abweichung Endposition	9,9 m	12,4 m	6,9 m
Mittlere Abweichung	12,9 m	8,3 m	4,4 m
Maximale Abweichung	39,5 m	18,0 m	11,2 m
Minimale Abweichung	0,8 m	n.a.	3,7 m
Abweichung in Schritten	**findstep()**	**INS-Algo.**	
mittlere Abweichung erkannter iCP	9,7	3,7	

Tabelle 10-14 INS-Algo. TL-H10

INS-Algorithmus Ergebnisse		
H10 Bahnhofnach7 S2		
Mittlere Abweichung Lon.	2,8	m
Mittlere Abweichung Lat.	7,5	m
Mittlere Abweichung	8,3	m
Abweichung Lon. Endpunkt	4,7	m
Abweichung Lat. Endpunkt	11,4	m
Abweichung Endpunkt	12,4	m
maximale Abweichung	18,0	m

Tabelle 10-15 iCP-INS- Algo. TL-H10

iCP-INS-Algorithmus Ergebnisse		
H10 Bahnhofnach7 S2		
Mittlere Abweichung Lon.	1,7	m
Mittlere Abweichung Lat.	3,8	m
Mittlere Abweichung	**4,4**	**m**
Abweichung Lon. Startpunkt	1,7	m
Abweichung Lat. Startpunkt	0,5	m
Abweichung Startpunkt	**1,8**	**m**
Abweichung Lon. Endpunkt	5,1	m
Abweichung Lat. Endpunkt	4,6	m
Abweichung Endpunkt	**6,9**	**m**
maximale Abweichung	11,2	m
minimale Abweichung	1,2	m
iCP - Abweichung	1 4 6 NaN	Schritte
Mittlere Abweichung iCP	3,7	Schritte

Tabelle 10-16 GPS TL-H10

GPS Aufzeichnung Ergebnisse		
H10 Bahnhofnach7 S2		
Mittlere Abweichung Lon.	5,9	m
Mittlere Abweichung Lat.	10,8	m
Mittlere Abweichung	**12,9**	**m**
Abweichung Lon. Startpunkt	8,5	m
Abweichung Lat. Startpunkt	35,8	m
Abweichung Startpunkt	**36,8**	**m**
Abweichung Lon. Endpunkt	9,8	m
Abweichung Lat. Endpunkt	1,1	m
Abweichung Endpunkt	**9,9**	**m**
maximale Abweichung	39,5	m
minimale Abweichung	0,8	m

10.3 iCP-INS-Algorithmus II im EI-Testgebiet

10.3.1 Testlauf EI1 im EI-Testgebiet im Eingangsbereich des Gebäudes

Abbildung 10-5 Testlauf EI1 GPS, INS- Algorithmus, iCP-INS-Algorithmus-II

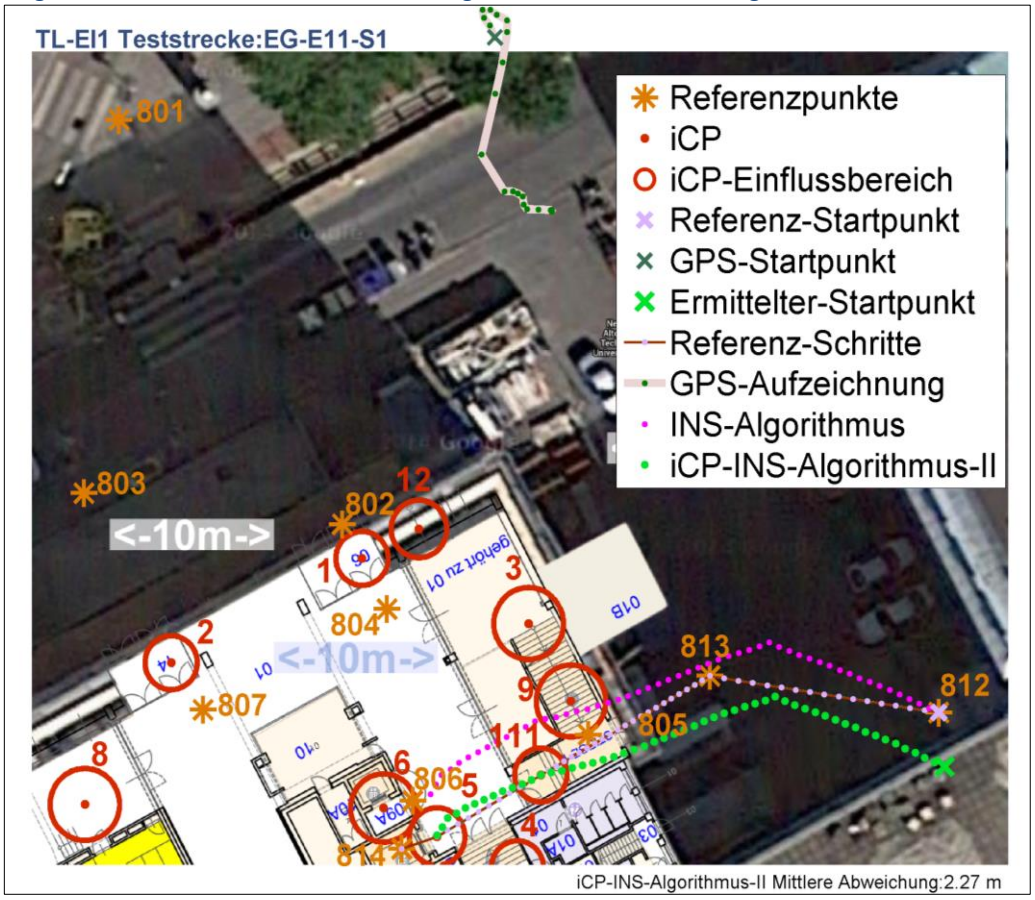


Tabelle 10-17 Zusammenfassung TL-EI1

Zusammenfassung der Ergebnisse			
EI1-EIEG1S1-HTC	**GPS**	**INS-Algo.**	**iCP-INS-Alog.**
Abweichung Startposition	45,5 m	n.a.	3,0 m
Abweichung Endposition	36,6 m	3,5 m	2,2 m
Mittlere Abweichung	34,5 m	2,9 m	2,3 m
Maximale Abweichung	45,7 m	4,3 m	4,5 m
Minimale Abweichung	29,2 m	n.a.	0,0 m

Abweichung in Schritten	**findstep()**	**INS-Algo.**
mittlere Abweichung erkannter iCP	9	4,5

Tabelle 10-18 INS-Algorithmus TL-EI1

INS-Algorithmus Ergebnisse		
EI1-EIEG1S1-HTC		
Mittlere Abweichung Lon.	1,5	m
Mittlere Abweichung Lat.	2,1	m
Mittlere Abweichung	2,9	m
Abweichung Lon. Endpunkt	1,7	m
Abweichung Lat. Endpunkt	3,1	m
Abweichung Endpunkt	3,5	m
maximale Abweichung	4,3	m

Tabelle 10-19 iCP-INS- Algorithmus-II TL-EI1

iCP-INS-Algorithmus-II Ergebnisse		
EI1-EIEG1S1-HTC		
Mittlere Abweichung Lon.	1,8	m
Mittlere Abweichung Lat.	1,1	m
Mittlere Abweichung	**2,3**	**m**
Abweichung Lon. Startpunkt	0,3	m
Abweichung Lat. Startpunkt	3,0	m
Abweichung Startpunkt	**3,0**	**m**
Abweichung Lon. Endpunkt	2,0	m
Abweichung Lat. Endpunkt	0,8	m
Abweichung Endpunkt	**2,2**	**m**
maximale Abweichung	4,5	m
minimale Abweichung	0,0	m
iCP - Abweichung	4 5	Schritte
Mittlere Abweichung iCP	4,5	Schritte

Tabelle 10-20 GPS TL-EI1

GPS Aufzeichnung Ergebnisse		
EI1-EIEG1S1-HTC		
Mittlere Abweichung Lon.	9,5	m
Mittlere Abweichung Lat.	32,6	m
Mittlere Abweichung	**34,5**	**m**
Abweichung Lon. Startpunkt	25,6	m
Abweichung Lat. Startpunkt	37,6	m
Abweichung Startpunkt	**45,5**	**m**
Abweichung Lon. Endpunkt	8,6	m
Abweichung Lat. Endpunkt	35,5	m
Abweichung Endpunkt	**36,6**	**m**
maximale Abweichung	45,7	m
minimale Abweichung	29,2	m

10.3.2 Testlauf EI2 im EI-Testgebiet im Eingangsbereich des Gebäudes

Abbildung 10-6 Testlauf EI2 GPS, INS- Algorithmus, iCP-INS-Algorithmus-II

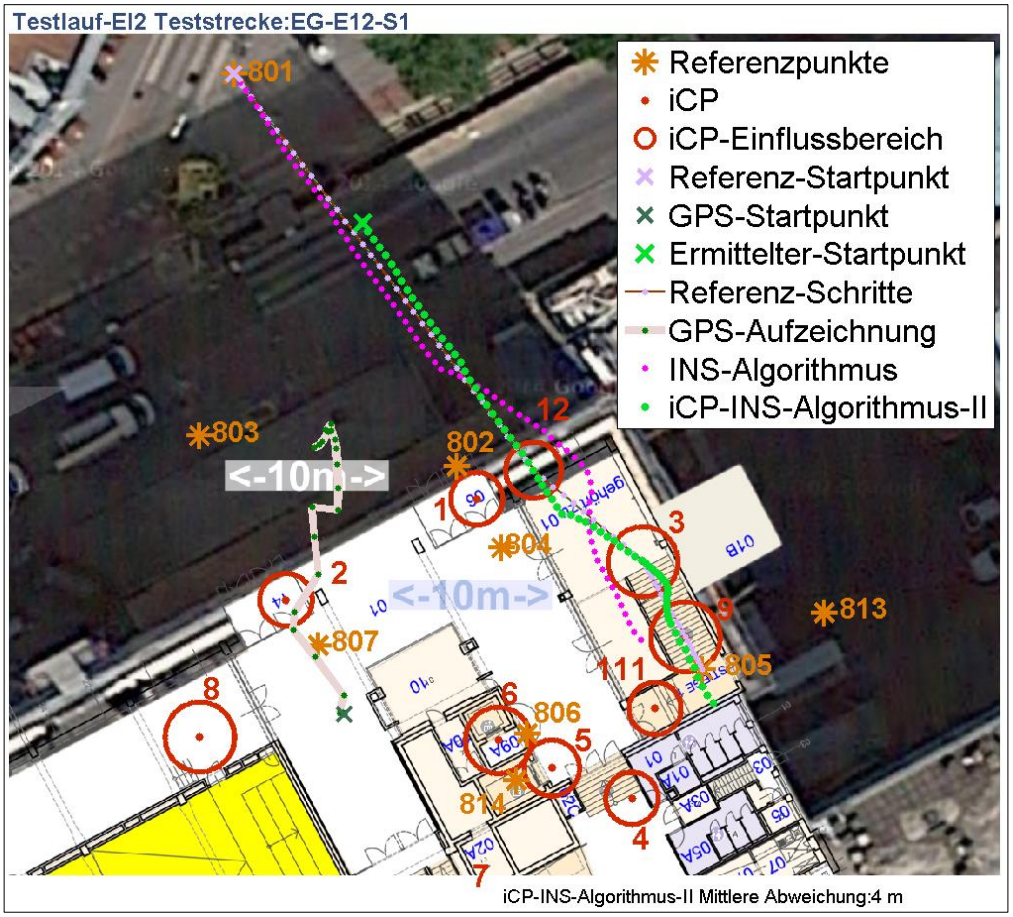


Tabelle 10-21 Zusammenfassung TL-EI2

Zusammenfassung der Ergebnisse			
EI2-EIEG12S1-HTC	**GPS**	**INS-Algo.**	**iCP-INS-Alog.**
Abweichung Startposition	37,4 m	n.a.	11,3 m
Abweichung Endposition	25,8 m	4,0 m	2,0 m
Mittlere Abweichung	20,7 m	6,0 m	4,0 m
Maximale Abweichung	37,4 m	10,0 m	11,3 m
Minimale Abweichung	8,4 m	n.a.	0,1 m

Abweichung in Schritten	**findstep()**	**INS-Algo.**
mittlere Abweichung erkannter iCP	15	16,0

Tabelle 10-22 INS-Algorithmus TL-EI2

INS-Algorithmus Ergebnisse		
EI2-EIEG12S1-HTC		
Mittlere Abweichung Lon.	4,0	m
Mittlere Abweichung Lat.	4,4	m
Mittlere Abweichung	6,0	m
Abweichung Lon. Endpunkt	3,6	m
Abweichung Lat. Endpunkt	1,7	m
Abweichung Endpunkt	4,0	m
maximale Abweichung	10,0	m

Tabelle 10-23 iCP-INS- Algorithmus-II TL-EI2

iCP-INS-Algorithmus-II Ergebnisse		
EI2-EIEG12S1-HTC		
Mittlere Abweichung Lon.	2,5	m
Mittlere Abweichung Lat.	3,1	m
Mittlere Abweichung	**4,0**	**m**
Abweichung Lon. Startpunkt	7,4	m
Abweichung Lat. Startpunkt	8,6	m
Abweichung Startpunkt	**11,3**	**m**
Abweichung Lon. Endpunkt	0,6	m
Abweichung Lat. Endpunkt	1,9	m
Abweichung Endpunkt	**2,0**	**m**
maximale Abweichung	11,3	m
minimale Abweichung	0,1	m
iCP - Abweichung	16 16	Schritte
Mittlere Abweichung iCP	16,0	Schritte

Tabelle 10-24 GPS TL-EI2

GPS Aufzeichnung Ergebnisse		
EI2-EIEG12S1-HTC		
Mittlere Abweichung Lon.	12,5	m
Mittlere Abweichung Lat.	12,9	m
Mittlere Abweichung	**20,7**	**m**
Abweichung Lon. Startpunkt	6,4	m
Abweichung Lat. Startpunkt	36,9	m
Abweichung Startpunkt	**37,4**	**m**
Abweichung Lon. Endpunkt	22,4	m
Abweichung Lat. Endpunkt	12,8	m
Abweichung Endpunkt	**25,8**	**m**
maximale Abweichung	37,4	m
minimale Abweichung	8,4	m

10.3.3 Testlauf EI3 im EI-Testgebiet im Eingangsbereich des Gebäudes

Abbildung 10-7 Testlauf EI3 GPS, INS- Algorithmus, iCP-INS-Algorithmus-II

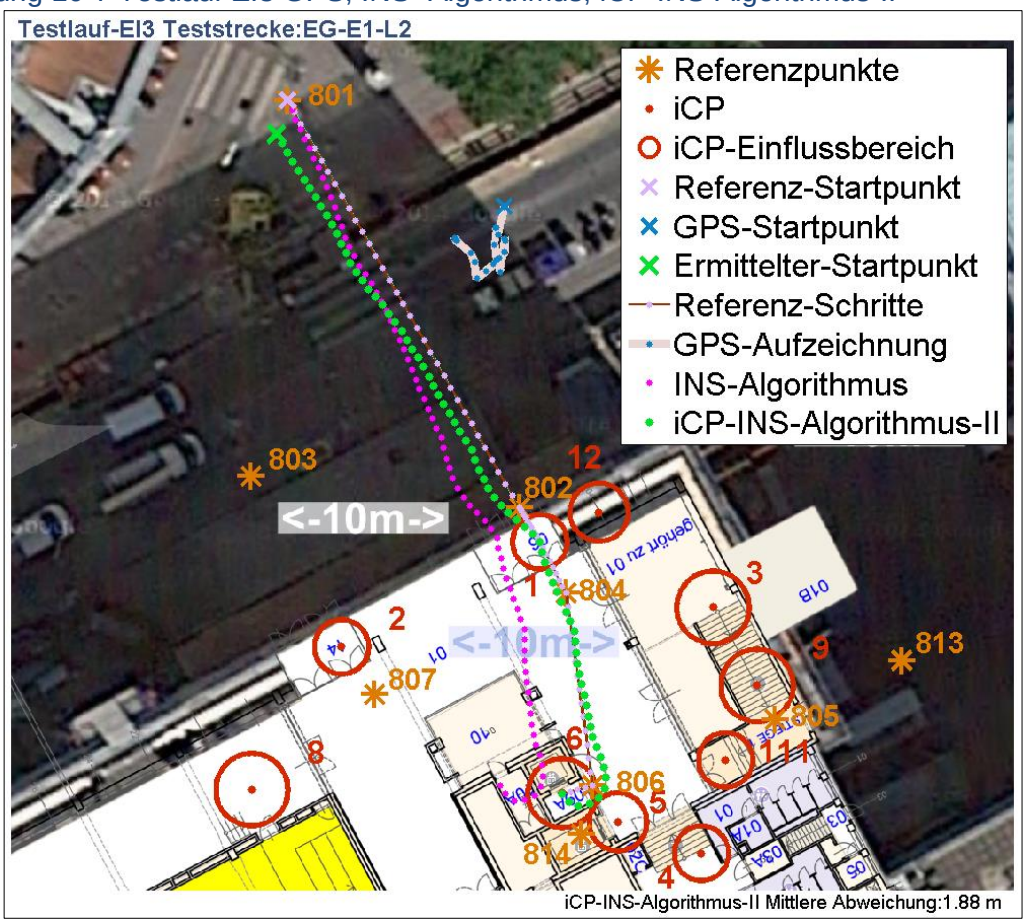


Tabelle 10-25 Zusammenfassung TL-EI3

Zusammenfassung der Ergebnisse			
EI3-EIEG1S1-HTC	**GPS**	**INS-Algo.**	**iCP-INS-Alog.**
Abweichung Startposition	13,4 m	n.a.	2,0 m
Abweichung Endposition	31,3 m	3,5 m	0,0 m
Mittlere Abweichung	16,4 m	3,2 m	1,9 m
Maximale Abweichung	31,3 m	6,4 m	4,8 m
Minimale Abweichung	5,6 m	n.a.	10, m

Abweichung in Schritten	**findstep()**	**INS-Algo.**
mittlere Abweichung erkannter iCP	0,5	1,5

Tabelle 10-26 INS-Algorithmus TL-EI3

INS-Algorithmus Ergebnisse		
EI3-EIEG1S1-HTC		
Mittlere Abweichung Lon.	2,7	m
Mittlere Abweichung Lat.	1,6	m
Mittlere Abweichung	3,2	m
Abweichung Lon. Endpunkt	3,5	m
Abweichung Lat. Endpunkt	0,4	m
Abweichung Endpunkt	3,5	m
maximale Abweichung	6,4	m

Tabelle 10-27 iCP-INS- Algo. TL-EI3

iCP-INS-Algorithmus Ergebnisse		
EI3-EIEG1S1-HTC		
Mittlere Abweichung Lon.	1,2	m
Mittlere Abweichung Lat.	1,3	m
Mittlere Abweichung	**1,9**	**m**
Abweichung Lon. Startpunkt	0,6	m
Abweichung Lat. Startpunkt	1,9	m
Abweichung Startpunkt	**2,0**	**m**
Abweichung Lon. Endpunkt	0,0	m
Abweichung Lat. Endpunkt	0,0	m
Abweichung Endpunkt	**0,0**	**m**
maximale Abweichung	4,8	m
minimale Abweichung	0,0	m
iCP - Abweichung	1-2	Schritte
Mittlere Abweichung iCP	1,5	Schritte

Tabelle 10-28 GPS TL-EI3

GPS Aufzeichnung Ergebnisse		
EI3-EIEG1S1-HTC		
Mittlere Abweichung Lon.	5,7	m
Mittlere Abweichung Lat.	14,3	m
Mittlere Abweichung	**16,4**	**m**
Abweichung Lon. Startpunkt	12,0	m
Abweichung Lat. Startpunkt	5,9	m
Abweichung Startpunkt	**13,4**	**m**
Abweichung Lon. Endpunkt	6,0	m
Abweichung Lat. Endpunkt	30,8	m
Abweichung Endpunkt	**31,3**	**m**
maximale Abweichung	31,3	m
minimale Abweichung	5,6	m

10.3.4 Testlauf EI4 im EI-Testgebiet im Eingangsbereich des Gebäudes

Abbildung 10-8 Testlauf EI4 GPS, INS- Algorithmus, iCP-INS-Algorithmus-II

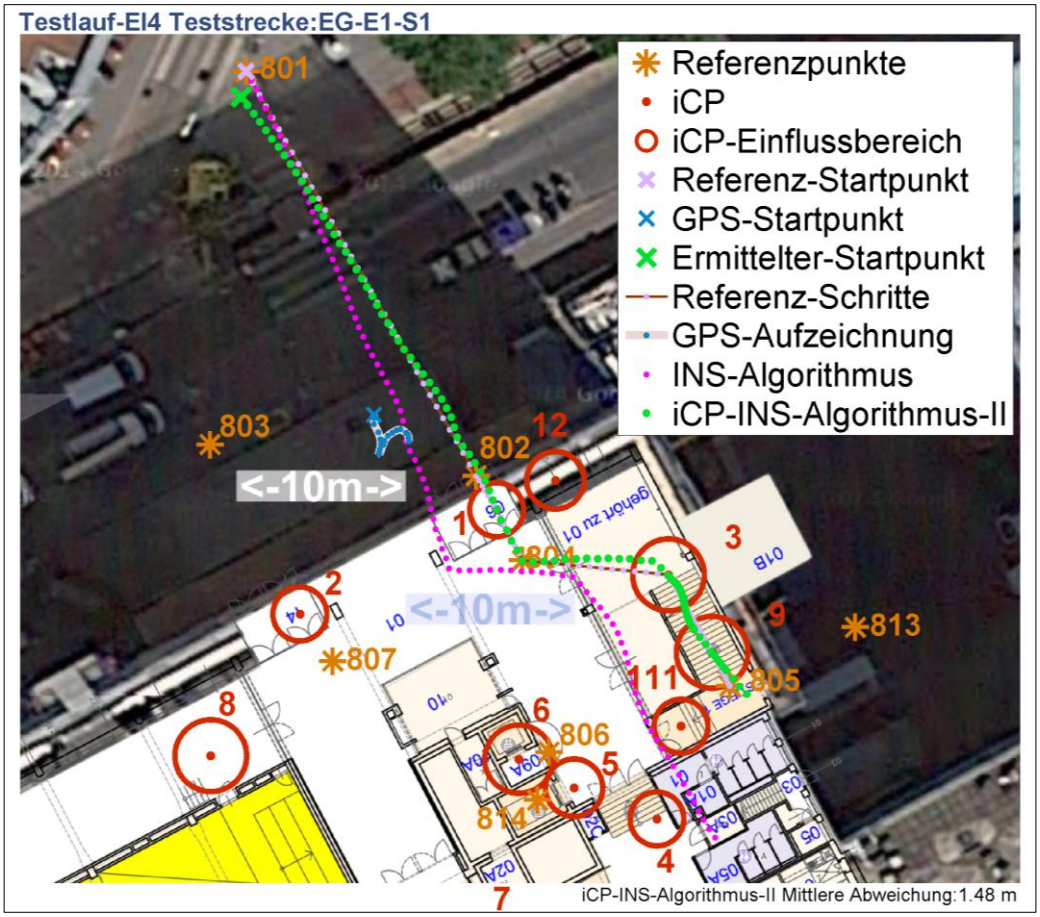


Tabelle 10-29 Zusammenfassung TL-EI4

Zusammenfassung der Ergebnisse			
EI4-EIEG1S1-HTC	**GPS**	**INS-Algo.**	**iCP-INS-Alog.**
Abweichung Startposition	20,4 m	n.a.	1,5 m
Abweichung Endposition	22,6 m	8,5 m	1,0 m
Mittlere Abweichung	12,8 m	4,3 m	1,5 m
Maximale Abweichung	22,6 m	8,5 m	4,7
Minimale Abweichung	3,5 m	n.a.	0,0 m

Abweichung in Schritten	**findstep()**	**INS-Algo.**
mittlere Abweichung erkannter iCP	5,3	1,0

Tabelle 10-30 INS-Algorithmus TL-EI1

INS-Algorithmus Ergebnisse		
EI4-EIEG1S1-HTC		
Mittlere Abweichung Lon.	3,1	m
Mittlere Abweichung Lat.	2,4	m
Mittlere Abweichung	4,3	m
Abweichung Lon. Endpunkt	0,8	m
Abweichung Lat. Endpunkt	8,5	m
Abweichung Endpunkt	8,5	m
maximale Abweichung	8,5	m

Tabelle 10-31 iCP-INS- Algorithmus-II TL-EI4

iCP-INS-Algorithmus-II Ergebnisse		
EI4-EIEG1S1-HTC		
Mittlere Abweichung Lon.	0,7	m
Mittlere Abweichung Lat.	1,3	m
Mittlere Abweichung	**1,5**	**m**
Abweichung Lon. Startpunkt	0,3	m
Abweichung Lat. Startpunkt	1,4	m
Abweichung Startpunkt	**1,5**	**m**
Abweichung Lon. Endpunkt	0,9	m
Abweichung Lat. Endpunkt	0,4	m
Abweichung Endpunkt	**1,0**	**m**
maximale Abweichung	4,7	m
minimale Abweichung	0,0	m
iCP - Abweichung	1-1-1	Schritte
Mittlere Abweichung iCP	1,0	Schritte

Tabelle 10-32 GPS TL-EI1

GPS Aufzeichnung Ergebnisse		
EI4-EIEG1S1-HTC		
Mittlere Abweichung Lon.	8,5	m
Mittlere Abweichung Lat.	8,5	m
Mittlere Abweichung	**12,8**	**m**
Abweichung Lon. Startpunkt	7,2	m
Abweichung Lat. Startpunkt	19,2	m
Abweichung Startpunkt	**20,4**	**m**
Abweichung Lon. Endpunkt	18,0	m
Abweichung Lat. Endpunkt	13,7	m
Abweichung Endpunkt	**22,6**	**m**
maximale Abweichung	22,6	m
minimale Abweichung	3,5	m

10.3.5 Testlauf EI5 im EI-Testgebiet im Eingangsbereich des Gebäudes

Abbildung 10-9 Testlauf EI5 GPS, INS- Algorithmus, iCP-INS-Algorithmus-II

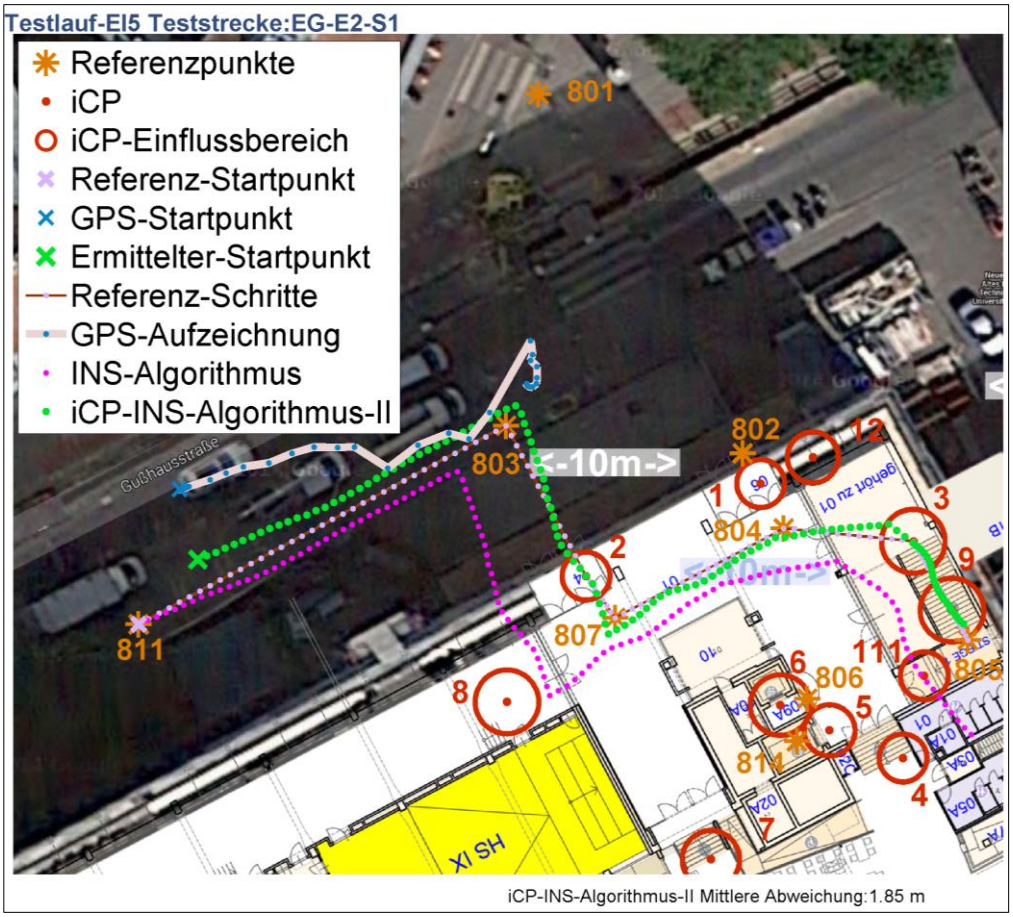


Tabelle 10-33 Zusammenfassung TL-EI5

Zusammenfassung der Ergebnisse			
EI5-EG1S1-HCT	**GPS**	**INS-Algo.**	**iCP-INS-Alog.**
Abweichung Startposition	8,9 m	n.a.	5,5 m
Abweichung Endposition	32,2 m	5,9 m	1,1 m
Mittlere Abweichung	14,7 m	4,2 m	1,9 m
Maximale Abweichung	32,2 m	6,4 m	5,5 m
Minimale Abweichung	1,1 m	n.a.	0,5 m

Abweichung in Schritten	**findstep()**	**INS-Algo.**
mittlere Abweichung erkannter iCP	8,7	10,0

Tabelle 10-34 INS-Algorithmus TL-EI5

INS-Algorithmus Ergebnisse		
EI5-EG1S1-HCT		
Mittlere Abweichung Lon.	2,9	m
Mittlere Abweichung Lat.	2,8	m
Mittlere Abweichung	4,2	m
Abweichung Lon. Endpunkt	0,2	m
Abweichung Lat. Endpunkt	5,9	m
Abweichung Endpunkt	5,9	m
maximale Abweichung	6,4	m

Tabelle 10-35 iCP-INS- Algorithmus-II TL-EI5

iCP-INS-Algorithmus-II Ergebnisse		
EI5-EG1S1-HCT		
Mittlere Abweichung Lon.	1,2	m
Mittlere Abweichung Lat.	1,4	m
Mittlere Abweichung	**1,9**	**m**
Abweichung Lon. Startpunkt	3,7	m
Abweichung Lat. Startpunkt	4,1	m
Abweichung Startpunkt	**5,5**	**m**
Abweichung Lon. Endpunkt	0,3	m
Abweichung Lat. Endpunkt	1,0	m
Abweichung Endpunkt	**1,1**	**m**
maximale Abweichung	5,5	m
minimale Abweichung	0,5	m
iCP - Abweichung	13-12-5	Schritte
Mittlere Abweichung iCP	10,0	Schritte

Tabelle 10-36 GPS TL-EI5

GPS Aufzeichnung Ergebnisse		
EI5-EG1S1-HCT		
Mittlere Abweichung Lon.	10,5	m
Mittlere Abweichung Lat.	9,0	m
Mittlere Abweichung	**14,7**	**m**
Abweichung Lon. Startpunkt	2,6	m
Abweichung Lat. Startpunkt	8,5	m
Abweichung Startpunkt	**8,9**	**m**
Abweichung Lon. Endpunkt	27,8	m
Abweichung Lat. Endpunkt	16,1	m
Abweichung Endpunkt	**32,2**	**m**
maximale Abweichung	32,2	m
minimale Abweichung	1,1	m

10.3.6 Testlauf EI6 im EI-Testgebiet im 3. Stock des Gebäudes

Abbildung 10-10 Testlauf EI6 GPS, INS- Algorithmus, iCP-INS-Algorithmus-II

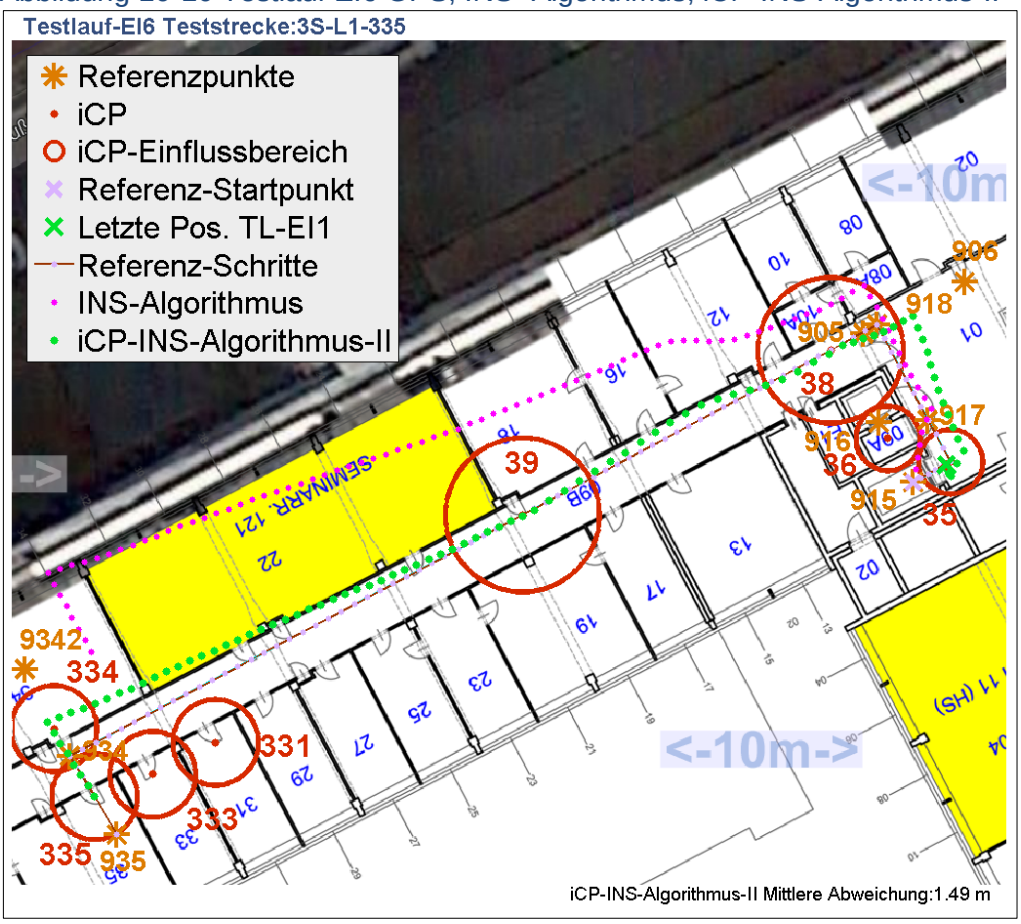


Tabelle 10-37 Zusammenfassung TL-EI6

Zusammenfassung der Ergebnisse			
EI6-EI3SL1335-HTC	**GPS**	**INS-Algo.**	**iCP-INS-Alog.**
Abweichung Startposition	N.A.	N.A.	1,8 m
Abweichung Endposition	N.A.	8,7 m	2,1 m
Mittlere Abweichung	N.A.	4,8 m	1,5 m
Maximale Abweichung	N.A.	9,2 m	2,7 m
Minimale Abweichung	N.A.	n.a.	0,3 m

Abweichung in Schritten	**findstep()**	**INS-Algo.**
mittlere Abweichung erkannter iCP	5,0	2,3

Tabelle 10-38 INS-Algorithmus TL-EI6

iCP-INS-Algorithmus Ergebnisse		
EI6-EI3SL1335-HTC		
Mittlere Abweichung Lon.	1,2	m
Mittlere Abweichung Lat.	0,7	m
Mittlere Abweichung	**1,5**	**m**
Abweichung Lon. Startpunkt	1,6	m
Abweichung Lat. Startpunkt	0,8	m
Abweichung Startpunkt	**1,8**	**m**
Abweichung Lon. Endpunkt	1,1	m
Abweichung Lat. Endpunkt	1,8	m
Abweichung Endpunkt	**2,1**	**m**
maximale Abweichung	2,7	m
minimale Abweichung	0,3	m
iCP - Abweichung	0-4-4-1	Schritte
Mittlere Abweichung iCP	2,3	Schritte

Tabelle 10-39 iCP-INS- Algo. TL-EI6

INS-Algorithmus Ergebnisse		
EI6-EI3SL1335-HTC		
Mittlere Abweichung Lon.	1,1	m
Mittlere Abweichung Lat.	4,7	m
Mittlere Abweichung	4,8	m
Abweichung Lon. Endpunkt	1,2	m
Abweichung Lat. Endpunkt	8,6	m
Abweichung Endpunkt	8,7	m
maximale Abweichung	9,2	m

10.3.7 Testlauf EI7 im EI-Testgebiet im 3. Stock des Gebäudes

Abbildung 10-11 Testlauf EI7 GPS, INS- Algorithmus, iCP-INS-Algorithmus-II

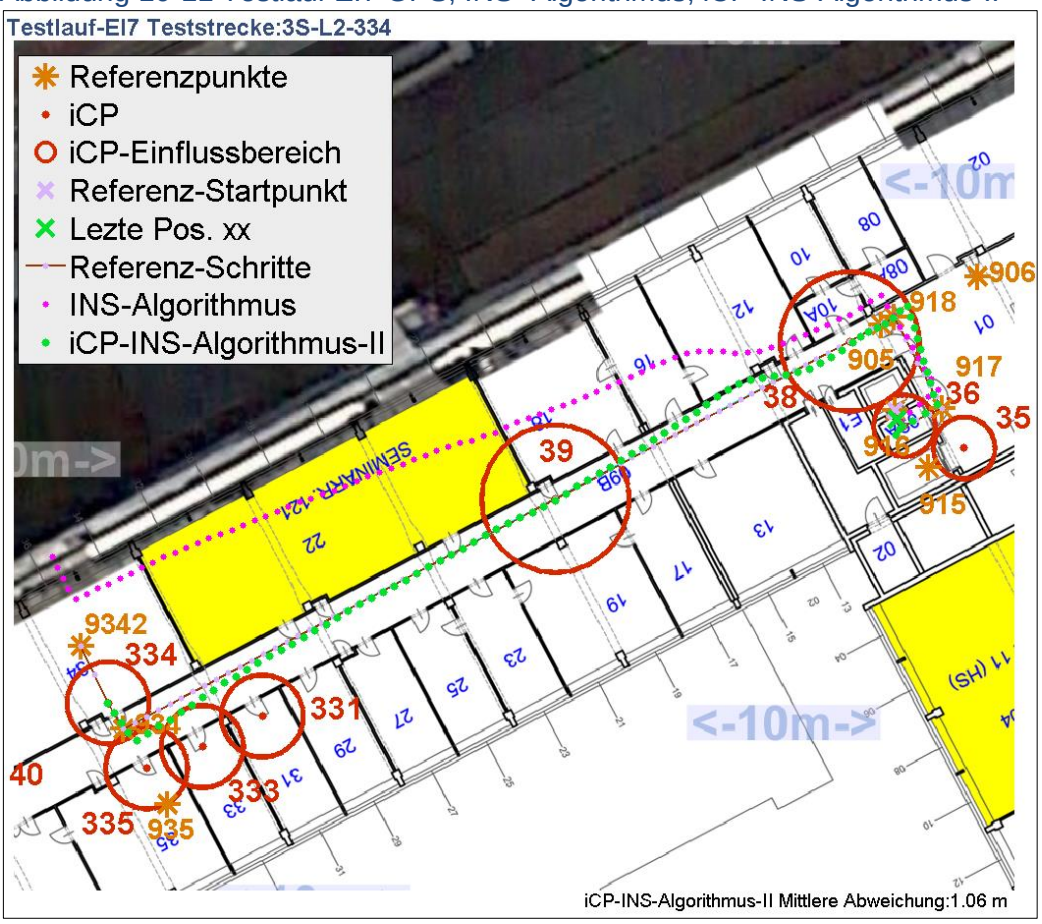


Tabelle 10-40 Zusammenfassung TL-EI7

Zusammenfassung der Ergebnisse			
EI7-EI3SL2334-HTC	**GPS**	**INS-Algo.**	**iCP-INS-Alog.**
Abweichung Startposition	N.A.	N.A.	0,5 m
Abweichung Endposition	N.A.	4,6 m	3,1 m
Mittlere Abweichung	N.A.	3,8 m	1,1 m
Maximale Abweichung	N.A.	6,9 m	3,7 m
Minimale Abweichung	N.A.	n.a.	0,1 m

Abweichung in Schritten	**findstep()**	**INS-Algo.**
mittlere Abweichung erkannter iCP	0,3	13,5

Tabelle 10-41 INS-Algorithmus TL-EI7

iCP-INS-Algorithmus-II Ergebnisse		
EI7-EI3SL2334-HTC		
Mittlere Abweichung Lon.	0,9	m
Mittlere Abweichung Lat.	0,5	m
Mittlere Abweichung	**1,1**	**m**
Abweichung Lon. Startpunkt	0,1	m
Abweichung Lat. Startpunkt	0,4	m
Abweichung Startpunkt	**0,5**	**m**
Abweichung Lon. Endpunkt	1,3	m
Abweichung Lat. Endpunkt	2,8	m
Abweichung Endpunkt	**3,1**	**m**
maximale Abweichung	3,7	m
minimale Abweichung	0,1	m
iCP - Abweichung	1-14-22-17	Schritte
Mittlere Abweichung iCP	13,5	Schritte

Tabelle 10-42 iCP-INS-Algorithmus-II TL-EI7

INS-Algorithmus Ergebnisse		
EI7-EI3SL2334-HTC		
Mittlere Abweichung Lon.	1,6	m
Mittlere Abweichung Lat.	3,4	m
Mittlere Abweichung	3,8	m
Abweichung Lon. Endpunkt	1,4	m
Abweichung Lat. Endpunkt	4,4	m
Abweichung Endpunkt	4,6	m
maximale Abweichung	6,9	m

10.3.8 Testlauf EI8 im EI-Testgebiet im 3. Stock des Gebäudes

Abbildung 10-12 Testlauf EI8 GPS, INS- Algorithmus, iCP-INS-Algorithmus-II

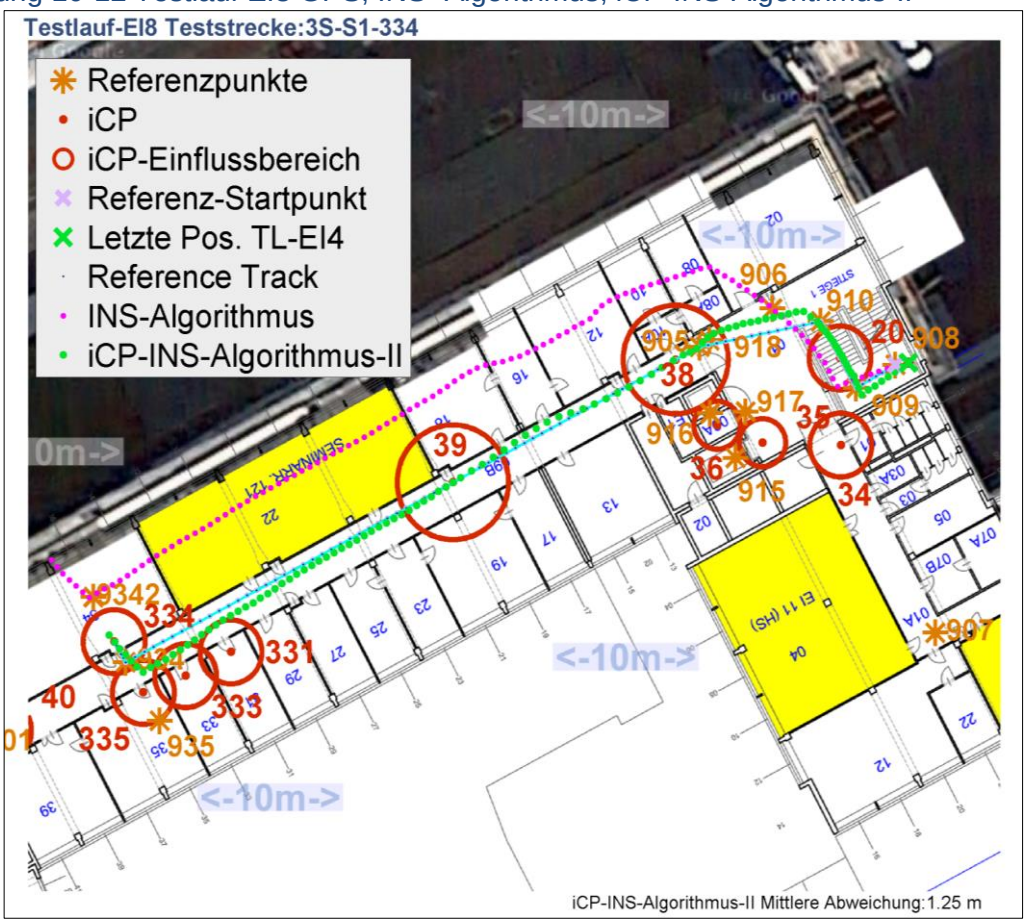


Tabelle 10-43 Zusammenfassung TL-EI8

Zusammenfassung der Ergebnisse			
EI8-EI3SZ1-HTC	**GPS**	**INS-Algo.**	**iCP-INS-Alog.**
Abweichung Startposition	N.A.	N.A.	0,9 m
Abweichung Endposition	N.A.	6,5 m	0,5 m
Mittlere Abweichung	N.A.	5,5 m	1,3 m
Maximale Abweichung	N.A.	8,0 m	3,4 m
Minimale Abweichung	N.A.	n.a.	0,0 m

Abweichung in Schritten	**findstep()**	**INS-Algo.**
mittlere Abweichung erkannter iCP	5,5	8,3

Tabelle 10-44 INS-Algorithmus TL-EI8

iCP-INS-Algorithmus Ergebnisse		
EI8-EI3SZ1-HTC		
Mittlere Abweichung Lon.	1,0	m
Mittlere Abweichung Lat.	0,7	m
Mittlere Abweichung	**1,3**	**m**
Abweichung Lon. Startpunkt	0,8	m
Abweichung Lat. Startpunkt	0,1	m
Abweichung Startpunkt	**0,9**	**m**
Abweichung Lon. Endpunkt	0,3	m
Abweichung Lat. Endpunkt	0,4	m
Abweichung Endpunkt	**0,5**	**m**
maximale Abweichung	3,4	m
minimale Abweichung	0,0	m
iCP - Abweichung	5-14-5-9	Schritte
Mittlere Abweichung iCP	8,3	Schritte

Tabelle 10-45 iCP-INS-Algorithmus-II TL-EI8

INS-Algorithmus-II Ergebnisse		
EI8-EI3SZ1-HTC		
Mittlere Abweichung Lon.	4,6	m
Mittlere Abweichung Lat.	2,8	m
Mittlere Abweichung	5,5	m
Abweichung Lon. Endpunkt	4,1	m
Abweichung Lat. Endpunkt	5,1	m
Abweichung Endpunkt	6,5	m
maximale Abweichung	8,0	m

Literaturverzeichnis

[1] Aassie, A, Omar, AS (2005): Time of Arrival Estimation for WLAN Indoor Positioning Systems using Matrix Pencil Super Resolution Algorithm. In: Omar A. M. Aly , A. S. Omar (Hrsg), *PROCEEDINGS OF THE 2nd WORKSHOP ON POSITIONING, NAVIGATION AND COMMUNICATION (WPNC'05) & 1st ULTRA-WIDEBAND EXPERT TALK (UET'05). University of Magdeburg, Magdeburg (Germany).* CiteSeerX digital library.

[2] Android Inc. (2011): Activity Lifecycle | Android Developers. Android SDK online documentation. http://developer.android.com/reference/android/app/Activity.html#ActivityLifecycle. Abgerufen am 09.05.2015.

[3] Android Inc. (Oktober 2015): Download Android Studio and SDK Tools | Android Developers. Android SDK. https://developer.android.com/sdk/index.html. Abgerufen am 10.05.2015.

[4] Android Inc. (16.10.2015 19:17): Activities | Android Developers. Android SDK online documentation. http://developer.android.com/guide/components/activities.html. Abgerufen am 18.10.2015.

[5] Android Inc. (16.10.2015 19:17): Activity | Android Developers. Android SDK online documentation. http://developer.android.com/reference/android/app/Activity.html. Abgerufen am 18.10.2015.

[6] Android Inc. (16.10.2015 19:17): android.view | Android Developers. Android SDK online documentation. http://developer.android.com/reference/android/view/package-summary.html. Abgerufen am 18.10.2015.

[7] Android Inc. (16.10.2015 19:17): BroadcastReceiver | Android Developers. Android SDK online documentation. http://developer.android.com/reference/android/content/BroadcastReceiver.html. Abgerufen am 18.10.2015.

[8] Android Inc. (16.10.2015 19:17): Dialogs | Android Developers. Android SDK online documentation. http://developer.android.com/guide/topics/ui/dialogs.html. Abgerufen am 18.10.2015.

[9] Android Inc. (16.10.2015 19:17): LocationListener | Android Developers. Android SDK online documentation. http://developer.android.com/reference/android/location/LocationListener.html. Abgerufen am 18.10.2015.

[10] Android Inc. (16.10.2015 19:17): LocationManager | Android Developers. Android SDK online documentation. http://developer.android.com/reference/android/location/LocationManager.html. Abgerufen am 18.10.2015.

[11] Android Inc. (16.10.2015 19:17): Position Sensors | Android Developers. Android SDK online documentation. http://developer.android.com/guide/topics/sensors/sensors_position.html. Abgerufen am 10.05.2015.

[12] Android Inc. (16.10.2015 19:17): ScanResult | Android Developers. Android SDK online documentation. http://developer.android.com/reference/android/net/wifi/ScanResult.html. Abgerufen am 18.10.2015.

[13] Android Inc. (16.10.2015 19:17): SensorEventListener | Android Developers. Android SDK online documentation. http://developer.android.com/reference/android/hardware/SensorEventListener.html. Abgerufen am 18.10.2015.

[14] Android Inc. (16.10.2015 19:17): SensorManager | Android Developers. Android SDK online documentation. http://developer.android.com/reference/android/hardware/SensorManager.html. Abgerufen am 18.10.2015.

[15] Android Inc. (16.10.2015 19:17): SensorManager | Android Developers. Android SDK online documentation. http://developer.android.com/reference/android/hardware/SensorManager.html. Abgerufen am 18.10.2015.

[16] Android Inc. (16.10.2015 19:17): SharedPreferences | Android Developers. Android SDK online documentation. http://developer.android.com/reference/android/content/SharedPreferences.html. Abgerufen am 18.10.2015.

[17] Android Inc. (16.10.2015 19:17): WifiManager | Android Developers. Android SDK online documentation. http://developer.android.com/reference/android/net/wifi/WifiManager.html. Abgerufen am 18.10.2015.

[18] Arnold, A, Freist, R (11.05.2013): Geniale NFC-Tags selber machen - so geht´s. PC Welt (online Ausgabe), IDG Tech Media GmbH, München (Detuschland). http://www.pcwelt.de/ratgeber/Ratgeber-Software-NFC-Tags-selber-machen-7769272.html. Abgerufen am 03.05.2015.

[19] Bahl, P, Padmanabhan, VN (2005): RADAR: An In-Building RF-Based User Location and Tracking System. In: Omar A. M. Aly , A. S. Omar (Hrsg), *PROCEEDINGS OF THE 2nd WORKSHOP ON POSITIONING, NAVIGATION AND COMMUNICATION (WPNC'05) & 1st ULTRA-WIDEBAND EXPERT TALK (UET'05). University of Magdeburg, Magdeburg (Germany).* CiteSeerX digital library.

[20] Bayes, Price (1763): An Essay towards Solving a Problem in the Doctrine of Chances. By the Late Rev. Mr. Bayes, F. R. S. Communicated by Mr. Price, in a Letter to John Canton, A. M. F. R. S. Philosophical Transactions of the Royal Society of London, 53(0):370–418.

[21] Bekkelien, A (2012): Bluetooth Indoor Positioning. Masterarbeit, Institute of Information Service Science, University of Geneva, http://tam.unige.ch/assets/documents/masters/bekkelien/Bekkelien_Master_Thesis.pdf 10. November 2015.

[22] Beom-Ju, S, Kwang-Won, L, Sun-Ho, C, Joo-Yeon, K, Woo, LJ, Hyung, KS (2010): Indoor WiFi positioning system for Android-based smartphone. In: Sung, W (Hrsg), *International Conference on Information and Communication Technology Convergence 2010 (ICTC'10). Sogang University, Jeju (South Korea).* IEEE Xplore digital library, Piscataway (NJ, USA).

[23] Bornstein, D (2008): Dalvik VM Internals. I/O Session Videos and Slides. https://sites.google.com/site/io/dalvik-vm-internals. Abgerufen am 13.05.2015.

[24] Di Wu, Yubin Xu, Lin Ma (2009): Research on RSS based Indoor Location Method. In: Xiong, F (Hrsg), *2009 Pacific-Asia Conference on Knowledge Engineering and Software Engineering. Shenzhen (China) - International Science and Engineering Center (Hong Kong).* IEEE Xplore digital library, Piscataway (NJ, USA).

[25] Dropbox Inc. (2007): Dropbox - Über uns - Info. https://www.dropbox.com/about. Abgerufen am 08.05.2015.

[26] Dropbox Inc. (28.10.2015): Dropbox für Android. https://play.google.com/store/apps/details?id=com.dropbox.android&hl=de. Abgerufen am 08.05.2015.

[27] Ebermann, E (18.06.2010): Arithmetisches Mittel. https://www.univie.ac.at/ksa/elearning/cp/quantitative/quantitative-73.html. Abgerufen am 18.10.2015.

[28] Eclipse Foundation Inc.: Eclipse Platform. 4.2.1, o. V. https://eclipse.org/, Ottawa (Ontario, Kanada).

[29] Esato (2011): HTC EVO 3D picture gallery. http://www.esato.com/phones/phonegallery.php?pid=820&page=3. Abgerufen am 02.04.2015.

[30] European Telecommunications Standards Institute (26.09.2014): Digital cellular telecommunications system (Phase 2+); Radio subsystem synchronization, (12). http://www.etsi.org/deliver/etsi_ts/145000_145099/145010/12.00.00_60/ts_145010v120000p.pdf. Abgerufen am 29.04.2015.

[31] Fu, Q, Retscher, G (2009): Using RFID and INS for Indoor Positioning. In: Gartner, G, Rehrl, K (Hrsg), *Location Based Services and TeleCartography II.* Springer Berlin Heidelberg.

[32] Gustafsson, F. and Gunnarsson, F. (2003): Positioning using time-difference of arrival measurements. In: Signal Processing Society (Hrsg), *IEEE International Conference on Acoustics, Speech, and Signal Processing 2003 (ICASSP '03). Hong Kong.* IEEE Xplore digital library, Piscataway (NJ, USA).

[33] Handy Deutschland GmbH (2015): HTC EVO 3D Datenblatt HTC EVO 3D technische Daten. http://www.handy-deutschland.de/htc-evo-3d-datenblatt.html. Abgerufen am 02.04.2015.

[34] Handy Deutschland GmbH (2015): Samsung I9100 Galaxy S2 Datenblatt Samsung Galaxy S2 technische Daten. http://www.handy-deutschland.de/samsung-galaxy-s2-datenblatt.html. Abgerufen am 11.11.2015.

[35] Herrera-May, Agustin, L., García-Ramírez, Pedro, J, Mota-Carrillo, Nelly, B., Padrón-Hernández, Wendy, Y., Figueras, E (2011): Development of Resonant Magnetic Field Microsensors: Challenges and Future Applications. In: Minin, O (Hrsg), *Microsensors.* InTech.

[36] Hnilica, E: Statistik - Lagemaße - Median. mathe-lexikon.at - Christian Kohout (Wien). http://www.mathe-lexikon.at/statistik/lagemasse/median.html. Abgerufen am 18.10.2015.

[37] Hofer, H (2014): Daten für Dimplomarbeit "Kombinierte Indoor/Outdoor Positionierung mit Smartphones". https://drive.google.com/folderview?id=0B2sMc_nmy1A1aTFlZXZYb2lTTUk&usp=sharing. Abgerufen am 24.07.2015.

[38] Honkavirta, V, Perala, T, Ali-Loytty, S, Piche, R (2009): A comparative survey of WLAN location fingerprinting methods. In: Kaiser, T, Di Benedetto, M (Hrsg), *6th Workshop on Positioning, Navigation and Communication 2009 (WPNC´09). Leibniz University of Hannover, Hannover (Germany).* IEEE Xplore digital library, Piscataway (NJ, USA).

[39] Hu, B (2013): Wi-Fi Based Indoor Positioning System Using Smartphones. Masterarbeit, School of Mathematical and Geospatial Sciences, Royal Melbourne Institute of Technology (RMIT) University, http://researchbank.rmit.edu.au/eserv/rmit:160749/Hu.pdf Zugegriffen: 10. November 2015.

[40] Ibraheem, IA, Schoebel, J (2007): Time of Arrival Prediction for WLAN Systems Using Prony Algorithm. In: Kaiser, T, Jobmann, K, Kyamaky, K (Hrsg), *4th Workshop on Positioning, Navigation and Communication 2007 (WPNC'07). Leibniz University of Hannover, Hannover (Germany).* IEEE Xplore digital library, Piscataway (NJ, USA).

[41] IDC Research Inc. (2015): IDC: Smartphone OS Market Share. International Data Corporation, Framingham (MA, USA). http://www.idc.com/prodserv/smartphone-os-market-share.jsp. Abgerufen am 13.05.2015.

[42] Ing. Betram BERISSON G.: Grundrisspläne TU Wien. http://www.gut.tuwien.ac.at/wir_fuer_sie/immobilienmanagement/grundrisse_objekte/. Abgerufen am 12.01.2015.

[43] Jarka, A, Aegidius, P, Stefan, M: Lagesensoren. Game-Apps für Smartphones und Tablets. http://www.jdroid.ch/index.php?inhalt_mitte=grundelemente/lagesensoren.inc.php. Abgerufen am 17.08.2015.

[44] Jun, M, Xuansong, L, Xianping, T, Jian, L (2008): Cluster filtered KNN: A WLAN-based indoor positioning scheme. In: Jain, R, Kumar, M (Hrsg), *International Symposium on a World of Wireless, Mobile and Multimedia Networks 2008 (WoWMoM´08). Newport Beach (CA, USA).* IEEE Xplore digital library, Piscataway (NJ, USA).

[45] KEE, C, PARKINSON, BW, AXELRAD, P (1991): Wide Area Differential GPS. Navigation, 38(2):123–145.

[46] Kinh, T, Tu, L, Tien, D (2012): A high-accuracy step counting algorithm for iPhones using accelerometer. In: Wu, J, A. V. Dinh (Hrsg), *IEEE International Symposium on Signal Processing and Information Technology 2012 (ISSPIT'12). Ho Chi Minh City (Vietnam), HCMC University of Technology.* IEEE Xplore digital library, Piscataway (NJ, USA).

[47] Köppe Enrico (18.07.2014): Lokalisierung sich bewegender Objekte innerhalb und außerhalb von Gebäuden. Dissertation, FB Mathematik und Informatik, Freie Universität Berlin.

[48] Kummer, J: Umrechnung von Geo-Koordinaten. Rechneronline - Nützliche Rechner. http://rechneronline.de/geo-koordinaten/. Abgerufen am 14.05.2015.

[49] Künneth, T (2012): Android 4. Apps entwickeln mit dem Android SDK. 2. Auflage. Galileo Press, Bonn (Deutschland).

[50] Markgraf, M (2015): GPS Navigation. https://wuecampus2.uni-wuerzburg.de/moodle/pluginfile.php/564965/mod_resource/content/1/GPS-Lecture_UniWuerzburg_SS2015.pdf. Abgerufen am 16.10.2015.

[51] MathWorks Deutschland: Cell Arrays - MATLAB & Simulink. http://de.mathworks.com/help/matlab/cell-arrays.html. Abgerufen am 14.05.2015.

[52] MathWorks Deutschland: Mean ignoring NaN values - MATLAB nanmean - MathWorks Deutschland. http://de.mathworks.com/help/stats/nanmean.html. Abgerufen am 18.05.2015.

[53] MathWorks Deutschland: Median ignoring NaNs - MATLAB nanmedian - MathWorks Deutschland. http://de.mathworks.com/help/finance/nanmedian.html. Abgerufen am 18.05.2015.

[54] MathWorks Deutschland: Most frequent values in array - MATLAB mode - MathWorks Deutschland. http://de.mathworks.com/help/matlab/ref/mode.html. Abgerufen am 20.05.2015.

[55] MathWorks Deutschland: Not-a-Number - MATLAB NaN - MathWorks Deutschland. http://de.mathworks.com/help/matlab/ref/nan.html. Abgerufen am 18.05.2015.

[56] MathWorks Deutschland: Smooth response data - MATLAB smooth - MathWorks Deutschland. http://de.mathworks.com/help/curvefit/smooth.html?refresh=true. Abgerufen am 10.06.2015.

[57] MathWorks Deutschland: Sort array elements - MATLAB sort - MathWorks Deutschland. http://de.mathworks.com/help/matlab/ref/sort.html. Abgerufen am 27.05.2015.

[58] MathWorks Deutschland: Structures - MATLAB & Simulink - MathWorks Deutschland. http://de.mathworks.com/help/matlab/structures.html. Abgerufen am 14.05.2015.

[59] MathWorks Deutschland (11.05.2015): Map values to unique keys - MATLAB - MathWorks Deutschland. http://de.mathworks.com/help/matlab/ref/containers.map-class.html. Abgerufen am 11.05.2015.

[60] MetaCtrl: Dropsync (Dropbox Autosync) – Android-Apps auf Google Play. https://play.google.com/store/apps/details?id=com.ttxapps.dropsync&hl=de. Abgerufen am 08.05.2015.

[61] Meyer, S (2012): SIGNALSTÄRKE-BASIERTE LOKALISIERUNG-BASIS FÜR ORTSABHÄNGIGE DIENSTE. Unterlagen zu Vortrag Indoor Navigation mit WLAN an HS Augsburg - Steffen Meyer (Fraunhofer Institut). http://www.hs-augsburg.de/~john/mobile-experience/workshops/awiloc/Vortrag_Lokalisierung.pdf. Abgerufen am 18.10.2015.

[62] Mok, E, Retscher, G, Chen Wen (2012): Initial test on the use of GPS and sensor data of modern smartphones for vehicle tracking in dense high rise environments. In: Kuusniemi, H (Hrsg), *Ubiquitous Positioning, Indoor Navigation, and Location Based Service 2012 (UPINLBS´12). Finnish Geodetic Institute, Helsinki (Finland).* IEEE Xplore digital library, Piscataway (NJ, USA).

[63] o. V.: MMA7361 Accelerometer ile Açı Ölçümü… | mikro dünya on WordPress.com. https://mikrodunya.wordpress.com/2011/10/21/mma7361-ivmeolcer/. Abgerufen am 17.08.2015.

[64] o. V. (29.04.2015): Ortung und Positionsbestimmung mit Mobilfunk. http://www.elektronik-kompendium.de/sites/kom/1201061.htm. Abgerufen am 29.04.2015.

[65] Oracle Corporation (07.09.2011): Class System - nanoTime. Java™ API Specification - java.lang.System. https://docs.oracle.com/javase/1.5.0/docs/api/java/lang/System.html#nanoTime. Abgerufen am 20.08.2015.

[66] Oracle Corporation (14.09.2015): Introduction to Event Listeners. JAVA Documentation - The Java™ Tutorials - Creating a GUI With JFC/Swing -Writing Event Listeners. https://docs.oracle.com/javase/tutorial/uiswing/events/intro.html. Abgerufen am 17.10.2015.

[67] Par, G (2011): Test : Samsung Galaxy S2. http://www.journaldugeek.com/2011/05/16/test-samsung-galaxy-s2-sii/. Abgerufen am 02.04.2015.

[68] Porteck, S (2011): Kompass im Android-Telefon. c't magazin 24/11, Heise Medien GmbH & Co. KG, Hannover (Deutschland). http://www.heise.de/ct/hotline/Kompass-im-Android-Telefon-1368092.html. Abgerufen am 11.11.2015.

[69] Raidl, G (2005): Folien zu LVA Heuristische Optimierungsverfahren. Institut für computergraphik und algorithmen, Technische Universität Wien (Wien). http://www.zid.tuwien.ac.at/tunet/wlan/. Abgerufen am 20.05.2015.

[70] Retscher, G, Hecht, T (2012): Investigation of location capabilities of four different smartphones for LBS navigation applications IPIN 2012, Sydney, Australia, November 13-15, 2012. In: Rizos, C, Andrew, DG, Li, B, Galla, T (Hrsg), *International Conference on Indoor Positioning and Indoor Navigation 2012 (IPIN´12). University of New South, Sydney (Australia).* IEEE Xplore digital library, Piscataway (NJ, USA).

[71] Retscher, G (2012): Wi-Fi Positioning with Smartphones. In: Technische Universität München (Hrsg), *9th International Symposium on Location-Based Services LBS 2012.* Springer Science, Berlin (Germany).

[72] Rötzer, F (2000): Global Positioning System ist jetzt für die zivile Nutzung genauer. heise online, Heise Medien GmbH & Co. KG, Hannover (Deutschland). http://www.heise.de/tp/artikel/6/6766/1.html. Abgerufen am 29.04.2015.

[73] Schnabel, P (2007): Elektronik-Fibel. Elektronik, Bauelemente, Schaltungstechnik, Digitaltechnik. 4. Auflage. Books on Demand, Norderstedt (Deutschland).

[74] Schnabel, P (03.05.2015): MEMS - Micro-Electro-Mechanical Systems. Elektronik-Kompendium.de. http://www.elektronik-kompendium.de/sites/bau/1503041.htm. Abgerufen am 03.05.2015.

[75] Tetz, E: Multiple SSIDs with a Single Access Point (AP). Cisco Networking All-in-One. http://www.dummies.com/how-to/content/multiple-ssids-with-a-single-access-point-ap.html. Abgerufen am 24.10.2015.

[76] Van Diggelen, Frank Stephen Tromp (2009): A-GPS. Assisted GPS, GNSS, and SBAS. Artech House, Boston.

[77] Vogel, L (2012): Android Sensor - Tutorial. http://www.vogella.com/tutorials/AndroidSensor/article.html. Abgerufen am 03.05.2015.

[78] Weber, R, Schek, H, Blott, S (1998): A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. In: Gupta, A, Shmueli, O, Widom, J (Hrsg), *Proceedings of the 24rd International Conference on Very Large Data Bases. New York, (NY, USA).* ACM Digital Library.

[79] Wikipedia (2015): GSM-Ortung - Wikipedia, Die freie Enzyklopädie. http://de.wikipedia.org/w/index.php?oldid=140535773. Abgerufen am 29.04.2015.

[80] Wikipedia (20.04.2015): Service set (802.11 network) - Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?oldid=657272127. Abgerufen am 18.05.2015.

[81] Wikipedia (30.04.2015): Azimut - Wikipedia, Die freie Enzyklopädie. http://de.wikipedia.org/w/index.php?oldid=138682300. Abgerufen am 08.05.2015.

[82] Wikipedia (01.05.2015): Mercator-Projektion - Wikipedia, Die freie Enzyklopädie. http://de.wikipedia.org/w/index.php?oldid=136280045. Abgerufen am 13.05.2015.

[83] Wikipedia (08.05.2015): Kartennetzentwurf - Wikipedia, Die freie Enzyklopädie. http://de.wikipedia.org/w/index.php?oldid=139279845. Abgerufen am 14.05.2015.

[84] Wikipedia (28.05.2015): Deklination (Geographie) - Wikipedia, Die freie Enzyklopädie. http://de.wikipedia.org/w/index.php?oldid=142570102. Abgerufen am 01.06.2015.

[85] Wikipedia (16.09.2015): Euklidischer Abstand - Wikipedia, Die freie Enzyklopädie. https://de.wikipedia.org/w/index.php?oldid=129598005. Abgerufen am 18.10.2015.

[86] Woodman, OJ (2007): An introduction to inertial navigation.

[87] Zandbergen, PA (2009): Accuracy of iPhone Locations: A Comparison of Assisted GPS, WiFi and Cellular Positioning. Transactions in GIS, 13:5–25.

[88] Zandbergen, PA, Barbeau, SJ (2011): Positional Accuracy of Assisted GPS Data from High-Sensitivity GPS-enabled Mobile Phones. Journal of Navigation, 64(03):381–399.