Clemens Reitbauer, BSc.

Graphen-basiertes Dead Reckoning für Fußgänger

MASTERARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

Masterstudium Geomatics Science

eingereicht an der

Technischen Universität Graz

Betreuer

Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. Manfred Wieser

Institut für Geodäsie

Graz, Juli 2017

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

Datum

Unterschrift

Danksagung

Zu Beginn möchte ich mich bei Herrn Ao.Univ.-Prof. Dipl.Ing. Dr.techn. Manfred Wieser für die Unterstützung und Betreuung während der Durchführung dieser Arbeit, sowie für die Möglichkeit als studentischer Mitarbeiter diese Arbeit durchzuführen, bedanken.

Ein besonderer Dank gilt Herrn Dipl.Ing. Thomas Moder für die ausgezeichnete Betreuung und das vielseitige Engagement im Verlauf dieser Arbeit. Für sämtliche, teils mit hohem Zeitaufwand verbundenen, Fragen und Problemstellungen konnte ich stets auf seine Expertise zurückgreifen.

Des Weiteren möchte ich mich bei den Mitarbeitern der gesamten Arbeitsgruppe Navigation des Institut für Geodäsie für die fachliche und moralische Unterstützung sowie für wertvolle Inputs bedanken.

Meinen Freunden, insbesondere meinen Studienkollegen, möchte ich für die außeruniversitären Reisen und Unternehmungen, welche die Studienzeit unvergesslich und einzigartig gemacht haben, danken.

Der größte Dank gilt schlussendlich meiner Freundin Anja und meiner Familie. Mit Anjas herzlicher und liebenswerter Art sowie ihrem sonnigen Gemüt bereichert sie mich jeden Tag und schafft dadurch den perfekten Ausgleich. Selbst in stressigen Zeiten konnte ich daher motiviert und gelassen die Anforderungen des Studiums bewältigen. Meinen lieben Eltern danke ich ebenso für die moralische und finanzielle Unterstützung und den wertvollen Rückhalt während meiner gesamten Ausbildungszeit. Dank ihrer Erziehung vermag ich auf eine unbeschwerte Kindheit, Schulzeit und Studienzeit zurückblicken. Stets mit Rat und Tat zur Seite standen mir ebenso meine Schwester Magdalena und ihr Verlobter Michael. Speziell für das Korrekturlesen dieser Masterarbeit möchte ich mich hier bei ihnen bedanken.

Zusammenfassung

Fußgänger können sich heutzutage im Outdoor-Bereich mit Hilfe des im Smartphone implementierten GNSS-Chip auf wenige Meter genau navigieren lassen. In Innenräumen versagt allerdings diese Methode aufgrund von Signalverlusten und Mehrwegeffekten. Gerade in größeren Gebäudekomplexen würde eine Positionierung und Zielführung von großem Nutzen sein. Neben kosten- und wartungsintensiven infrastrukturabhängigen Positionierungsmethoden können Smartphones aufgrund zusätzlicher Sensoren eine autonome Positionslösung für den Fußgänger mit Hilfe eines Schritt-basierten PDR-Algorithmus berechnen. Diesbezüglich werden mit Beschleunigungsmessungen Schrittevents detektiert. Des Weiteren dienen Gyroskopdaten für die Berechnung der Bewegungsrichtung des Fußgängers. Durch eine bekannte Schrittlänge kann somit die Positionsänderung durch einen vollzogenen Schritt berechnet werden. Die daraus resultierende Trajektorie weicht jedoch aufgrund der low-cost Charakteristiken der Messeinheiten stark von der tatsächlichen Trajektorie ab, da diese Sensoren nicht für navigationsspezifische Anwendungen konzipiert wurden. Dies gibt Anlass die PDR-Trajektorie durch zusätzliche Verfahren zu stützen.

In dieser Masterarbeit wurde diesbezüglich ein Graphen-basierter PDR-Algorithmus entwickelt. Dabei sind die Positionslösungen auf einen Graphen, welcher aufbauend auf einem Gebäudeplan erstellt wurde, beschränkt. Dementsprechend können keine widersprüchlichen Trajektorien durch Wände oder außerhalb des Gebäudes berechnet werden. Die Positionen werden dabei über fünf Zustandsabfragen, welche auf der momentanen Richtungsabweichung in Bezug auf die Kantenausrichtung und der aktuellen Position auf der Kante beruhen, auf den Graphen gezwungen. Ein weiterer Vorteil besteht in der Driftkompensierung durch den Graphen-basierten Ansatzes. Da nur die Richtungsabweichung von der Kante benötigt wird, müssen die Drehraten nicht über den gesamten Messzeitraum aufkumuliert werden, sondern nur über den Zeitraum einer Kante. Der Graphen-basierte Ansatz wurde anhand von Trajektorien in zwei Testumgebungen (Steyrergasse 30 und Inffeldgasse 16 in Graz) verifiziert. Durch zusätzliche Barometermessungen konnten zudem stockwerksübergreifende Trajektorien ermöglicht werden.

Abstract

Nowadays, pedestrians mainly use smartphones with implemented GNSS-Chips for navigation within an accuracy of a few meters. Due to multipath and signal loss, this positioning method fails in indoor areas. Especially for large building complexes a suitable positioning and guidance method would be of great benefit. Instead of the high-maintenance and cost-intensive infrastructure-based indoor positioning methods, smartphones with their implemented sensors can provide autonomous positions for pedestrians with the help of a step-based PDR-algorithm. Regarding this, step-events are being detected with the help of acceleration measurements. Furthermore, the heading of the pedestrian can be calculated with gyroscope data. With this two information and a known step length, the change of position due to one step can be estimated. The resulting trajectory differs from the actual trajectory because of the low-cost characteristics of the measurement units, since these sensors are not designed for navigation-specified applications. This causes the necessity to support the PDR-trajectory through additional techniques.

For this master thesis, a graph-based PDR-algorithm was developed. Therefore the position solution is restricted to a graph, which is designed on the basis of a building plan. Consequently, no contradictory trajectories through walls or outside of a building can be calculated. The positions are being restricted on the graph by five status inquiries. They are based on the current difference between the edge direction and the accumulated heading rate as well as on the current position on the edge. Furthermore, the drift compensation of the graph-based approach is another benefit. It has to be mentioned that the angular rate data will only be accumulated during one edge and not during the entire measurement time. The advantages of this approach were verified by the use of trajectories at two testing environments (Steyrergasse 30 and Inffeldgasse 16 in Graz). Through additional barometer measurements trajectories across different floors were also made possible.

Abkürzungsverzeichnis

API	Application Programming Interface
APSP	All Pairs Shortest Paths
BIG	BundesImmobilienGesellschaft
BLE	Bluetooth Low Energy
CAD	Computer Aided Design
DME	Distance Measuring Equipment
DR	Dead Reckoning
DXF	Drawing Interchange Format
EWMA	Exponentially Weighted Moving Average
FFT	Fast Fourier Transformation
GK	Gauß-Krüger
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
IMU	Inertial Measurement Unit
KNN	Künstliche Neuronale Netzwerke
LOP	Line Of Position
MEMS	MikroElektroMechanische Systeme
PDR	Pedestrian Dead Reckoning
SPSP	Single Pair Shortest Path
SSSP	Single Source Shortest Paths
WLAN	Wireless Local Area Network

Inhaltsverzeichnis

Kapitel 1

Einführung

Heutzutage ist die Navigation für Fußgänger im Outdoor-Bereich bereits fest im Alltag etabliert. Durch GNSS (Global Navigation Satellite System) gestützte Positionslösungen in Verbindung mit open source oder kommerziellen Kartendiensten lassen sich am Smartphone oder Navigationsgerät Fußgänger auf wenige Meter genau zu gewünschten Zielen leiten. Da sich die meisten User jedoch größtenteils in Innenräumen aufhalten, versagen diese Positionierungsmethoden aufgrund von starken Signalabschwächungen bzw. Signalverlusten oder Mehrwegeffekte.

Gerade in Innenräumen kann eine akkurate Positionsinformation von großer Bedeutung sein. Beispielsweise würde in Flughäfen, Einkaufszentren oder Krankenhäusern ein Indoor-Navigationssystem von wesentlichem Vorteil sein, um die Organisation und Effizienz in derartigen Gebäuden verbessern zu können. Dementsprechend gibt es ein starkes Bestreben robuste Indoor-Navigationslösungen zu entwickeln.

Die Mehrheit der sich im Umlauf befindlichen Smartphones haben Sensoren implementiert, die eine Indoor-Positionslösung ermöglichen. Durch MikroElektroMechanische Systeme (MEMS), Wireless Local Area Network (WLAN) oder Bluetooth ist es möglich, die Trajektorie eines Fußgängers zu bestimmen. Diese Messeinheiten sind jedoch nicht für navigationsspezifische Anwendungen konzipiert, sondern dienen an erster Stelle einem anderen Zweck. So fungieren beispielsweise WLAN und Bluetooth primär als Kommunikationsschnittstellen. Die inertialen MEMS-Einheiten, welche aus Akzelerometern und Gyroskopen bestehen, sind hauptsächlich für die Bildschirmausrichtung oder Smartphone-Spiele verbaut. Die möglichst kleinen und kostengünstigen Sensoren weisen demnach nur eine geringe Präzision und Sensitivität in Bezug auf Navigationsanwendungen auf. Dies hat zum Nachteil, dass die Navigationsperformance darunter stark leidet. Aufgrund

dessen werden Techniken und Verfahren entwickelt, die trotz der low-cost Sensoren eine bestmögliche Navigationslösung erlauben.

Ein möglicher Ansatz der Indoor-Trajektorienbestimmungen basierend auf Smartphone Sensoren nennt sich Schritt-basiertes Pedestrian Dead Reckoning (PDR). Diese Positionierungsmethode beruht auf dem Koppelnavigationskonzept. Dabei wird die relative Positionsänderung fortlaufend zur letzten bekannten Position addiert. Die elementaren Komponenten eines PDR werden wie folgt zusammengefasst:

- Schritterkennung
- Richtungsschätzung
- Schätzung der Schrittlänge

Aufgrund der Schritterkennung ist dem PDR-Algorithmus bekannt, zu welchem Zeitpunkt der Fußgänger einen Schritt tätigt. Durch die Schrittlänge und Bewegungsrichtung zum Zeitpunkt des Schrittes kann die relative Positionsänderung berechnet werden. Beginnend an einem bekannten Punkt wird die Trajektorie somit schrittweise bestimmt. Solche Dead Reckoning (DR) Algorithmen haben den Nachteil, dass sich die Sensorfehler immer weiter fortpflanzen, weil die aktuelle Position von allen vorherigen Positionen und folglich auch deren Ungenauigkeit abhängig ist. Charakteristisch ist vor allem der Drift im Heading einer PDR-Trajektorie. Dies gibt Anlass, die Lösungen durch zusätzlichen Verfahren zu stützen.

Eine Methode, um DR Algorithmen zu verbessern, besteht darin, sie mit Informationen über das Gebäude, in dem sich der Fußgänger befindet, zu versehen. Aufbauend auf einem Gebäudeplan kann ein Graph bestehend aus Knoten und Kanten erstellt werden. Dieser Graph beschreibt mit seiner simplen Form die zugrundeliegende Gebäudestruktur wie z.B. Korridore, Stiegen oder Eingänge. Ähnlich wie in Fahrzeugnavigationsgeräten, bei denen sich Autos nur auf den in der Karte eingezeichneten Straßen bewegen können, so sind auch die Positionslösungen des PDR-Algorithmus auf die Knoten und Kanten des Netzwerks beschränkt. Dadurch wird die Trajektorie des Fußgängers stark generalisiert dargestellt.

Entscheidungen, wie man sich auf dem Graphen fortbewegt, werden über die Richtungsabweichung zur aktuellen Kantenausrichtung getroffen. Zeigt diese Richtungsinformation an, dass man sich bei einem Schritt von der aktuellen Kante wegbewegt, muss eine zur momentanen Richtung passende Kante gefunden werden. Ansonsten bewegt man sich entlang der aktuellen Kante. Es wird somit fortwährend berechnet, wie sich die Bewegungsrichtung

des Fußgängers zur aktuellen Kantenrichtung verhält. Diese Richtungsabweichung wird dabei durch die Kumulation der Gyroskopdaten berechnet. Erreicht der User eine neue Kante wird die Abweichung wieder auf Null gesetzt. Der große Vorteil dieses gestützten Verfahrens besteht somit darin, dass die Drehraten nicht über den gesamten Messzeitraum aufkumuliert werden, sondern nur für den Zeitraum einer Kante. Dadurch kann der Drift im Heading kompensiert werden.

Ziel dieser Masterarbeit ist die Erstellung eines in dieser Weise konzipierten Graphen-basierten PDR-Algorithmus mit Smartphone Sensoren. Wesentliche Bestandteile der Arbeit gliedern sich einerseits in der Aufbereitung eines geeigneten Graphen für die Testumgebungen Steyrergasse 30 und Inffeldgasse 16 und andererseits in der Implementierung des PDR-Algorithmus, welcher eine driftfreie Trajektorie durch ein ganzes Gebäude ermöglichen soll.

Diese Masterarbeit ist in 8 Kapitel aufgebaut. Zu Beginn werden allgemeine Grundlagen der Positionsbestimmung und die Funktionsweise der vorhandenen Smartphone-Sensoren erklärt. Diese zwei Kapitel sollen neben der Graphentheorie in Kapitel 6 für die Masterarbeit eine grundlegende Wissensbasis schaffen. In Kapitel 4 werden die Grundlagen des Schritt-basierten PDR-Algorithmus erläutert, sowie die Modifizierung durch zusätzliche Techniken. Der Hauptteil der Arbeit beschreibt den Aufbau und die Funktionsweise des entwickelten Graphen-basierten PDR-Algorithmus. Die Ergebnisse und Schlussfolgerungen, welche in den Testgebieten erzielt wurden, befinden sich im darauffolgenden Kapitel. Den Abschluss bildet ein Resümee und Ausblick.

Kapitel 2

Grundlagen der Positionsbestimmung

Die Ursprünge der Navigation liegen in der maritimen Seefahrt. Denn früher war es für Seefahrer essentiell sich auf dem offenen Meer orientieren zu können, um die Steuerung der Schiffe auf die geplante Route abzustimmen. Diese grundlegende Aufgabe von Positionierung, Routing und Guidance hat sich bis heute nicht verändert. Jedoch beschränkt sich die Navigation heute nicht nur auf die Schifffahrt, sondern ist bei navigationsspezifischen Anwendungen auf dem Festland sowie in der Luft- und Raumfahrt fest verankert. Die Positionsbestimmung kann dabei, wie in Kapitel 2.1 und 2.2 genauer beschrieben, in zwei fundamentalen Techniken unterschieden werden: einerseits eine absolute Positionierung, welche die Koordinaten eines unbekannten Standpunktes unabhängig von den vorherigen Positionslösungen direkt bestimmt, und andererseits ein relatives Verfahren. Diese Methode berechnet mit Hilfe des zuletzt bekannten Punktes und des Differenzvektors zwischen neuer und alter Position den gesuchten Standpunkt. Abschließend ist anzumerken, dass sich die Ausführungen in Kapitel 2 auf Hofmann-Wellenhof et al. [9], [10] stützen.

2.1 Absolute Positionierung

Wie bereits in der Einleitung erwähnt, beschäftigt sich die absolute Positionierung mit der direkten Bestimmung der unbekannten Positionen ohne dabei vorherige Positionslösungen in die Berechnung mit einzuschließen. Dabei wird durch Messungen von oder zu fixen Referenzpunkten auf die Position des Neupunktes geschlossen. Der Vorteil dieser Methode

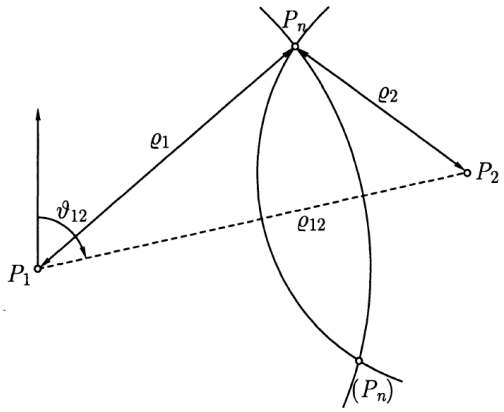


Abbildung 2.1: Rho-rho fixing aus Hofmann-Wellenhof et al. [9]

liegt vor allem in der Langzeitstabilität der Positionslösungen, da sich Fehler nicht über die Zeit fortpflanzen.

Heutzutage verwenden die meisten Positionierungsmethoden im Navigationsbereich Streckenbeobachtungen aus Laufzeitmessungen. Je nach Messtechniken ergeben sich verschiedene Methoden, wobei die nachfolgenden Techniken zur Vereinfachung für den 2D Fall erklärt werden.

Rho-rho fixing

Eine Positionierungsmethode mit fehlerfreien Streckenmessungen ist die in Abbildung 2.1 dargestellte *Rho-rho fixing* Methode. Bei diesem Ansatz wird der Neupunkt P_n durch Streckenbeobachtungen ϱ_1, ϱ_2 von zwei Fixpunkten P_1, P_2 berechnet. Die LOP einer gemessenen Distanz wird durch einen Kreis mit Radius ϱ und dem jeweiligen Fixpunkt als Mittelpunkt beschrieben. Der Schnittpunkt beider Kreise repräsentiert den gesuchten Neupunkt. Ein Defizit tritt bei dieser Methode dann auf, wenn die Punkte P_1, P_2, P_n nahezu auf einer Linie liegen. Dadurch entstehen schleifende Schnitte, welche zu einem enormen Genauigkeitsverlust führen. Bestmögliche Ergebnisse würden bei einem 90° Schnittwinkel der LOPs erzielt werden. Ein weiterer Nachteil entsteht durch die Mehrdeutigkeit der Positionslösung, da sich die Kreise in zwei Schnittpunkten schneiden. Benützt man zusätzliche Informationen, wie z.B. Näherungskoordinaten, kann jedoch ein Schnittpunkt ausgeschlossen werden.

Der *Rho-rho fixing* Ansatz kann beispielsweise durch simultane Messungen mehrerer Tachymeter oder DistanceMeasuring Equipments (DME) Verwendung finden. Charak-

teristisch dabei sind die Zweiwegmesssysteme. Dadurch erfolgt die Laufzeitmessung des ausgesendeten Signals durch ein und dieselbe Uhr.

Pseudorange position fixing

Bei Einwegmessungen wie z.B. bei GNSS wird die Laufzeit durch die Senderuhr und Empfängeruhr erfasst. Diese beiden Uhren sind jedoch nicht synchron und besitzen in den meisten Navigationssystemen auch unterschiedliche Genauigkeiten. Der sich daraus ergebende Fehler in der Streckenmessung muss bei einer Positionsermittlung berücksichtigt werden. Beispielsweise würde bei einer GPS (Global Positioning System) Streckenmessung ein Fehler von einer Mikrosekunde durch die Ausbreitungsgeschwindigkeit von 300.000.000m/s (Lichtgeschwindigkeit) in einem Fehler von 300m in der Streckenmessung resultieren.

Wenn bei der Positionsberechnung ein Uhrenfehler als unbekannter Parameter beachtet werden muss, spricht man von einem *Pseudorange position fixing*. Dabei werden von Fixpunkten Pseudoranges R_i zu dem unbekannten Punkt gemessen. Die Beobachtungen bestehen dabei aus einer geometrischen Strecke ϱ_i. Diese ist zusätzlich um einen Streckenfehler $\Delta\varrho$, verursacht aus dem Uhrenfehler, verfälscht:

$$R_i = \varrho_i + \Delta\varrho \tag{2.1}$$

Wie in Abbildung 2.2 ersichtlich, ist die LOP einer Pseudorange ein Kreis mit dem Radius R_i. Die Mittelpunkte der Kreise sind durch die Festpunkte P_i definiert. Im Unterschied zu dem *Rho-rho fixing* Ansatz wird bei der *Pseudorange position fixing* Methode jedoch der Neupunkt nicht durch den Schnittpunkt der Kreise repräsentiert. Die unbekannte Position P_n kann als Mittelpunkt eines Innenkreises mit dem Radius $|\Delta\varrho|$ interpretiert werden.

Für diese Problemstellung, muss somit neben den zwei unbekannten Koordinaten des Neupunktes der Uhrenfehler mitgeschätzt werden. Dementsprechend benötigt man Pseudorangemessungen von drei Fixpunkten, damit das Gleichungssystem mit den drei Unbekannten nicht unterbestimmt ist. Im Falle von GNSS müssen mindestens vier Pseudoranges gemessen werden, um die 3D Koordinaten und den Fehler $\Delta\varrho$ zu bestimmen.

Hyperbolische Positionierung

Neben dem Schätzen des Uhrenfehlers kann als Berechnungsalternative auch der Fehler $\Delta\varrho$ durch das Bilden von Streckendifferenzen eliminiert werden. Dadurch ergibt sich aus der

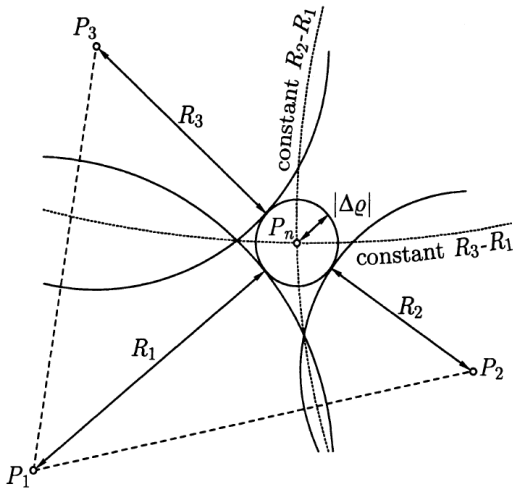


Abbildung 2.2: Pseudorange position fixing und Hyperbolische Positionierung aus Hofmann-Wellenhof et al. [9]

Gleichung 2.1 durch Differenzbildung zweier Streckenbeobachtungen von unterschiedlichen Festpunkten folgende Formel:

$$R_i - R_1 = \varrho_i - \varrho_1 \tag{2.2}$$

Die geometrische Form, welche die Menge aller Punkten repräsentiert, für die der Betrag der Differenz der Entfernungen zu zwei Festpunkten konstant ist, nennt sich Hyperbel. Da dies genau durch Formel 2.2 ausgedrückt wird, repräsentiert die LOP in diesem Fall eine Hyperbel, siehe Abbildung 2.2. Bildet man von einem zusätzlichen bekannten Punkt mit einem bereits verwendeten Festpunkt eine zweite Streckendifferenz, kann der Neupunkt als Schnittpunkt der Hyperbeln berechnet werden. Keine Lösung würde sich bei diesem Ansatz, sowie bei der *Pseudorange position fixing* Methode, ergeben, wenn alle drei Fixpunkte auf einer Linie situiert wären. Diese hyperbolische Positionierung benötigt somit genauso wie die *Pseudorange position fixing* Methode eine zusätzliche dritte Messung um die gesuchten Neupunktkoordinaten zu ermitteln. Abschließend ist zu erwähnen, dass analytisch betrachtet, beide Methoden die gleichen Ergebnisse für die zwei gesuchten Neupunktkoordinaten liefern.

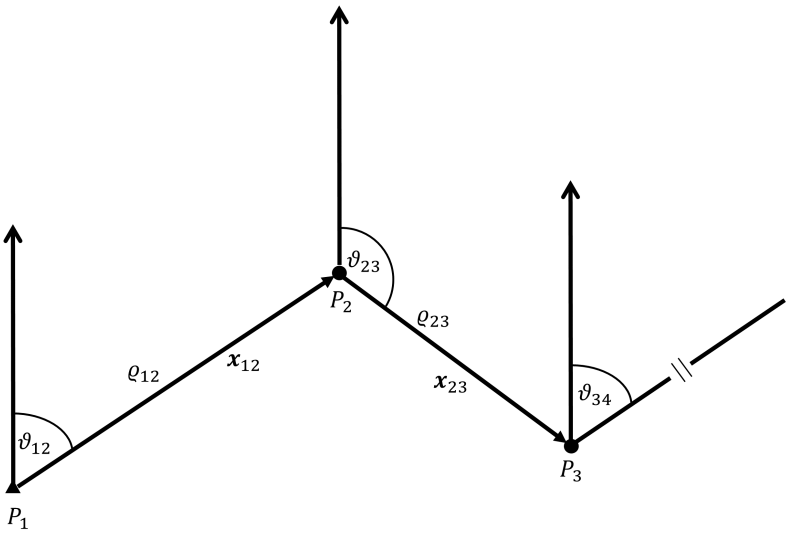


Abbildung 2.3: Konzept relative Positionsbestimmung

2.2 Relative Positionierung

Die zweite fundamentale Art, eine Position zu bestimmen, ist die relative Positionierung. Der gesuchte Neupunkt $\mathbf{x}_2$ wird dabei durch eine Vektoraddition zwischen den Koordinaten eines Fixpunktes $\mathbf{x}_1$ und einem bekannten Basislinienvektor $\mathbf{x}_{12}$ (Vektor zwischen Neupunkt Fixpunkt) berechnet.

$$\mathbf{x}_2 = \mathbf{x}_1 + \mathbf{x}_{12} \tag{2.3}$$

Des Weiteren wird bei der relativen Positionierung das Schema der Formel 2.3 nicht nur einmal ausgeführt, sondern man startet von einem initialen Punkt und fügt fortlaufend zu den berechneten Positionen die neuen Basislinienvektoren hinzu, siehe Abbildung 2.3. Dementsprechend beruht der aktuell berechnete Punkt auf allen vorherigen Punkteigenschaften, insbesondere Fehlern.

Der Unterschied zwischen verschiedenen DR Methoden liegt in der Berechnung des Basislinienvektors. Nachfolgend werden zwei Arten erklärt:

Rho-theta Technik
Bei dieser Methode wird die Basislinie durch eine Distanzmessung ϱ und eine orientier-

te Richtung ϑ beschrieben. Wie in Abbildung 2.3 ersichtlich, kann beispielsweise der Basislinienvektor $\mathbf{x}_{12}$ über die Formel 2.4 berechnet werden:

$$\mathbf{x}_{12} = \varrho_{12} \begin{bmatrix} \cos \vartheta_{12} \\ \sin \vartheta_{12} \end{bmatrix} \tag{2.4}$$

Die benötigten Messgrößen können dabei auf verschiedene Art und Weise generiert werden. In der Vermessung liefert meist ein Tachymeter die Distanz und Richtungsinformation. Anwendung findet die *Rho-theta Technik* dann bei Polygonzügen, welche für die meisten vermessungstechnischen Aufnahmen die grundlegenden Fixpunkte bereitstellen. Im gegensätzlichen Fall der relativen Positionsbestimmung eines Fußgängers kann die Distanz und die orientierte Richtung auch durch die Schrittlänge und die Bewegungsrichtung eines Fußgängers beschrieben werden, siehe Kapitel 4.1.

Inertialnavigation

Die Inertialnavigation basiert auf dem fundamentalen Prinzip, dass der Basislinienvektor durch eine zweimalige Integration von Beschleunigungen entlang der Achsen eines Koordinatensystems bestimmt werden kann. Die Ausrichtung wird dabei durch Drehratenmessungen eines Gyroskops berücksichtigt. Integriert man einmal die gemessenen Beschleunigungen $\ddot{\mathbf{x}}$ über die Zeit, erhält man den Geschwindigkeitsvektor $\dot{\mathbf{x}}(t)$, siehe Formel 2.5.

$$\dot{\mathbf{x}}(t) - \dot{\mathbf{x}}_1 = \int_{t_1}^{t} \ddot{\mathbf{x}}(\tau)\, \mathrm{d}\tau \tag{2.5}$$

Aufbauend auf der oben genannten Gleichung kann durch eine erneute Integration der Basislinienvektor $\mathbf{x}_{12}$ berechnet werden:

$$\mathbf{x}_{12} = \mathbf{x}_2 - \mathbf{x}_1 = \int_{t_1}^{t_2} \dot{\mathbf{x}}(t)\, \mathrm{d}t \tag{2.6}$$

Da durch die Integration zwei Integrationskonstanten entstehen, müssen zusätzliche Informationen einfließen. Diesbezüglich wird in Formel 2.5 eine initiale Geschwindigkeit $\dot{\mathbf{x}}_1$ benötigt und in Formel 2.6 muss für die absoluten Koordinaten des Neupunktes eine Startposition $\mathbf{x}_1$ gegeben sein. Benötigt man nur den Basislinienvektor kann die Startposition vernachlässigt werden.

Wie bereits in der Einführung erwähnt, haben relative Positionierungsmethoden den großen Nachteil, dass sie aufgrund der Abhängigkeit aller zuvor berechneter Positionslösungen nicht langzeitstabil sind. Dementsprechend sind DR-Methoden Drift-behaftet. Wie und in welcher Form sich die Fehler bei DR-Lösungen fortpflanzen, wird in Kapitel 6.4 genauer erörtert.

2.3 Koordinatensysteme

Eine Position wird durch Koordinaten eines klar definierten Referenzsystems beschrieben. Festgelegt wird ein Koordinatensystem durch sein geodätisches Datum, welches die Angabe des Koordinatenursprungs und die Ausrichtung der Achsen, umfasst. Diese Systeme bilden die Grundlage, sodass Positionierungsmethoden sinnvolle und reproduzierbare Koordinaten in einem kartesischen n-dimensionalen Raum liefern können. In der Geodäsie werden jedoch nicht nur kartesische Koordinaten angegeben, sondern auch ellipsoidische. Dabei wird die Erdfigur als Ellipsoid angenähert. Ein Punkt auf der Oberfläche wird dann durch die Lageparameter φ und λ sowie durch die ellipsoidische Höhe h beschrieben. Je nach Region gibt es verschiedene Referenzellipsoide, die die Erde in den jeweiligen Gebieten bestmöglich approximiert.

Erdfestes Äquatorsystem
Ein satellitengestütztes Positionierungssystem mit einer globalen Abdeckung benötigt für die Angabe von Punkten als Referenzsystem ein erdfestes Äquatorsystem, siehe Abbildung 2.6. Dabei dient das Geozentrum der Erde als Ursprung, die X_3-Achse weist in Richtung der Rotationsachse, die X_1-Achse nach Greenwich und die X_2-Achse komplettiert das Referenzsystem zu einem rechtsdrehenden Koordinatensystem. Durch die X_1-Achsendefinition rotiert es mit der Erde um die Rotationsachse mit, ansonsten würde ein und der selbe Punkt auf der Erdoberfläche seine Koordinaten aufgrund der Erdrotation zeitlich ändern.

Die Untersuchungen in dieser Masterarbeit benötigen jedoch hauptsächlich lokale Koordinatensysteme, welche nachfolgend erläutert werden:

Sensorsystem
Messungen eines Kamerasystems, Smartphones oder z.B. einer Inertial Measurement Unit (IMU) beziehen sich auf das dreidimensionale kartesische Sensorsystem. Dieses

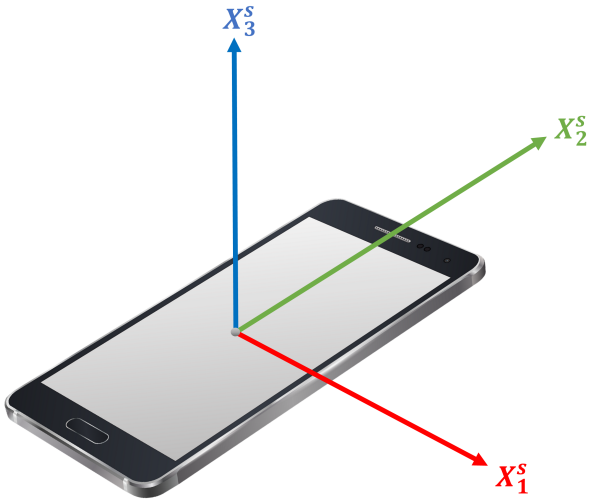


Abbildung 2.4: Sensorsystem

Koordinatensystem ist durch den Ursprung fest mit dem Messsystem verbunden und zeichnet die Messungen in den drei Achsrichtungen auf. Da sich die im Zuge des Kapitel 7 präsentierten Ergebnisse auf Smartphone-Messungen stützen, wird in Abbildung 2.4 das zugrunde liegende Sensorkoordinatensystem des verwendeten Smartphone visualisiert. Die X_1-Achse ist in Querrichtung, die X_2-Achse in Längsrichtung und die X_3-Achse ist orthogonal zur Bildschirmoberfläche definiert. Dementsprechend wird das Sensorsystem in einem rechtsdrehenden Koordinatensystem beschrieben.

Bodysystem

Das Bodysystem, oder auch Körpersystem, ist ebenfalls ein rechtsdrehendes dreidimensionales kartesisches Koordinatensystem, welches fix mit einem Objekt (Schiff, Auto, Flugzeug, Fußgänger...) verbunden ist. Die Abbildung 2.5 zeigt eine Realisierung dieses Koordinatensystems anhand einer Rakete. Dabei ist die X_1-Achse mit der Längsachse, die X_2-Achse mit der Querachse und die X_3-Achse mit der Vertikalachse des Flugobjektes gleichgesetzt. Aufgrund der Kopplung mit den Rotationsachsen des Objektes ist es möglich, die relative Orientierung der Rakete im Raum zu bestimmen. Die Lage im Raum kann dabei durch folgende drei Attitude-Parameter beschrieben werden:

- Roll: Drehung der Längsachse (X_1-Achse)
- Pitch: Drehung der Querachse (X_2-Achse)
- Yaw: Drehung der Vertikalachse (X_3-Achse)

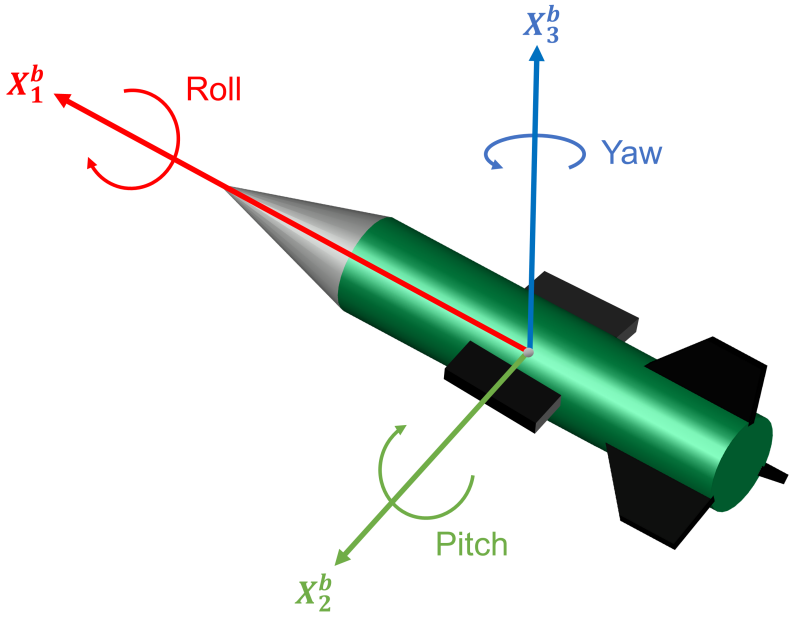


Abbildung 2.5: Bodysystem

Der Ursprung dieses System ist je nach Plattform ein spezifischer Punkt innerhalb des Objektes, in den meisten Fällen wird der Schwerpunkt verwendet.

Lokales Horizontsystem

Neben den bereits erörterten Koordinatensystemen ist es in den meisten Fällen notwendig, ein lokales kartesisches Koordinatensystem zu definieren, welches einen Bezug zur Erdoberfläche besitzt. Die Relevanz eines solchen Systems kann am besten anhand von Tachymetermessungen begründet werden. Dabei erfolgen die Beobachtungen in einem System, dessen Ursprung der aktuelle Standpunkt ist. Durch die Horizontierung des Gerätes weist dessen Stehachse in Richtung der Lotlinie. Die eingemessenen Punkte werden dann in Bezug auf die Nordrichtung aufgenommen. Genau auf diesem grundlegenden Konzept beruht die Definition des lokalen Horizontsystems. Wie in Abbildung 2.6 ersichtlich, ist der Ursprung dieses Koordinatensystems der momentane Standpunk P auf der Erdoberfläche. Die X_1^l-Achse weist nach Norden und die X_2^l-Achse nach Osten, dadurch wird an dem Standpunkt eine Tangentialebene aufgespannt. Je nachdem ob das lokale Horizontsystem rechts- oder linksdrehend definiert wird, weist die X_3^l-Achse Richtung Zenit oder Nadir. In Abbildung 2.6 wird durch die Nadirrichtung das System zu einem rechtsdrehenden Koordinatensystem komplettiert.

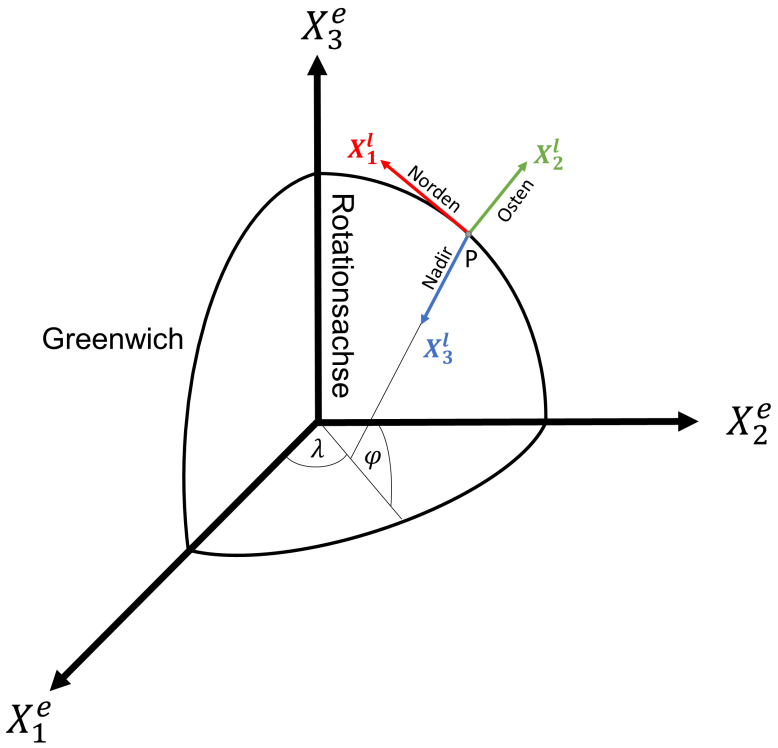


Abbildung 2.6: Lokales Horizontsystem

Gauß-Krüger Abbildung

Der in der Arbeit forcierte Graphen-basierte PDR-Algorithmus benötigt als Grundlage Gebäude- bzw. Lagepläne der Testgebiete. Diese Pläne werden von der BundesImmobilienGesellschaft (BIG) bereitgestellt und beziehen sich auf das österreichische Bezugssystem MGI (Militärgeographisches Institut). Als Abbildungsvorschrift verwendet dieses System eine Gauß-Krüger Abbildung, demnach beziehen sich die 2D Koordinaten der BIG-Pläne auf das Gauß-Krüger (GK) System.

Diese fundamentale Abbildung ermöglicht es, Punkte eines die Erdfigur beschreibenden Ellipsoids in die Ebene abzubilden, sodass ebene Kartendarstellungen sowie trivialere Berechnungsformen durchführbar sind. Wie in Abbildung 2.7 ersichtlich, ist die GK-Abbildung eine transversale Zylinderprojektion. Dieser Zylinder berührt dabei das Ellipsoid nur entlang des Bezugsmeridians.

Diese Abbildungsvorschrift gewährt eine winkeltreue Abbildung. Dadurch hat der Winkel in der Projektion den gleichen Wert wie auf dem Ellipsoid. Längentreu kann nur der Bezugsmeridian abgebildet werden. Diesbezüglich entstehen Verzerrungen je weiter der Punkt von diesem Meridian entfernt ist. Um diese Verzerrungen möglichst gering zu

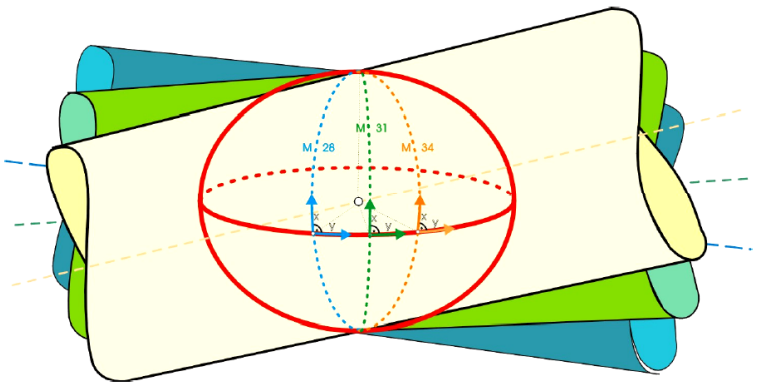


Abbildung 2.7: Gauß-Krüger Abbildung aus Otter et al. [14]

halten, wird nur ein 3° breiter Streifen des Bezugsmeridians abgebildet. Für die komplette Abdeckung Österreichs werden demnach drei Bezugsmeridiane (M28, M31 und M34) benötigt. Anzumerken ist, dass sich der fundamentale Bezugsmeridian nicht wie z.B. beim deutschen Bezugssystem auf Greenwich bezieht, sondern auf Ferro (westlichste Kanarische Insel). Im Falle von Greenwich würde für Österreich eine schlechte Konstellation entstehen, da vier Meridianstreifen nötig wären, um das ganze Land abzudecken und die Hauptstadt Wien genau an der Grenze zwischen zwei Streifen liegen würde.

Bezüglich der Ausrichtung der Koordinatenachsen repräsentiert die waagrechte Achse das Bild des Äquator und die senkrechte Achse das Bild des jeweiligen Bezugsmeridian. Weitere Informationen bezüglich des MGI-Systems werden in Otter et al. [14] angeführt.

2.4 Koordinatentransformationen

In der Regel werden Messungen in einem System ausgeführt, welches für die anschließenden Berechnungen oder Vergleichszwecke ungeeignet ist. Wie in Abbildung 2.8 ersichtlich, werden beispielsweise von einem auf einem Flugzeug montierten Messinstrument bestimmte Größen gemessen. Schlussendlich möchte man die Ergebnisse jedoch nicht im Sensorkoordinatensystem, sondern im lokalen Horizontsystem vorliegen haben.

Um dies bewerkstelligen zu können, muss zuerst zwischen Sensorsystem und Bodysystem transformiert werden. Der Vektor in $\mathbf{x}^s$ im Sensorsystem kann mit Hilfe einer Rotationsmatrix $\mathbf{R}_s^b$ in das Bodysystem transformiert werden, siehe Formel 2.7. (Ein möglicher Koordinatenursprungsoffset und ein Maßstabsfaktor wird der Einfachheit halber vernachlässigt.)

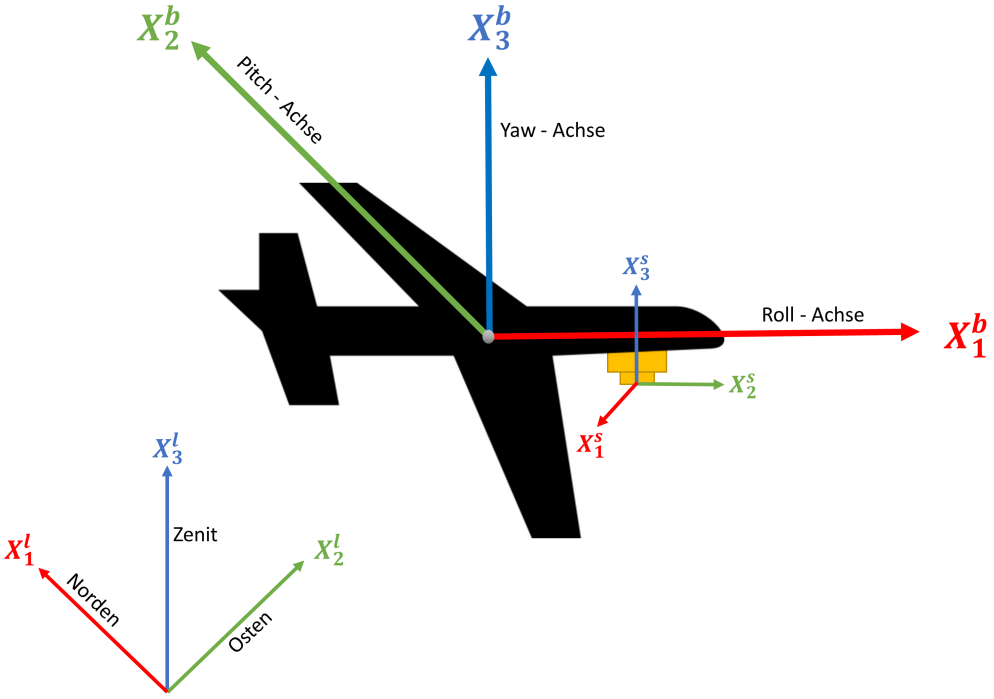


Abbildung 2.8: Kombination Sensorsystem, Bodysystem und lokales Horizontsystem

$$\mathbf{x}^b = \mathbf{R}_s^b \mathbf{x}^s \tag{2.7}$$

Eine allgemeine Rotation um alle Achsen eines 3D Koordinatensystems kann durch die Kombination der folgenden elementaren Rotationsmatrizen beschrieben werden. Die Matrizen in Formel 2.8 repräsentieren dabei jeweils die Rotation um eine Achse:

$$\begin{aligned}
\mathbf{R}_1(\alpha) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) \\ 0 & -\sin(\alpha) & \cos(\alpha) \end{bmatrix} \\
\mathbf{R}_2(\alpha) &= \begin{bmatrix} \cos(\alpha) & 0 & -\sin(\alpha) \\ 0 & 1 & 0 \\ \sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \\
\mathbf{R}_3(\alpha) &= \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{2.8}$$

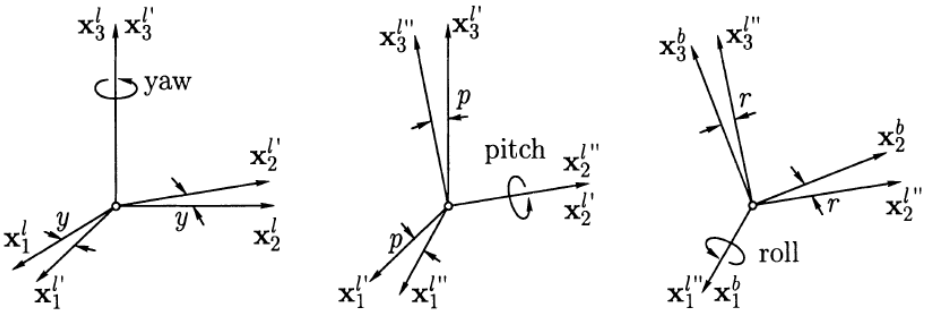


Abbildung 2.9: Rotationen um roll, pitch und yaw aus Hofmann-Wellenhof et al. [9]

Für die Transformation zwischen Sensor- und Bodysystem müssen somit die Drehwinkel in $\mathbf{R}_s^b$ bekannt sein. Die Information über diese Winkel erhält man über Kalibrierungsverfahren. In den meisten Fällen werden jedoch die Messsensoren so montiert, dass die Achsen des Sensorsystems mit den Achsen des Bodysystems äquivalent sind. Dadurch entfällt die Transformation, weil die Messungen direkt im Bodysystem getätigt werden.

Die Transformation zwischen dem Bodysystem und dem lokalen Horizontsystem wird durch die bereits in Kapitel 2.3 erwähnten Drehwinkel (*roll, pitch, yaw)* beschrieben. Diese Winkel legen die relative Orientierung des Objektes im Bezug zum lokalen Horizontsystem fest. Definiert wird die Rotationsmatrix nach Hofmann-Wellenhof et al. [9] als Transformation vom lokalen Horizontsystem in das Bodysystem:

$$\mathbf{R}_l^b = \mathbf{R}_1(roll)\mathbf{R}_2(pitch)\mathbf{R}_3(yaw) \tag{2.9}$$

Durch

$$\mathbf{R}_b^l = \mathbf{R}_l^{b\mathsf{T}} \tag{2.10}$$

kann jedoch auf die inverse Operation geschlossen werden.

Teilt man $\mathbf{R}_l^b$ in seine zugrundeliegenden Matrizen auf, wird zuerst mit dem Winkel *yaw* um die X_3-Achse, anschließend mit *pitch* um die X_2-Achse und abschließend durch *roll* um die X_1-Achse rotiert. Zur Verdeutlichung dieser Kombination von Drehwinkel befindet sich in Abbildung 2.9 die Illustration dazu.

Kapitel 3

Smartphone-Sensoren

Heutzutage ist die Mehrheit der Fußgänger mit einem Smartphone ausgestattet. Diese mobilen Endgeräte besitzen neben den standardmäßigen Kommunikationsfunktionen viele zusätzliche Sensoren, die eine Vielzahl an Einsatzmöglichkeiten bieten. Ein mögliches Anwendungsfeld ist, wie bereits in den vorigen Kapiteln erwähnt, die Navigation.

Laut Banville und Diggelen [1] wurde bereits 1999 das erste Smartphone mit einem GPS-Empfänger ausgestattet. Mittlerweile gibt es in der Smartphone-gestützten Fußgängernavigation nicht nur Sensoren für die satellitengestützte Positionierung, sondern es werden in Bezug auf die Innenraumpositionierung hauptsächlich Funktechnologien (WLAN, Bluetooth) und MEMS Sensoren eingesetzt.

Der Algorithmus aus Kapitel 6 benötigt für die relative Positionsbestimmung die in den folgenden Kapitel beschriebenen MEMS Sensoren und für die Startposition Bluetooth-Messungen. Der Aufbau und die Funktionsweise der beschriebenen Sensoren basiert auf den Ausführungen aus Titterton und Weston [17], Groves [6], Willemsen [20] und Wild-Pfeiffer und Schäfer [18].

3.1 Mikroelektromechanische Systeme

Mikroelektromechanische Systeme (MEMS) sind eine Technologie, die eine Miniaturisierung von Sensoren durch winzige kleine Bauteile ermöglicht. In diesen Systemen werden in einem Chip durch integrierte Schaltkreise mikromechanische, elektrische, optische und chemische Komponenten mit Logikelementen vereint. Durch Integration der einzelnen Komponenten in einem MEMS-Sensor lassen sich physikalische Größen, wie z.B. Drehmomente,

Beschleunigungen, magnetische Flussdichte oder der Druck, in elektrische Spannung umwandeln. Der ganze Sensor hat nach Willemsen [20] in der Regel eine Abmessung zwischen 0,02mm bis einem Millimeter. Die untergeordneten Bauteile sind dabei selbst nur zwischen 0,001mm und 0,1mm groß.

Durch das hohe Funktionsspektrum bilden MEMS die Grundlage für eine Vielzahl von Anwendungsmöglichkeiten. Die Vor- und Nachteile dieser Technologie werden in Tabelle 3.1 angeführt, wobei die Vorteile durch diesen innovativen Lösungsansatz überwiegen.

Tabelle 3.1: Vor- und Nachteile von MEMS

Vorteile	**Nachteile**
Kostenersparnis	geringe Messgenauigkeit
Minituarisierung	
Kompaktheit	
hohes Funktionsspektrum	
Energie- und Leistungseffizienz	

Smartphones sind prädestiniert für das Anwendungsgebiet dieser low-cost Sensoren. Dabei sind hochpräzise Messungen nicht vorrangig, sondern die geringe Sensorgröße und die Kosten-, Energie- und Leistungseffizienz.

Zu Demonstrationszwecken befindet sich in Abbildung 3.1 die Platine eines Samsung Galaxy S5 Smartphones mit den eingezeichneten MEMS-Sensoren. Anzumerken ist, dass das in Kapitel 3.3 vorgestellte Smartphone der Vorgänger des Galaxy S5 ist. Die Funktionsweisen dieser Smartphones sind hinsichtlich der in dieser Arbeit vorgestellten MEMS-Sensoren gleichwertig. In den nachfolgenden Unterkapiteln werden jene Smartphone-MEMS beschrieben, die für die Positionsschätzung in dieser Arbeit Verwendung finden:

3.1.1 Akzelerometer

Ein Akzelerometer misst die auf eine bekannte Probemasse m wirkende Trägheitskraft F in Form einer linearen Beschleunigung $\bar{a}$. Dieser Zusammenhang beruht auf dem zweiten Newton‘schen Gesetz (*lex secunda*):

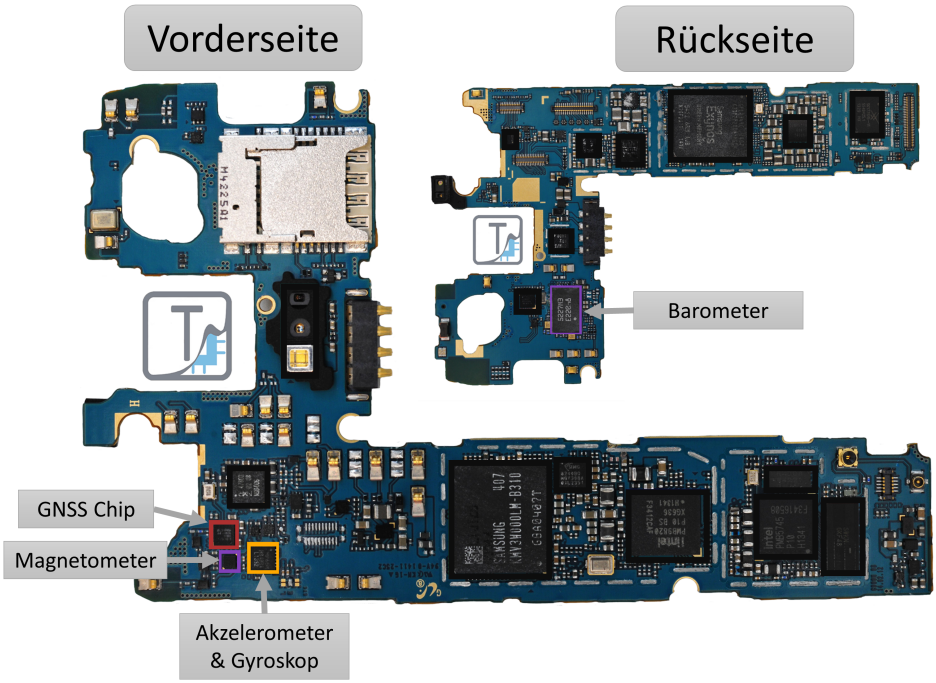


Abbildung 3.1: Übersicht der MEMS-Chips und des GNSS-Chip auf einer Samsung Galaxy S5 Platine aus Teardown.com [15]

$$F = m\bar{a} \tag{3.1}$$

Benützt man ein Feder-Masse Prinzip als Messtechnik, beinhaltet der Funktionsaufbau, wie in Abbildung 3.2 ersichtlich, zwei Federn, eine Prüfmasse und ein Gehäuse. Die Federn sind dabei durch die Prüfmasse verbunden und an einem Ende des Gehäuses fix verankert. Wirkt auf die Prüfmasse eine Kraft, so wird diese aus ihrer Nullposition proportional zur einwirkenden Kraft um Δl ausgelenkt. Durch die bekannte Auslenkung und je nach Federkonstante k kann die einwirkende Kraft durch Formel 3.2 beschrieben werden:

$$F = k\Delta l \tag{3.2}$$

Setzt man Formel 3.1 und 3.2 gleich, erhält man den Zusammenhang zwischen Auslenkung Δl und Beschleunigung $\bar{a}$:

$$\bar{a} = \frac{k\Delta l}{m} \tag{3.3}$$

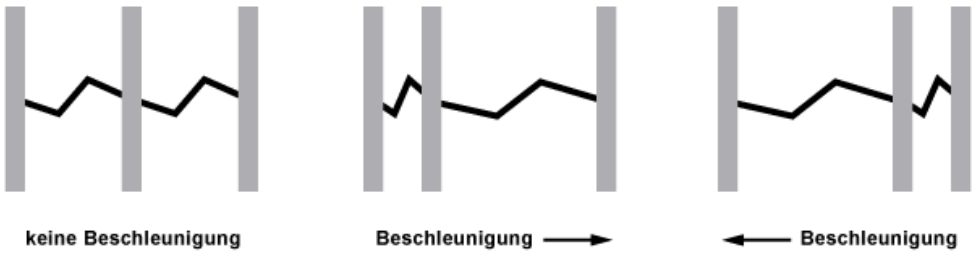


Abbildung 3.2: Funktionsaufbau eines Masse-Feder MEMS-Akzelerometer aus www.elektronik-kompendium.de [22]

Anzumerken ist, dass in den oben genannten Formeln ein inertialer Referenzrahmen angenommen wird. Dadurch müssen keine Scheinkräfte in den Beobachtungsgleichungen beachtet werden. Des Weiteren misst das Akzelerometer durch die permanente Erdbeschleunigung g nicht direkt die erzwungene kinematische Beschleunigung a. Durch die Differenzbildung

$$a = \bar{a} - g \tag{3.4}$$

kann jedoch auf die gesuchte Beschleunigung a geschlossen werden.

Laut Wild-Pfeiffer und Schäfer [18] lassen sich zur Messung von Beschleunigungen folgende MEMS-Systeme realisieren:

Kapazitive-MEMS: Die Messung der Beschleunigung wird durch die Änderung der Kapazität zweier Kondensatoren, welche durch die Auslenkung der Probemasse entsteht, bewerkstelligt.

Vibrations-MEMS: Zwei Quarze werden aufgrund der beschleunigten Probemasse in Schwingung versetzt. Anhand der Frequenzdifferenz wird die Beschleunigung gemessen

Piezoelektrische-MEMS: Realisiert wird die Beschleunigungsmessung durch eine Oberflächendeformation eines piezoelektrischen Materials.

Piezoresitive-MEMS: Bei dieser Methode wird aufgrund der Beschleunigung der sich ändernde Widerstand eines Materials gemessen.

3.1.2 Gyroskop

Ein Gyroskop dient dazu, Rotationsgeschwindigkeiten einer Prüfmasse zu messen. Durch diese gemessenen Winkeländerungen können Drehratensensoren die Lage des Objektes im Raum (Attitude-Parameter) bestimmen.

Früher wurden Drehratensensoren durch mechanische Kreisel in Kombination mit kardanischen Aufhängungen realisiert. Heute werden in hochwertigen IMU-Systemen meist optische Kreisel, welche auf dem Sagnac-Effekt beruhen, eingesetzt. Diese sehr präzisen Sensoren finden jedoch aufgrund der kostspieligen Erzeugung in der MEMS-Technologie keine Verwendung. Für diese Systeme werden deutlich unpräzisere aber kostengünstigere Vibrationskreisel eingesetzt.

Das grundlegende Prinzip der MEMS-Drehratensensoren beruht auf dem Corioliseffekt, welcher jedoch nur auftritt, wenn die Prüfmasse künstlich durch Vibrationen, in eine bestimmte Richtung, in Bewegung versetzt wird. Wie beispielsweise in Abbildung 3.3 ersichtlich, wird die Prüfmasse mit einer bestimmten Grundfrequenz in X-Richtung in Schwingung versetzt. Dadurch ergibt sich eine periodische Veränderung in der Geschwindigkeit v in diese Richtung. Rotiert man die Prüfmasse mit der Rotationsgeschwindigkeit Ω um die Z-Achse, entsteht durch den Coriolis Effekt ein Moment $\mathbf{F}$ in Y-Richtung:

$$\mathbf{F} = 2m\,(\mathbf{v} \times \mathbf{\Omega}) \tag{3.5}$$

Durch die Rotation und die infolge hervorgerufene Corioliskraft $\mathbf{F}$ wird nun auch in Y-Richtung eine Schwingung induziert, deren Amplitude ein Maß für die letztendlich gesuchte Rotationsgeschwindigkeit Ω ist.

Nach Wild-Pfeiffer und Schäfer [18] gibt es die folgenden Realisierungen von Drehratensensoren die auf dem physikalischen Grundprinzip des Corioliseffektes beruhen:

- Prismatische Biegebalken
- Stimmgabelstrukturen
- Rotationssymmetrische Resonanzkörper
- Feder-Masse Systeme

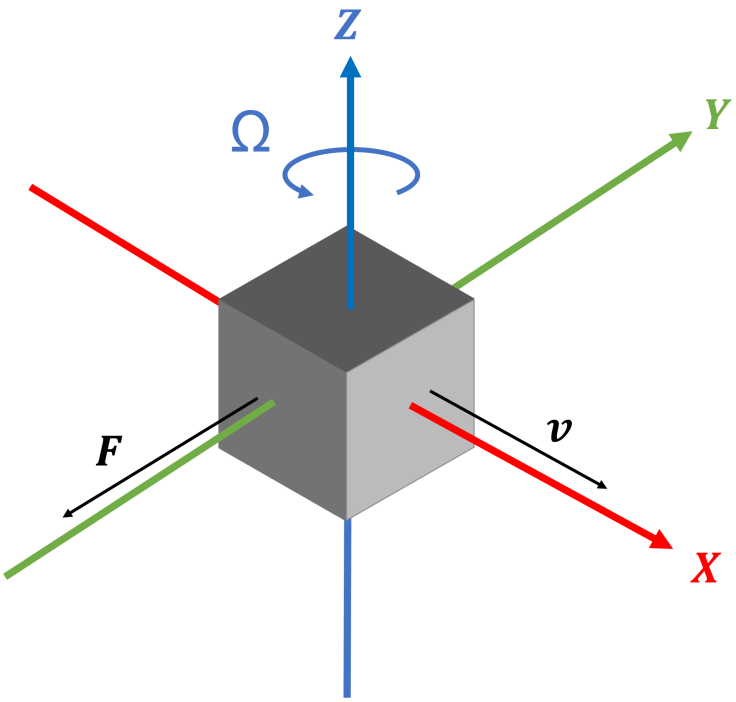


Abbildung 3.3: Vibrationskreisel - Coriolis Effekt

3.1.3 Barometer

Der Luftdruck hat die Eigenschaft, dass er mit zunehmender Höhe aufgrund des geringer werdenden Gewichts der Atmosphäre abnimmt. Vergleicht man den Druck, welcher durch ein Barometer gemessen wird, zwischen zwei Atmosphärenschichten, kann aufgrund der Druckdifferenz der Höhenunterschied ermittelt werden. Der Luftdruck selbst ist jedoch für einen spezifischen Punkt auf der Erdoberfläche nicht konstant. Je nach Wetterlage (Hoch- oder Tiefdruckgebiet), Temperatur und Sensorsystem variiert der gemessene Luftdruck. Bei MEMS Barometern in Smartphones kommt zusätzlich noch hinzu, dass die Messwerte aufgrund der Temperaturschwankungen innerhalb des Smartphones sich verändern. Der Temperatur- und Wetterdrift kann jedoch nach Willemsen [20] durch ein Referenzgerät, welches über den Messzeitraum an konstanter Position bleibt, herausgefiltert werden.

Die Höhenangabe eines Barometers kann vereinfacht durch die Internationale Höhenformel

$$h = \frac{T_0}{T_{grad}} \left(1 - \sqrt[5.225]{\frac{p(h)}{p_0}} \right) \tag{3.6}$$

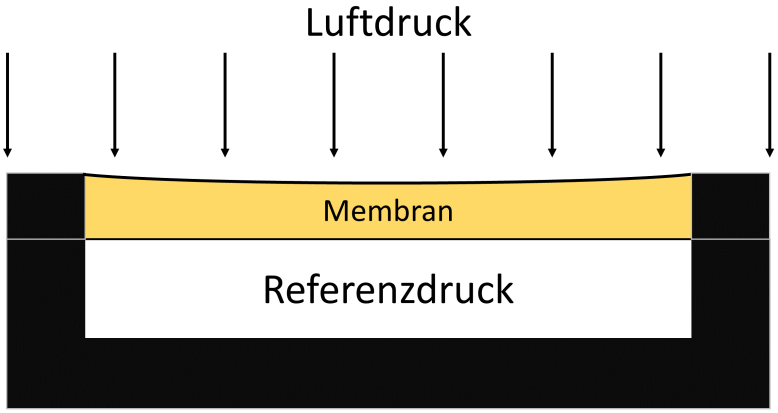


Abbildung 3.4: Messprinzip eines MEMS Barometer

bestimmt werden. Temperaturänderung mit zunehmender Höhe sowie die regionalen unterschiedlichen Zusammensetzungen der Atmosphäre werden in Formel 3.6 vernachlässigt.

Damit aus der Formel 3.6 eine Höheninformation entnommen werden kann, dient die Meereshöhe als Referenzhöhe sowie die internationale Standardatmosphäre für die Atmosphäre, welche durch die Parameter aus Tabelle 3.2 beschreiben wird.

Tabelle 3.2: Standardatmosphäre

T_0	Temperatur	15°
T_{grad}	Temperaturgradient	0,0065 K/m
p_0	Luftdruck	1013,25 hPa

Durch die vereinfachte Höhenformel sind die Genauigkeiten für die resultierenden Höhen begrenzt, genügen jedoch, um die angestrebten Anforderungen dieser Masterarbeit, insbesondere den Ausführungen aus Kapitel 6.11, gerecht zu werden.

Der Funktionsaufbau aus Abbildung 3.4 eines MEMS Barometers besteht aus einem fixen Gehäuse und einer veränderlichen Membran. Die Membran ist am Gehäuse fest verankert und an der Oberseite dem umgebenden Luftdruck und an der Unterseite einem bekannten Referenzdruck innerhalb des Gehäuses ausgesetzt. Je nach atmosphärischem Luftdruck verformt sich die Membran an der Oberseite. Dadurch ändert sich der spezifische Widerstand des Materials, welcher zur Berechnung des einwirkenden Luftdrucks dient.

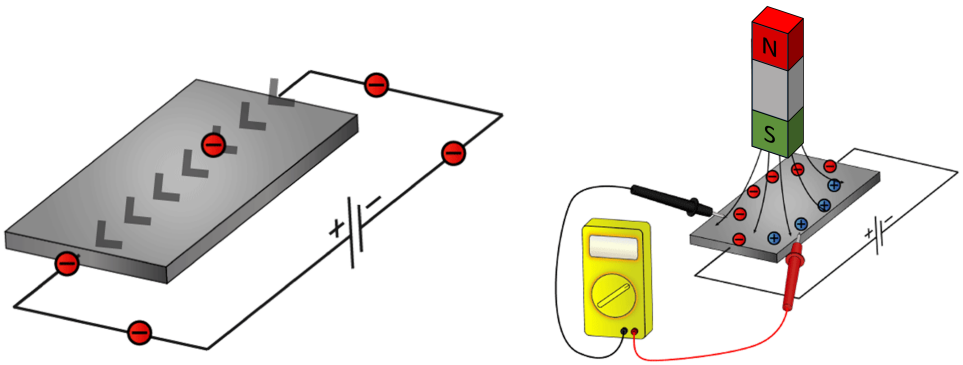


Abbildung 3.5: Hall Effekt und Messprinzip eines MEMS Magnetometers abgeleitet aus www.howtomechatronics.com [23]

3.1.4 Magnetometer

Zur Bestimmung der magnetischen Flussdichte (bzw. die daraus abgeleitete Nordrichtung) besitzen die meisten Smartphones ein Magnetometer, welches das umgebende Magnetfeld durch den Hall Effekt misst. Dazu wird eine leitende Platte, nach Willemsen [20] in der Größenordnung von $0.2 \times 0.2 \times 0.01$ mm^3, unter Strom versetzt. Die Elektronen fließen dadurch, wie in Abbildung 3.5 ersichtlich, gerade von einer Seite zur anderen. Wenn sich die Platte in einem Magnetfeld befindet, wird der geradlinige Strahl der Elektronen zu einer Seite abgelenkt. Dadurch entsteht eine messbare Spannungsänderung, welche zu einer bestimmten Magnetfeldstärke korrespondiert.

Das Erdmagnetfeld wird in der Einheit Tesla [T] angegeben und ist abhängig von der Position und der Zeit. Die größten Differenzen in der magnetischen Flussdichte liegen dabei zwischen dem Äquator $30\mu T$ und den Polen $60\mu T$.

3.2 Bluetooth

Neben den Sensoren aus Kapitel 3.1, besitzen Smartphones Komponenten zur Netzwerkkommunikation wie z.B. WLAN und Bluetooth. Hinsichtlich der Positionsbestimmung arbeiten beide Funktechnologien nach den gleichen Prinzipien. Der im Zuge dieser Masterarbeit umgesetzte Algorithmus benötigt lediglich für die Anfangsposition eine Positionsschätzung aus diesen Sensoren. Dabei wird, wie in Kapitel 6.2.3 beschrieben, die Bluetooth-Technologie

genützt. Aus diesem Grund und dem ähnlichen Funktionsprinzip wird im Folgenden nur Bluetooth beschrieben.

Bluetooth wurde entwickelt um verschiedene Devices kabellos miteinander zu verknüpfen. Die neueste Generation dieser Netzwerkkommunikation ist Bluetooth 4.0 welches ähnlich wie WLAN in einem Frequenzbereich von 2.4 GHz arbeitet. Bei dem Bluetooth 4.0 Release ist neben dem klassischen Bluetooth zusätzlich die Bluetooth Low Energy (BLE) Technologie eingeführt worden.

Ein Vorteil von BLE Devices in Bezug auf nicht-autonome Indoorpositionierungen ist die kleine und kostengünstige Bauweise sowie der minimale Stromverbrauch. Der sich daraus ergebende Nachteil ist die geringer werdende Reichweite. Nach Wilfinger [19] kann je nach Datenübertragungsrate ein BLE-System über mehrere Monate bis sogar Jahre nur durch eine simple Knopfbatterie ($\sim$ 3 Volt) betrieben werden. Anzumerken ist jedoch, dass das reguläre Bluetooth nicht mit BLE kompatibel ist. Demnach ist nicht jedes Bluetooth 4.0 Device automatisch für die BLE Technologie geeignet.

Zur Positionsbestimmung werden ein Bluetooth-Chip im Smartphone und, je nach Positionierungsmethode, ein oder mehrere Bluetooth-Beacons benötigt. Diese Beacons senden kontinuierlich ein standardisiertes Signal aus, welches von einer unbegrenzten Anzahl an Usern registriert werden kann. In der folgenden Auflistung werden Ansätze zur Bluetooth- wie auch WLAN-Positionsbestimmung vorgestellt, wobei nur das erste genannte *Cell of Origin* in dieser Arbeit Verwendung findet.

Cellof Origin: Empfängt man das Signal eines Senders ist die aktuelle Position jener Ort, welcher zuvor speziell für diesen Sender definiert wurde. Dieser kann beispielsweise durch einen Raum, einen Eingang oder die Koordinaten des Senders gegeben sein. Überlappen sich Signale mehrerer Sender wird der Ort des Senders mit der höchsten registrierten Signalstärke herangezogen.

WeightedCentroidLocalization: In Anlehnung an das *Cell of Origin* Prinzip wird bei dieser Methode jedoch nicht die Position des stärksten Senders gewählt, sondern es wird die aktuelle Position aus allen Sendestationen je nach Signalstärke gewichtet gemittelt.

Lateration Die Lateration beschreibt den Vorgang der Positionsbestimmung durch die in Kapitel 2.1 vorgestellte *Rho-rho fixing* Methode. Bei Bluetooth oder WLAN können die Strecken in Abhängigkeit der Signalstärke gemessen werden, wobei

Signalabschwächungen durch temporäre Störeffekte die Messungen stark verfälschen können.

Fingerprinting Für diese Methode muss ein Referenzpunktefeld als Grundlage erstellt werden. An diesen Punkten wird das Signalmuster aus allen empfangenen Bluetooth oder WLAN Signalen gemessen und in einer Datenbank gespeichert. Die aktuelle Position des Users wird dann durch den Vergleich zwischen dem aktuell gemessenen Signalmuster und jenen der Referenzpunkte geschätzt. Der Punkt mit dem ähnlichsten Muster ist die aktuelle Userposition.

3.3 Device

Das für die Testmessungen verwendete Smartphone ist, wie in Abbildung 3.6 ersichtlich, ein Samsung Galaxy S4. Dieses Smartphone besitzt einen Qualcomm Snapdragon 600 Quad-Core (1,9GHz) Prozessor mit zwei GByte Arbeitsspeicher. Das verwendete Betriebssystem ist das von Google entwickelte Android 5.0.1 Lollipop. (www.samsung.com [25])

Alle Sensoren aus den Kapiteln 3.1 und 3.2 sind in diesem Smartphone implementiert. Dementsprechend kann das Samsung Galaxy S4 auf eine Vielzahl an Sensoren für die Positionsbestimmung zurückgreifen und ist demnach für den entwickelten Algorithmus aus Kapitel 6 geeignet.

Die Arbeitsgruppe Navigation am Institut für Geodäsie an der TU Graz hat für dieses Smartphone eine Software entwickelt, mit der man auf die Rohdaten dieser Sensoren zugreifen kann. Die Tabelle 3.3 beinhaltet die für diese Arbeit verwendeten Sensoren mit den zugehörigen Datenraten.

Tabelle 3.3: Verwendete Sensoren und aufgezeichnete Datenrate

Sensor	**Einheit**	**Datenrate**
Akzelerometer	$[m/s^2]$	100 Hz
Gyroskop	$[rad/s]$	100 Hz
Magnetometer	$[\mu T]$	100 Hz
Druck	$[mbar]$	5 Hz
BLE Signal	$[dBm]$	1 Hz

Abbildung 3.6: Samsung Galaxy S4 aus www.samsung.com [25]

Kapitel 4

Pedestrian Dead Reckoning (PDR)

Eine speziell für Fußgänger entwickelte relative Positionierungsmethode mit inertialen Sensoren nennt sich PDR. Dabei wird nach der in Kapitel 2.2 beschriebenen *Rho-theta Technik* zur letzten bekannten Position fortwährend die relative Positionsänderung hinzugefügt. Dies entspricht der schrittweisen Fortbewegung des Fußgängers.

Zur Ermittlung des Differenzvektors $\mathbf{x}_{12}$ aus Formel 2.4 wird für PDR-Algorithmen zwischen einem Schritt-basierten und einem inertialen Ansatz unterschieden. Für die zugrundeliegende Präzision von Smartphone-Sensoren liefert der erstgenannte Ansatz robustere Lösungen. Dementsprechend beruht der entwickelte Graphen-basierte PDR-Algorithmus aus Kapitel 6 auf den Grundprinzipien eines Schritt-basierten PDR.

4.1 Schritt-basierter PDR

Ein Schritt-basierter PDR Ansatz besteht aus den in den nachfolgenden Unterkapiteln beschriebenen drei Teilen. Zuerst wird durch eine Schritterkennungsmethode der Zeitpunkt des Schrittevents ermittelt. Zusätzlich wird durch eine Richtungsschätzung das aktuelle Heading des Users bestimmt. Durch die Schrittlänge des Fußgängers kann die zurückgelegte Distanz pro Schritt repräsentiert werden und die neue Position des Fußgängers berechnet werden.

4.1.1 Schritterkennung

Für die Schritterkennung gibt es eine Vielzahl an Methoden die sich nach Brajdic und Harle [3] in eine Schrittdetektion in der Zeitdomäne, in der Frequenzdomäne und durch Feature classification unterteilen lassen. Die Grundlage für diese Methoden bilden dreiachsige Akzelerometermessungen.

Frequenzdomäne
Die Grundidee bei einer Schrittdetektion im Frequenzbereich ist es, die periodischen Schwingungen aufgrund der Beschleunigungen während einer Schrittabfolge in den Akzelerometerdaten zu analysieren. Dadurch kann man je nach Schrittfrequenz die Zeitreihe der Beschleunigungen in Intervalle zerlegen. Pro Intervall kann dann der maximale Wert als Schrittevent angesehen werden. Nach Gupta et al. [7] wird dabei die Fast Fourier Transformation (FFT) benützt um aus der Akzelerometerzeitreihe die Schrittfrequenzen heraus zu filtern. Probleme bei diesem Frequenz-basierten Ansätzen treten auf, wenn die Bewegung des Fußgängers nicht konstant bleibt. Sie kann beispielsweise durch schnelles Gehen oder Stehphasen verändert werden.

Feature classification
Bei dieser Methode werden zur Schrittdetektion Features in den Akzelerometermessungen erkannt. Diese Mustererkennung von Schritten findet in machine-learning-Algorithmen mit Hilfe von künstlichen neuronalen Netzwerken (KNN) Verwendung. Diese Algorithmen erzielen aufgrund ihrer komplexen Struktur sehr gute Ergebnisse, eignen sich allerdings je nach Ausmaß des KNN für Smartphone-Anwendungen aufgrund der geringen Berechnungskapazität nur bedingt. Zusätzlich benötigt diese Methode ein a-priori Verfahren, welches den Algorithmus durch Testdatensätze von Fußgängern hinsichtlich der Schritterkennung trainiert.

Zeitdomäne
Bei Schritterkennungen in der Zeitdomäne werden die Schritte durch zuvor definierte Bedingungen bzw. Grenzwerte erkannt. Ein Ansatz wird in Bauer [2] vorgestellt und nennt sich Exponentially Weighted Moving Average (EWMA). Die grundlegende Datenreihe bildet bei diesem Algorithmus die Länge des Beschleunigungsvektors. Dies hat zum Vorteil, dass der Algorithmus unabhängig von den Achsausrichtungen des Akzelerometers ist.

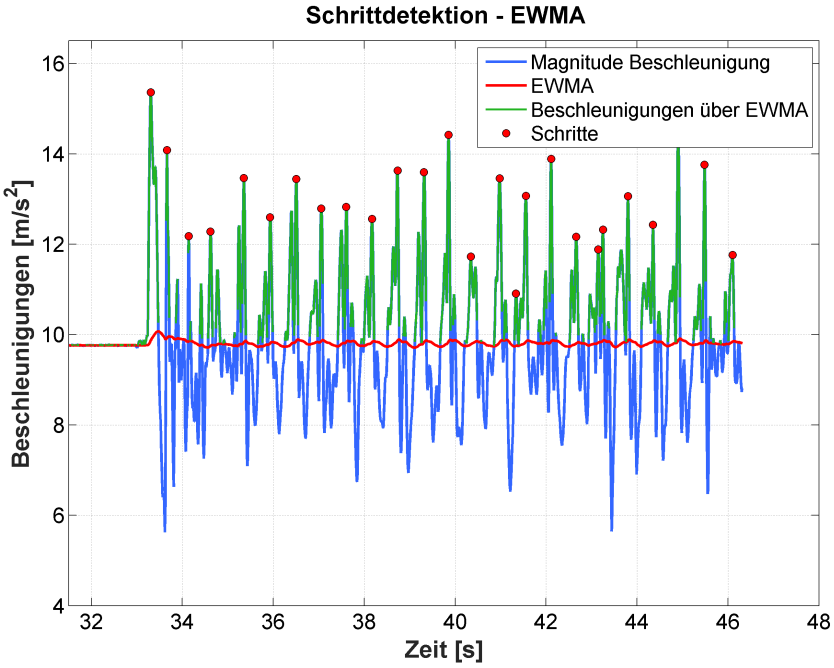


Abbildung 4.1: Schrittdetektion durch EWMA

Die Abbildung 4.1 beinhaltet eine Schritterkennung durch den EWMA-Algorithmus. Dabei werden zu Beginn die Messwerte durch eine Tiefpassfilterung geglättet. Anschließend wird durch einen EWMA-Filter ein exponentiell geglättetes Mittel berechnet, welches für die Schrittdetektion in Folge als Schranke dient. Die beschriebene Schranke wird dabei in Abbildung 4.1 in Rot dargestellt. Die Beschleunigungswerte über dieser Schranke (Grün) bilden Intervalle in denen jeweils der höchste Ausschlag repräsentativ für einen Schritt steht (Rote Punkte).

Für diese Arbeit wurde hinsichtlich der Schrittdetektion, neben der in Kapitel 6.3 vorgestellten Schritterkennung von Google, zu Testzwecken ein eigener Algorithmus in dieser Masterarbeit implementiert. Dieser wird jedoch nicht primär für den Graphen-basierten Algorithmus verwendet, sondern dient lediglich als Backup falls die Google Schritterkennung nicht zur Verfügung steht. Durch die einfache Realisierung wurde dabei auf einen Algorithmus in der Zeitdomäne zurückgegriffen.

Benötigt werden für diese Schritterkennung zwei Grenzwerte. Der erste Wert beinhaltet eine Schranke Z, welche angibt, ab wann ein Beschleunigungswert einem Schritt entsprechen kann. Damit der Algorithmus für verschiedene User und Datensätze ein sinnvolles Ergebnis liefern kann, berechnet sich dieser Grenzwert Z durch den Mittelwert und die

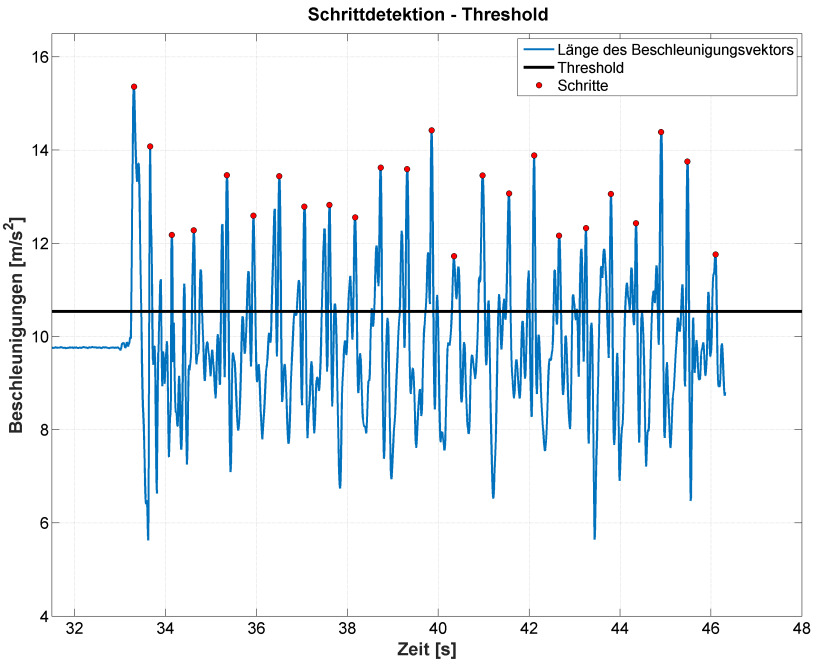


Abbildung 4.2: Schrittdetektion durch „find peaks“

Standardabweichung der Beschleunigungszeitreihe. Die zugrundeliegende Zeitreihe besteht dabei wieder aus der Länge des Beschleunigungsvektors.

$$Z = \mu + \sigma \tag{4.1}$$

Innerhalb der Beschleunigungswerte, welche über dem Grenzwert Z liegen, werden die maximalen Ausschläge so detektiert, dass im Bereich von $0,3$s nur ein Peak möglich ist. Diese zweite Bedingung beinhaltet somit eine Information, dass nur ein Schritt alle $0,3$s getätigt werden kann. Der Nachteil dieser Methode ist, dass eine höhere Schrittfrequenz, wie beispielsweise beim Laufen, nicht in die Schrittdetektion einfließen kann. Für die Anwendungen in dieser Masterarbeit reicht dieser definierte Grenzwert von $0,3$s jedoch aus, da auf wechselnde Bewegungsmuster nicht näher eingegangen wird sondern eine konstante Bewegung vorausgesetzt wird. Die Abbildung 4.2 beinhaltet die Schrittdetektion durch diese Methode.

4.1.2 Schrittlänge

Die Schrittlänge des Fußgängers ist dafür verantwortlich, dass die zurückgelegte Distanz während eines Schrittes bekannt ist. Nach Bauer [2] gibt es viele verschiedenen Methoden zur Schrittlängenschätzung. Diese beruhen auf fixen Konstanten, Kalibrierungsparametern, Schrittfrequenzen und in manchen Ansätzen auf der Länge der Beschleunigungsvektoren.

Chen und Guinness [4] zeigen einen Ansatz, bei dem die Schrittlänge SL aufgrund der Größe des Fußgängers h, der Schrittfrequenz SF, der Modellparameter a und b sowie dem Kalibrierungsfaktor c berechnet wird:

$$SL = \left(0.7 + a(h - 1.75) + \frac{b(SF - 1.79)h}{1.75}\right) c \tag{4.2}$$

Die Formel 4.2 implementiert die Schlussfolgerung, dass die Schrittlänge sich proportional zur Schrittfrequenz erhöht.

In dieser Masterarbeit werden hinsichtlich der Schrittlänge keine vertiefenden Untersuchungen angestrebt, wobei eine auf die Testperson abgestimmte Standardschrittlänge von 0,68cm festgelegt wurde. Für Stiegen ändert sich die Schrittlänge des Fußgängers. Da die meisten Stufen innerhalb von Gebäuden eine standardisierte Länge von 30cm aufweisen, kann die Schrittlänge beim Treppensteigen auf diesen Wert verringert werden.

4.1.3 Richtungsschätzung

Die Lage eines Körpers im Raum wird, wie bereits in Kapitel 2.3 erwähnt, durch die Attitude-Parameter bestimmt. Das Heading für zweidimensionale DR-Algorithmen entspricht dem Drehwinkel um die horizontierte Z-Achse des Objektes, welche durch den Winkel yaw beschrieben wird. Diese Richtungsinformation gibt somit den Winkel zwischen der Nordrichtung und der aktuellen Bewegungsrichtung an.

Zur Richtungsschätzung werden Akzelerometerdaten und Gyroskop- oder Magnetometerdaten benötigt. Der erste Schritt beinhaltet eine Transformation um roll und pitch vom Bodysystem in das lokale Horizontsystem, sodass die X_3 Achsen beider Systeme parallel sind. Dies wird in weiterer Folge als Horizontierung bezeichnet. Graphisch wird diese Horizontierung in Abbildung 2.9 durch die Rotationen um roll und pitch visualisiert. Diese zwei Drehwinkel werden nach Groves [6] über Beschleunigungsmessungen berechnet:

$$\begin{aligned} \text{roll:} \quad & r = \arctan\left(\frac{a_y^b}{a_z^b}\right) \\ \text{pitch:} \quad & p = \arctan\left(\frac{a_x^b}{\sqrt{\left(a_y^b\right)^2 + \left(a_z^b\right)^2}}\right) \end{aligned} \tag{4.3}$$

Der Winkel yaw, welcher das aktuelle Heading repräsentiert, kann entweder durch Magnetometer- oder durch Gyroskopdaten berechnet werden. Für die Richtungsschätzung durch Drehratenmessungen wird die erwähnte Horizontierung mit Hilfe der Rotationsmatrix aus Formel 4.4 durchgeführt. Dabei fließen die roll und pitch Parameter aus den Gleichungen 4.3 in die Rotationsmatrix

$$\mathbf{R}_l^b = \begin{bmatrix} \cos(p)\cos(y) & \cos(p)\sin(y) & -\sin(p) \\ \sin(r)\sin(p)\cos(y) - \cos(r)\sin(y) & \sin(r)\sin(p)\sin(y) + \cos(r)\cos(y) & \sin(r)\cos(p) \\ \cos(r)\sin(p)\cos(y) + \sin(r)\sin(y) & \cos(r)\sin(p)\sin(y) - \sin(r)\cos(y) & \cos(r)\cos(p) \end{bmatrix} \tag{4.4}$$

ein. Diese Matrix beinhaltet allerdings auch den yaw Parameter, welcher jedoch für die Horizontierung keine Rolle spielt. Dementsprechend kann yaw in dieser Matrix mit null angenommen werden. Durch $\mathbf{R}_b^l = \mathbf{R}_l^{b\mathsf{T}}$ können die Drehraten aus dem Bodysystem in das lokale Horizontsystem transformiert werden:

$$\begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix}^l = \mathbf{R}_b^l \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix}^b \tag{4.5}$$

Die gesuchten Drehwinkel aus den gemessenen Winkeländerungen erhält man durch die Integration der horizontierten Drehraten:

$$\begin{aligned} r_i &= r_{i-1} + \omega_1^l \Delta t \\ p_i &= p_{i-1} + \omega_2^l \Delta t \\ y_i &= y_{i-1} + \omega_3^l \Delta t \end{aligned} \tag{4.6}$$

Die Integration aus Formel 4.6 benötigt für die rekursive Berechnung als Integrationskonstante eine initiale Startausrichtung. Diese kann aus Magnetometerdaten oder aus einer

Referenzrichtung gewonnen werden. Im Kapitel 6 wird dabei für einen Graphen-basierten PDR-Algorithmus auf die Richtung einer Startkante zurückgegriffen.

Bei einem magnetischen Heading

$$\text{yaw:} \quad y_{mag} = \arctan\left(\frac{-m_y\cos(r) + m_z\sin(r)}{m_x\cos(p) + m_y\sin(p)\sin(r) + m_z\sin(p)\cos(r)}\right) \tag{4.7}$$

muss zusätzlich die Variation (Unterschied zwischen magnetisch Nord- und geographisch Nord) beachtet werden. Der Wert der Variation verändert sich aufgrund der Magnetischen Polbewegungen und ist je nach Region verschieden. In Graz beträgt er derzeit nach www.ngdc.noaa.gov [24] $3,69°$.

Das Heading aus Magnetometerdaten hat den Vorteil, dass es für jeden Messzeitpunkt, unabhängig von vorigen Messungen, absolut bestimmt wird. Dem entgegengesetzt unterliegt ein Heading aus Drehratenmessungen (nur Winkeländerungen) einem Drift aufgrund der rekursiven Berechnung, siehe Formel 4.6.

In Abbildung 4.3 ist das Ergebnis der Richtungsschätzung durch Gyroskop- und durch Magnetometerdaten visualisiert. Nach Willemsen [20] können die starken Schwankungen in den Magnetometermessungen auf die geringe Sensibilität und die hohe Temperaturempfindlichkeit des Sensors zurückzuführen sein. Zusätzlich verfälschen lokale Magnetfelder das Messergebnis des Magnetometers. Diese Störfaktoren werden durch Baustrukturen, Maschinen und vielen weiteren Komponenten hervorgerufen. Der Verlauf des Headings bei den präziser gemessenen Drehraten weist einen glatteren und robusteren Verlauf auf. Die Drehungen des Users sind dabei eindeutig erkennbar. Initialisiert wurde das Heading durch das erste yaw aus den Magnetometermessungen.

Abschließend ist anzumerken, dass sich die Smartphone-Messungen prinzipiell auf das Sensorsystem beziehen. Dementsprechend müsste eine Verdrehung zwischen dem Bodysystem und dem Sensorsystem zusätzlich in der Richtungsschätzung beachtet werden. Richtet man jedoch die Achsen des Sensorsystems so aus, dass sie mit denen des Bodysystems kongruent sind, entfällt diese Transformation. In dieser Arbeit wird demzufolge das Smartphone bei den Testmessungen in der sogenannten *hand-held* Tragweise gehalten (horizontal in der Hand und mittig vor dem Körper). Dadurch entspricht das Sensorsystem dem Bodysystem und die inertialen Messungen werden im Bodysystem gemessen.

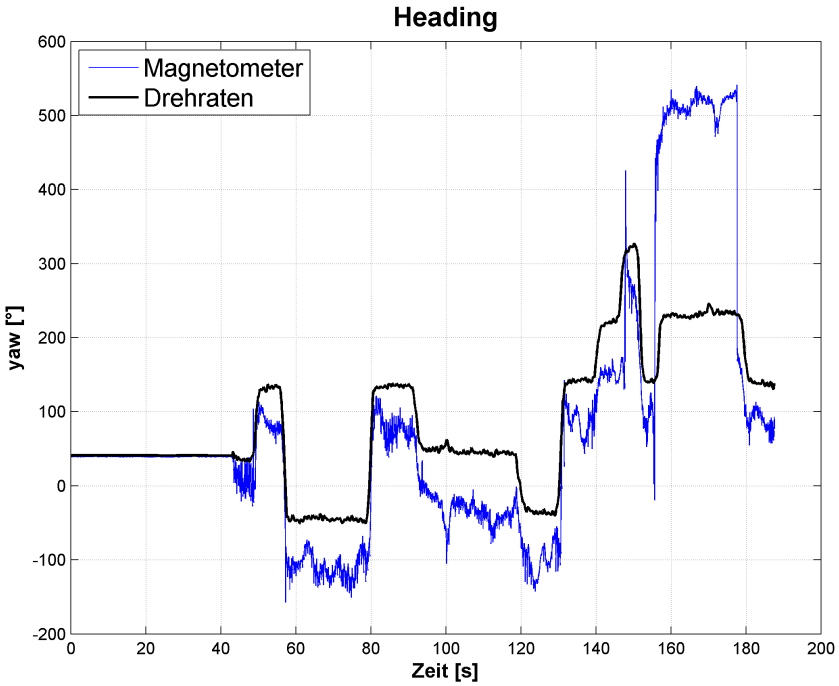


Abbildung 4.3: Richtungsschätzung durch Magnetometer und Gyroskop

4.1.4 PDR-Trajektorie

Wenn alle Komponenten des Schritt-basierten PDR bekannt sind, können durch

$$\begin{aligned} y_i &= y_{i-1} + d \sin \alpha \\ x_i &= x_{i-1} + d \cos \alpha \end{aligned} \tag{4.8}$$

die Positionen fortwährend berechnet werden. Zu Beginn muss bei dieser Positionierungsmethode eine initiale Startposition bekannt sein, damit ein Bezug zu einem speziellen Koordinatensystem, wie z.B. GK-System, hergestellt werden kann.

Die Abbildung 4.4 visualisiert die Positionslösungen eines Schritt-basierten PDR, wobei sich die beiden Trajektorien durch die Richtungsschätzung unterscheiden. Wie auch bereits in Abbildung 4.3 erkennbar, weist die Lösung durch Drehratenmessungen einen glatteren Verlauf auf. Die stark verrauschten Magnetometerdaten repräsentieren kein robustes Ergebnis der Soll-Trajektorie. Da sich die Positionslösungen auf das GK-Koordinatensystem beziehen, müssten die Nordrichtungen zusätzlich um die Meridiankonvergenz korrigiert werden. Dies kann jedoch aufgrund der angestrebten Genauigkeit in den Positionslösungen vernachlässigt werden.

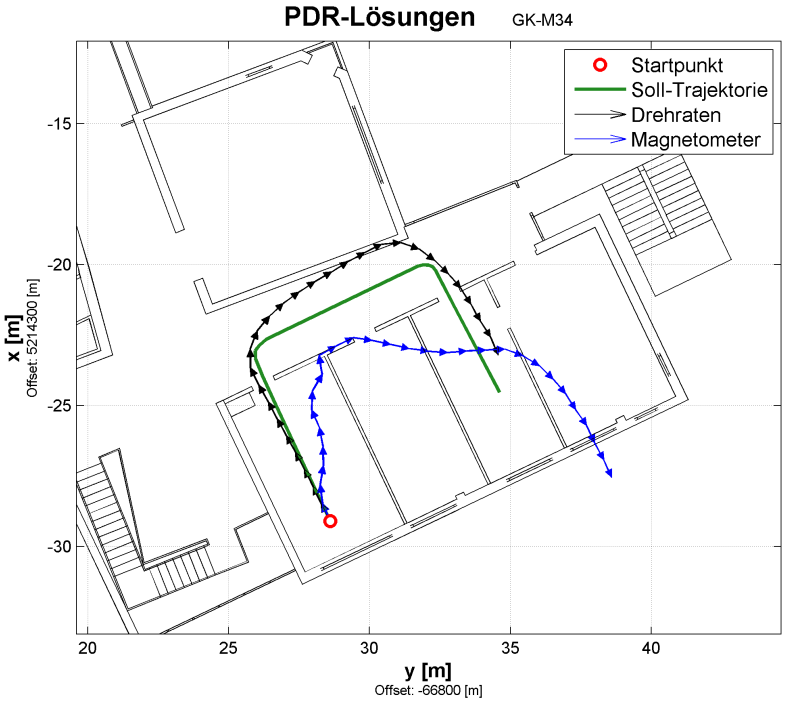


Abbildung 4.4: PDR-Positionslösungen mit Richtungsschätzungen aus Magnetometer und Gyroskopdaten

4.2 PDR Modifikationen

Reine PDR-Lösungen, wie sie in Kapitel 4.1.4 berechnet werden, neigen bei längeren Trajektorien dazu, dass sie einen Drift beinhalten. Demzufolge werden ungenaue und unrealistische Trajektorien berechnet. Dies gibt Anlass die PDR-Lösungen durch zusätzliche Verfahren zu stützen.

4.2.1 Kalman Filter

Bauer [2] zeigt in ihrer Arbeit einen Algorithmus, mit den sie die PDR-Lösung durch die Kombination von Magentometer- und Drehratenmessungen mit Hilfe eines diskreten Kalman Filters verbessert. Damit der Kalman Filter diese Sensorfusion realisieren kann, liegen ihm eine Beobachtungsgleichung sowie ein dynamisches Modell zu Grunde. Die Beobachtungsgleichung dient dazu, eine Beziehung zwischen den gemessenen Größen und dem Zustands-/Parametervektor herzustellen. Diese Gleichung wird wie folgt gebildet,

wobei k die Zeitabhängigkeit und $\mathbf{H}$ den Zusammenhang zwischen Messungen und Parametern repräsentiert. Zusätzlich unterliegen die Beobachtungen einem normalverteilten Rauschen $\mathbf{v}_k$, welches durch eine Kovarianzmatrix $\mathbf{R}_k$ beschrieben werden kann.

$$\begin{aligned} \mathbf{z}_k &= \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \\ \mathbf{v}_k &\sim N(0, \mathbf{R}_k) \end{aligned} \tag{4.9}$$

Die Beobachtungen $\mathbf{z}$ und der Parametervektor $\mathbf{x}$ bestehen aus den Attitude Parametern und ihren zeitlichen Änderungen (Drehraten). In $\mathbf{z}$ wird roll und pitch, wie in Kapitel 4.1.3, aus Beschleunigungsmessungen berechnet. Das yaw wird durch Formel 4.7 aus Magnetometerdaten gebildet und die Drehraten fließen bereits nach Formel 4.5 transformiert ein.

$$\mathbf{z} = \mathbf{x} = \begin{bmatrix} r & p & y & \dot{r} & \dot{p} & \dot{y} \end{bmatrix}^\mathsf{T} \tag{4.10}$$

Zusätzlich zu den Beobachtungen wird im Kalman Filter auch die Bewegung durch das dynamische Modell mit modelliert. Dieses unterliegt ebenfalls einem normalverteilten Rauschen $\mathbf{w}_k$, welches durch die Matrix $\mathbf{Q}_k$ beschrieben wird:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{\Phi}_k \mathbf{x}_k + \mathbf{w}_k \\ \mathbf{w}_k &\sim N(0, \mathbf{Q}_k) \end{aligned} \tag{4.11}$$

In der Formel 4.11 repräsentiert $\mathbf{\Phi}$ die sogenannte Übertragungsmatrix. Dieser Matrix liegt eine gleichförmige Bewegung zugrunde und sie beschreibt wie sich der Zustandsvektor von einem Zeitpunkt zum nächsten Zeitpunkt verändert.

Die Implementierung des Kalman Filters basiert auf drei Komponenten. Diese Schritte schaffen es, den Parametervektor $\hat{\mathbf{x}}$ nach dem BLUE Prinzip zu schätzen. Der erste Teil beinhaltet die Berechnung des Kalman Gewichtes $\mathbf{K}$:

$$\mathbf{K}_k = \tilde{\mathbf{P}}_k \mathbf{H}_k^\mathsf{T} (\mathbf{H}_k \tilde{\mathbf{P}}_k \mathbf{H}_k^\mathsf{T} + \mathbf{R}_k)^{-1} \tag{4.12}$$

Diese Größe kontrolliert den Filter. Sie bestimmt, ob den Messungen oder der Systemdynamik mehr Gewicht im Zuge der Berechnung des Zustandsvektors zugeordnet wird.

Der zweite Teil wird als Korrekturschritt bezeichnet und berechnet den gesuchten Parametervektor.

$$\hat{\mathbf{x}}_k = \tilde{\mathbf{x}}_k + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k\tilde{\mathbf{x}}_k) \tag{4.13}$$

Für diesen Zustandsvektor kann eine bestimmte Unsicherheit in Form der folgenden Kovarianzmatrix angegeben werden.

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\tilde{\mathbf{P}}_k \tag{4.14}$$

Der letzte Schritt beinhaltet den Prädiktionsschritt. In diesem Teil kann eine Voraussage auf den nächsten Zustandsvektor getätigt werden.

$$\tilde{\mathbf{x}}_{k+1} = \mathbf{\Phi}_k\hat{\mathbf{x}}_k \tag{4.15}$$

Für diesen prädizierten Zustandsvektor kann ebenfalls eine Unsicherheit berechnet werden.

$$\tilde{\mathbf{P}}_{k+1} = \mathbf{\Phi}_k\mathbf{P}_k\mathbf{\Phi}_k^{\mathsf{T}} + \mathbf{Q}_k \tag{4.16}$$

Durch die Fusion der Drehraten mit den Magnetometermessungen kann die in Abbildung 4.5 dargestellte PDR-Lösung erzielt werden. Hierbei ist deutlich zu erkennen, dass die Magnetometerdaten durch Kombination mit dem dynamischen Modell und den Drehraten einen deutlich glatteren Verlauf beschreiben. Probleme liefert dieses Modell auch weiterhin, wenn künstliche Magnetfelder die Magnetometermessungen stören. Diesbezügliche müsste ein adaptiver Kalman Filter implementiert werden, der die Standardabweichung nach bestimmten Bedingungen während des Berechnungsverlauf verändert.

4.2.2 Partikelfilter

Ein diskreter Kalman Filter, wie er in Kapitel 4.2.1 beschrieben wird, ist auf die Linearität der Beobachtungs- und Bewegungsgleichung beschränkt. Eine Normalverteilung wird ebenfalls vorausgesetzt. Aufgrund der stochastischen Zusammenhänge in einem Kalman Filter lassen sich somit Karteninformationen, zur Stützung von PDR Lösungen, nur schwer implementieren. Dem entgegengesetzt kann ein Partikelfilter nichtlineare Zusammenhänge

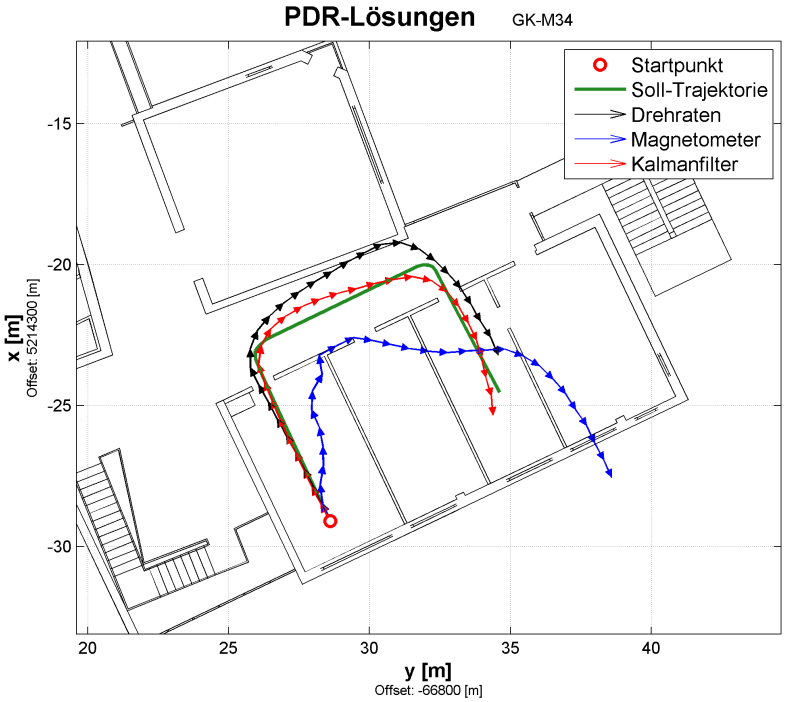


Abbildung 4.5: Fusion von Drehraten mit Magnetometermessungen durch einen diskreten Kalman Filter

sowie beliebige Verteilungen behandeln. Die Unsicherheiten der Zustandsschätzungen werden bei einem Partikelfilter, nicht wie bei einem Kalman Filter in Form von Kovarianzen angegeben, sondern durch die Verteilung von gewichteten Partikeln. Die Anzahl der Partikel in einem bestimmten Bereich und deren Gewichtung sind somit ein Indikator für den aktuellen Zustand. Durch diese Voraussetzungen eignet sich ein Partikelfilter für die Integration von Karteninformationen. Eine mögliche geschätzte Trajektorie eines Partikelfilters aus der Steyrergasse 30 ist in Kapitel 7.5 festgehalten.

Hafner [8] zeigt solch einen Filter in Bezug auf eine Smartphone-basierte Positionsschätzung. Zu Beginn werden die Partikel zufällig über den ganzen möglichen Zustandsraum verteilt, siehe Abbildung 4.6 (a). Anschließend wird jedem Partikel aufgrund der Messungen z_t durch das Beobachtungsmodell mit der zugehörigen Wahrscheinlichkeitsfunktion $p(z_t|x_t)$ ein Gewicht zugeordnet. Dieser Schritt beschreibt somit, ob ein bestimmter Partikel, in Bezug auf die Messungen, ein mögliches Ergebnis der Positionsschätzung wäre $bel(x)$, Bild (b). Dadurch können somit Karteninformationen für die Positionsschätzung eingebunden werden. Entsprechend den berechneten Partikelgewichtungen werden neue gleich-gewichtete Partikel generiert. Partikel die zuvor eine hohe Gewichtung besaßen, werden in diesem

Schritt vervielfacht, jene mit geringer Gewichtung gelöscht, Bild (c). Das Vervielfachen ist deshalb notwendig, damit der Filter über den gesamten Messzeitraum nicht divergent wird. Durch das angenommene Bewegungsmodell $p(x_t|x_{t-1})$ werden im darauffolgenden Schritt die Partikel verschoben, Bild (d). Anschließend beginnt der Filter wieder von Neuem, um mit den aktuellen Messungen die einzelnen Partikel zu gewichten, Bild (e).

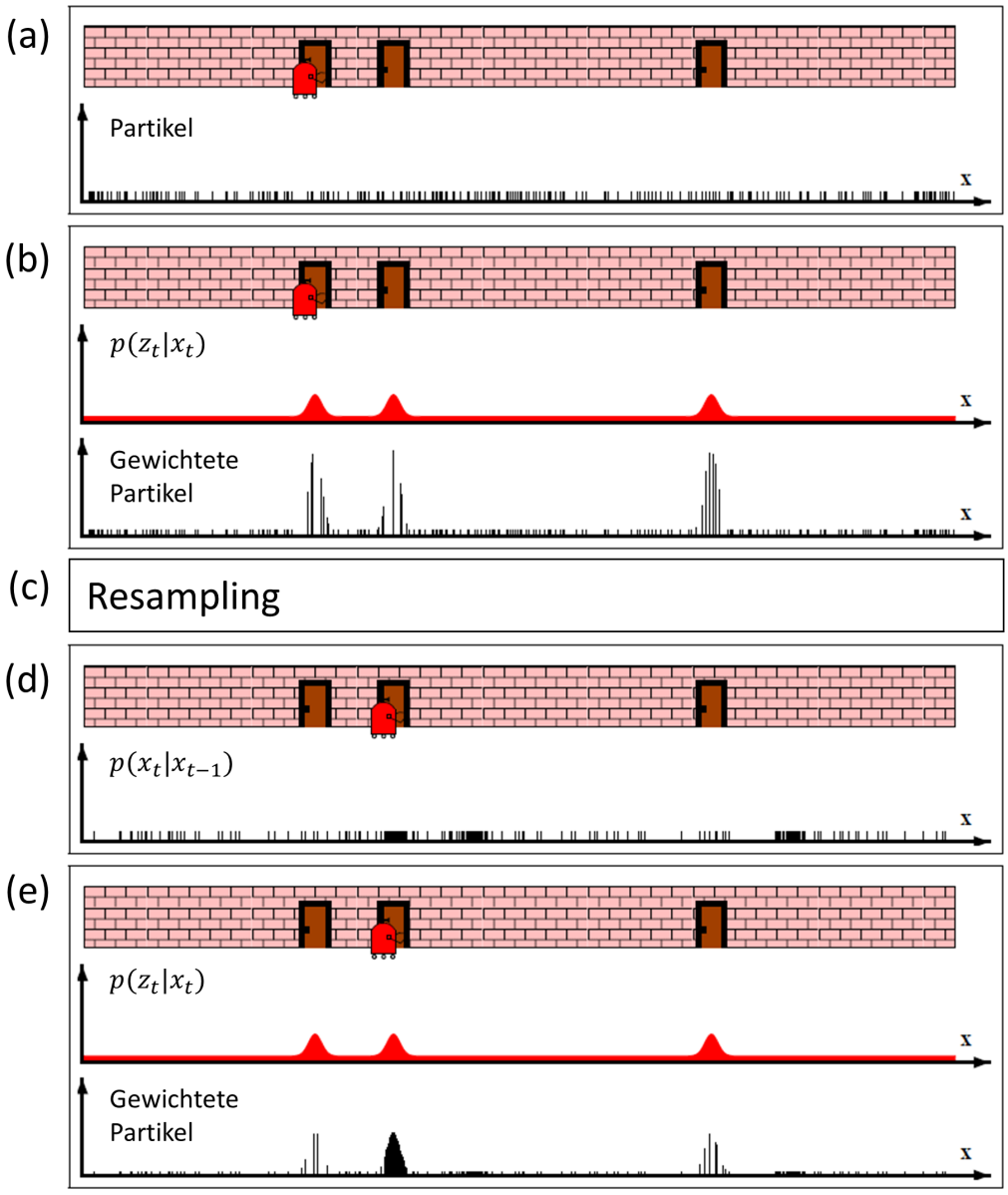


Abbildung 4.6: Arbeitsablauf eines Partikelfilters abgeleitet aus Thrun et al. [16]

4.2.3 Graphen-basierender Ansatz

Eine weitere Möglichkeit die PDR-Trajektorie speziell in Innenräumen zu verbessern, ist die Stützung durch einen Graphen. Ein Graph, bestehend aus Knoten und Kanten, kann aufbauend auf Gebäudeplänen erstellt werden. Dadurch repräsentiert der Graph die Gebäudestruktur wie z.B. Korridore, Stiegen oder Eingänge.

Die Positionslösungen des Fußgängers werden durch einen Graphen-basierten PDR-Algorithmus auf dieses Knoten- und Kanten-System beschränkt. Demzufolge wird die Trajektorie des Fußgängers stark generalisiert dargestellt, jedoch können unrealistische Ergebnisse ausgeschlossen werden. Auf dem gleichen Grundprinzip werden die Positionslösungen von Fahrzeugnavigationsgeräten berechnet. Autos bewegen sich prinzipiell nur auf den in der Karte eingezeichneten Straßen. Die GNSS-Positionslösungen sind aufgrund ihrer Messgenauigkeit jedoch nicht zwingend darauf. Damit dennoch eine Position auf dem Straßennetz (repräsentiert durch einen Graphen) angezeigt wird, müssen durch ein geeignetes Map-Matching-Verfahren die gemessenen Positionen auf die Knoten und Kanten des Netzwerks projiziert werden.

Einen topologischen Ansatz auf einem Graphen, rein durch Smartphone-MEMS, zeigen Willemsen et al. [21]. Dabei wird zu Beginn durch Bekanntgabe einer initialen Position die wahrscheinlichste Startkante ausgewählt. Dies wird über eine Raum- oder Türkoordinate, welche durch QR-Codes oder über eine Eingabe einer Raumnummer registriert wird, realisiert. Die fortwährenden Positionsschätzungen werden durch Zustandsabfragen auf dem Graphen durchgeführt. Je nach Heading und Schrittlänge wird ein Schritt auf der aktuellen Kante oder auf einer neuen Kante getätigt. Die Drehraten werden dabei nicht über den gesamten Messzeitraum integriert, sondern nur über den Zeitraum einer Kante.

Der Vorteil dieser Methode liegt in der Driftkompensierung, dem geringen Rechenaufwand und dass die Positionslösungen in der Berechnung keine weiteren Stützinformationen benötigen. Der Nachteil ist die starke Generalisierung der Trajektorie. Durch die Bewegungsfreiheit des Users können dadurch große Diskrepanzen zwischen der berechneten Kantenabfolge und der tatsächlichen Trajektorie entstehen. Speziell auf großen freien Flächen eignet sich dieser Ansatz im Gegensatz zu z.B. schmalen Gängen weniger.

Kapitel 5

Graphentheorie

Der Schweizer Mathematiker Leonhard Euler war der erste der navigationsspezifische Probleme mit Hilfe eines Graphen gelöst hat. 1736 beschäftige er sich mit dem Königsberger Brückenproblem. Dabei sollte eine Route gefunden werden, bei der alle sieben Brücken im Zuge einer Tour einmal überquert werden. Euler modellierte als Lösungsansatz die Brücken durch Kanten und das Ufer bzw. die Inseln als Knoten. Dadurch konnte bewiesen werden, dass die angestrebte Tour aufgrund der Brückenkonstellationen nicht möglich ist.

Heutzutage beschreiben Graphen in Karten je nach Anwendung die vektortypischen Strukturen wie Straßen, Flugrouten oder Hafeneinfahrten. Neben Visualisierungszwecken eignen sich Graphen in der Navigation für Positionsberechnungen und für Routing/Guidance Aufgaben. Abschließend ist anzumerken, dass sich die Ausführungen in diesem Kapitel auf Hofmann-Wellenhof et al. [9] stützen.

5.1 Graphenmodellierung

Knoten und Kanten bilden durch Kombinationen und Verknüpfungen miteinander einen Graph. Diese Bausteine eines Graphen werden in den nachfolgenden Ausführungen beschrieben. Zur grafischen Veranschaulichung sind die einzelnen Definitionen in Abbildung 5.1 illustriert.

Kanten und Knoten Ein Graph $G = G(V, E)$ ist definiert aus einem Satz von $V = \{v_1, v_2, ..., v_n\}$ Knoten (Kreuzungspunkte) und $E = \{e_1, e_2, ..., e_m\}$ Kanten (Verbindungen zwischen Knoten). Spricht man von einem gerichteten Graphen sind

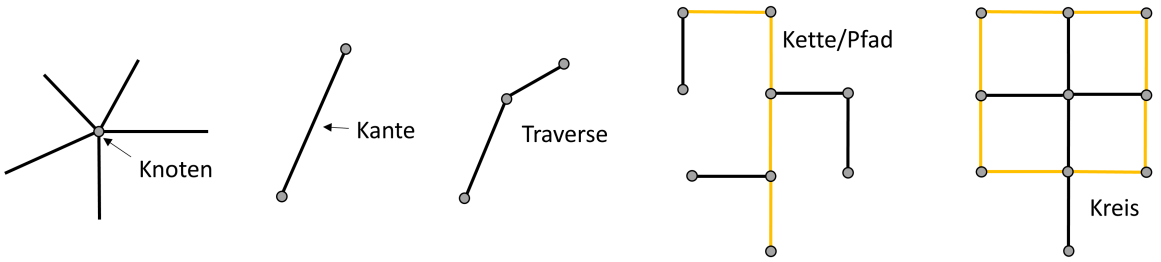


Abbildung 5.1: Definitionen der Graphenelemente

die Kanten auf eine Richtung beschränkt. Somit ist z.B. die Verbindung von Knoten v_1 zu v_2 durch die gerichtete Kante, oder auch Bogen genannt, modelliert, jedoch nicht die umgekehrte Verknüpfung von v_2 zu v_1.

Adjazent und Inzident Zwei Knoten sind adjazent, wenn sie durch eine Kante verbunden sind, wohingegen zwei Kanten inzident sind, wenn sie einen gemeinsamen Knoten besitzen. Beginnt und endet eine Kante in nur einem Knoten, repräsentiert die Kante eine Schleife.

Traverse Eine komplexe Verbindung zwischen mehreren Knoten kann durch eine Traverse beschreiben werden. Dieses topologische Element beschreibt eine Verbindung zwischen einem Anfangsknoten v_1 über den Knoten v_2 bis zu einem Endknoten v_3. Dadurch kann beispielsweise eine Abbiegung bei Kreuzungen modelliert werden.

Kette und Pfad Eine Kette repräsentiert eine Kantenfolge, welche eine Verbindung zwischen zwei nicht adjazenten Knoten darstellt. Dabei dürfen die Kanten in der Abfolge nur einmal gewählt werden. Ein Pfad beschreibt den gleichen Sachverhalt, jedoch nur mit gerichteten Kanten.

Kreis Ein Kreis ist definiert als eine geschlossene Kette. Falls neben den Kanten auch alle Knoten nur einmal gewählt werden, dann spricht man von einem simplen Kreis. Ein gerichteter Kreis entsteht, wenn Kanten durch Bögen ersetzt werden.

Ein Graph besitzt je nach Konstruktion verschiedene Eigenschaften:

Simpler (einfacher) Graph Besteht ein Graph aus keinen parallelen Kanten (idente Anfangs- und Endknoten) und aus keinen Schleifen dann wird der Graph als einfach

bezeichnet. Repräsentiert man ein Straßennetzwerk durch eine Knoten- und Kantenstruktur, dann kann ein simpler Graph aufgrund von 180° Richtungsänderungen und hinsichtlich von parallelen Straßen nicht realisiert werden.

Regulärer Graph Der Grad eines Knotens wird angegeben durch die Anzahl an inzidenten Kanten in einem Knoten. Ist der Grad in jedem Knoten eines Netzwerkes ident, wird der Graph als regulär bezeichnet. Diese Eigenschaft würde den Berechnungsaufwand von Algorithmen deutlich vereinfachen, jedoch entspricht solch ein Graph in den meisten Anwendungen nicht der Realität.

Vollständiger Graph In einem vollständigen Graph ist jeder Knoten mit jedem Knoten durch eine Kante verbunden. Bei n Knoten würden dadurch mindestens

$$m = \begin{bmatrix} n \\ 2 \end{bmatrix} = \frac{n(n-1)}{2} \tag{5.1}$$

Kanten benötigt.

Planarer Graph Ein Planarer Graph ist ein Netzwerk, welches nur in einer Ebene liegt. Die Kanten schneiden sich nur in den Knoten. In maritimen Karten werden beispielsweise nur planare Graphen verwendet, weil sich Schiffe ausschließlich in einer Ebene befinden können. In der Luftfahrt oder bei Straßennetzwerken mit Unterführungen können sich Kanten aufgrund der unterschiedlichen Höhen in der 2D Darstellung kreuzen.

Dünner Graphen Durch ihre geringe Anzahl an Kanten

$$m \ll \frac{n(n-1)}{2} \tag{5.2}$$

weisen dünne Graphen kein dichtes Netzwerk auf. Durch diese Eigenschaft eignen sich diese Graphen besonders gut für schnelle und effiziente Berechnungen. Ein Beispiel für dünne Graphen ist ein planarer Graph.

Bewerteter (gewichteter) Graph Graphen haben neben der topologischen- oder auch in manchen Fällen geometrischen Information zusätzliche semantische Komponenten. Dementsprechend erweitert sich der Graph zu $G = G(V, E, c)$. Durch eine Kostenfunktion c erhält jede Kante einen spezifischen Wert. Dies kann beispielsweise die Länge der Kante oder wie in Straßennetzwerken eine Geschwindigkeitsbeschränkung sein. Anstelle von Kostenparametern können auch Wahrscheinlichkeiten für die Gewichtung von Graphenelementen benützt werden.

5.2 Speichermodalitäten

Für computergestützte Berechnungen bedarf es einer effizienten Speicherung des Graphen. Je nach Größe des Graphen und Speicherkapazitäten existieren verschieden Methoden:

Adjazenzmatrix
Bei dieser Speichermethode wird der Graph durch eine $\mathbf{A} = (n \times n)$ Matrix repräsentiert. Demnach ist eine Zeile i und eine Spalte j für einen Knoten n vorgesehen. Existiert eine Kante e_{ij} besitzt das Element A_{ij} den Wert eins ansonsten null. Wenn die Kanten ungerichtet sind, dann ist die Matrix $\mathbf{A}$ symmetrisch. Ersetzt man den Zellwert eins durch den Kostenparameter einer Kante spricht man von einer Kostenmatrix:

$$\mathbf{A}_{ij} = \begin{cases} c_{ij} & \text{falls } e_{ij} \text{ existert} \\ \infty & \text{ansonsten} \end{cases} \tag{5.3}$$

Durch diese Speicherung wird, unabhängig davon wie viele Knoten in einem Netzwerk vorhanden sind, immer eine $(n \times n)$ Matrix gespeichert. Dieser Nachteil tritt speziell bei dünnen Graphen auf.

Inzidenzmatrix
Eine Inzidenzmatrix wird mehrheitlich bei gerichteten Graphen eingesetzt und hat eine $(n \times m)$ Dimension. Infolgedessen werden die Zeilen durch Knoten und die Spalten durch Kanten repräsentiert. Gebildet wird die Inzidenzmatrix $\mathbf{B}$ durch folgende Vorschrift:

$$\mathbf{B}_{ij} = \begin{cases} +1 & \text{wenn } v_i \text{ der Startknoten von } e_j \text{ ist} \\ -1 & \text{wenn } v_i \text{ der Endknoten von } e_j \text{ ist} \\ 0 & \text{wenn } e_j \text{ nicht inzident mit } v_i \text{ ist} \end{cases} \tag{5.4}$$

Adjazenzliste
Die Adjazenzliste besteht aus m Zeilen und drei Spalten. Eine Zeile ist somit für jede Kante vorgesehen. In der ersten Spalte stehen die jeweiligen Anfangsknoten und in der zweiten die Endkonten. Die dritte Spalte enthält, falls vorhanden, den Kostenparameter der Kante. Effizient ist diese Methode falls $m < n^2/3$ gilt. Dementsprechend müssen speziell

bei dünnen oder planaren Graphen, nicht n^2 Element gespeichert werden, sondern lediglich $3m$.

Indizierte Adjazenzliste

Die Speichermodalität für eine indizierte Adjazenzliste besteht aus zwei Listen. Die Knotenliste besitzt eine Information über die inzidenten Kanten pro Knoten. Dabei wird jedoch nicht direkt die Anzahl der inzidenten Kanten angegeben, sondern die kumulative Summe mit den inzidenten Kanten der vorigen Knoten. Durch Differenzbildungen können rückwirkend aus der kumulativen Summe die Anzahl der abgehenden Kanten pro Knoten ermittelt werden. In der Bogenliste werden in der ersten Spalte die Endkonten und in der Zweiten die Kostenparameter der jeweiligen Kanten angeführt. Inzidente Kanten eines Knotens werden in der Bogenliste sequentiell gespeichert. Durch die Information aus der Knotenliste können somit die Kanteneinträge in der Bogenliste zu den zugehörigen Anfangsknoten aus der Knotenlisten in Verbindung gebracht werden. Die Funktionsweise und der Aufbau dieser beiden Listen wird in der Abbildung 5.2 visualisiert.

Diese Methode kann im Gegensatz zur Adjazenzliste wiederum die Speicherkapazität verringern. Anstatt $3m$ Matrixeinträge müssen bei einer indizierten Adjazenzliste nur $n + 2m$ Elemente gespeichert werden.

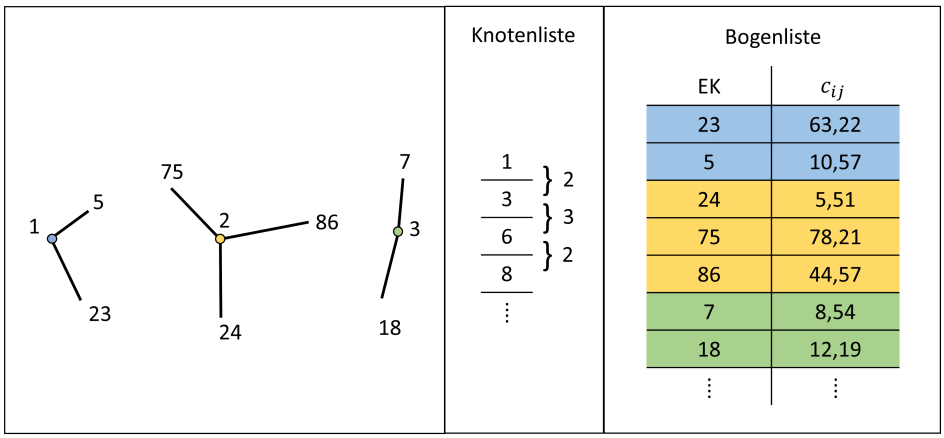


Abbildung 5.2: Indizierte Adjazenzliste

5.3 Routing

Eine grundlegende Aufgabe in der Navigation ist die Berechnung der bestmöglichen Route (shortest path) zwischen einem Anfangspunkt und einem Zielpunkt. Voraussetzung dafür ist ein durch Kostenparameter bewerteter Graph. Aufbauend auf einer geeigneten Speichermodalität wird die Route im Sinne einer Kantenabfolge durch Routingalgorithmen ermittelt. Nachfolgend werden die bekanntesten Algorithmen vorgestellt, wobei nur der Dijkstra- und ein k-shortest-path-Algorithmus in Kapitel 6 Verwendung finden. Die Kostenparameter werden dabei durch die Längen der Kanten definiert. Dementsprechend wird in den nachfolgenden Ausführungen immer von einer kürzesten Route gesprochen.

Single Source Shortest Paths (SSSP)
SSSP-Algorithmen berechnen ausgehend von einem bekannten Startpunkt bzw. Startknoten v_s zu allen anderen Knoten in einem Graphen $G(V, A, c)$ mit positiven Kosten c_{ij} die kürzeste Route. Der bekannteste Algorithmus wurde von Edsger W. Dijkstra entwickelt.

Die Routenberechnung von diesem Algorithmus stützt sich auf eine *breadth-first search* Methode. Dabei wird ein Suchbaum generiert, welcher, wie in Abbildung 5.3 ersichtlich, iterativ in alle Richtungen wächst. In einem idealisierten Fall, bei denen der Graph eine regelmäßige Struktur aufweist und alle Kanten gleiche Kosten besitzen, würde der Suchbaum sich symmetrisch in alle Richtungen erweitern.

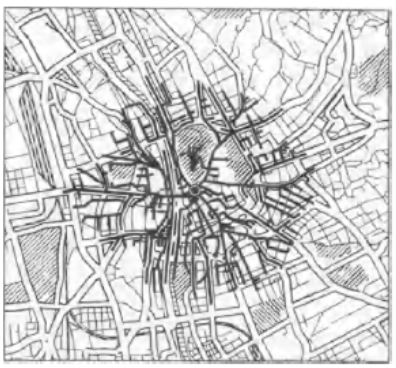


Abbildung 5.3: Breadth-first Suchbaum aus Hofmann-Wellenhof et al. [9]

Die prinzipielle Vorgehensweise des Dijkstra-Algorithmus beruht während der Routenberechnung darauf, dass Knoten entweder ein permanentes oder ein temporäres Label

bekommen. Ein temporäres Label eines Knotens v_j beinhaltet die zurückgelegte Strecke vom Startknoten v_s bis zu dem Knoten v_j, diese muss jedoch noch nicht die kürzeste Distanz zwischen den beiden Knoten sein. Erst wenn v_j ein permanentes Label im Zuge des Dijkstra-Algorithmus erhält, beschreibt das Label die kürzeste Distanz.

Im ersten Schritt wählt der Algorithmus jenen Knoten aus, welcher mit dem Startknoten durch die Kante mit der kürzesten Distanz verbunden ist. Dieser Knoten erhält ein permanentes Label, alle anderen erhalten ein temporäres Label mit den zugehörigen Distanzen von dem Startknoten. Anschließend wird zu den adjazenten Knoten des zuvor ausgewählten Knotens wiederum die Distanzen vom Startknoten berechnet und als Label zu den Knoten hinzugefügt. In diesem Schritt können somit die temporären Label, im Falle einer kürzeren Route, überschrieben werden. Anschließend wird im gesamten Netzwerk jener Knoten ausgewählt, der zu dem Startknoten die geringste Distanz aufweist. Zusätzlich erhält dieser Knoten ein permanentes Label. Dadurch schafft es der Dijkstra-Algorithmus, dass, wenn ein Knoten erreicht wird, die Distanz vom Startknoten zu diesem immer die geringste ist. Die Zuteilung dieser Labels wird solange fortgesetzt, bis alle Knoten ein permanentes Label erlangt haben. Neben den Labels werden zusätzlich noch die Knotenvorgänger der jeweiligen Routen gespeichert. Dadurch kann die Kantenabfolge für die kürzeste Route nachvollzogen werden.

Abschließend ist anzumerken, dass der Dijkstra-Algorithmus grundsätzlich nur mit gerichteten Kanten arbeitet. Damit auch ungerichtete Kanten in diesem Algorithmus eingebunden werden können, müssen diese durch parallele Kanten (jeweils in entgegengesetzte Richtung gerichtete) modelliert werden.

k-shortest-path

Ein k-shortest-path-Algorithmus berechnet ausgehend von einem Startpunkt zu allen anderen Knoten nicht nur den kürzesten, sondern auch den zweit-, dritt-, ... kürzesten Weg. Diese Routenberechnungen können folglich alle k-kürzesten Wege von einem Startpunkt zu einem Zielpunkt berechnen.

Yen [26] entwickelte solch einen Algorithmus. Dabei wird zu Beginn, wie in Abbildung 5.4 Bild I ersichtlich, durch den Dijkstra-Algorithmus die kürzeste Verbindung zwischen einem Startknoten und einem gewünschten Endknoten berechnet. Für die zweitkürzeste Route wird dann im nächsten Schritt diejenige abgehende Kante des Startpunktes, welche in der kürzesten Route gewählt wird, in die Routenberechnung nicht miteinbezogen. Anschließend wird ohne diese Kante wiederum mit dem Dijkstra-Algorithmus die kürzeste Verbindung

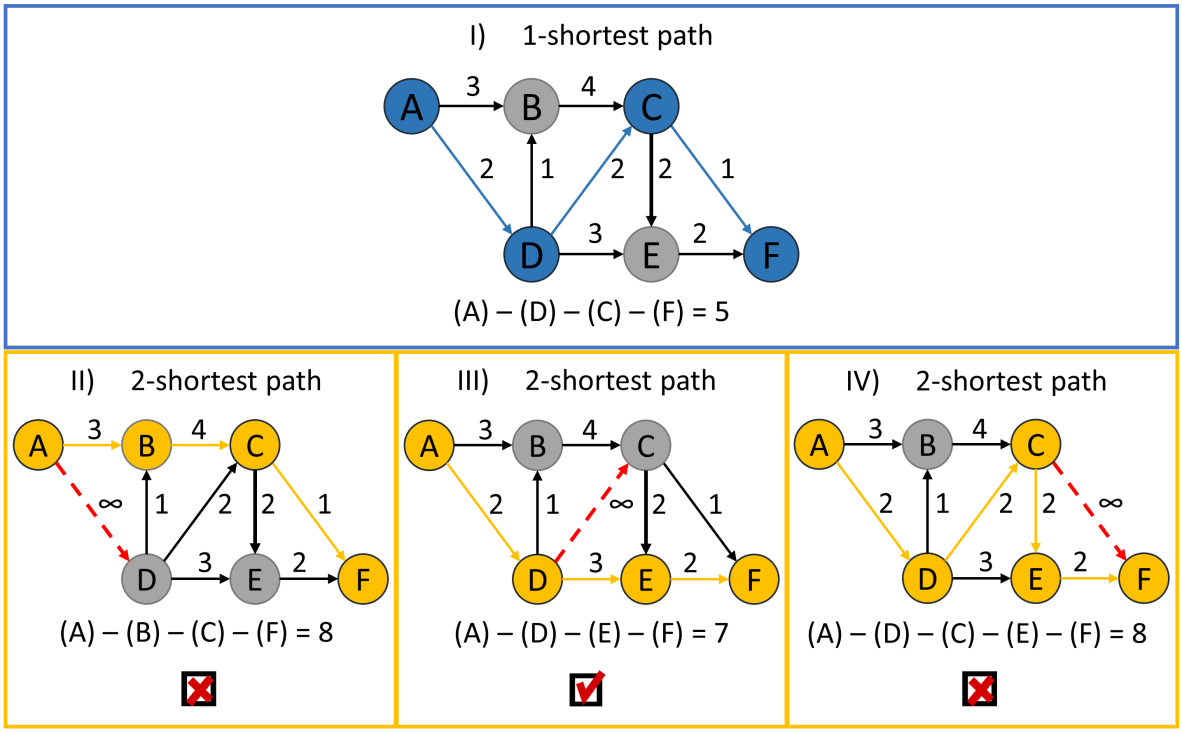


Abbildung 5.4: Funktionsprinzip k-shortest-path Yen Algorithmus

zwischen den zwei Punkten gewählt, siehe Abbildung 5.4 Bild II. Diese Route ist somit eine potentielle zweitkürzeste Route. Anschließend wird die zweite Kante aus der zu Beginn ermittelten kürzesten Route nicht in die Routenberechnung mit einbezogen und es entsteht wiederum eine mögliche zweitkürzeste Route (Abbildung 5.4 Bild III). Vernachlässigt man, wie in Abbildung 5.4 Bild IV ersichtlich, fortwährend immer eine Kante der kürzesten Route, erhält man alle möglichen potentiellen zweitkürzesten Routen. Diejenige mit den geringsten Kosten repräsentiert schlussendlich die gesuchte zweitkürzeste Route. Im Falle der Abbildung 5.4 würde dies die Route aus dem Bild III sein. Wiederholt man diese Vorgehensweise kann man alle k-shortest-path Routen eines Graphen erhalten.

Zur Vollständigkeit werden in der nachfolgenden Auflistung einige weitere Routingalgorithmen mit ihren speziellen Eigenschaften vorgestellt:

All Pairs Shortest Paths (APSP) APSP-Algorithmen sind sehr rechenaufwändig, da sie von jedem Knoten zu jedem anderen Knoten die kürzeste Route berechnen. Diese Aufgabenstellung kann beispielsweise durch den Floyd Algorithmus behandelt werden. Durch diese Berechnungsart ist abschließend eine Matrix bekannt, aufgrund derer man alle Routen innerhalb des Netzwerkes nachvollziehen kann.

Single Pair Shortest Path (SPSP) Bei einer SPSP Routenberechnung ist der Startpunkt und der Endknoten vorgegeben. Der Vorteil ergibt sich in der Berechnungszeit, da nur eine einzige Route berechnet wird.

Bidirectional Eine Bidirectionale Routenberechnung versucht die Rechenzeit über eine parallele Berechnung zu verkürzen. Dabei soll nicht innerhalb des ganzen Netzwerkes ein Suchbaum aufgespannt werden, sondern einer vom Startknoten und in einem inversen Graphen einer vom Endknoten. Der Suchbereich beider Suchbäume soll dabei bis zu einem bestimmten Umkreis definiert werden. Dieser darf allerdings nicht zu klein gewählt werden. Ansonsten würden sich die beiden Kreise nicht schneiden und die gesuchte Route vom Startpunkt zum Endpunkt könnte nicht generiert werden.

Heuristic Heuristische Algorithmen stützen sich wie Bidirektionale auf *depth-first search* Methoden. Dies bedeutet, dass ein bestimmter Suchraum für die kürzeste Route definiert wird. Im Falle von heuristischen Ansätzen wird ein bestimmter Suchbereich aufgespannt, in der sich höchstwahrscheinlich die kürzeste Route befindet. Beispielsweise werden für die Routenberechnung Kanten bevorzugt die hoch mit der *line of sight* (Luftlinie zwischen Start- und Zielknoten) korrelieren. Ein bekannter Algorithmus für diese Herangehensweise ist der A*-Algorithmus. Dabei wird eine Beurteilungsfunktion verwendet die einen elliptischen Suchbereich aufspannt.

5.4 Map-Matching

In vielen navigationsspezifischen Anwendungen wie z.B. dem Guidance ist es wichtig, dass die gemessene Position (GNSS, Bluetooth, ...) in Verbindung mit dem Knoten- und Kanten-System gebracht wird. Dadurch ist es möglich, mit Hilfe einer Manöverliste aus dem Routing, dem User aufgrund seiner aktuellen Position eine Anweisung zu geben, welche Abzweigung er bei der nächsten Kreuzung nehmen soll. Die Methode, um aus einer gemessenen Position die zugehörige auf dem Graphen zu finden, nennt sich Map-Matching. Somit ist es, wie in Abbildung 5.5 ersichtlich, durch diese Technik möglich, die gemessene Trajektorie auf den Graphen zu projizieren und dadurch eine topologische Information zwischen der aktuellen Position und dem Knoten- und Kanten-System zu erhalten.

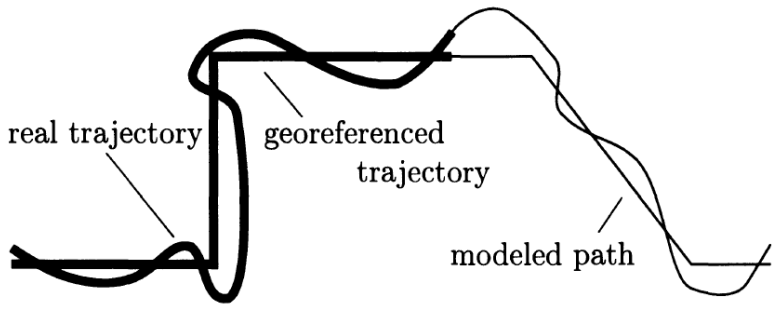


Abbildung 5.5: Funktionsprinzip des Map-Matchings aus Hofmann-Wellenhof et al. [9]

Konventionelles Map-Matching

Zu Beginn eines Map-Matching-Verfahrens werden aufgrund der aktuell gemessenen Position mögliche Kantenkandidaten ausgewählt, auf welche die Position projiziert werden könnte. Die Auswahl der Kandidaten erfolgt dabei z.B. über eine Suchregion welche durch die Fehlerellipse der Positionslösung definiert wird. Im nächsten Schritt wird ein Plausibilitätstest durchgeführt. Dabei werden Kanten, die beispielsweise eine Einbahn repräsentieren, ausgeschlossen, da sie aufgrund ihrer semantischen Information nicht für einen möglichen Kandidaten in Fragen kommen können. Anschließend erfolgt eine Korrelation mit der gemessenen Trajektorie einerseits und der bisherigen Trajektorie auf dem Graphen plus den jeweiligen Kantenkandidaten andererseits:

$$f^{corr}(s_j) = \sum_{s_i} f^{traj}(s_i) f^{map}(s_j + s_i) \tag{5.5}$$

Für die Korellationsfunktionen f^{traj} und f^{map} können Krümmungen oder Winkel in Abhängigkeit der Bogenlänge s verwendet werden. Durch diese Faltung zwischen Trajektorie und Kantenabfolge wird, bei einer gewissen Verschiebung s_j, die bestmögliche Kantenabfolge durch ein Maximum in f^{corr} detektiert. Durch die Formel 5.5 ist es somit möglich, durch die nicht-normalisierte Korrelationsfunktion zwei Trajektorien auf ihre Ähnlichkeit zu überprüfen. Nach diesem Schritt ist die für das Map-Matching bestmögliche Kante bekannt. Die aktuelle Position wird im Anschluss auf diese Kante durch einen Ausgleich nach kleinsten Fehlerquadraten projiziert. Zu Beginn dieses Ausgleichs wird ein Shiftfaktor η_s und ein Skalierungsfaktor μ_s durch folgende lineare Gleichung zwischen den Bogenlängen der gemessenen Trajektorie und jener aus dem bisherigen Map-Matching plus der Kante mit dem höchsten Kreuzkorrelationspeak definiert.

$$s_i^{map} = \mu_s s_i^{traj} + \eta_s \tag{5.6}$$

Zusätzlich wird, mit gleicher Herangehensweise wie in Formel 5.6, durch einen Shiftfaktor η_f und einen Skalierungsfaktor μ_f ein linearer Zusammenhang zwischen den Krümmungen oder Richtungen der beiden Trajektorien definiert:

$$f^{traj}(s_i^{traj}) = \mu_f f^{map}(s_i^{map}) + \eta_f \tag{5.7}$$

Setz man Gleichung 5.6 in 5.7 ein, erhält man die Beobachtungsgleichung des Ausgleichs:

$$f^{traj}(s_i^{traj}) = \mu_f f^{map}(\mu_s s_i^{traj} + \eta_s) + \eta_f + \epsilon(s_i^{traj}) \tag{5.8}$$

Die zu schätzenden Shiftparameter η_s und η_f beinhalten den Offset in der Ordinate und Abszisse zwischen der gemessenen Trajektorie und der Trajektorie auf dem Graphen. Die Skalierungsparameter μ_s und μ_f kompensieren etwaige Maßstabsfehler zwischen den beiden Trajektorien, um eine bestmögliche Einpassung der gemessenen Trajektorie auf den Graphen zu bewerkstelligen.

Die Fehlergrößen müssen deshalb geschätzt werden, weil der Graph die Realität nur vereinfacht darstellt und die Positionslösungen selbst, je nach Messsystem, mit einem Messrauschen ϵ behaftet sind.

Für den in dieser Arbeit entwickelten Graphen-basierten Algorithmus aus Kapitel 6 sind im Wesentlichen nur die Schritte des konventionellen Map-Matching bis zum Ausgleich von Bedeutung, da in den Ausführungen in Kapitel 6.10.2 nur die bestmögliche Kantenabfolge von Bedeutung ist und nicht die exakte Position auf dem Graphen.

Alternativ zu der Kreuzkorrelation kann die zugehörige Kantenabfolge zu der gemessenen Trajektorie nach Czommer [5] und Mayerhofer [11] durch eine Helmert- oder Affin Transformation ermittelt werden. Die Ähnlichkeitsinformationen werden dabei nicht über abgeleitete Größen wie Krümmungen oder Richtungen der Trajektorien berechnet sondern direkt auf der Koordinatenebene.

Kapitel 6

Graphen-basierter PDR Algorithmus

Zur Verbesserung von ungestützten PDR-Lösungen kann, wie bereits in Kapitel 4.2.3 vorgestellt, ein Graph verwendet werden. Das zugrundeliegende Knoten- und Kantenstruktur wird anhand eines Gebäudeplanes definiert und repräsentiert den möglichen Zustandsraum der Fußgängerpositionslösungen. Die Herausforderung besteht somit darin, die Positionen durch einen geeigneten Algorithmus auf den Graphen zu zwingen. In dieser Masterarbeit werden für diese Problemstellung die Smartphone-Sensoren aus Kapitel 3 verwendet. Nach den Messungen wird die Trajektorie auf dem Graphen durch eine computergestützte Berechnungssoftware, welche in Matlab 2013a programmiert wurde, berechnet.

Hierbei ist jedoch anzumerken, dass der in Folge beschriebene Graphen-basierte PDR-Algorithmus nicht auf der Funktionsweise eines konventionellen Map-Matchings aus Kapitel 5.4 beruht. Es wird dementsprechend die geeignete Kante nicht durch eine Kreuzkorrelation und die Position auf dieser durch einen Ausgleich berechnet, sondern die Kantenauswahl und die Position werden durch vordefinierte Zustände/Fälle ermittelt, siehe Abbildung 6.3. Diese Zustände hängen hauptsächlich davon ab, wie sich der Fußgänger von der aktuellen Kante wegbewegt und ob es zu dieser Bewegung eine zugehörige Kante im Graphen gibt.

Der Aufbau und das Grundprinzip des in dieser Arbeit entwickelten Graphen-basierten PDR-Algorithmus wird in diesem Kapitel zu Beginn vorgestellt. Nach den Ausführungen der benötigten Initialisierungsschritte werden die einzelnen Zustände im Detail erläutert. Da der implementierte Algorithmus eine Trajektorie über mehrere Stockwerke ermöglichen soll, wird zusätzlich ein Stockwerkswechsel mit und ohne Barometer beschrieben. Abschließend ist anzumerken, dass der Einfachheit halber in weiterer Folge alle gerichteten Kanten als Kanten bezeichnet werden, da in der Darstellung des Graphen nie die parallelen gerichteten

Kanten dargestellt werden sondern ungerichtete Kanten. Der entwickelte Graphen-basierte PDR-Algorithmus arbeitet jedoch, wie bereits erwähnt, mit parallel gerichteten Kanten.

6.1 Übersicht - Workflow

Der Worklfow des Algorithmus wird in Abbildung 6.1 und 6.2 durch ein Flussdiagramm beschrieben. Dabei kann der Aufbau in eine Initialisierungsphase, fünf Zustände (blaue Ellipsen) und vier Entscheidungen (gelbe Rechtecke) gegliedert werden. Zusätzlich visualisiert Abbildung 6.3 die in den Unterkapitel beschriebenen Zustände. Die vier Entscheidungen werden in keinem extra Kapitel erwähnt. Um dennoch Bezug darauf zu nehmen, werden sie durch römische Ziffern nummeriert.

Der Algorithmus bezieht zu Beginn den Gebäudeplan und den zugehörigen Graphen, welcher das Wegenetz modelliert, aus einer Datenbank, siehe Kapitel 6.2.1 und 6.2.2. Neben den Messdaten des Smartphones wird zusätzlich aufgrund der relativen Positionierungsmethode des Algorithmus in der Initialisierungsphase ein Startknoten als Anfangsposition und eine Startkante für die Startausrichtung benötigt.

Wenn ein Schritt detektiert wird, kann es fünf Zustände geben, wie die nächste Position berechnet wird. Die Auswahl hängt dabei von der Richtungsabweichung δ ab. Dieser Winkel wird durch die Drehraten berechnet und gibt an, wie stark sich der Fußgänger von der aktuellen Kante wegbewegt.

1. Geradeausgehen: Je nach Schrittlänge wird entlang der Kante ein Schritt getätigt.
2. 180° Drehung: Durch diese Drehung wird die inverse Kante der momentanen Kante ausgewählt und die neue Position auf dieser berechnet.

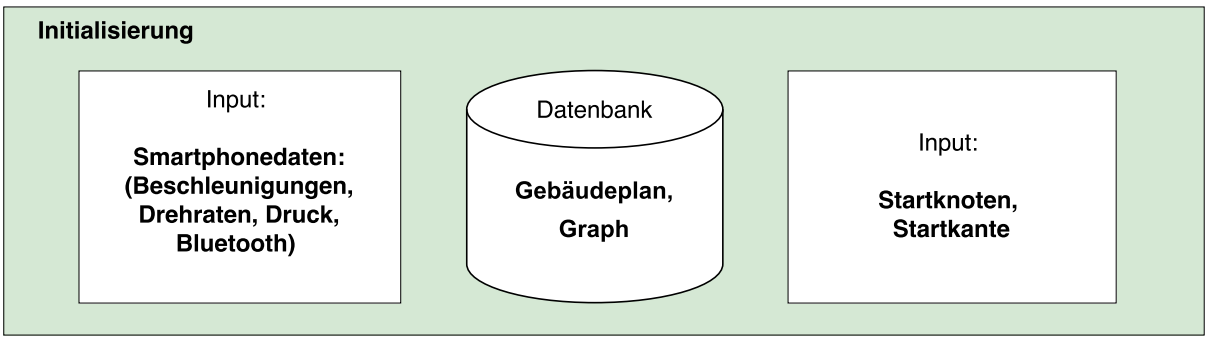


Abbildung 6.1: Initialisierungsschritt des Graphen-basierten PDR-Algorithmus

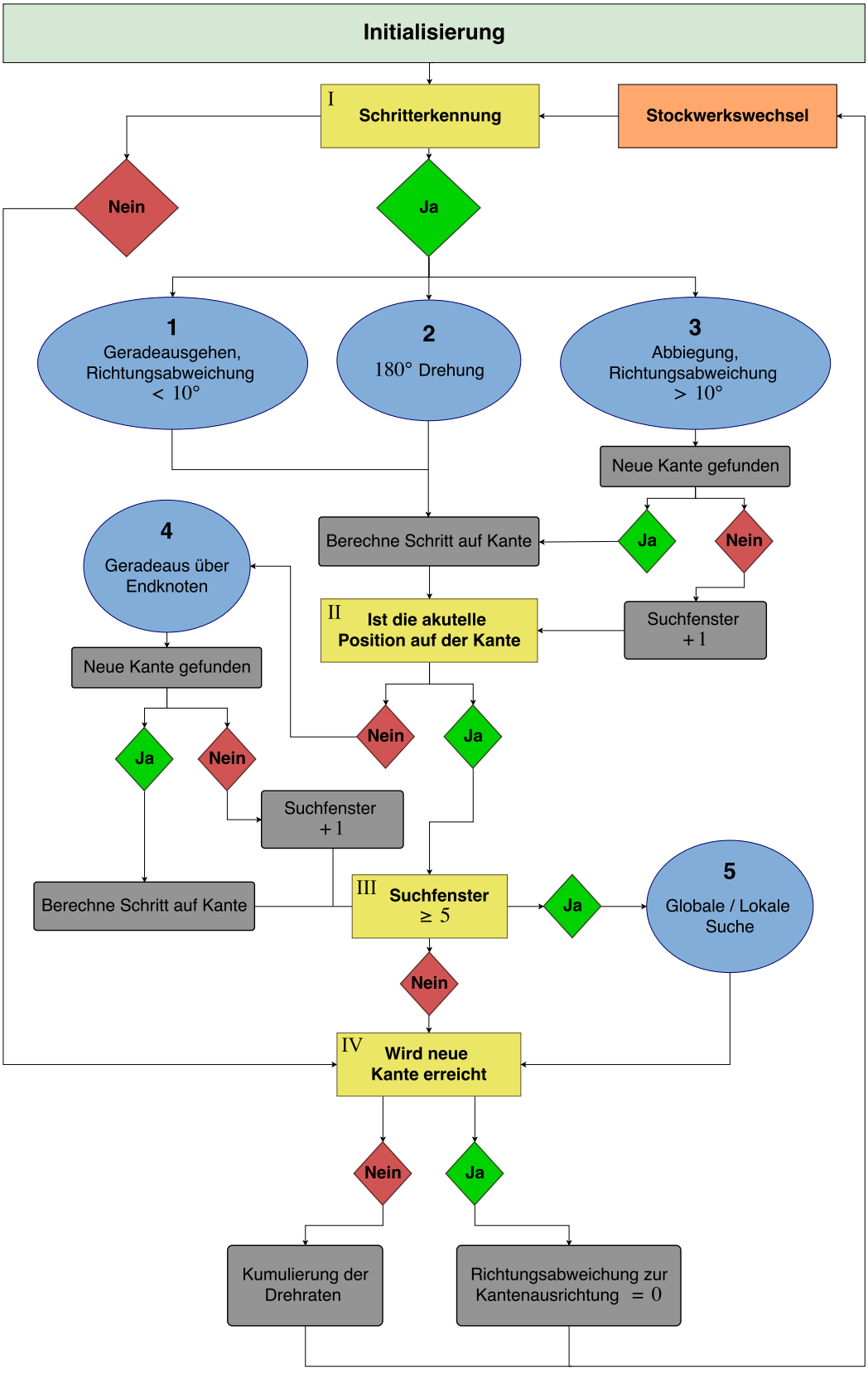


Abbildung 6.2: Workflow des Graphen-basierten PDR-Algorithmus

3. Abbiegung: Falls die Bewegungsrichtung des Fußgängers der Kantenausrichtung nicht mehr entspricht, muss eine neue Kante gefunden werden. Wenn bei dem nächstgelegenen Knoten keine zum aktuellen Heading passende Kante detektiert wird, soll der Algorithmus bei diesem Schritt keine neue Position berechnen und weitere Schritte abwarten bis eine klare Entscheidung getroffen werden kann. Dieses Suchfenster darf allerdings maximal fünf Schritte betragen.

4. Geradeaus über Endknoten: Wenn das Ende einer Kante erreicht wird und der Fußgänger geradeaus weitergeht, muss jene inzidente Kante ausgewählt werden, welche eine geradlinige Bewegung über den Knoten hinaus beschreibt.

5. Globale/Lokale Suche. Überschreitet das Suchfenster fünf Schritte wird der bestmögliche Kantenkandidat innerhalb eines stochastisch definierten Bereiches oder im gesamten Graphen gesucht.

Erreicht man in Folge der Positionsberechnungen eine neue Kante, wird die Richtungsabweichung δ zur Kantenausrichtung wieder auf Null gesetzt. Dadurch kann der Drift in der Richtungsberechnung kompensiert werden, da die Drehraten nur mehr über den Zeitraum einer Kante und nicht über den gesamten Messzeitraum aufkumuliert werden. Falls die Barometerdaten einen Stockwerkswechsel detektieren, wird der zugehörige Graph des neuen Stockwerkes geladen.

6.2 Modellierung und Initialisierung

In der Modellierungs- und Initialisierungsphase werden alle benötigten Größen entweder erstellt oder für den Algorithmus entsprechend vorbereitet. Dazu zählt die Erstellung eines Gebäudeplanes in einem geeigneten Koordinatensystem und darauf aufbauend eine Konstruktion eines Graphen. Die Definition der Startposition und Startausrichtung charakterisiert einen essentieller Punkt der Initialisierungsphase. Aufgrund des Fehlerverhaltens des Gyroskops kann zusätzlich eine Kalibrierung der Drehraten vorgenommen werden.

6.2.1 Gebäudeplan

Der Graph legt durch die Knoten- und Kantenstruktur jene Linien fest, auf denen die Positionslösungen des Fußgängers möglich sind. Dementsprechend soll der Graph alle für

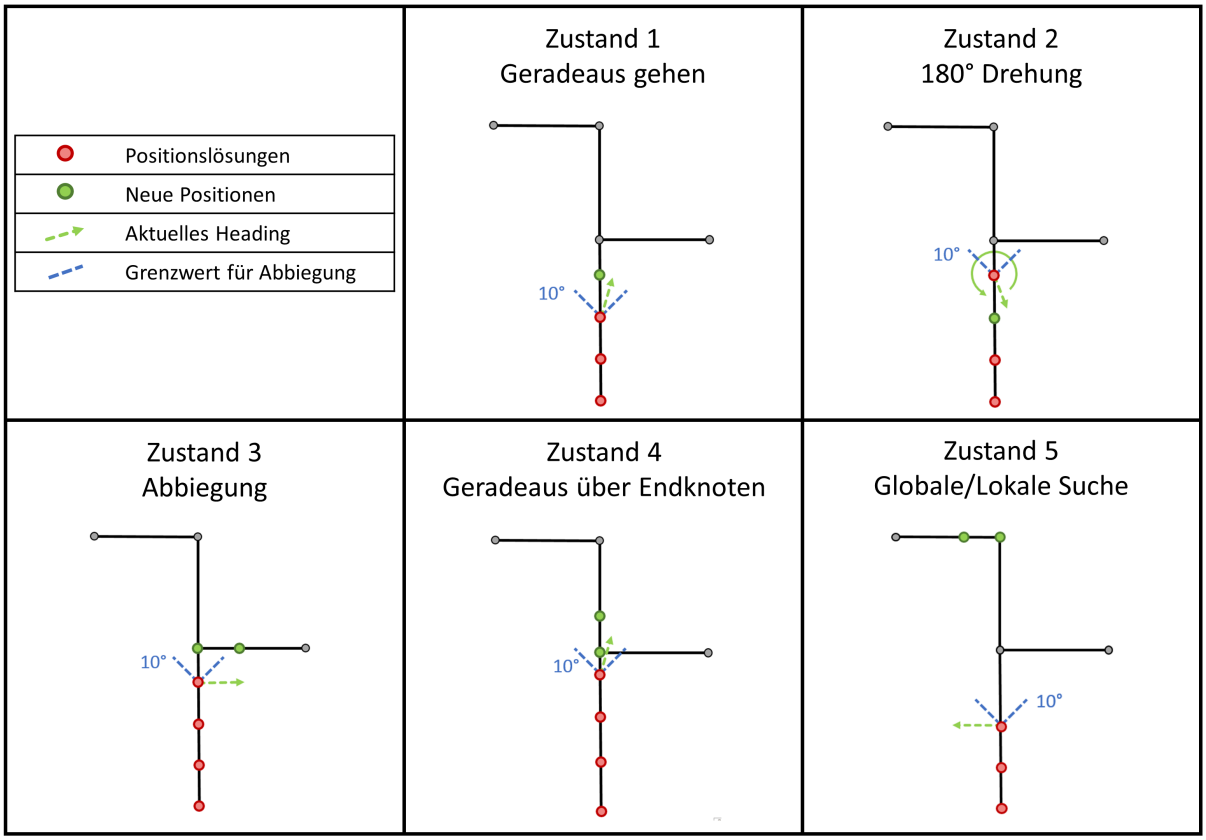


Abbildung 6.3: Fallunterscheidungen bei der Positionsbestimmung

den Fußgänger zugänglichen Bereiche abbilden. Aus diesem Grund muss für die Erstellung der Graphen ein Gebäudeplan vorliegen, damit Korridore, Stiegen, Eingänge usw. modelliert werden können.

Der in dieser Arbeit forcierte Graphen-basierte PDR-Algorithmus wird in zwei Gebäuden der Technischen Universität Graz getestet. Eine Testumgebung befindet sich in der Steyrergasse 30. In diesem Gebäude werden einschließlich des Kellers und des Dachgeschosses (DG) sechs Stockwerke modelliert. Aus diesem Grund können stockwerksübergreifende Trajektorien getestet werden. Die zweite Testumgebung befindet sich in der Inffeldgasse 16 (Graz). Dieses Gebäude wird aufgrund seiner Größe für lange Trajektorien innerhalb eines Stockwerkes verwendet.

Die Erstellung des Gebäudeplanes wird anhand der Testumgebung Steyrergasse 30 erläutert. Als Grundlage dafür dienen die Stockwerkspläne der einzelnen Geschosse und ein Lageplan. Diese Pläne sind in einem Drawing Interchange Format (DXF) gespeichert und können mit einer Computer Aided Design (CAD) Software bearbeitet werden. In dieser Masterarbeit

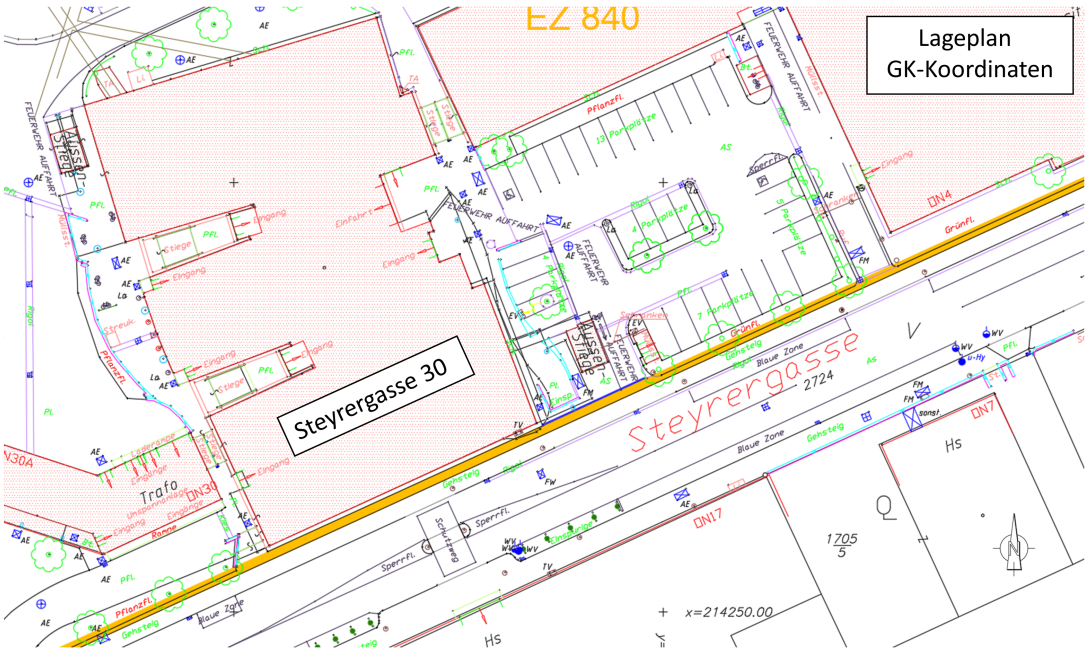


Abbildung 6.4: Lageplan der Testumgebung Steyrergasse 30

wurde dabei das AutoCAD 2017 Programm verwendet. Der Lageplan, siehe Abbildung 6.4, wird von der BIG zur Verfügung gestellt und bezieht sich auf GK-Koordinaten. Dieser Plan dient dazu, dass die Stockwerkspläne eingenordet, skaliert und mit GK-Koordinaten versehen werden können. Folglich wird aus dem Lageplan nur der Umriss der Steyrergasse 30 entnommen.

Die Stockwerkspläne, welche ebenfalls von der BIG bereitgestellt wurden, sind in einem lokalen Koordinatensystem festgehalten, siehe Abbildung 6.5. Aufgrund der Skalierung kann dieses System nicht als Grundlage für den Algorithmus dienen, da beispielsweise die Gänge im Verhältnis zu einer gemessenen PDR-Trajektorie verkürzt dargestellt werden. Damit die Stockwerkspläne dennoch als Grundlage für den Graphen dienen, werden sie mit Hilfe des Lageplanes georeferenziert. Dazu wird zu Beginn eine markante Gebäudeecke des Stockwerkplans in die entsprechende Ecke des Lageplanes gelegt. Anschließend wird der Stockwerksplan in den Lageplan hineingedreht. Vergleicht man die Längen zweier zugehöriger Gebäudekanten erhält man den Skalierungsfaktor. Der sich daraus ergebende Gebäudeplan für beispielsweise das Erdgeschoss (EG) ist in Abbildung 6.6 ersichtlich. Durch diese Methode stimmen die Stockwerkspläne im Zentimeterbereich mit dem Lageplan überein. Dieser Fehler kann hinsichtlich der erzielten Genauigkeiten des Graphen-basierten PDR-Algorithmus vernachlässigt werden.

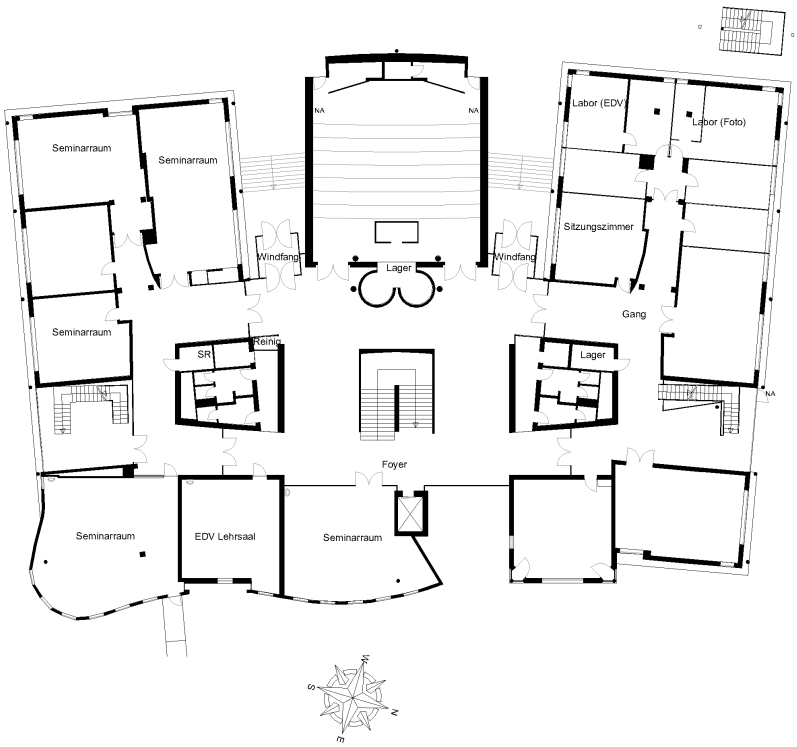


Abbildung 6.5: Stockwerksplan des Erdgeschosses der Testumgebung Steyrergasse 30

Abschließend ist anzumerken, dass die Linien des Gebäudeplanes durch Polylinien in einem CAD Zeichenprogramm erstellt werden müssen. Dies ist deshalb notwendig, damit der Algorithmus zwischen der Gebäudestruktur und den darauf aufbauenden Linien des Graphen unterscheiden kann.

6.2.2 Graph

Der Graph wird aufbauend auf dem in Kapitel 6.2.1 beschriebenen Gebäudeplan direkt in einer CAD Software erstellt. Durch Linien und Punkte können je nach Stockwerk und zugänglichen Räumen die Knoten und Kanten des Graphen generiert und georeferenziert werden.

Die Abbildung 6.6 visualisiert den Graphen des Erdgeschosses der Steyrergasse 30. Dieser Graph besteht aus 138 Knoten und 384 Kanten. Durch diese Konstellation ergibt sich nach der Formel 5.2 $\left(m \ll \frac{n(n-1)}{2} \;\hat{=}\; 384 \ll 9453\right)$ ein spärlicher Graph. Für einen kompletten Graphen müssten somit nach Formel 5.1 9453 Kanten vorhanden sein. Eine weitere Eigenschaft ist die Planarität des Netzwerkes. Falls sich die Kanten nicht nur

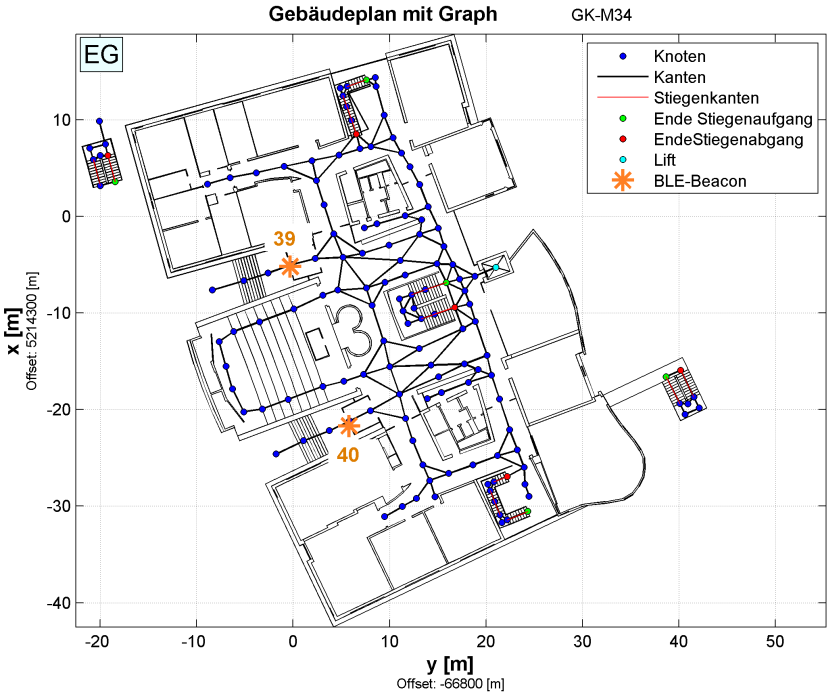


Abbildung 6.6: Endresultat des editierten Stockwerkplanes mit einem Graphen und dessen semantische Kennzeichnungen

in den Knoten schneiden, würde dies allerdings für den Algorithmus keinen Unterschied bedeuten.

Der Graph wird als ein simpler Graph mit ungerichteten Kanten konstruiert. Der Algorithmus benötigt allerdings die Information der aktuellen Kantenausrichtung. Diesbezüglich entstehen in der Generierung der Adjazenzliste aus einer ungerichteten Kante zwei parallele Kanten. Diese besitzen vertauschte Anfangs- und Endknoten und sind somit in entgegengesetzte Richtungen gerichtet. Gewichtet werden die einzelnen Kanten durch die zugehörige euklidische Norm. Anzumerken ist, dass die erläuterten Grapheneigenschaften für dieses EG für alle in dieser Arbeit konstruierten Graphen zutreffend sind.

Die Modellierung der Abbiegungen und Kurven sowie die Kantenlängen hat eine essentielle Auswirkung auf den Algorithmus und seine Wirkungsweise. Bei zu langen Kanten kann es vorkommen, dass der Drift in den Gyroskopdaten und geringfügige Abweichungen in der Bewegungsrichtung (z.B. kleine Veränderung der *hand-held* Position) aufgrund der langen Kumulationszeiten der Drehraten als Abbiegung wahrgenommen werden, wohingegen die eigentliche Bewegung des Fußgängers geradlinig verläuft. Bei zu kurzen Kanten kann

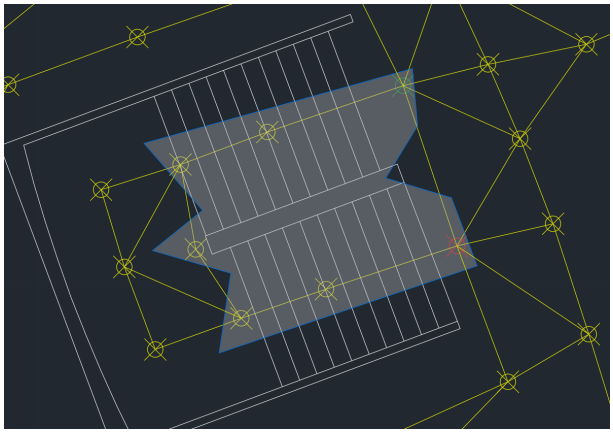


Abbildung 6.7: Kennzeichnung von Stiegenkanten

es passieren, dass bei einer detektierten Abbiegung der nächstgelegene Knoten keine zur Bewegungsrichtung passende Kante besitzt, da diese sich erst bei dem übernächsten Knoten befindet. Die Ausführungen zu dieser Problemstellung werden in Kapitel 7.1 genauer beschrieben.

Gespeichert wird der konstruierte Graph und der zugehörige Gebäudeplan in einem DXF-File. Aus diesem File können die Knoten und Kanten in der Berechnungssoftware extrahiert und in eine erweiterten Adjazenzliste umgewandelt werden. Im Gegensatz zu einer Adjazenzmatrix hat diese Speichermodalität wesentlicher weniger Einträge. Zusatzinformationen wie z.B. die Kantenlänge oder die Kantenausrichtung können in dieser Speichermodalität ebenfalls mitgespeichert werden. Aufgebaut ist die in dem Graphen-basierten PDR-Algorithmus verwendete Adjazenzliste wie folgt:

Tabelle 6.1: Aufbau der erweiterten Adjazenzliste

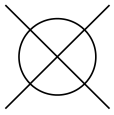
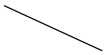
Anfangknoten ID	y [m]	x [m]	Stockwerk	Endknoten ID	y [m]	x [m]	Stockwerk	Kantenausrichutng	Kantenlänge
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Bei der Erstellung des Graphen müssen zusätzlich gewissen Knoten und Kanten eine semantische Information erhalten. Dadurch ist es dem Algorithmus möglich, Stiegenkanten, die Endknoten des Stiegenaufgangs und -abgangs, sowie Liftknoten während der Bewegung des Fußgängers zu detektieren. Für Stiegenkanten wird, wie in Abbildung 6.7 ersichtlich, um die entsprechenden Kanten ein Polygon gezeichnet.

Die Berechnungssoftware des Algorithmus ermittelt anhand des Graphen und des Polygons alle Knoten die innerhalb des Polygons liegen. Diese erhalten das Attribut Stiegenknoten. Anschließend wird überprüft, ob die Kanten der jeweils adjazenten Stiegenknoten das

Polygon schneiden. Falls dies zutrifft, werden diese Kanten nicht als Stiegenkanten ausgewiesen. Dieser Fall muss deshalb behandelt werden, da ansonsten beispielsweise die Kante zwischen den Knoten am Stiegenaufgang und -abgang (roter und grüner Knoten) ebenfalls als Stiegenkanten detektiert wird, obwohl sie auf einer Ebene liegen. Der rote Knoten beschreibt den letzten Punkt des Stiegenabgangs und somit den letzten Knoten vor dem Wechsel in das darunterliegende Stockwerk. Gleiches gilt für den grünen Knoten, wobei hier der Wechsel in das darüber liegende Stockwerk behandelt wird. Damit diese Knoten und der Liftknoten die zugehörige Semantik erhalten, müssen sie auf den entsprechenden Layer innerhalb des CAD Zeichenprogrammes gelegt werden. Die abschließende Tabelle fasst alle Größen zusammen, die in der CAD Software speziell konstruiert bzw. gekennzeichnet werden müssen.

Tabelle 6.2: Konstruktionsvorschriften bei der Erstellung eines Graphen

Abbildung		**Zeichentyp**	**Layer**
	Gebäude	Polylinie	Stockwerk
	Knoten	Punkt	Graph
	Kanten	Linie	Graph
	Stiegen	Polygon	Stiegen
	Ende des Stiegenaufgangs	Punkt	upstairs
	Ende des Stiegenabgangs	Punkt	downstairs
	Lift	Punkt	Lift

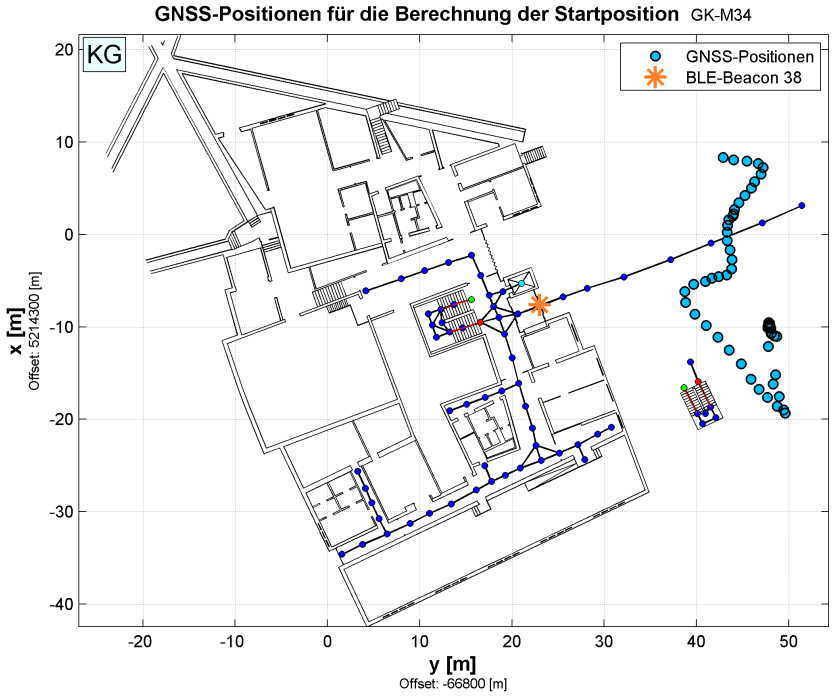


Abbildung 6.8: GNSS-Messungen entlang des Graphen bis ins Gebäudeinnere

6.2.3 Startposition und Startausrichtung

Für die Initialisierung der Startposition und Startausrichtung wurden in dieser Arbeit zwei Methoden entwickelt. Die erste Methode beinhaltet eine simple manuelle Eingabe des Startknotens und der Startkante in der Berechnungssoftware. Die zweite Methode beinhaltet eine automatisierte Erkennung der Startposition und Startausrichtung. Smartphone-basierte GNSS Messungen können für diese Aufgabenstellung, wie in Abbildung 6.8 ersichtlich, jedoch nicht herangezogen werden. Die GNSS-Positionslösungen (hellblauen Kreise) in dieser Abbildung sollten eigentlich den Verlauf entlang des eingezeichneten Graphen außerhalb des Gebäudes beschreiben. Aufgrund der starken Streuung $\sigma \approx 15$m und der Mehrwegeffekte, bedingt durch die umliegenden Gebäude, kann der Zeitpunkt des Betretens des Gebäudes nicht erkannt werden.

Ein verbesserter und robuster automatisierter Ansatz wurde durch BLE-Beacons implementiert. Diesbezüglich wurden an den Eingängen über den Türen der Testumgebung Steyrergasse 30 im Erdgeschoss und im Kellergeschoss, siehe Abbildung 6.6 und 6.8, *Accent Advanced System* BLE-Beacons angebracht. Nach der in Kapitel 3.2 beschriebenen Cell of Origin Methode kann ab einer bestimmten Signalstärke schlussgefolgert werden, dass

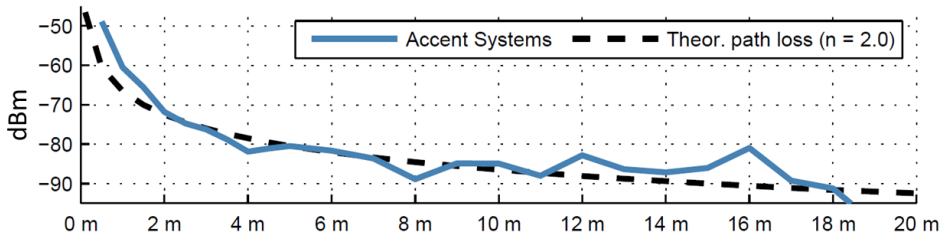


Abbildung 6.9: Theoretischer und empirisch bestimmter Verlauf der Signalenergie des BLE-Beacons Accent Advanced Systems, aus Wilfinger [19]

der Fußgänger durch die spezifische Eingangstüre das Gebäude betritt. Demnach wird der entsprechende Knoten des Einganges als Startposition ausgewählt. Die Kantenausrichtung der zugehörige Kante die in das Gebäude hineinführt wird als Startausrichtung verwendet.

Wilfinger [19] testet in seiner Arbeit den Verlauf der Signalleistung von den benützten *Accent Advanced System* BLE-Beacons, siehe Abbildung 6.9. Demnach kann ein gemessener Wert nahe -50dBm für diese Art von BLE-Beacons dann erwartet werden, wenn sich der Fußgänger direkt bei der Signalquelle befindet.

Zu Vergleichszwecken wurde ein Test in der Steyrergase 30 durchgeführt. Wie in Abbildung 6.10 ersichtlich, startete die Testperson sechs Schritte vor dem Eingang mit dem BLE-Beacon 40, siehe Abbildung 6.6. Die maximale registrierte Signalleistung konnte in unmittelbarer Umgebung des sechsten Schrittes empfangen werden (grüner Punkt in Abbildung 6.10). Die gemessene Signalleistung liegt im Vergleich zu Abbildung 6.9 ebenfalls bei ca. $-50dBm$. Zusätzlich konnte im Vergleich zu dem BLE-Beacon 39 der richtige Eingang identifiziert werden. Dementsprechend eignet sich für die Identifikation der Startposition dieser automatisierte Ansatz, wenn der Fußgänger das Gebäude betritt. Als Grenzwert für den Beginn des Graphen-basierten PDR-Algorithmus wurde eine empfangene Signalleistung von -60 dBm festgelegt.

6.2.4 Kalibrierung

Moder et al. [12] untersuchen die Sinnhaftigkeit der Kalibrierung von Smartphone-Sensoren hinsichtlich PDR-Lösungen. Dabei wurden die Erkenntnisse erzielt, dass bei Gyroskopdaten aufgrund der systematischen Fehler bzw. des vorhanden Drifts eine Kalibrierung einen

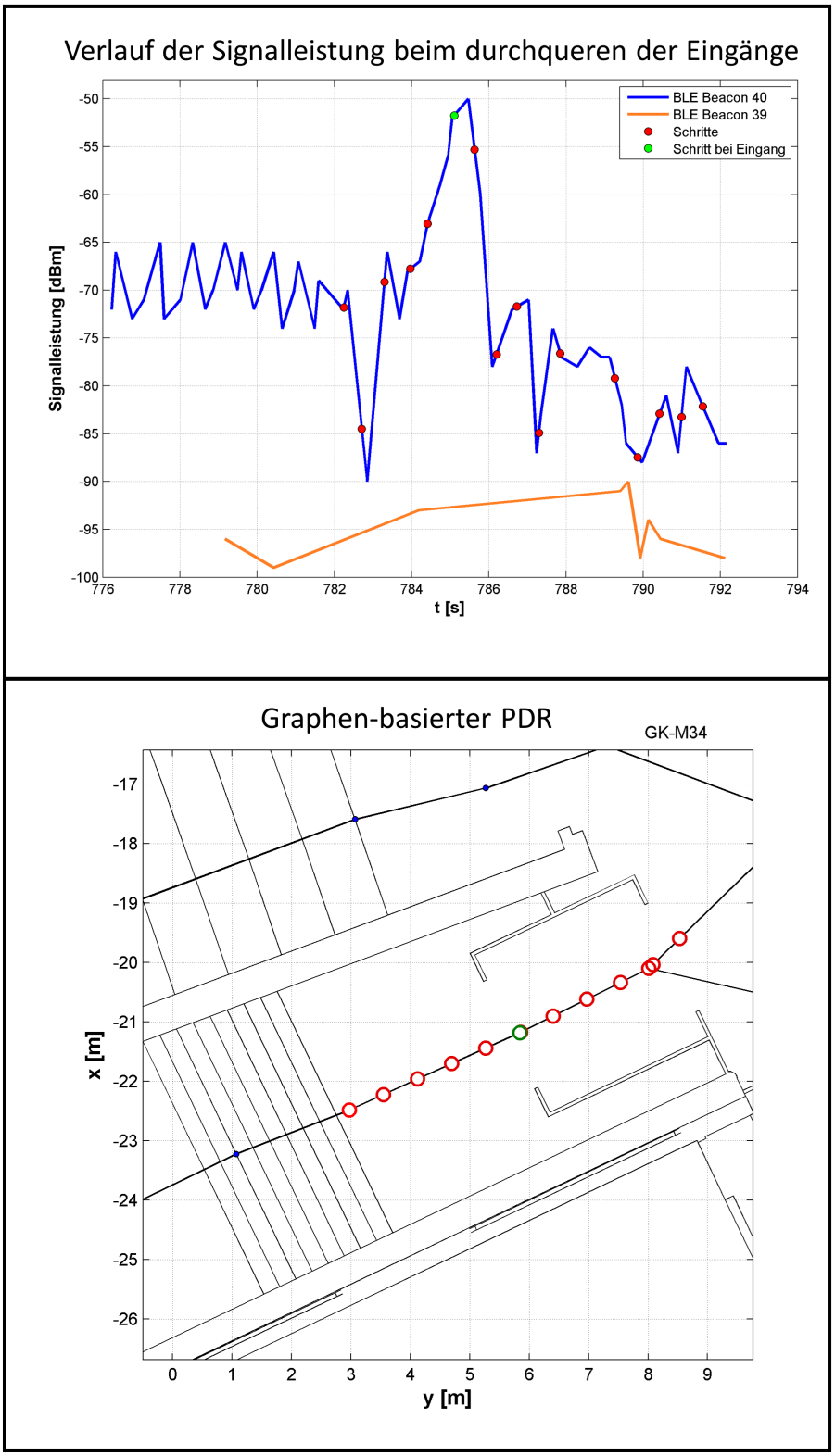


Abbildung 6.10: Vergleich zwischen maximaler empfangener Signalstärke und tatsächlicher Position während des Betretens des Gebäudes

Vorteil bringt. Bei den Beschleunigungsmessungen konnte durch eine Kalibrierung keine wesentlichen Verbesserungen erzielt werden. Demnach wird in diesem Graphen-basierten PDR-Algorithmus nur eine Kalibrierung der Drehraten vorgenommen.

Ohne zusätzliches Equipment beruht das grundlegende Konzept der Sensorkalibrierung auf statischen und unabhängigen Messungen. Diesbezüglich kann das Smartphone auf einen Tisch oder den Boden gelegt werden. Die dabei gemessenen Drehraten $\tilde{\boldsymbol{\omega}}$ werden über einen Zeitraum von ca. einer Minute durch Formel 6.1 mit den fehlerfreien Drehraten $\boldsymbol{\omega}$ und dem gesuchten Bias $\mathbf{b}$ in Relation gebracht:

$$\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega} + \mathbf{b} \tag{6.1}$$

Die fehlerfreien Drehraten $\boldsymbol{\omega}$ sind über folgende Formel definiert:

$$\boldsymbol{\omega} = \begin{pmatrix} (\dot{\lambda} + \omega_E)\cos(\varphi) \\ -\dot{\varphi} \\ -(\dot{\lambda} + \omega_E)\sin(\varphi) \end{pmatrix} \tag{6.2}$$

Für den stationären Fall würden die Positionsänderungen $\dot{\varphi}$ und $\dot{\lambda}$ den Wert Null annehmen.

$$\boldsymbol{\omega} = \begin{pmatrix} \omega_E \cos(\varphi) \\ 0 \\ -\omega_E \sin(\varphi) \end{pmatrix} \tag{6.3}$$

Die Größe ω_E beschreibt die Erdrotation mit $\sim 0,004167°/s$. Da dieser Wert im Verhältnis zu den gesuchten Kalibrierungsparametern wesentlich geringer ist (eine Zehnerpotenz kleiner), kann dieser vernachlässigt werden. Demnach entsprechen die gemessenen Drehraten in der statischen Phase dem gesuchten Bias:

$$\tilde{\boldsymbol{\omega}} = \mathbf{b} \tag{6.4}$$

Durch eine Mittelbildung über den gesamten statischen Zeitraum können somit der Kalibrierungsparameter für jede Achse berechnet und an den Messungen angebracht werden. Anzumerken ist, dass dieser berechnete Bias nicht konstant ist. Dementsprechend muss diese Kalibrierung vor jedem Messvorgang neu durchgeführt werden.

6.3 Google Schrittdetektion

Neben den Methoden aus Kapitel 4.1.1 kann aufgrund des Android-basierten Smartphone-Betriebssystem auch auf die Application Programming Interface (API) der Google Schrittdetektion zugegriffen werden. Diese Funktion ist ebenfalls in der von der Arbeitsgruppe Navigation der TU Graz bereitgestellten Software implementiert. Als Resultat erhält man jene Zeitpunkte bei denen ein Schrittevent stattgefunden hat.

Zur Überprüfung dieser Schrittdetektion dient auf einer *foot-mounted* IMU basierte Schritterkennung als Referenzlösung. Durch die Montierung der Messeinheit auf dem Fußrücken können bei dieser Methode nur Doppelschritte detektiert werden. Dieser Schritt bezeichnet den Vorgang von der Ruhephase des Fußes bis zur nächsten Ruhephase des gleichen Fußes. Ein einzelner Schritt beinhaltet den Prozess zwischen den Ruhephasen beider Beine. (z.B. die Bewegung ab der Ruhephase des rechten Fußes bis der linke Fuß wieder am Boden aufsetzt und umgekehrt). Demnach müssen immer zwei einzelne Schritte während eines Doppelschrittes getätigt werden.

Die Abbildung 6.11 vergleicht die *foot-mounted* Schrittdetektion und die *hand-held* Smartphone-basierte Schritterkennung von Google. Dabei kann schlussgefolgert werden, dass die Schrittdetektion von Google zu Beginn die ersten sechs Schritte nicht erkennt. Im weiteren Verlauf werden jedoch alle Einzelschritte detektiert. Mit Hilfe zusätzlicher Testmessungen kann schlussgefolgert werden, dass am Start und nach jeder Stehphase ca. die ersten fünf Schritte nicht detektiert werden können, da die Schrittdetektion von Google eine Art Einschwingphase besitzt.

Diese fehlenden Schritte am Start werden im Zuge des Algorithmus einfach durch eine entsprechend veränderte Startposition kompensiert. Falls der User jedoch während der Trajektorie stehen bleibt, können die fehlenden Schritte nicht rekonstruiert werden. Für Trajektorien mit Stehphasen kann somit nicht auf die Schrittdetektion von Google zurückgegriffen sondern auf die in Kapitel 4.1.1 vorgestellte Methode. Da für die Testmessungen jedoch auf Stehphasen verzichtet wurde, beruhen die Ergebnisse aus Kapitel 7 auf der Schritterkennung von Google.

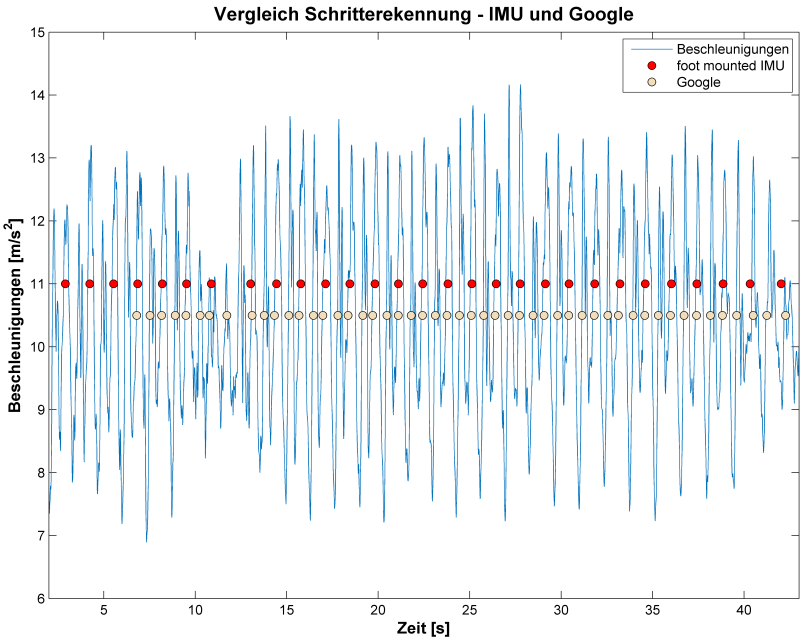


Abbildung 6.11: Vergleich der Schrittdetektion mittels *foot-mounted* IMU und jener von Google

6.4 Fehlerschätzung

Informationen über die erzielten Genauigkeiten der Positionslösungen des Algorithmus können aufgrund fehlender Referenztrajektorien nur über eine Varianzfortpflanzung der Formel 4.8 aus Kapitel 4.1.4 approximiert werden. Dementsprechend kann die Genauigkeitsinformation nicht unabhängig und exakt berechnet, sondern nur durch bestimmte Annahmen modelliert werden.

Die Varianzfortpflanzung aus Formel 6.6 basiert auf den Gleichungen aus 4.8, wobei diese zusätzlich um einen Graphenbias G erweitert werden müssen:

$$\begin{aligned} f_1(y_{i-1}, d, \alpha, G) &= y_i = y_{i-1} + d\sin\alpha + G \\ f_2(x_{i-1}, d, \alpha, G) &= x_i = x_{i-1} + d\cos\alpha + G \end{aligned} \tag{6.5}$$

$$\begin{bmatrix} \sigma_{y_i}^2 & \sigma_{y_i x_i} \\ \sigma_{x_i y_i} & \sigma_{x_i}^2 \end{bmatrix} = \mathbf{B}\boldsymbol{\Sigma}\mathbf{B}^{\mathsf{T}} \tag{6.6}$$

In der Matrix $\mathbf{B}$ befinden sich die partiellen Ableitungen nach den fehlerhaften Größen:

$$\mathbf{B} = \begin{bmatrix} \frac{\partial f_1}{\partial d} & \frac{\partial f_1}{\partial \alpha} & \frac{\partial f_1}{\partial y_{i-1}} & \frac{\partial f_1}{\partial x_{i-1}} & \frac{\partial f_1}{\partial G} \\ \frac{\partial f_2}{\partial d} & \frac{\partial f_2}{\partial \alpha} & \frac{\partial f_2}{\partial y_{i-1}} & \frac{\partial f_2}{\partial x_{i-1}} & \frac{\partial f_2}{\partial G} \end{bmatrix} \tag{6.7}$$

Neben den Messgrößen in Matrix **B** müssen zusätzlich die Standardabweichungen der letzten Position, der Schrittlänge und der Richtung für die Matrix **Σ** gegeben sein, sowie eine weitere Unsicherheit durch die Beschränkung der Position auf den Graphen. Das Problem dieser Fehlergrößen ist, dass einerseits die Standardabweichungen der Smartphone-Sensoren für einen Schritt nicht exakt bekannt sind und andererseits kann der Fehler, welcher durch die Generalisierung der Trajektorie auf den Graphen entsteht, nicht präzise und allgemeingültig definiert werden (Fehler variiert z.B. je nach Flurbreite). Die angeführten Standardabweichungen aus Tabelle 6.3 können somit nur empirisch und aus Erfahrungswerten bestimmt werden.

Tabelle 6.3: Standardabweichungen der PDR-Varianzfortpflanzung

	Größe	**Standardabweichung**
Letzte Position	σ_y,σ_x	nicht konstant
Schrittlänge	σ_d	0,05m
Richtung	σ_α	0,5°
Unsicherheit durch Graph	σ_G	1/2 Flurbreite $\sim$ 1,5m

Die Kovarianzmatrix **Σ** beinhaltet je nach Reihenfolge der Ableitung in der Matrix 6.7 die zugrundeliegenden Varianzen:

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_d^2 & 0 & 0 & 0 & 0 \\ 0 & \sigma_\alpha^2 & 0 & 0 & 0 \\ 0 & 0 & \sigma_y^2 & \sigma_{yx} & 0 \\ 0 & 0 & \sigma_{xy} & \sigma_x^2 & 0 \\ 0 & 0 & 0 & 0 & \sigma_G^2 \end{bmatrix} \tag{6.8}$$

Innerhalb des relativen Positionierungsalgorithmus wird somit diese Varianzfortpflanzung für jeden Schritt berechnet. Je nach Bias erhalten die Positionslösungen einen zugehörigen Grauwert, wobei weiß einen niedrigen (1,5 m) und schwarz einen hohen Fehler (drei Meter) repräsentiert. Die Abbildung 6.12 visualisiert die Positionslösungen (rot umrandete Kreise)

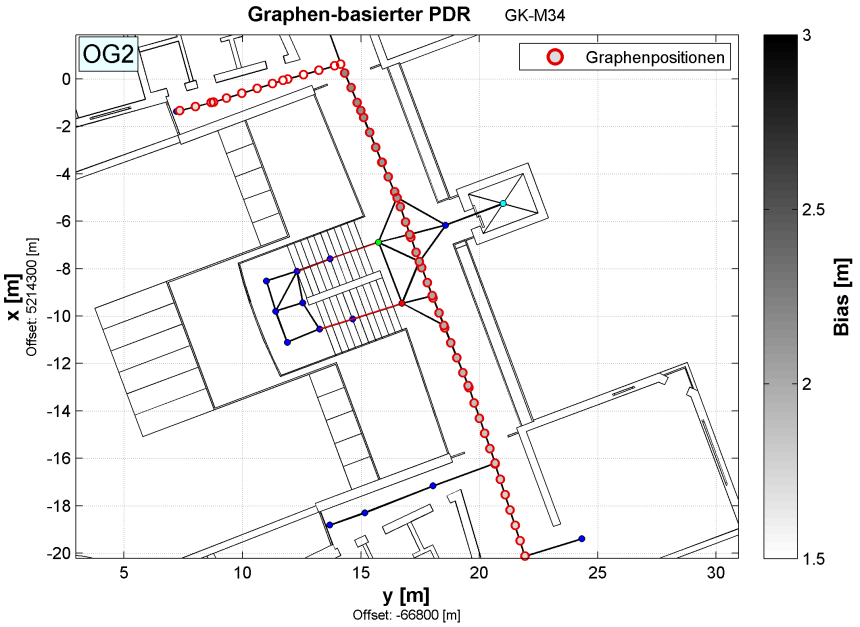


Abbildung 6.12: Visualisierung der Fehlerschätzung

mit den zugehörigen Standardabweichungen (Grauwert der Füllung der Kreise). Dabei ist außerdem ersichtlich, dass sich die Standardabweichungen der Positionen, je mehr Schritte getätigt werden, verschlechtern. Schafft es der Algorithmus jedoch eine Abbiegung, ohne den in Kapitel 6.10 vorgestellten Zustand, zu detektieren und auf die entsprechende Kante zu verweisen, wird die erste Position nach der Abbiegung wieder als fehlerfrei angenommen, siehe Abbiegung in Abbildung 6.12. Anzumerken ist dabei, dass die Kovarianzmatrix der Startposition ebenfalls als Nullmatrix initialisiert wird.

6.5 Richtungsschätzung während des Algorithmus

Der große Vorteil des Graphen-basierten PDR Algorithmus besteht, wie bereits erwähnt, in der Kompensierung des Drifts in der Richtung durch kurze Kumulationszeiten (nur eine Kante) der Drehraten. Nachfolgend wird anhand des Flussdiagrammes 6.2 aus Kapitel 6.1 beschrieben, wie die Richtungsschätzung innerhalb des Algorithmus implementiert wurde:

Zu Beginn muss in der Initialisierungsphase die Startausrichtung durch die Startkante

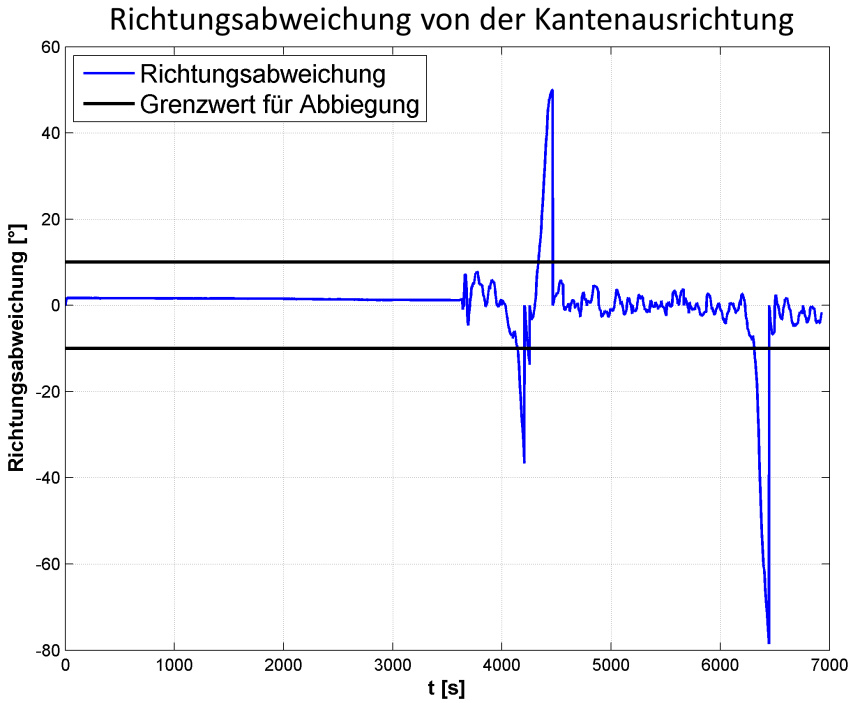


Abbildung 6.13: Richtungsabweichung von der Kantenausrichtung durch Drehratenmessungen

festgelegt werden. Anschließend wird fortwährend in I (Schritterkennung) überprüft, ob zu dem Zeitpunkt i ein Schrittevent stattfindet. Falls nicht, wird direkt auf die Entscheidung IV (neue Kante erreicht) verwiesen, welche für den Zeitpunkt i negativ ausfällt, da eine neue Kante nur durch einen Zustand, siehe Abbildung 6.3, während eines Schrittes erreicht werden kann. Dementsprechend kumuliert man die Drehraten wie in Kapitel 4.1.3 auf. Diesbezüglich erhält man fortwährend eine Information wie sich der Fußgänger von der Kante wegbewegt.

Aufgrund dieser Richtungsinformation basiert die Auswahl der fünf Zustände während eines Schrittes. Erreicht man dadurch eine neue Kante, werden die Drehraten nicht weiter aufkumuliert sondern die zuvor ermittelte Richtungsabweichung δ wird wieder auf Null gesetzt. Nach diesem Schritt werden somit die Drehraten unabhängig der vorigen von neuem aufkumuliert bis wieder eine neue Kante erreicht wird. Die Abbildung 6.13 visualisiert die sich daraus ergebenden Richtungsabweichungen δ über den gesamten Messzeitraum. Hierbei kann das Zurücksetzen durch das Erreichen einer neuen Kante deutlich erkannt werden. Zusätzlich ist der Grenzwert von 10° angegeben, welcher die Schranke für Abbiegungen und Geradeausgehen repräsentiert. Die Definition dieses Grenzwertes wird in Kapitel 6.6 genauer erläutert.

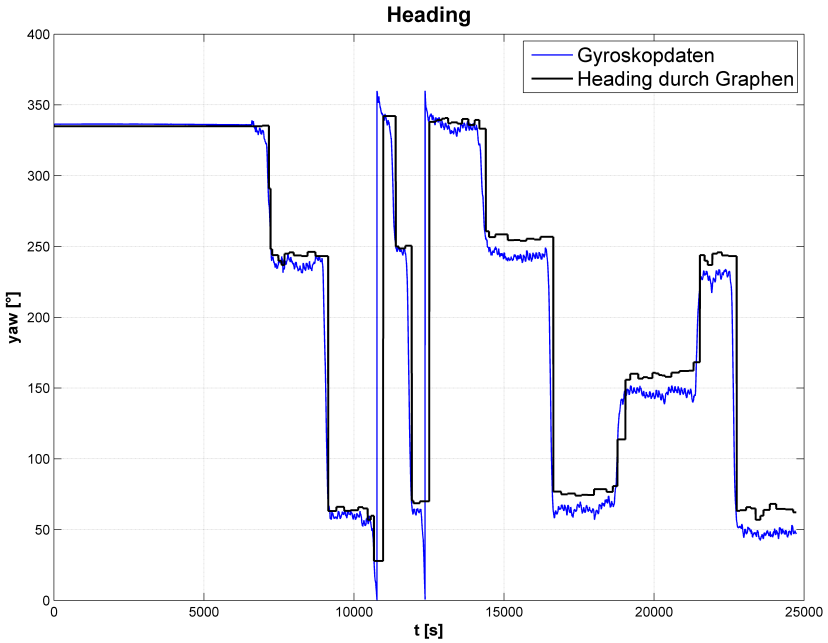


Abbildung 6.14: Unterschied zwischen dem Heading während des Algorithmus und jenem aus den aufkumulierten Drehraten über den gesamten Messzeitraum

Die Richtung α, mit der die Positionen auf dem Graphen (Formel 4.8) berechnet werden können, bezieht sich auf die aktuelle Kantenausrichtung. Dementsprechend garantiert man, dass alle Positionslösungen immer auf dem Knoten- und Kanten-System liegen. Die abschließende Abbildung 6.14 vergleicht das Heading, welches durch die Kantenausrichtungen des Graphen-basierten PDR-Algorithmus gewählt wird mit den aufkumulierten Drehraten über den gesamten Messzeitraum. Dabei wird der Drift in den Gyroskopdaten im Laufe des Messzeitraumes deutlich, da zu Beginn die beiden Linien noch nicht so stark voneinander abweichen als am Ende.

6.6 Fall 1: Geradeausgehen

Wie in Abbildung 6.2 ersichtlich, wählt der Algorithmus zum Zeitpunkt eines Schrittevents, je nach Richtungsabweichung δ von der aktuellen Kante, zwischen drei Zuständen. Falls $\delta < 10°$ ist, trifft der Zustand 1 zu. Dieser Fall beschreibt, dass sich der Fußgänger entlang der aktuellen Kante bewegt, da der Unterschied zwischen der Bewegungsrichtung und der Kantenausrichtung innerhalb des Grenzwertes von 10° liegt. Diese Schranke hat sich bedingt

durch die Graphen in den beiden Testumgebungen anhand 40 gemessenen Trajektorien als geeignet heraus gestellt. Aufgrund der Toleranz von 10° bei der Kantendetektion und bei dem Grenzwert zwischen Abbiegung und Geradeausgehen, können minimal 20° Abbiegungen modelliert werden. Falls der Graph jedoch beispielsweise nur 90° Abbiegungen beinhaltet, kann ein höherer Grenzwert angenommen werden.

Die Funktionsweise des Zustand 1 wird in Abbildung 6.15 im Detail durch ein Flussdiagramm beschrieben.

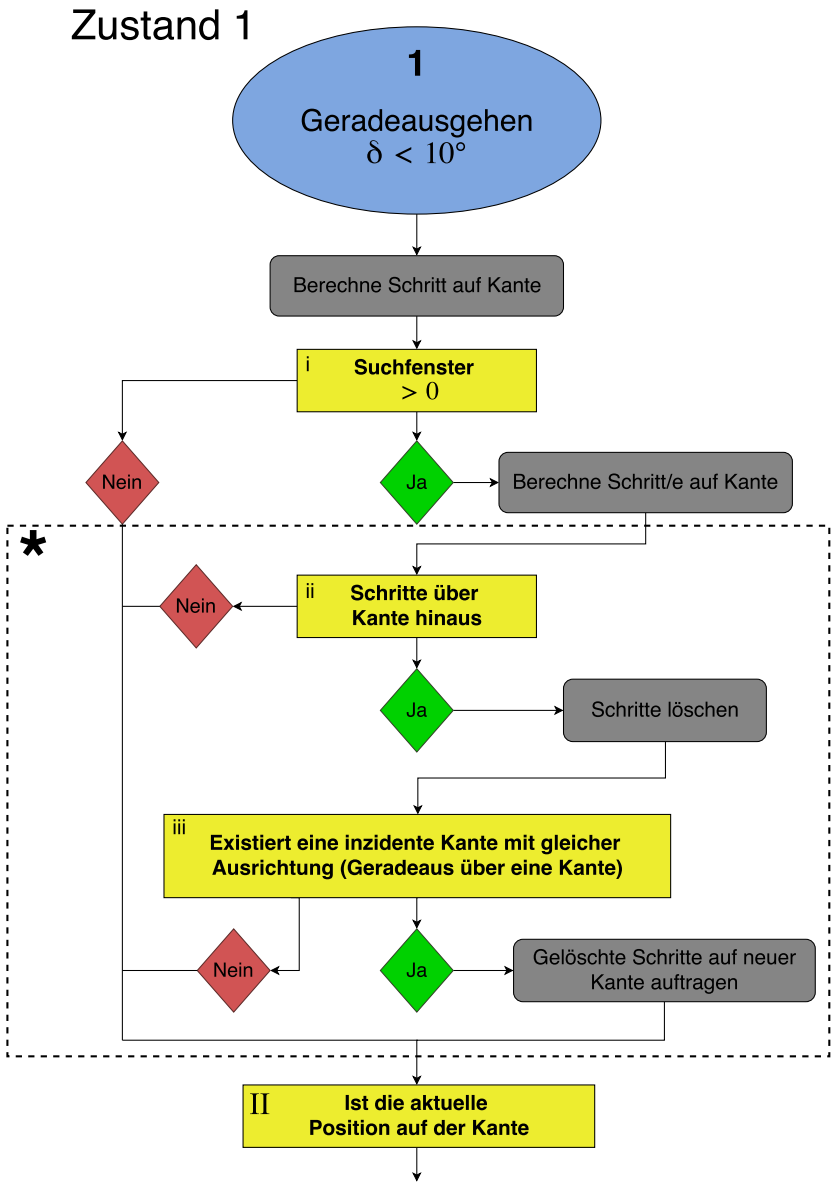


Abbildung 6.15: Flussdiagramm für den Zustand 1 - Geradeausgehen

Zu Beginn wird die nächste Position auf der Kante durch die Formel 4.8 berechnet, beispielsweise letzter rotumrandeter Kreis in Abbildung 6.16. Die Schrittlänge wird dabei, wie in Kapitel 4.1.2 erwähnt, für die Testmessungen mit 0,68cm angenommen und der Richtungswinkel α entspricht der Kantenausrichtung der aktuellen Kante. Anschließend wird überprüft, ob gegebenenfalls zuvor Schritte keiner Kante zugewiesen werden konnten. Beispielsweise wurde beim vorigen Schritt durch die Drehratenmessungen eine Richtungsabweichung δ von über 10° registriert. Dies wird, wie bereits erwähnt, als Abbiegung interpretiert. Allerdings konnte nach dem Ansatz in Zustand 3 (Kapitel 6.8) keine zu dieser Bewegungsrichtung passende Kante gefunden werden. Diesbezüglich wird für diesen Schritt keine Position auf dem Graphen aufgetragen, sondern es werden weitere Schritte abgewartet, um eine eindeutige Entscheidung zu treffen.

Dieses Suchfenster darf allerdings nur fünf Schritte betragen, andernfalls wird auf Zustand 5 verwiesen (Kapitel 6.10). Die Größe dieses Suchfensters entspricht ungefähr der Anzahl an Schritten, die der Fußgänger benötigt um eine Abbiegung zu tätigen. Dieser Grenzwert wurde wiederum durch Testmessungen empirisch bestimmt und für die Aufgabenstellungen des Algorithmus als geeignet eingestuft.

Durch die Restriktion auf fünf Schritte setzt man jedoch auch voraus, dass der User eine Abbiegung binnen dieser Schritte vollzieht. Falls er mehr Schritte benötigt, kann dieses Verhalten durch den Algorithmus nicht als eine einzige Abbiegung eingestuft werden. Die Abbildung 6.16 visualisiert diese Problemstellung. Die übergangenen Schritte werden dabei durch blaue Kreise repräsentiert, die Richtung für die Berechnung dieser Scheinpositionen entspricht der aktuellen Kantenausrichtung plus Richtungsabweichung.

Nach dem zuvor beschriebenen übersprungenen Schritt fällt im darauffolgenden δ wieder unter 10°. Dementsprechend ist die Entscheidung i (Suchfenster > 0) zutreffend. Das Verhalten, dass die Richtungsabweichung zuerst $> 10°$ und anschließend wieder $< 10°$ wird, entspricht der Fortbewegung in Schlangenlinien. Oder es kann beispielsweise daran liegen, dass der Grenzwert von 10° aufgrund von einer abweichenden Tragweise des Smartphones einmal nur minimal überschritten wird (z.B. $10,001°$) und anschließend sich wieder unter 10° einpendelt.

Sinngemäß müssen die übersprungenen Schritte auf der Kante zusätzlich aufgetragen werden. Demzufolge kann jedoch die Situation eintreten (Entscheidung ii), dass durch die zusätzlichen Schritte die Kante, aufgrund ihrer limitierten Länge, überschritten wird. In diesem Fall, müssen die Positionslösungen, die nicht mehr auf der Kante Platz finden, gelöscht werden. Damit diese Schritte in der Trajektorie jedoch nicht vernachlässigt werden,

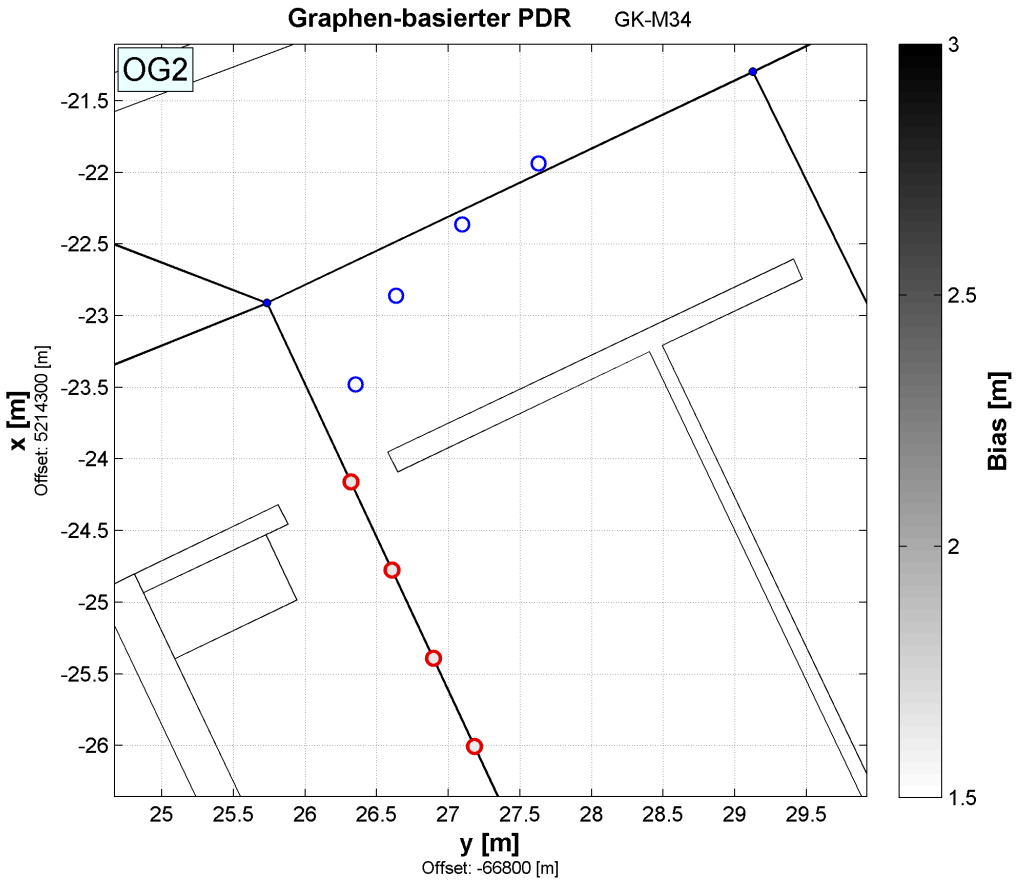


Abbildung 6.16: Suchfenster während der Kantenfindung

untersucht der Algorithmus ob eine inzidente Kante zur aktuellen Kante vorhanden ist, welche die gleiche Kantenausrichtung besitzt (Entscheidung iii). Für die Auswahl von neuen Kanten wird ebenfalls der Grenzwert von 10° benützt. Diesbezüglich lässt man eine Toleranz bei der Kantenfindung von ±10° zu.

Dies bedeutet für den Zustand 1, dass zur aktuellen Kante eine inzidente Kante existiert die ein Geradeausgehen über einen Knoten ermöglicht. Falls eine derartige Konstellation vorliegt, werden die zuvor gelöschten Positionen auf die neue Kante projiziert. Die Abbildung 6.17 visualisiert diese Funktionsweise des Algorithmus. Anschließend ist dieser Zustand 1 abgeschlossen und der nächste Schritt im Algorithmus ist Entscheidung II aus dem Übersichtsflussdiagramm 6.2. Abschließend ist noch anzumerken, dass der mit dem Stern (*) gekennzeichnete gestrichelte Block des Diagrammes 6.15 dieses Zustandes sich in den Flussdiagrammen 6.21 und 6.27 ebenfalls wiederfindet. Dementsprechend wird bei diesen zwei Diagrammen auf diese Erklärung verwiesen.

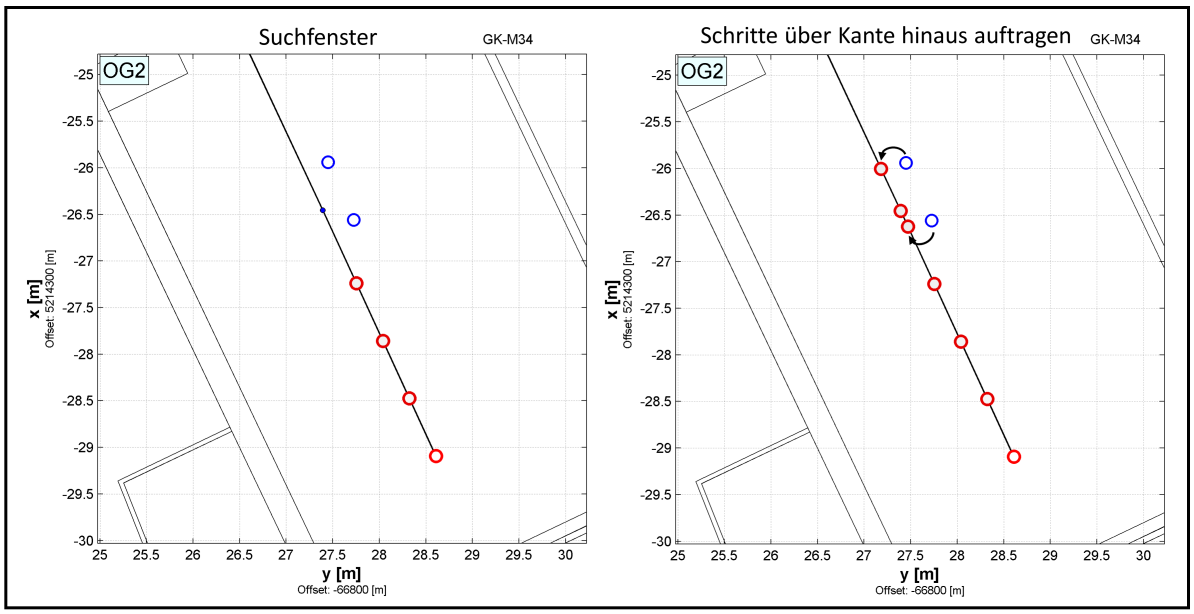


Abbildung 6.17: Mehrere Schritte beim Geradeausgehen über eine Kante hinaus auftragen

6.7 Fall 2: 180° Drehung

Ein weiteres Verhalten des Fußgängers, welches durch einen separaten Zustand abgedeckt werden muss, ist eine 180° Drehung. Die Funktionsweise dieses Zustandes bezieht sich auf das Flussdiagramm aus Abbildung 6.18. Dieses Umkehren der Bewegungsrichtung erkennt der Algorithmus, wenn die aktuelle Richtungsabweichung δ eine Schranke von 145° überschreitet.

Der Grenzwert dient dazu, dass der User für diese Art von "Abbiegung" mehr als fünf Schritte benötigen darf, um eine tatsächliche 180° Drehung zu vollziehen. Als Toleranz gilt dabei wieder ±10°. Falls die Richtungsabweichung δ zwar über 145° ist, jedoch die Entscheidung iv aus Abbildung 6.18 nicht zutrifft, wird ein weiterer Schritt abgewartet. Der Counter für das Suchfenster erhöht sich dabei nicht, dementsprechend kann die Grenze von fünf Schritten ausgedehnt werden.

Setzt man voraus, dass der User die Kehrtwendung binnen fünf Schritten schafft, kann der Grenzwert von 145° auf 170° erhöht werden. Benötigt der User jedoch beispielsweise sechs Schritte für die Wende, würde nach dem Ansatz von Zustand 5 aus Kapitel 6.10 eine Abbiegung für einen Bruchteil der 180° Drehung gesucht und die 180° Wende nicht erkannt werden. Ebenfalls würde bei einem Grenzwert von 170° angenommen, dass sich der Fußgänger für eine Wende mindestens 170° dreht.

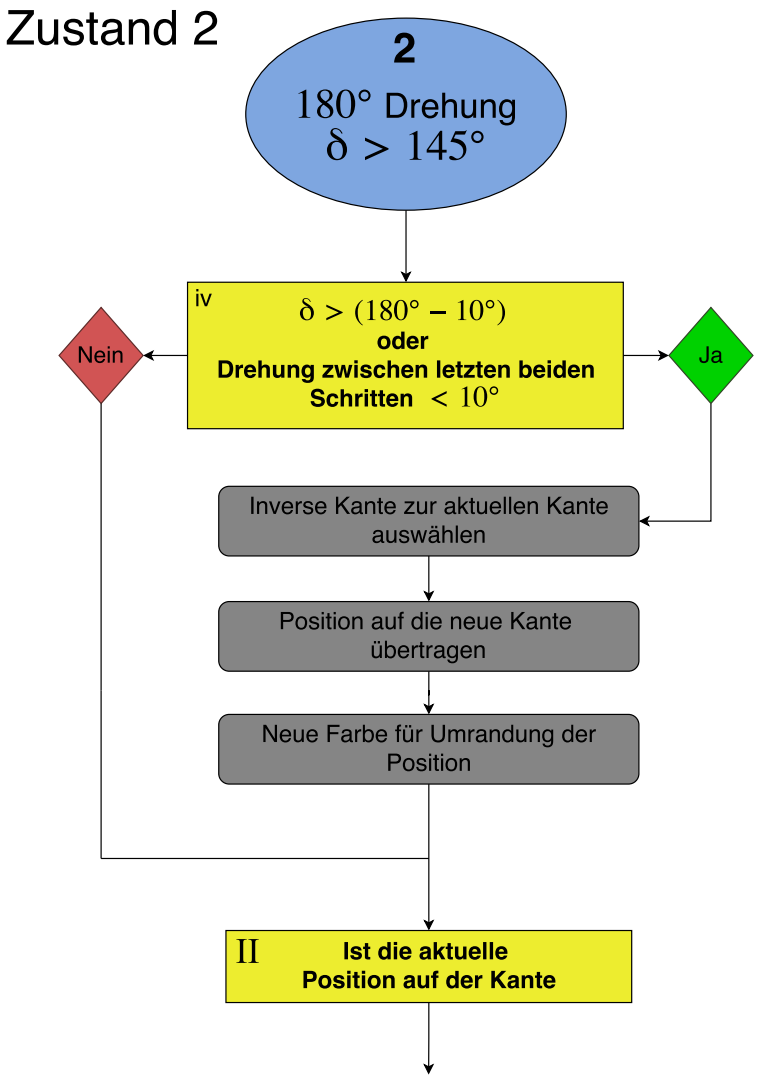


Abbildung 6.18: Flussdiagramm für den Zustand 2 - 180° Drehung

Eine nochmalige Überprüfung der Richtungsabweichung δ in Entscheidung iv ist deshalb notwendig, da die Wende erst vollzogen werden darf, wenn die 180° Drehung abgeschlossen ist. Dies ist der Fall, wenn die Bewegungsrichtung, abzüglich der Toleranz von 10°, der inversen Kantenausrichtung entspricht (170° (180° − 10°)). Dabei ist anzumerken, dass die inverse Kante die gleichen Knoten wie die aktuelle Kante besitzt, jedoch ist Start- und Endknoten vertauscht. Dadurch ist diese Kante in die entgegengesetzte Richtung gerichtet. Anderseits kann die 180° Drehung auch abgeschlossen sein, wenn sich die Richtungsabweichung zwischen dem aktuellen Schritt und dem letzten Schritt nur innerhalb des Grenzwertes von 10° verändert hat. Die letzte Bedingung deckt denn Fall ab, dass der Fußgänger die Wende nicht sauber im Sinne einer 180° Drehung vollzieht, sondern sich

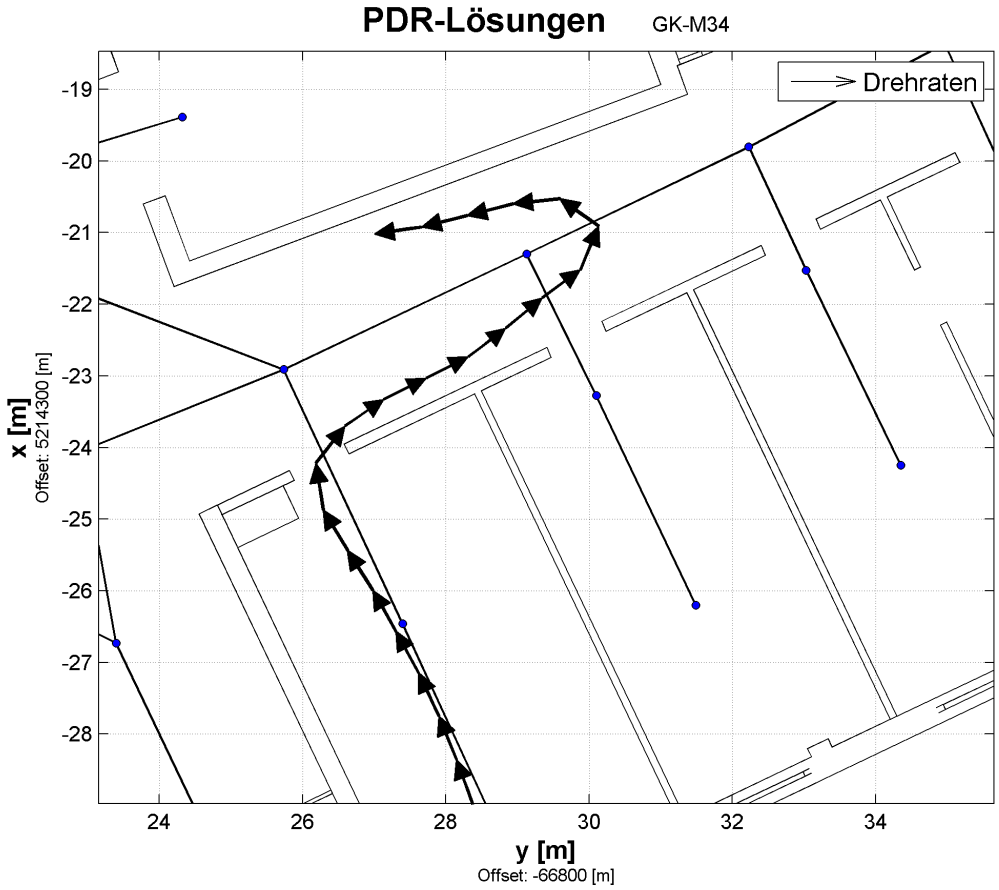


Abbildung 6.19: Wende, jedoch keine vollständige 180° Drehung vollzogen

beispielsweise nur 165° dreht. Diesbezüglich muss der Zustand 2 schon zu Beginn bei 145° erkannt werden. Die Abbildung 6.19 illustriert den Verlauf einer derartigen 180° Wende.

Falls eine der beiden Bedingungen in Entscheidung iv zutrifft, wird die momentane Kante durch die inverse Kante ersetzt und die Position auf dieser berechnet. Diesbezüglich wird allerdings kein Schritt getätigt, sondern die letzte Position vor der Drehung wird als Startpunkt nach der Wende verwendet. Die einzige Änderung ist somit nur die neue Kante. Zusätzlich wird allerdings auch für eine bessere Übersicht in den Plots die Farbe der Umrandung in den Positionslösungen geändert. Diesen Vorgang einer vollzogenen 180° Drehung visualisiert die Abbildung 6.20. Abschließend wird wiederum nach diesem Zustand auf die Entscheidung II verwiesen.

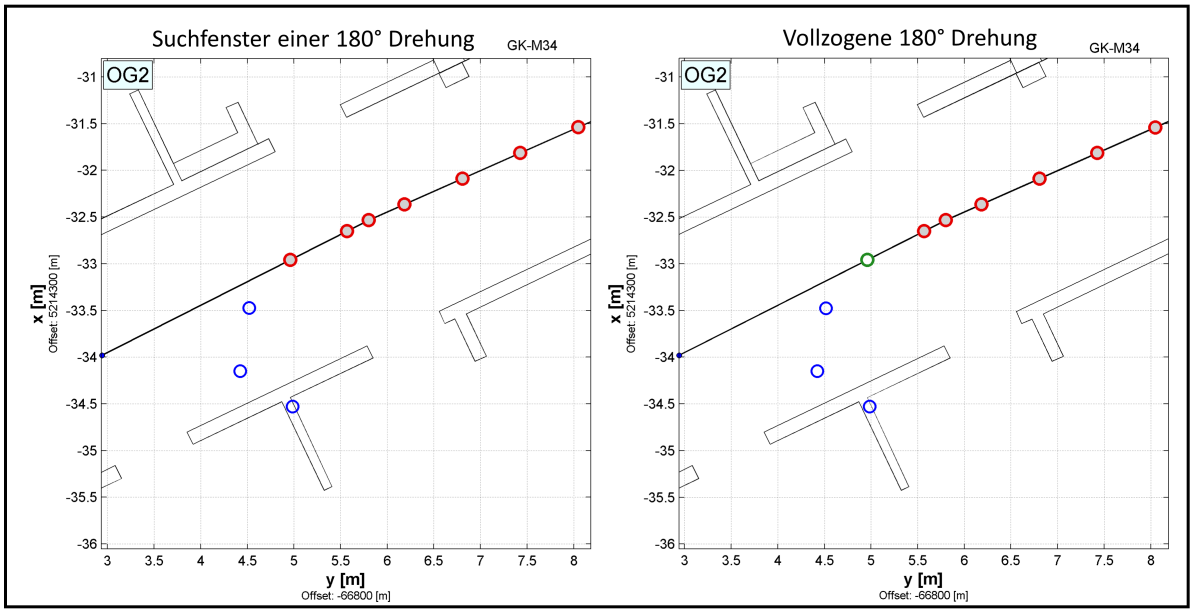


Abbildung 6.20: Visualisierung der 180° Drehung, die neue Position wird durch eine grüne Umrandung gekennzeichnet

6.8 Fall 3: Abbiegung

Der dritte Zustand, welcher eintreten kann wenn ein Schritt detektiert wird, ist, wie in Flussdiagramm 6.2 ersichtlich, eine Abbiegung. Dieser Fall trifft dann zu, wenn sich der Fußgänger über 10° von der Kante wegbewegt. Anschließend wird bei dem nächstgelegenen Knoten eine Kante gesucht, die zu dieser Bewegungsrichtung passt. Diese Funktionsweise des Algorithmus wird im Detail in Flussdiagramm 6.21 visualisiert und nachfolgend beschrieben:

Zu Beginn muss in diesem Zustand 3 in Entscheidung v überprüft werden ob in vorigen Schritten eine 180° Drehung vollzogen wurde. Falls dies zutrifft, beinhaltet die gemessene Richtung δ in diesem Schritt noch einen Bruchteil der 180° Wende. Dies kann durch die Tatsache hervorgerufen werden, dass aufgrund der beschriebenen Toleranz in Kapitel 6.7 bereits bei 170° eine 180° Drehung getätigt wird. Wenn allerdings der Fußgänger für das Umkehren der Bewegungsrichtung beispielsweise sich 190° dreht, sind nach der Wende noch Anteile dieser 180° vorhanden und werden im Algorithmus als separate Abbiegung interpretiert. In diesem Fall wird die detektierte Abbiegung ignoriert und ein Schritt entlang der Kante aufgetragen. Anschließend wird der Zustand 3 beendet und auf Entscheidung II verwiesen.

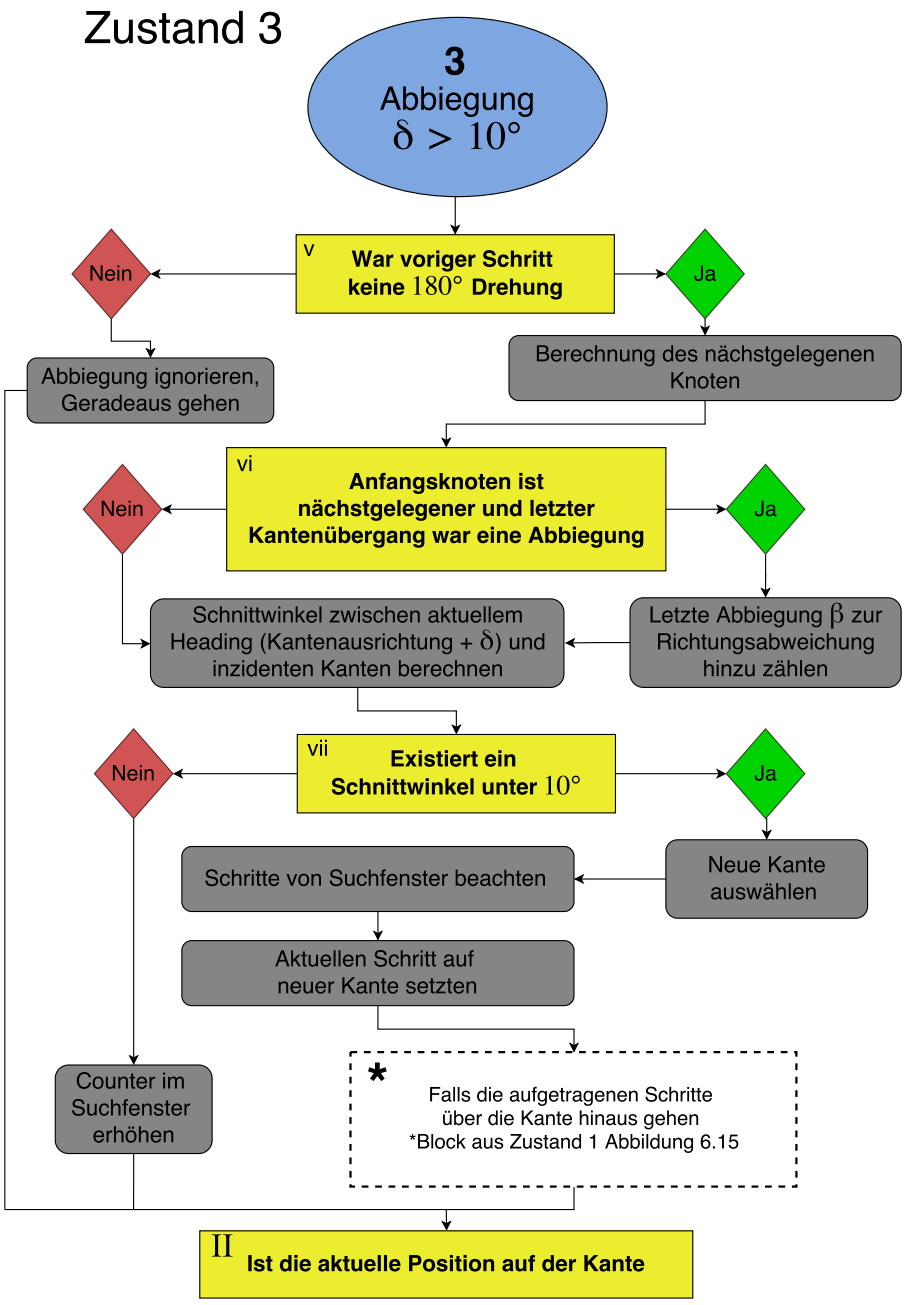


Abbildung 6.21: Flussdiagramm für den Zustand 3 - Abbiegung

Ist Entscheidung v jedoch positiv wird die Richtungsabweichung δ als Abbiegung interpretiert. Dazu berechnet der Algorithmus zuerst welcher Knoten der aktuellen Kante der nächstgelegene zur momentanen Position ist. Diesbezüglich kann die Berechnung der kürzeren Distanz auch gewichtet erfolgen. Nach mehreren Testmessungen (40) wurden für die in dieser Masterarbeit entworfenen Testumgebungen dabei die besten Ergebnisse

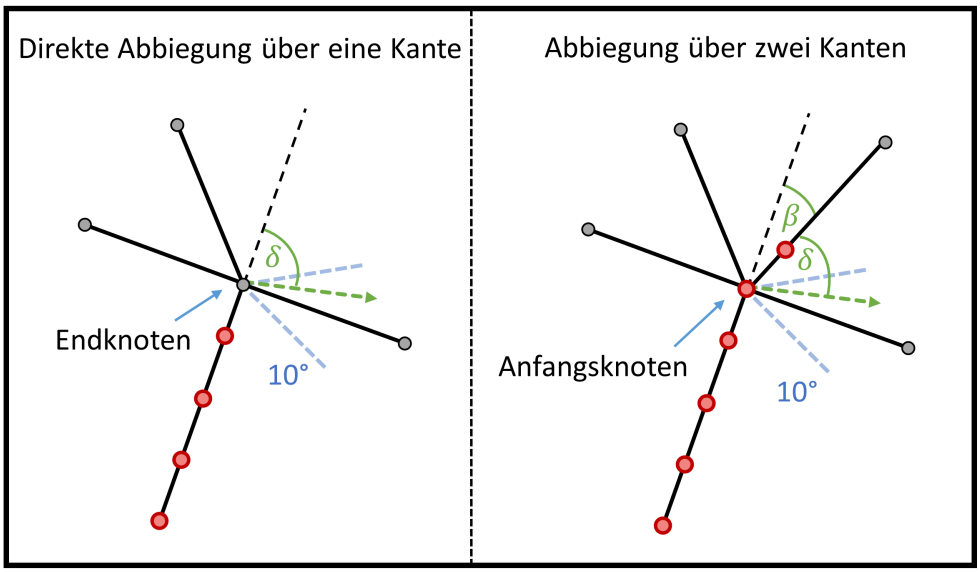


Abbildung 6.22: Szenarien durch Entscheidung vi: Abbiegung wird direkt gefunden oder Abbiegung bei der zuerst eine Kante gewählt wird, die einem Bruchteil der gesamten Abbiegung entspricht

erzielt, wenn eine Gewichtung von $1,5$ Schritten zu Gunsten des Endknotens implementiert wurde. Dementsprechend wird bei der Distanz zu dem Endkonten die Länge von $1,5$m abgezogen. Anschließend wird diese reduzierte Distanz und jene zu dem Anfangsknoten miteinander verglichen.

Diese Verbesserung durch die Gewichtung kann dadurch begründet werden, dass bei der natürlichen Bewegung des Fußgängers mehrere Schritte von Nöten sind um eine Abbiegung zu detektieren (Suchfenster). Dementsprechend bewegt man sich in dieser Zeit von dem Anfangsknoten weg und näher zum Endknoten hin. Durch die Gewichtung kann dieses Verhalten angenähert werden.

Die Knotenauswahl legt den Kreuzungsbereich innerhalb des Graphen fest, bei der eine zur Abbiegung passende inzidente Kante gefunden werden soll. Eine neue Kante wird dann als geeignete Kante eingestuft, wenn der Schnittwinkel zwischen der Kantenausrichtung des Kantenkandidaten und dem aktuellen Heading (Kantenausrichtung der aktuellen Kante plus Richtungsabweichung δ) innerhalb der Toleranz von $10°$ liegt, siehe linkes Szenario in Abbildung 6.22. Falls jedoch die Entscheidung vi zutrifft, sodass der nächstgelegene Knoten der Anfangsknoten ist und dass der letzte Kantenwechsel aufgrund einer Abbiegung zustande gekommen ist, muss für die Schnittwinkelberechnung zu der aktuellen Richtungsabweichung δ der Winkel β hinzugezählt werden, siehe rechtes Szenario in Abbildung

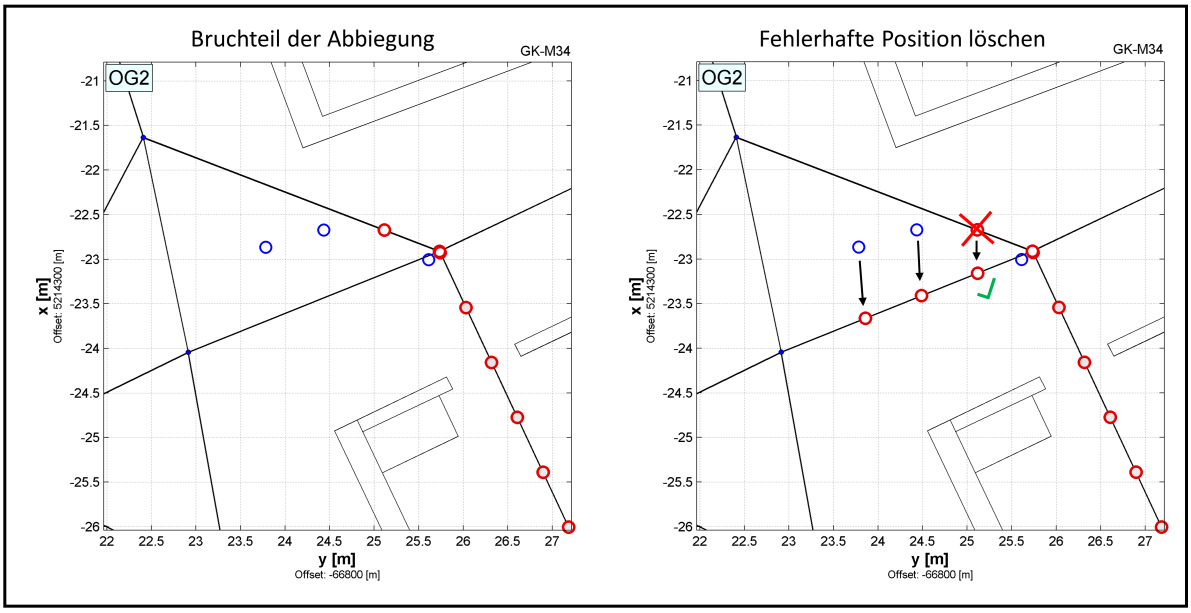


Abbildung 6.23: Kanten und Positionskorrektur aufgrund einer fälschlicherweise detektierten Kante die einem Bruchteil der gesamten Abbiegung entspricht

6.22. Dieser Winkel β beschreibt die Änderung der Kantenausrichtung welche durch den Kantenwechsel bei der letzten Abbiegung entsteht.

Diese Konstellation entsteht dadurch, dass beispielsweise bei der Kreuzung im rechten Bild der Abbildung 6.22 eine 30° und eine 90° Abbiegung vorhanden sind. Erreicht der Fußgänger die angestrebte 90° Drehung erst durch mehrere Schritte, so wird zu Beginn zuerst die Kante der 30° Abbiegung gewählt, weil sich der User beispielsweise bei dem ersten Schritt genau um diesen Bruchteil der 90° Abbiegung gedreht hat. In den nächsten Schritten bemerkt der Algorithmus jedoch die weiteren Anteile der 90° Drehung. Diesbezüglich wird die 30° Abbiegung als fehlerhaft interpretiert und gelöscht. Zudem wird die aktuelle Richtungsabweichung von $\delta = 60°$ durch die 30° Abbiegung β zu der tatsächlichen 90° Drehung komplettiert ($90° = \beta + \delta$). Abbildung 6.23 visualisiert diese Problemstellung.

Die Schnittwinkelberechnungen zwischen den Kantenkandidaten (inzidente Kanten in dem nächstgelegenen Knoten) können aufgrund von 360° Sprüngen nicht einfach über eine Differenzbildung der Winkel erfolgen. Beispielsweise würde bei Richtungen von 358° und 1° durch eine Subtraktion nicht auf die gesuchte Größe von 3° sondern auf 357° geschlossen werden. Um diese Problematik zu umgehen, werden die zwei Richtungen in ihre entsprechenden Einheitsvektoren x_1, x_2 umgewandelt. Die schlussendliche Berechnung der Schnittwinkel γ_1 und γ_2 aus Abbildung 6.24 zwischen diesen zwei Vektoren basiert

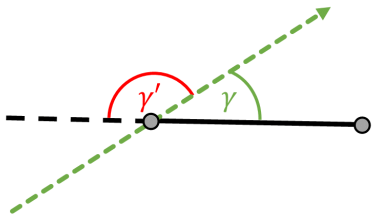


Abbildung 6.24: Schnittwinkel zwischen der Kantenausrichtung eines Kantenkandidaten und dem aktuellen Heading

auf Formel 6.9. Abschließend muss noch der richtige Schnittwinkel in Bezug auf die Ausrichtungen der zwei Geraden ausgewählt werden.

$$\gamma = \arccos\left(\frac{\mathbf{x_1 x_2}}{|\mathbf{x_1}||\mathbf{x_2}|}\right) \tag{6.9}$$

Als nächsten Schritt untersucht der Algorithmus, ob einer der berechneten Schnittwinkel zwischen den Kantenkandidaten und dem aktuellen Heading unter 10° liegt (Entscheidung vii). Falls nicht, kann der Algorithmus zur aktuellen Bewegungsrichtung keine geeignete Kante zuordnen und es wird diesbezüglich der Counter im Suchfenster erhöht, um weitere Schritte abzuwarten.

Existiert jedoch bei der nächstgelegenen Kreuzung eine Kante welche der momentanen Richtung des Fußgängers entspricht, wird diese als neue Kante gewählt. Anschließend werden, wie in Abbildung 6.25 ersichtlich, mögliche Schritte des Suchfensters auf der alten Kante und auf der neuen Kante aufgetragen. Würden allerdings auf der alten Kante mehr Schritte bis zum nächsten Knoten Platz finden als tatsächliche Schritte getätigt wurden, füllt der Algorithmus durch "virtuelle" Schritte die alte Kante auf und setzt einen Schritt auf der neuen Kante. In der Abbildung 6.25 ist ebenfalls der Nachteil der Generalisierung des Graphen ersichtlich. Dadurch, dass der Fußgänger im Verhältnis zu dem Graphen die Kurve enger schneidet, sind die Positionen auf dem Graphen ungefähr einen Schritt hinter der "tatsächlichen" Position.

Die nächste Phase des Algorithmus ist, wie in Flussdiagrammes 6.21 ersichtlich, der mit dem Stern (*) gekennzeichnete gestrichelte Block. Dieser Block verweist auf die Ausführungen des Flussdiagrammes 6.15 des Kapitels 6.6 und tritt dann ein, wenn die aufzutragenden Schritte über die neu detektierte Kante hinausgehen. Diesbezüglich wird wiederum überprüft, ob es nach der neuen Kante eine inzidente Kante gibt, die ein Geradeausgehen ermöglicht. Die Funktionsweise ist dabei dieselbe wie in Zustand 1 und

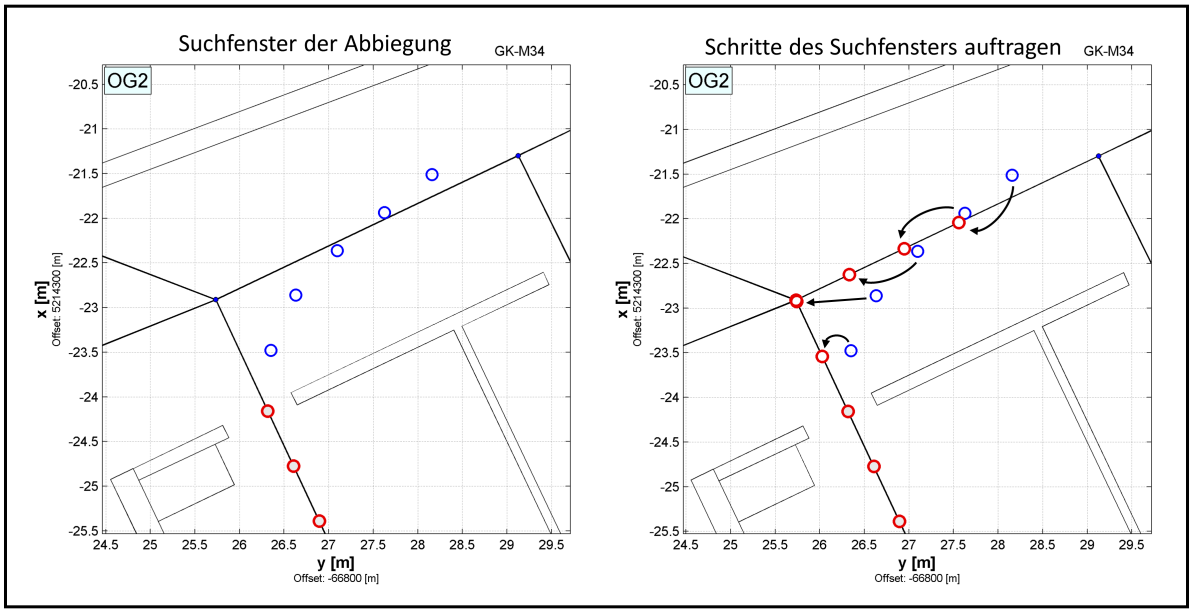


Abbildung 6.25: Schritte welche für die Beschreibung des Abbiegeverhaltens benötigt werden auf der alten und der neuen Kante auftragen

wird in Abbildung 6.26 visualisiert. Abgeschlossen wird der Zustand 3 ident zu den beiden vorherigen Zuständen mit dem Verweis auf die Entscheidung II.

6.9 Fall 4: Geradlinige Bewegung über den Endknoten

Nach den drei beschriebenen Zuständen überprüft der Graphen-basierte PDR-Algorithmus, wie in dem Übersichtsflussdiagramm 6.2 ersichtlich, in Entscheidung II ob die aktuelle Position auf der Kante ist. Dies ist nicht der Fall, wenn in Zustand 1 ein Schritt entlang der Kante berechnet wird, dieser jedoch aufgrund der limitierten Länge der Kante über den Endknoten hinausgeht.

Diese Untersuchung basiert auf der vektoriellen 2D Lagebeziehungsberechnung zwischen einem Punkt und einer Geraden aus der analytischen Geometrie. Durch die Koordinaten des Anfangsknoten (OA) und des Endknoten (OB) der Kante wird eine Gerade in Parameterform aufgestellt und mit den Koordinaten der aktuellen Position (OP) gleichgesetzt:

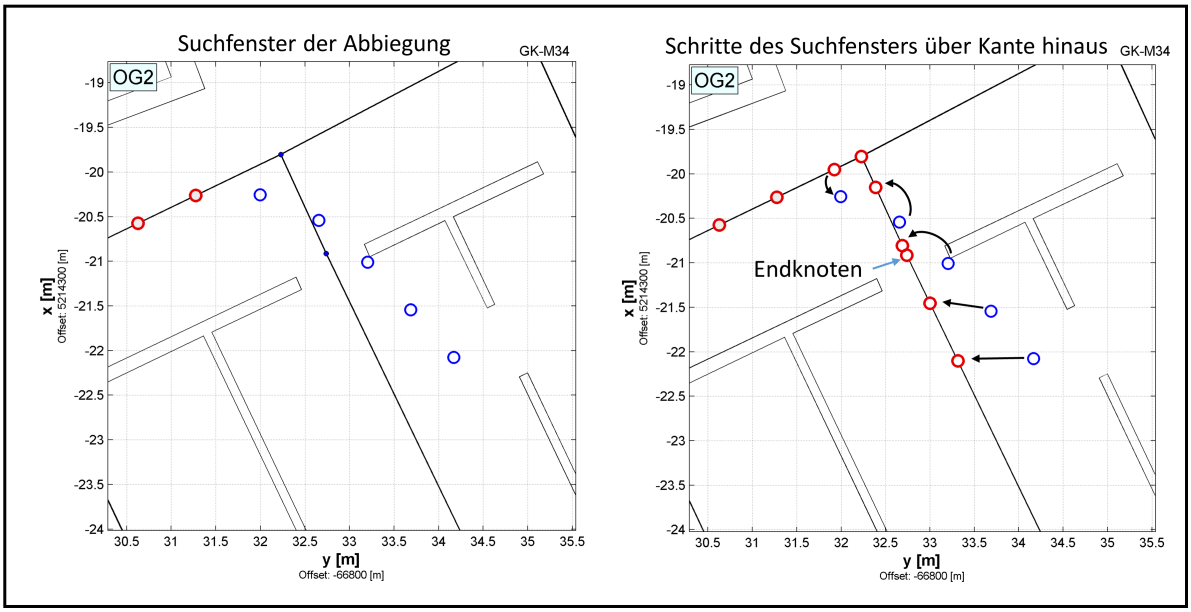


Abbildung 6.26: Aufgetragene Schritte gehen über die detektierte Kante der Abbiegung hinaus

$$OP = OA + \lambda(OB - OA) \tag{6.10}$$

Formt man die Vektorgleichung 6.10 nach λ um und werden die x und y Zeile separat betrachtet, erhält man für die erste und zweite Zeile einen numerischen Wert für λ. Sind die Werte dieselben, liegt der Punkt auf dieser Geraden, andernfalls nicht. Wird bei identen Lambdas $\lambda = 0$ in die Vektorgleichung eingesetzt, würde der Punkt OP dem Anfangsknoten OA entsprechen, bei $\lambda = 1$ repräsentiert OP den Endkonten OB. Diesbezüglich erhält man alle Punkte auf der Kante/Strecke, wenn λ zwischen null und eins liegt. Falls die λ Werte ident sind allerdings die Grenzwerte von null und eins übersteigen, ist der Punkt zwar auf der Geraden jedoch nicht auf der Strecke.

Liegt die momentane Position aufgrund des Überschreitens des Endknotens durch eine geradlinige Bewegung nicht auf der aktuellen Kante wird der Zustand 4 aufgerufen. Dementsprechend soll eine Kante gefunden werden, welch die aktuelle Kantenausrichtung fortsetzt und somit dem User eine geradlinige Bewegung über den Endknoten ermöglicht. Dieser Fall wird in Abbildung 6.3 in Zustand 4 und im Flussdiagramm 6.27 visualisiert.

Zusätzlich wird in diesem Zustand noch der Fall abgedeckt, dass ein Ende eines Ganges erreicht wird, jedoch die Messdaten eine weitere geradlinige Bewegung feststellen und erst nach zusätzlichen Schritten eine Abbiegung registrieren. Diese Situation wird in Abbildung

6.28 visualisiert und trifft dann ein, wenn beispielsweise die Schrittlänge zu groß gewählt wird oder durch die Generalisierung des Graphen.

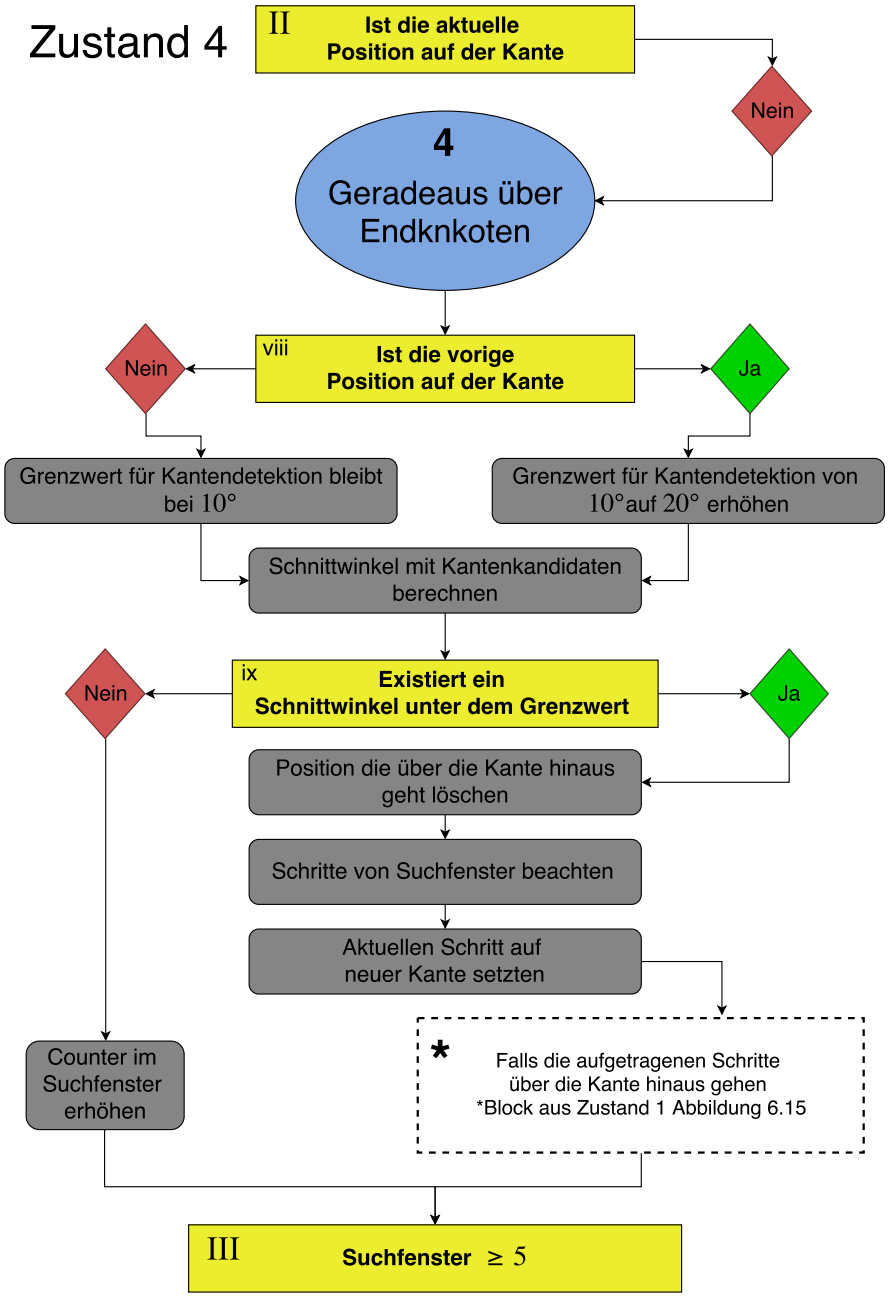


Abbildung 6.27: Flussdiagramm für den Zustand 4 - Geradlinige Bewegung über den Endknoten

Die Unterscheidung dieser zwei Fälle erfolgt durch Entscheidung viii. Nur wenn die vorherige Position auf der aktuellen Kante noch Platz gefunden hat, tritt der Fall ein, dass

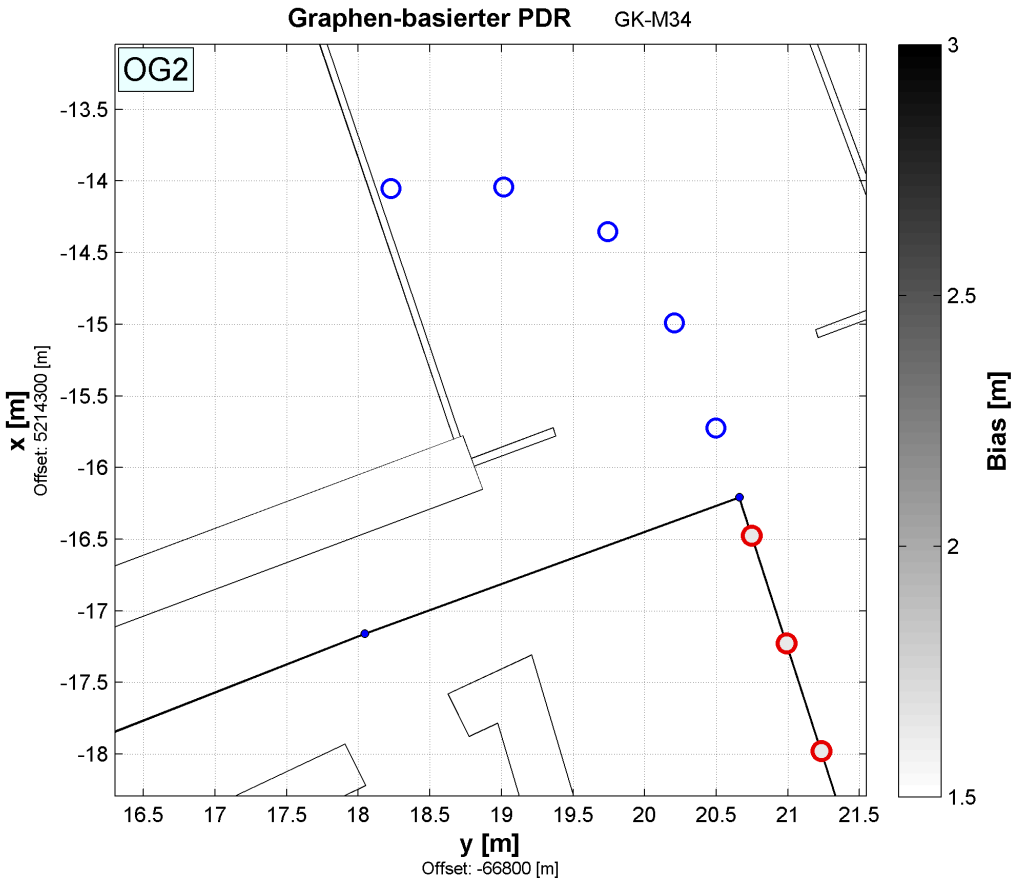


Abbildung 6.28: Abbiegung nach dem Ende eines Ganges/Kante

ein Geradeausgehen über einen Endknoten stattfindet. Diesbezüglich wird der Grenzwert für die Auswahl möglicher Kantenkandidaten von 10° auf 20° erhöht. Dies ist darauf zurückzuführen, dass der Graph nicht immer sauber im Sinne einer geradlinigen Bewegung erstellt werden kann. Abbildung 6.29 visualisiert diesen Fall. In dem rot markierten Bereich kann aufgrund der Gebäudestruktur ein Geradeausgehen nicht exakt unter 10° modelliert werden. Der Fußgänger geht allerdings nicht entlang dieser Kantenstruktur, sondern vollzieht eine geradlinige Bewegung. Aus diesem Grund kann durch den erhöhten Grenzwert diese unsaubere Graphendefinitionen kompensiert werden.

Falls die Entscheidung viii nicht zutrifft, tritt der Fall aus Abbildung 6.28 ein. Diesbezüglich wird der Grenzwert für die Kantendetektion nicht erhöht. Die Schnittwinkel bei der Auswahl des bestmöglichen Kantenkandidaten werden anhand des gleichen Funktionsprinzips wie in Zustand 3 berechnet. Anschließend trifft je nach gesetzten Grenzwert die Entscheidung ix zu.

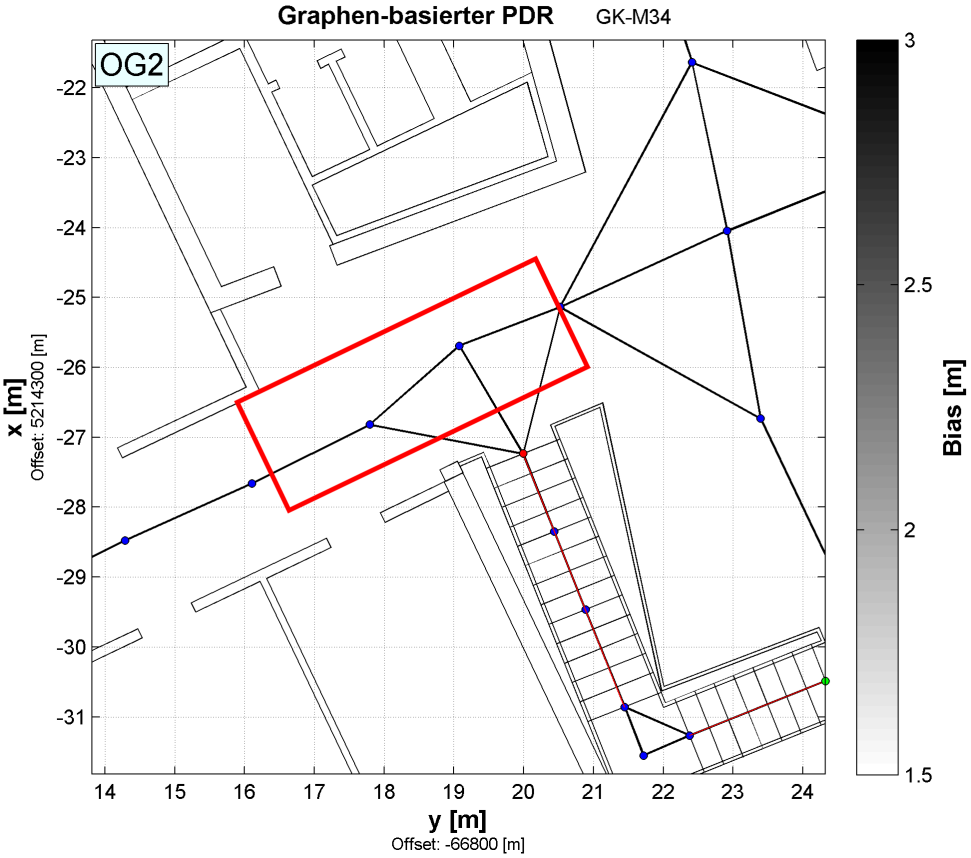


Abbildung 6.29: Ungenaue Konstruktion des Graphen für geradlinige Bewegungen über mehrere Kanten

Kann keine passende Kante zum aktuellen Heading gefunden werden, erhöht sich der Counter des Suchfensters und weitere Schritte werden für eine eindeutige Kantendetektion abgewartet. Existiert ein Schnittwinkel der unter dem Grenzwert liegt, wird im Falle einer geradlinigen Bewegung die Position die über die Kante hinausgeht gelöscht. Anschließend wird diese gelöschte Position auf der neuen Kante aufgetragen, sodass ein Geradeausgehen über einen Knoten ermöglicht wird. Die Abbildung 6.30 visualisiert diese Funktionsweise des Algorithmus.

Detektiert der Algorithmus anhand einer Abbiegung eine neue Kante (Abbildung 6.28), werden zusätzlich die Schritte des Suchfenster aufgetragen. Dadurch kann wiederum, wie bereits in Zustand 1 und Zustand 3 durch den mit dem Stern (*) markierten gestrichelten Block, der Fall eintreten, dass die aufgetragenen Schritte über die neue Kante hinaus gehen. Dementsprechend wird der gleiche Ablauf getätigt wie in den zwei erwähnten Zuständen. Abgeschlossen wird der Zustand 4 auf den Verweis auf die Entscheidung III.

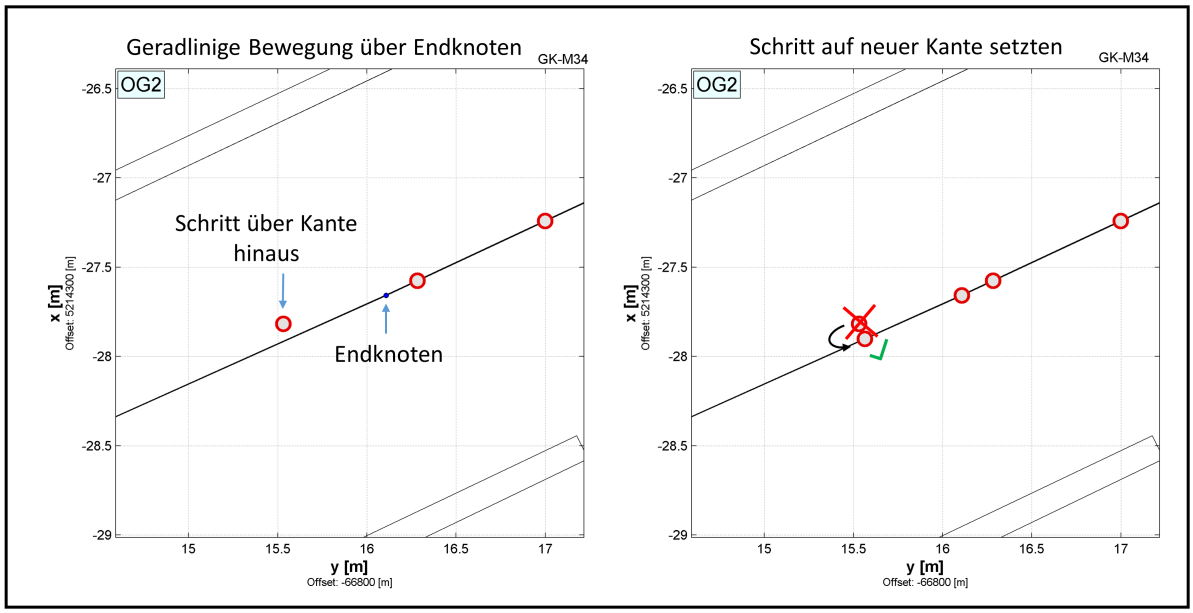


Abbildung 6.30: Kantendetektion der neuen Kante die einer geradlinigen Bewegung entspricht

6.10 Fall 5: Globale/Lokale Suche

Anhand der Entscheidungen aus den Zuständen 3 und 4 kann es, wie bereits in Kapitel 6.1 erwähnt, eintreten, dass nicht binnen fünf Schritten ein geeigneter Kantenkandidat gefunden werden konnte. Demzufolge existiert bei der nächstgelegenen Kreuzung keine Kante, die zu der gemessenen Bewegungsrichtung innerhalb der Toleranzgrenze liegt. Für diesen Fall wurden zwei unterschiedliche Herangehensweisen in dieser Arbeit entwickelt, wobei sich anhand der 40 Testmessungen das Verfahren aus Kapitel 6.10.1 als geeignetere Lösung herausgestellt hat und als Standardverfahren für den Graphen-basierten PDR-Algorithmus implementiert wurde.

6.10.1 Auswahlkriterium Richtungsabweichung δ und Distanz

Das Standardverfahren für den Zustand 5 sucht innerhalb eines statistisch definierten Suchradius oder im gesamten Graphen nach einem geeigneten Kantenkandidaten. Die Auswahl einer passenden Kante erfolgt über den Vergleich des Headings und den Kantenausrichtungen der Kandidaten. Für den Fall, dass mehrere Kanten in Frage kommen, wird die Kante, welche zu der Ausgangsposition des Suchvorganges den nächstgelegenen

Anfangsknoten besitzt, ausgewählt. Die Umsetzung innerhalb des Algorithmus ist für diese Aufgabenstellung in Flussdiagramm 6.31 visualisiert.

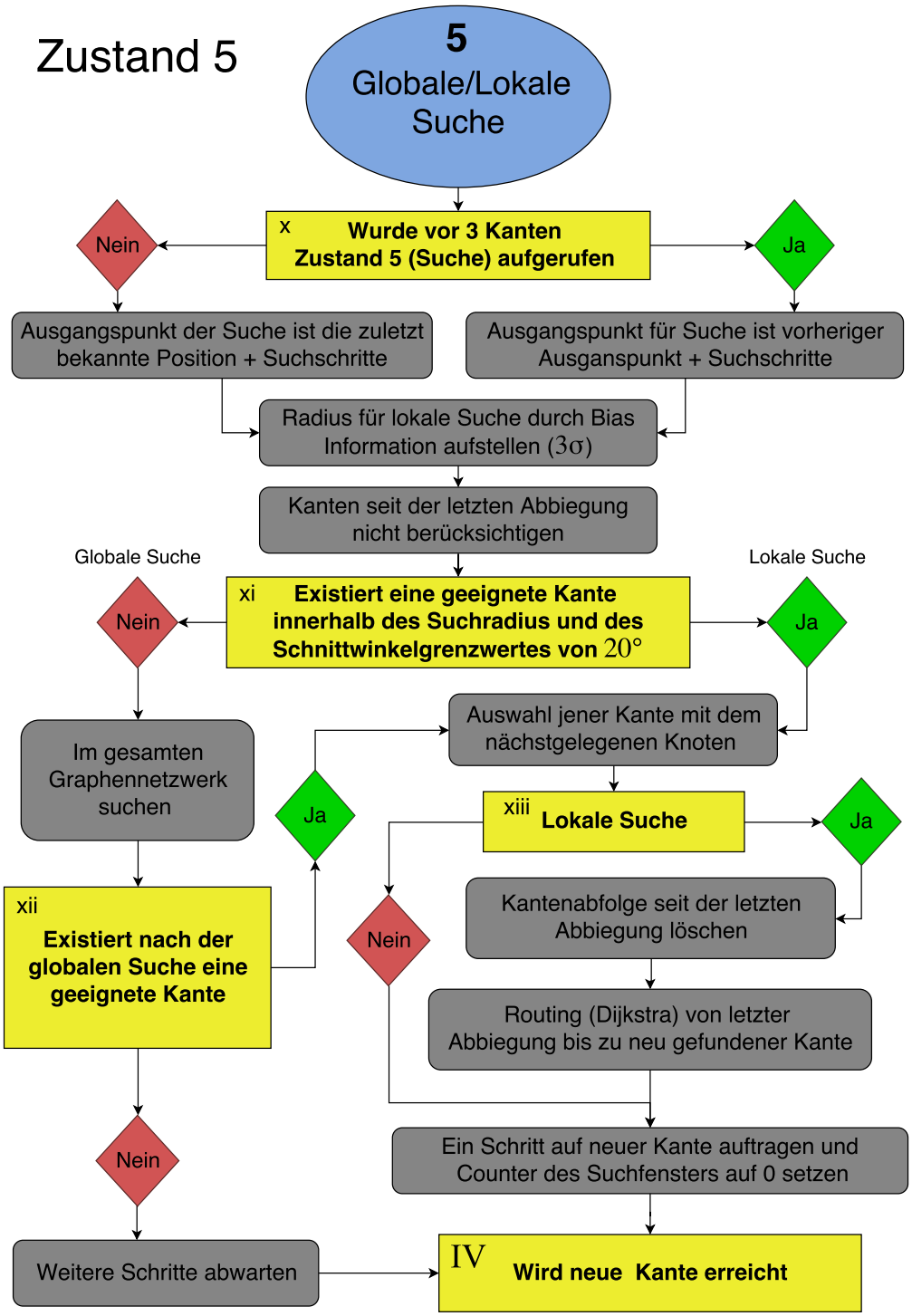


Abbildung 6.31: Flussdiagramm für den Zustand 5 - Globale/Lokale Suche

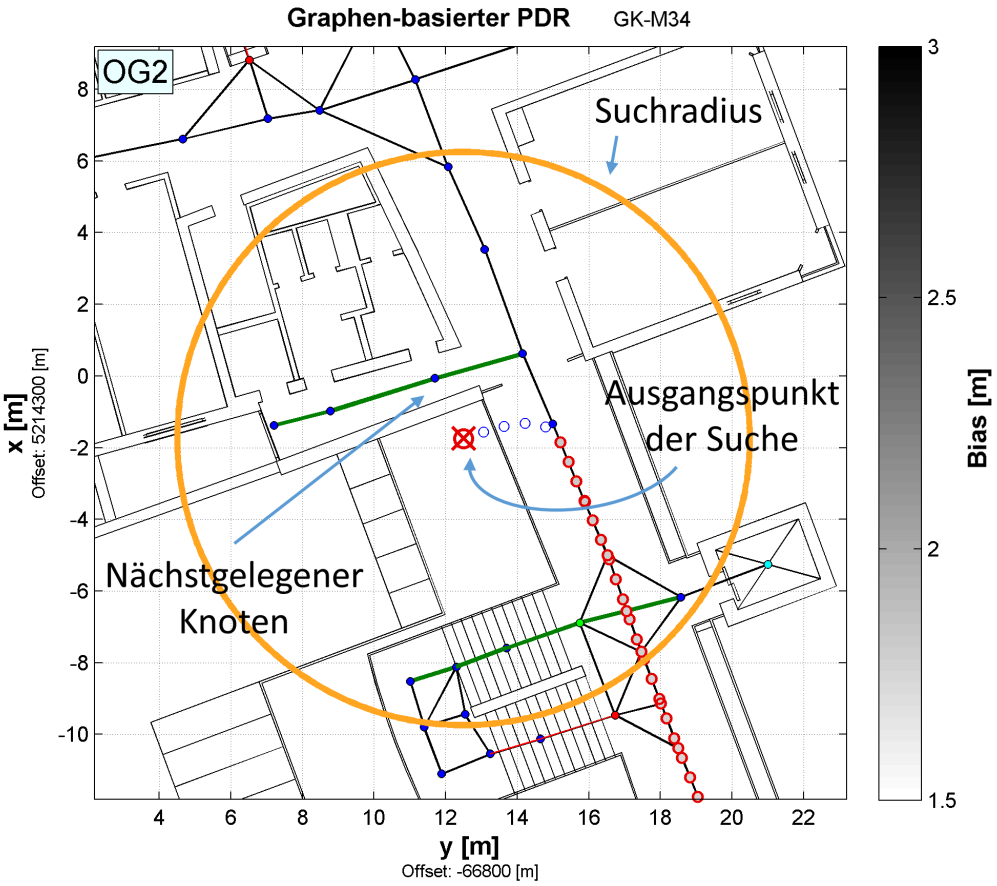


Abbildung 6.32: Zustand 5 - lokale Suche mit Suchradius nach geeigneten Kantenkandidaten, grüne Kanten haben eine zum Heading passende Kantenausrichtung

Zu Beginn muss der Ausgangspunkt für die Suche nach geeigneten Kanten festgelegt werden. Im simplen Fall ist die Bedingung in Entscheidung x nicht zutreffend und der Ausgangspunkt setzt sich, wie in Abbildung 6.32 ersichtlich, aus der zuletzt bekannten Position auf dem Graphen plus den Positionen aus den Suchschritten zusammen. Wenn allerdings der Zustand 5 bereits vor drei Kanten ebenfalls aufgerufen wurde, kann davon ausgegangen werden, dass diese Suche nicht das gewünschte Ergebnis geliefert hat. Dies kann darauf zurückzuführen sein, dass beispielsweise bei langgezogenen Kurven schon bei einem Bruchteil der gesamten Drehung die Suche gestartet wird. Die komplette Abbiegung würde jedoch erst bei der darauffolgenden Suche vollendet sein. In diesem Fall setzt sich der Ausgangspunkt der zweiten Suche aus dem Ausgangspunkt der vorherigen Suche plus den neuen Suchschritten zusammen, siehe Abbildung 7.9 aus Kapitel 7.2.

Anschließend berechnet der Algorithmus anhand des in Kapitel 6.4 beschriebenen Bias der

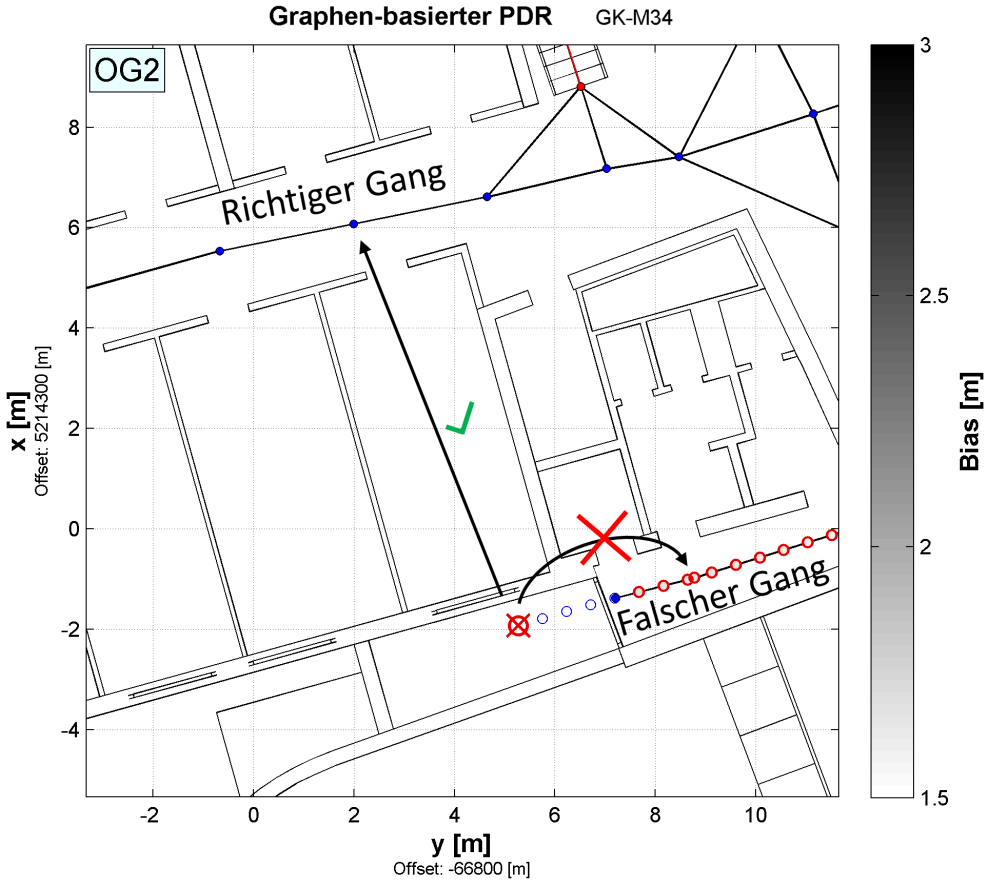


Abbildung 6.33: Falsch detektierter Gang. Kanten mit Positionslösungen bei den möglichen Kantenkandidaten nicht berücksichtigen

letzten Position auf dem Graphen einen Suchradius (3σ), innerhalb dessen geeignete Kanten für die neue Position liegen könnten, siehe Abbildung 6.32. Dabei ist anzumerken, dass alle Kanten, die seit der letzten Abbiegung von dem Algorithmus gewählt wurden, nicht berücksichtigt werden. Diese Einschränkung ist dadurch zu begründen, dass beispielsweise bei einem falsch detektierten Gang, wie in Abbildung 6.33 ersichtlich, immer aufgrund des nächstgelegenen Knotens die letzte Kante des Ganges gewählt werden würde und der richtige Gang nie.

Im Anschluss berechnet der Algorithmus mit allen möglichen Kantenkandidaten und der aktuellen Bewegungsrichtung den eingeschlossenen Schnittwinkel und wählt geeignete Kandidaten anhand eines Grenzwertes aus. Dieser Grenzwert ist jedoch nicht, wie in den vorigen Zuständen, mit 10° definiert, sondern wird auf 20° erhöht. Diese Annahme dient dazu, die Diskrepanz zwischen der Bewegungsfreiheit des Fußgängers und den vordefinierten

Richtungen der Kanten des Graphen auszumerzen.

Falls trotz des erhöhten Grenzwertes keine passende Kante innerhalb des Suchradius gefunden werden kann, wird eine globale Suche gestartet. Diesbezüglich vergleicht der Algorithmus alle Kantenausrichtungen des Graphen mit dem aktuellen Heading. Tritt der Fall in Entscheidung xii ein, dass trotz der globalen Suche keine Kante im gesamten Graphen gefunden wird, müssen zusätzliche Schritte abgewartet werden, bis das aktuelle Heading einer Kante im Graphen entspricht.

Liefert die globale oder lokale Suche aus den Entscheidungen xi und xii geeignete Kanten, wählt der Algorithmus, wie in Abbildung 6.32 ersichtlich, jene Kante aus, die den zur Ausgangsposition der Suche nächstgelegenen Anfangskonten besitzt. Anschließend untersucht Entscheidung xiii, ob die neue Kante durch eine lokale Suche gefunden wurde.

Trifft diese Entscheidung zu, löscht der Algorithmus die Kantenabfolge seit der letzten signifikanten Abbiegung, welche durch den Zustand 3 detektiert wurde. Darauffolgend berechnet der Algorithmus anhand des Dijkstra shortest-path Routing-Algorithmus aus Kapitel 5.3 die kürzeste Route zwischen der letzten Abbiegung (Anfangsknoten der Kante nach der Abbiegung) und dem Anfangsknoten der neu gefundenen Kante. Als nächstes füllt der Graphen-basierte PDR-Algorithmus, je nach Schrittlänge, die Kantenabfolge mit Positionslösungen auf. Folglich stimmt nach diesem Schritt die Anzahl der Positionen auf dem Graphen nicht mehr mit der Anzahl der gemessenen Schrittevents überein, da die Positionslösungen auf diesem, durch Routing ermittelten Teilstück, nicht zu einem detektierten Schritt passen.

Bei einer globalen Suche wird davon ausgegangen, dass der Algorithmus seit längerer Zeit keine sinnvollen Ergebnisse mehr produziert. Durch die neu gefundene Kante wird bei der globalen Suche der Algorithmus neu initialisiert und es wird somit von der letzten Abbiegung kein zusammenhängender Weg durch einen Routing-Algorithmus generiert.

Abschließend wird, wie in Abbildung 6.34 im Zustand 5, als aktuelle Position nach der globalen/lokalen Suche ein Schritt auf der neu gefundenen Kante gesetzt. Nach diesem Zustand 5 sind somit alle Bereiche und Funktionen des Algorithmus beschrieben, die bei einem durch die Schritterkennung detektiert Schritt auftreten können. Anschließend wird auf die Entscheidung IV aus dem Übersichtsflussdiagramm 6.2 verwiesen und überprüft, ob durch die beschriebenen Zuständen eine neue Kante erreicht wurde.

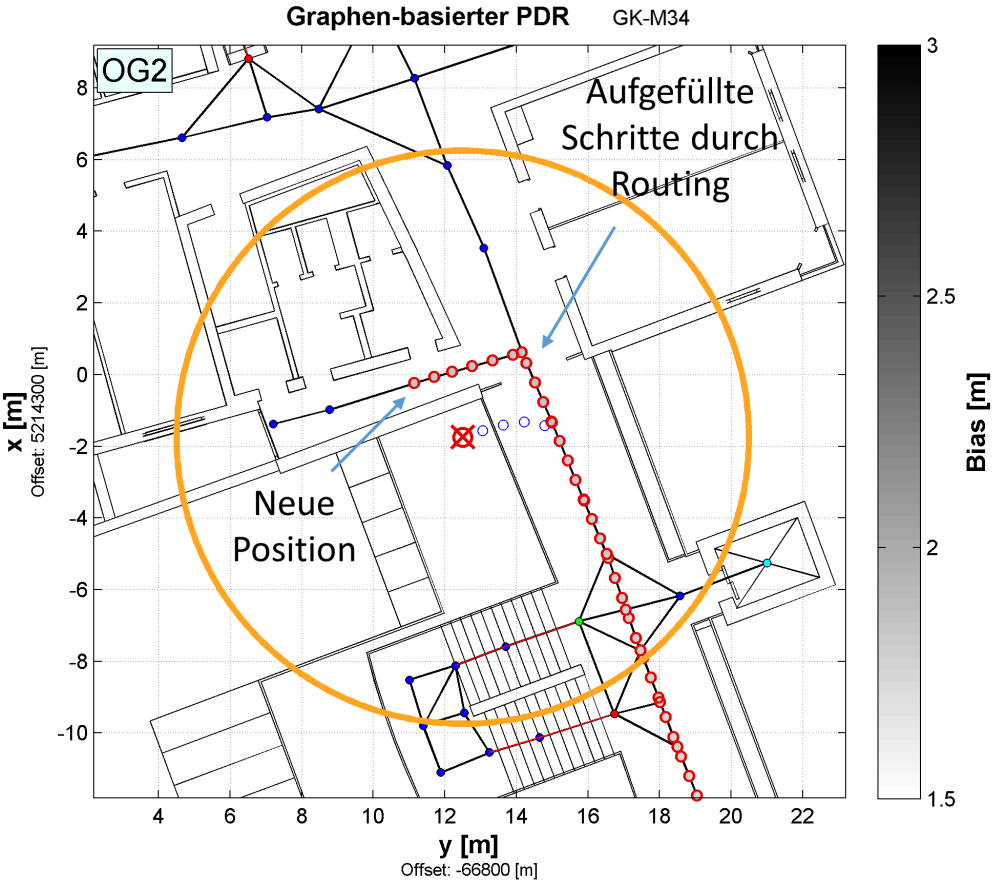


Abbildung 6.34: Aktuelle Positionslösung nach der lokalen Suche

6.10.2 Auswahlkriterium Map-Matching

Für die zugrundeliegende Problemstellung der Suche eines geeigneten Kantenkandidaten nach den fünf Suchschritten wurde in dieser Arbeit ein zweites Verfahren entwickelt, welches auf den Prinzipien des aus Kapitel 5.4 vorgestellten Map-Matchings beruht. Dabei geht es primär um die Detektion einer Kantenabfolge, welche der ungestützten PDR-Trajektorie seit der letzten Abbiegung bestmöglich entspricht. Auf der Endkante wird schlussendlich die neue Position gesetzt und der Algorithmus beginnt von neuem die Zustände 1-4 auszuführen.

Vergleicht man die Ergebnisse dieses Ansatzes, anhand von Testtrajektorien mit dem vorherigen aus Kapitel 6.10.1, kann das Auswahlkriterium durch Map-Matching nicht so robuste Ergebnisse liefern. Dementsprechend benützt der umgesetzte Graphen-basierte PDR-Algorithmus primär das Auswahlkriterium mit Richtungsabweichung δ und Distanz.

Falls dieser Ansatz jedoch Versagen sollte und die Position des Fußgängers nicht mehr sinngemäß wiedergegeben wird, kann auf den Map-Matching Ansatz zurückgegriffen werden.

Das Flussdiagramm 6.35 beschreibt die Funktionsweise des entwickelten Map-Matching Ansatzes. Zu Beginn werden dabei die Distanzen anhand der Historie der bisherigen von dem Algorithmus gewählten Kantenabfolge zu den nach Zustand 3 vollzogenen Abbiegungen berechnet. Ab der letzten Abbiegung wird, wie in Abbildung 6.36 ersichtlich, die ungestützte PDR-Lösung als Referenztrajektorie für das Map-Matching, berechnet. Anzumerken ist hierbei, dass die ungestützte PDR-Trajektorie diesbezüglich im weiteren Verlauf als Referenztrajektorie bezeichnet wird. Diese Bezeichnung bedeutet allerdings nicht, dass die ungestützte PDR-Lösung fehlerfrei ist und die wahre Trajektorie widerspiegelt.

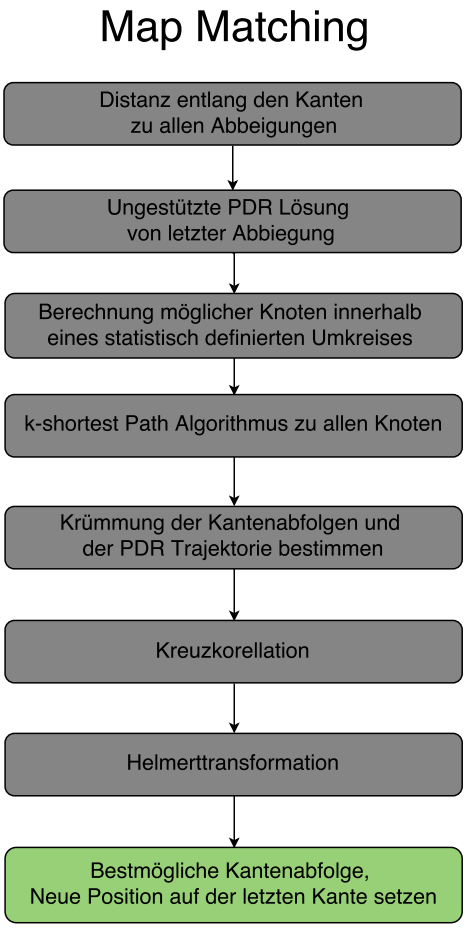


Abbildung 6.35: Flussdiagramm für den Suchansatz mittels Map-Matchings

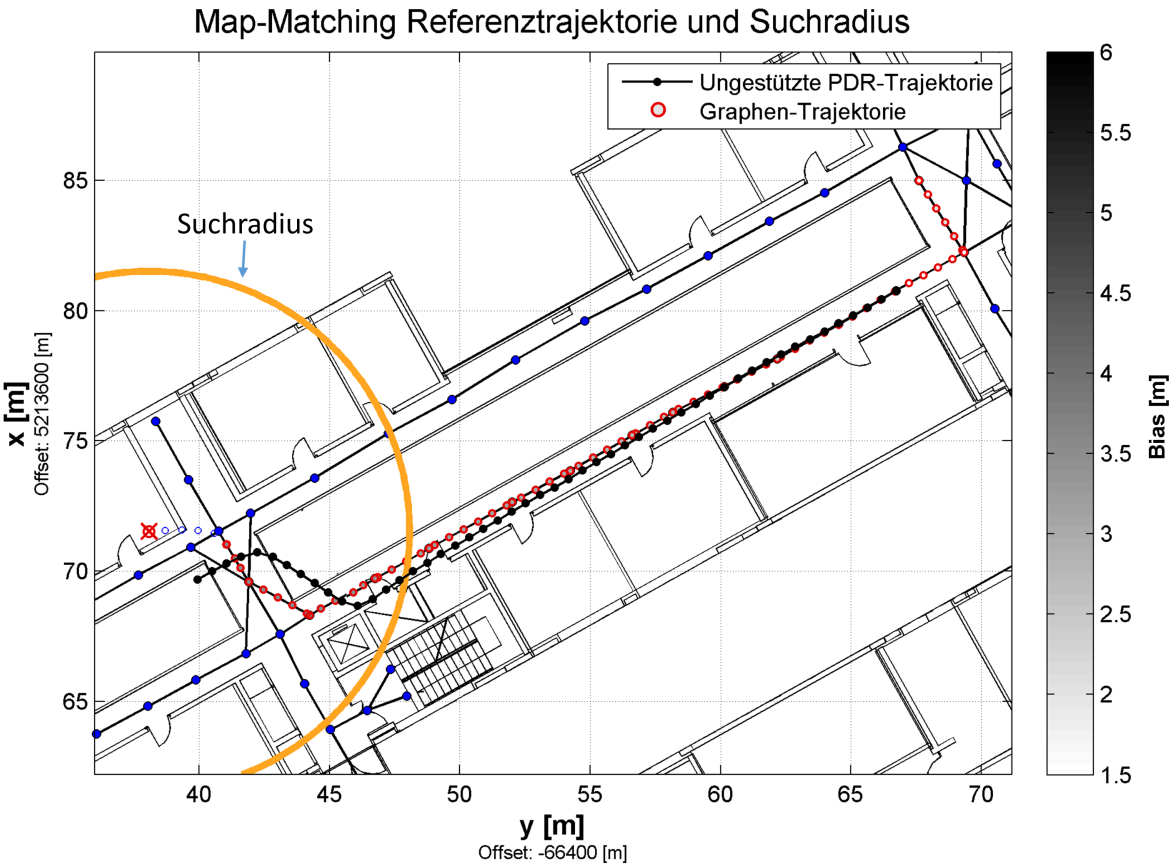


Abbildung 6.36: Suche einer neuen Kante mittels Map-Matching. Referenztrajektorie durch ungestütze PDR-Lösung

Die Länge dieser Trajektorie und in weiterer Folge die Längen der zu vergleichenden Kantenabfolgen spielen für das Resultat des Map-Matching eine wesentliche Rolle. Für eine bestmögliche Aussage über die Ähnlichkeit zwischen der PDR-Trajektorie und den Kantenabfolgen darf die Referenztrajektorie nicht zu kurz sein. Diesbezüglich wählt der Algorithmus frühere Abbiegungen für den Startpunkt dieser PDR-Trajektorie aus. Als Grenzwert hat sich durch Testmessungen eine Mindestlänge von 15m herausgestellt. Anschließend wird in einem Umkreis von zehn Metern zu allen darin liegenden Knoten ein k-shortest path Algorithmus (siehe Kapitel 5.3) berechnet. Den Mittelpunkt des Suchkreises definiert die zuletzt bekannte Position des ungestützten PDR-Trajektorie.

Der Graph in der Testumgebung Steyrergasse 30 des Kellergeschosses (KG) aus Abbildung 6.8 beinhaltet 61 Knoten und 140 gerichtete Kanten. Diesbezüglich ergeben sich von einem Startknoten zu allen anderen Knoten 1084 mögliche Routen. Dabei werden jedoch keine Routen mitgezählt, die invers gerichtete Kanten von bereits gewählten Kanten beinhalten

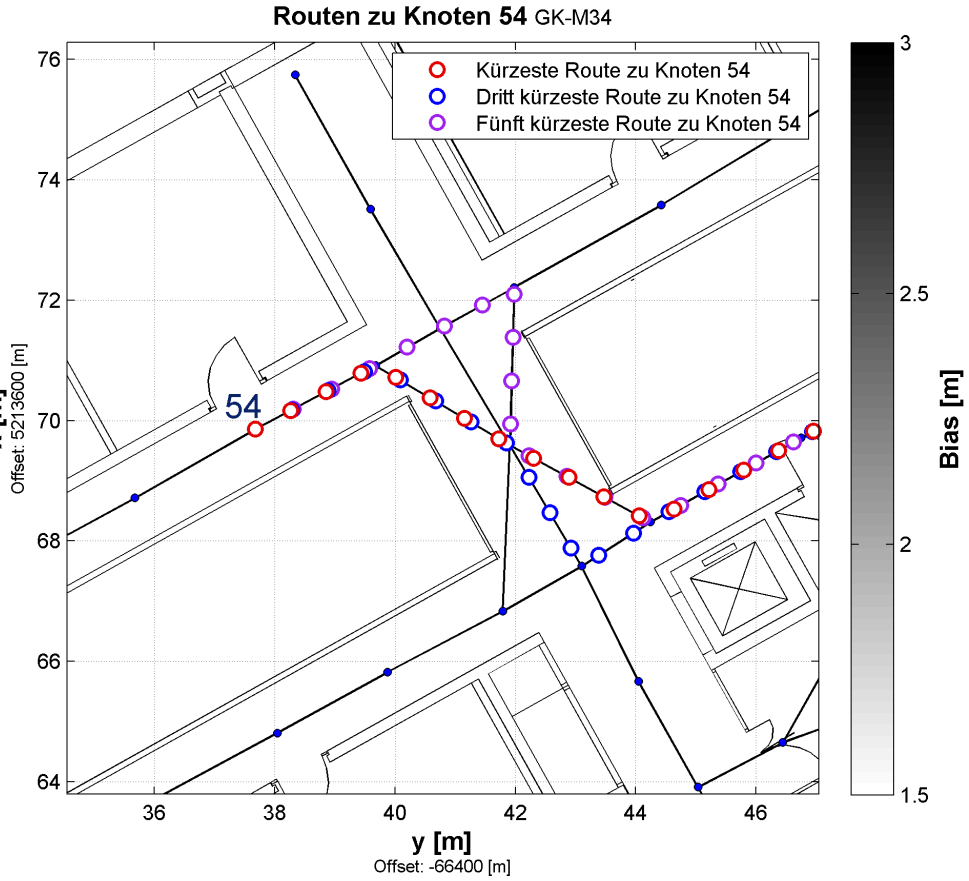


Abbildung 6.37: Aktuelle Positionslösung nach der lokalen Suche

würden. In diesem Fall kommt jedoch die Struktur des kaum verzweigten Graphen und die geringe Anzahl an Knoten dieses Stockwerks der Berechnungszeit und der relativ geringen Anzahl an Routen zugute.

In der Testumgebung Inffeldgasse 16 ist der Graph mit 320 Knoten und 730 Kanten wesentlich größer. Wie in Abbildung 7.5 ersichtlich, gibt es in diesem Gebäude eine Vielzahl an verbundenen parallelen Gängen, die es dem User ermöglichen, das Gebäude hin und zurück zu durchqueren ohne dabei einmal umdrehen zu müssen. Durch diese Konstellation ergibt sich schon zwischen zwei Knoten eine Anzahl von 3932 unterschiedlichen Routen. Diesbezüglich können aufgrund der Rechenperformance nicht zu allen Knoten alle möglichen Routen berechnet werden, sondern nur die fünf kürzesten Routen zu den im Umkreis liegenden Knoten. Die Abbildung 6.37 visualisiert bezüglich der Suchsituation aus Abbildung 6.36 drei der fünf kürzesten Routen von der letzten Abbiegung zu einem bestimmten Knoten (KnotenID 54) innerhalb des Suchradius.

Nachdem alle möglichen Routen bekannt sind, werden zu lange und zu kurze Kantenabfolgen eliminiert, da sie im Verhältnis zu der Länge der Referenztrajektorie nicht für die bestmögliche Kantenabfolge in Frage kommen. Diesbezüglich wurde eine Schranke von ±10m definiert. Nach dieser Einschränkung werden die berechneten Routen durch Positionen auf den Kanten diskretisiert. Dabei variiert die Schrittlänge je nach Länge der Kantenabfolgen, sodass in Summe auf einer Kantenabfolge gleich viele Positionen wie Schritte in der ungestützten PDR-Trajektorie vorhanden sind.

Im Anschluss benötigt der Algorithmus eine Information über die Ähnlichkeit der Referenztrajektorie zu den berechneten Routen. Dies kann in Analogie zu dem konventionellen Map-Matching aus Kapitel 5.4 mittels einer Kreuzkorrelation bewerkstelligt werden. Durch diesen Berechnungsschritt wird bei einer gewissen Verschiebung Δ die bestmögliche Kantenabfolge durch einen maximalen Peak in der Kreuzkorrelationsfunktion detektiert. Durch den Drift in der Referenztrajektorie und durch die vereinfachte Darstellung der Bewegung des Fußgängers anhand der Kantenabfolgen der ermittelten Routen, kann bei der Kreuzkorrelation nicht auf den Vergleich der orientierten Richtungen zurückgegriffen werden. Stattdessen wird das Richtungsbild nach der Bogenlänge abgeleitet um die Krümmung ($\kappa = d\varphi/ds$) der Trajektorie zu erhalten. Diese Größe ist von der absoluten Richtung unabhängig.

Nach Czommer [5] sollen für die Berechnung der Krümmung zu Beginn die orientierten Richtungen an den Stützstellen/Positionen ermittelt werden:

$$\varphi = \arctan\left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i}\right) \tag{6.11}$$

Abbildung 6.38, visualisiert das Winkelbild der Referenztrajektorie, der kürzesten- und der fünft kürzesten Trajektorie zu dem Knoten 54 aus Abbildung 6.37.

Eine Differenzbildung der Richtung und eine simple Division durch die Bogenlängen würde als "Ableitung" aufgrund der Länge der Bogenlängensegmente für die Krümmung unbrauchbare Ergebnisse liefern. Dementsprechend müssen die absoluten Richtungen, welche nur an den Stützstellen bekannt sind, für eine analytische Ableitung der Krümmung interpoliert werden. Dafür wurde eine stückweise kubische Splineinterpolation des diskreten Winkelbildes angewendet. Darauffolgend wird, wie in Abbildung 6.39 ersichtlich, diese kontinuierliche Funktion abgeleitet.

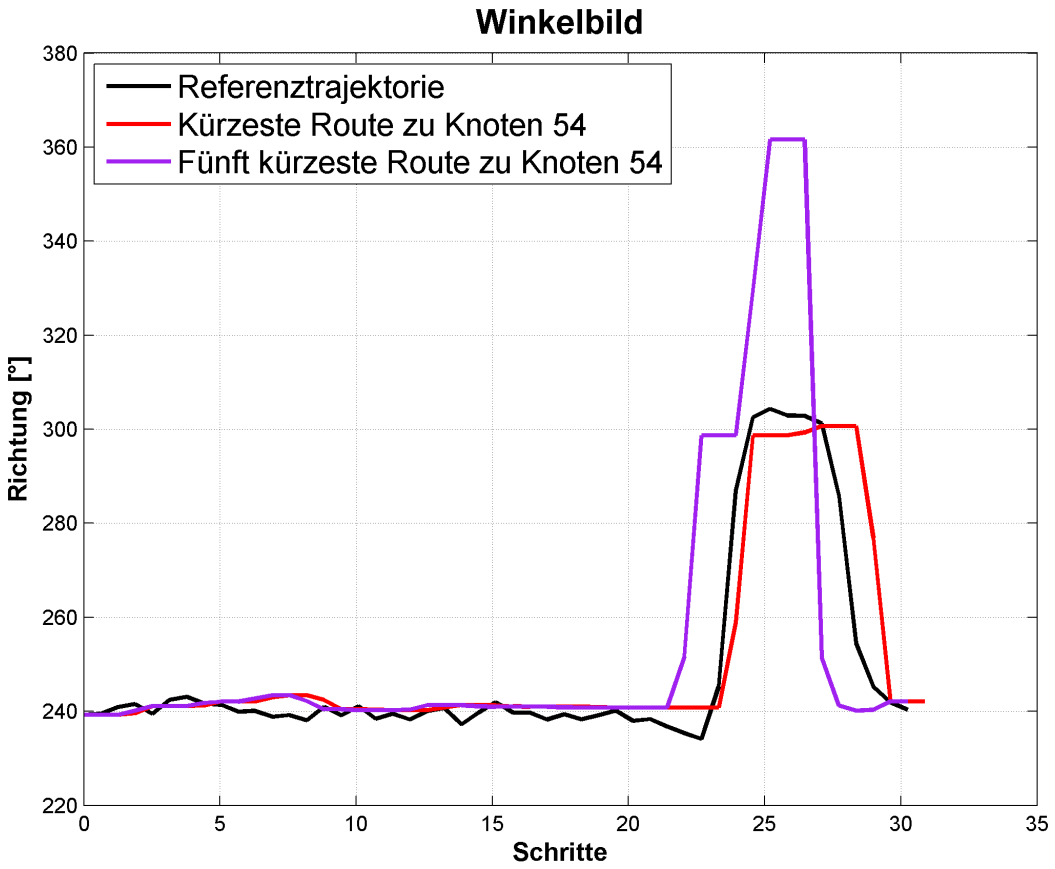


Abbildung 6.38: Winkelild der Referenztrajektorie, der kürzesten- und der fünft kürzesten Trajektorie zu Knoten 54

Anschließend existierte eine kontinuierliche Funktion der Krümmung für die Referenztrajektorie und für alle möglichen Kantenkombinationen, die innerhalb der erwähnten Schranken liegen. Die Abbildung 6.40 visualisiert die Kreuzkorrelation zwischen der Referenztrajektorie und der kürzesten bzw. der fünft kürzesten Route zu dem Beispielknoten 54.

Den maximalen Wert der Kreuzkorrelation erhält die kürzeste Route, wenn sie um einen Schritt nach links verschoben wird. Bei der fünft kürzesten Route ergibt sich ein maximaler Wert bei einer Verschiebung von zwei Schritten nach rechts. Diese Verschiebung kann als weiteres Ausschlusskriterium angesehen werden. Falls eine Route für ein bestmögliches Ergebnis zu weit verschoben werden muss, entspricht diese Route nicht dem Bewegungsablauf des Fußgängers, da sich nur ein geringer Anteil der beiden Trajektorien in der Kreuzkorrelation deckt. Des Weiteren visualisiert die Abbildung 6.40 die Problematik, dass

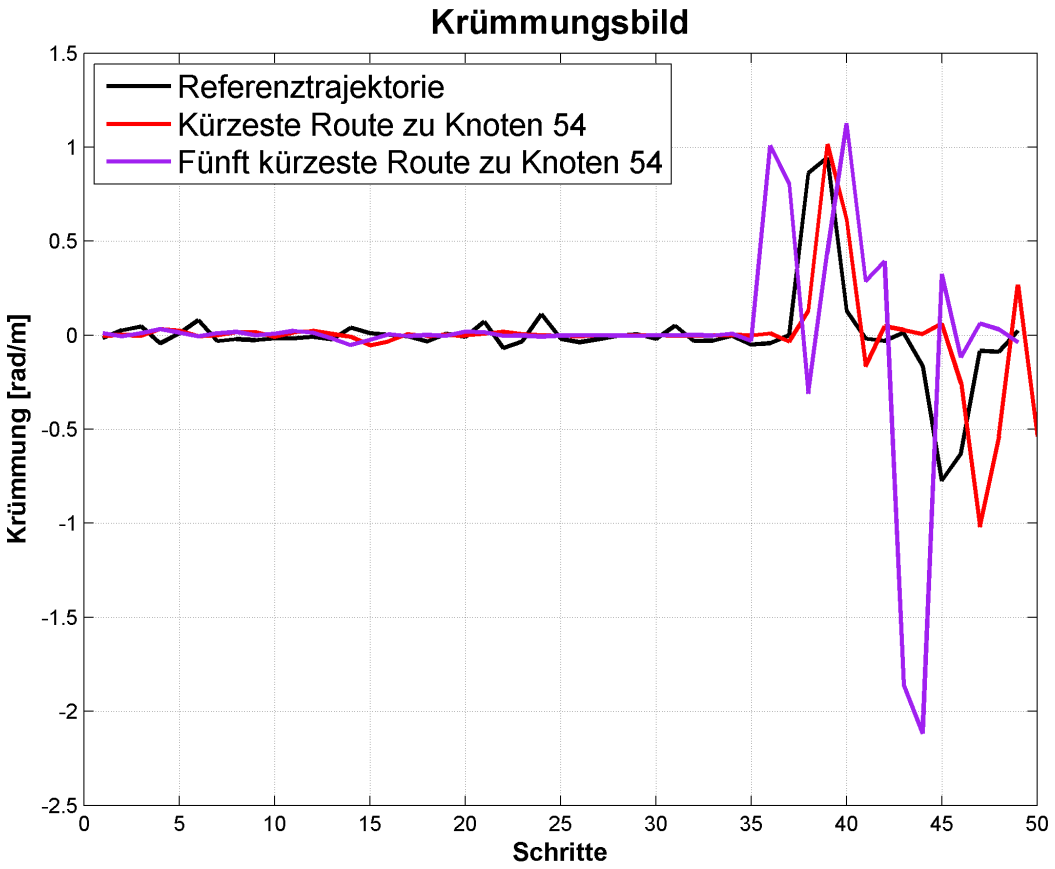


Abbildung 6.39: Krümmungsbild der Referenztrajektorie, der kürzesten- und der fünft kürzesten Trajektorie zu Knoten 54

die Kreuzkorrelation kein eindeutiges Ergebnis zu Gunsten der kürzesten Route liefert. Auf die Gründe wird in Kapitel 7.3 näher eingegangenen.

Nach Czommer [5] und Mayerhofer [11] kann als weiteres Hilfsmittel zur Ähnlichkeitsuntersuchung eine Helmert-Transformation durchgeführt werden. Diese Transformation wird primär dazu verwendet, um Koordinaten eines Systems in ein anderes zu transformieren. Bei der 2D Helmert-Transformation bewerkstelligen diesen Vorgang zwei Verschiebungsparameter a, b, ein Rotationsparameter α und ein Maßstabsfaktor m:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix} + m \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{6.12}$$

Falls die Passpunkte (idente Koordinaten in beiden Systemen) gegeben und die

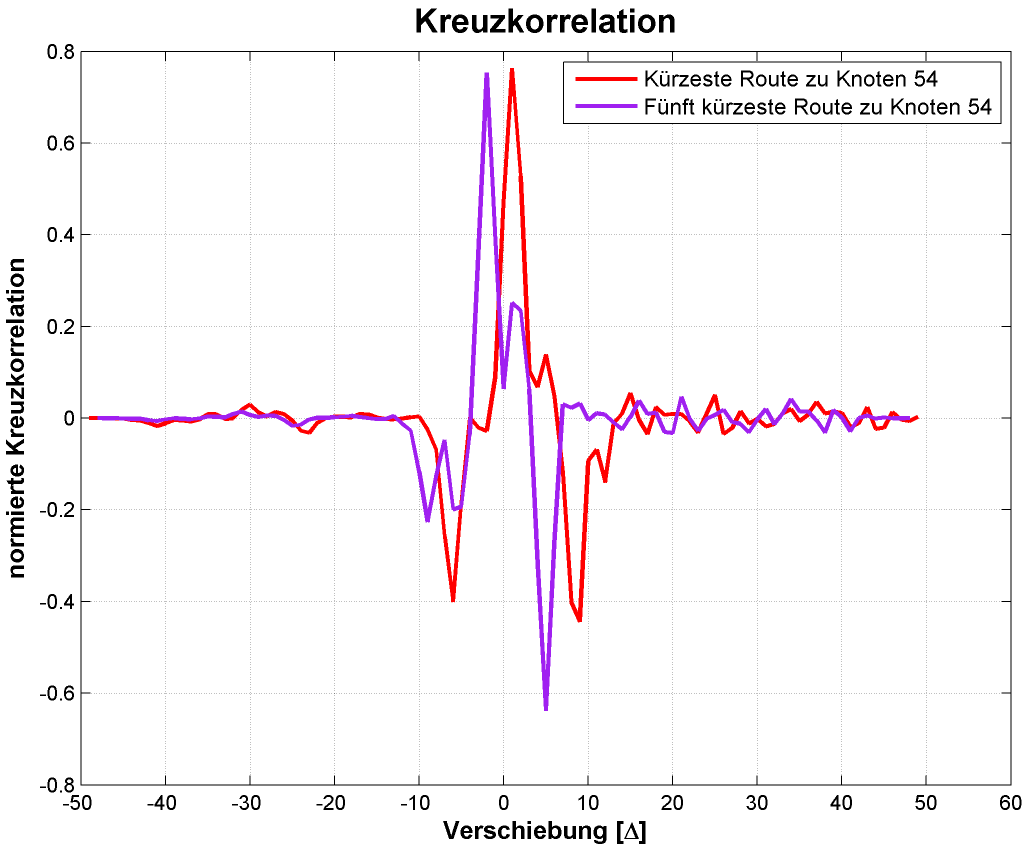


Abbildung 6.40: Kreuzkorrelation zwischen der Referenztrajektorie und der kürzesten bzw. fünft kürzesten Trajektorie zu Knoten dem 54

Transformationsparamater gesucht sind, können diese Parameter, nach Navratil und Staudinger [13], durch einen Ausgleich geschätzt werden. Diesbezüglich wird zu Beginn die Gleichung 6.12 in vier unbekannte Parameter $\mathbf{x} = [a\ b\ c\ d]^\mathsf{T}$ umgeformt, sodass lineare Beobachtungsgleichungen entstehen.

$$\begin{aligned} X &= a + m\cos(\alpha)x - m\sin(\alpha)y &&= a + cx - dy \\ Y &= b + m\sin(\alpha)x + m\cos(\alpha)y &&= b + dx + cy \end{aligned} \tag{6.13}$$

$$\mathbf{l} + \mathbf{v} = \mathbf{A}\mathbf{x} \tag{6.14}$$

Der Beobachtungsvektor $\mathbf{l}$ des Ausgleichsmodells beinhaltet die Koordinaten des Zielkoordinatensystems (X_i, Y_i). Die Designmatrix $\mathbf{A}$ kann aus Gleichung 6.13 abgeleitet werden:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & x_1 & -y_1 \\ 0 & 1 & y_1 & x_1 \\ 1 & 0 & x_2 & -y_2 \\ 0 & 1 & y_2 & x_2 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & x_i & -y_i \\ 0 & 1 & y_i & x_i \end{bmatrix} \tag{6.15}$$

Löst man die Gleichung 6.14 im Sinne eines Ausgleichs nach kleinsten Fehlerquadraten, erhält man den gesuchten Parametervektor $\hat{\mathbf{x}}$. Mit diesen geschätzten Parametern werden die Koordinaten des Ausgangssystems bestmöglich mit jenen des Zielsystem übereinstimmt. In Bezug auf das Map-Matching sind jedoch nicht die numerischen Werte dieser Parameter von Bedeutung, sondern die Quadratsumme der Verbesserungen aus Formel 6.16. Ein Wert nahe Null beschreibt hierzu, dass die Trajektorie im Ausgangssystem und die Trajektorie im Zielsystem sehr gut aneinander angepasst sind.

$$\begin{aligned} \mathbf{v} &= \mathbf{A}\hat{\mathbf{x}} - \mathbf{l} \\ min &= \mathbf{v}^\mathsf{T}\mathbf{v} \end{aligned} \tag{6.16}$$

Der Vorteil bei dieser Ähnlichkeitsüberprüfung liegt darin, dass die Generalisierung der Kantenabfolgen durch die Verschiebungsparameter, der Drift in der Referenztrajektorie durch den Rotationsparameter und die ungleichen Schrittlängen durch den Maßstabsfaktor kompensiert werden. Diesbezüglich vergleicht man mit dieser Methode rein die Topologie/Form der Trajektorien ohne sie durch Störgrößen zu beeinflussen. Die Abbildung 6.41 visualisiert die beiden Trajektorien aus der Kreuzkorellation vor und nach der Transformation.

Anzumerken ist, dass der Ausgleich gleich viele Punkte in jedem System voraussetzt. Falls in einem System mehr Koordinaten vorhanden sind, können diese nicht beachtet werden. Dementsprechend wurden, wie bereits erwähnt, die möglichen Kantenabfolgen zu gleich vielen Schritten wie in der Referenztrajektorie diskretisiert. Andernfalls würde die Zuordnung identer Passpunkte ein Problems darstellen.

Der Graphen-basierte PDR-Algorithmus verwendet für den Map-Matching Ansatz zuerst eine Kreuzkorrelation, bei dem mögliche Kantenabfolgen aufgrund minimaler Korrelationswerte ausgeschlossen werden können. Anschließend wird das hinsichtlich der Matrizenin-

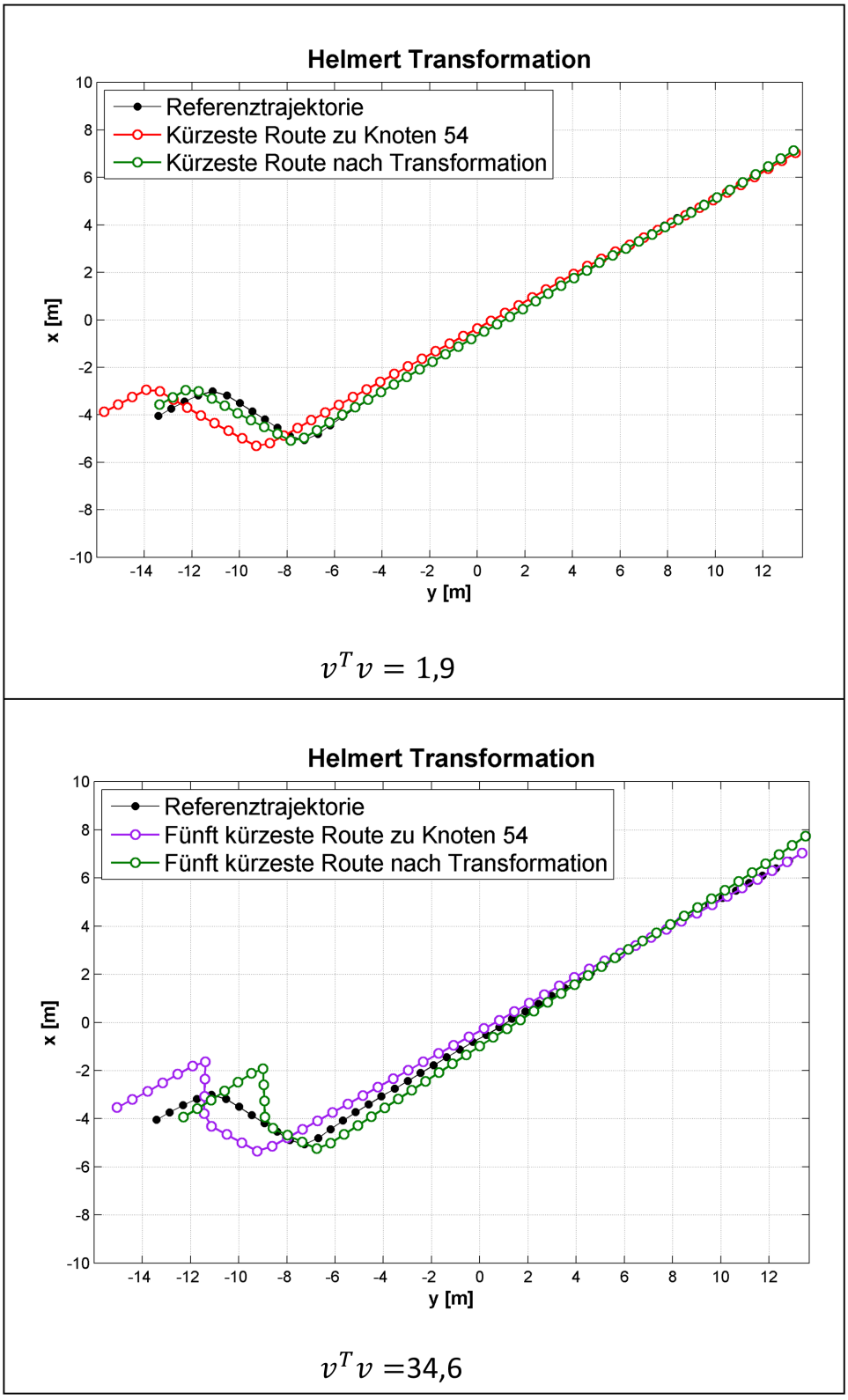


Abbildung 6.41: Vergleich der kürzesten und fünft kürzesten Route zu Knoten 54 nach der Helmert-Transformation mit der Referenztrajektorie

version rechenintensivere Transformationsverfahren für eine eindeutige Identifizierung der bestmöglichen Route zur Referenztrajektorie durchgeführt.

Bei dem konventionellen Map-Matching wird abschließend die aktuelle Position auf dem Graphen durch einen Ausgleich bestimmt. Dies ist aufgrund der erzielten Genauigkeit und der kurzen Knotenlängen nicht notwendig. Dementsprechend wird, falls möglich, der letzte Punkt der Referenztrajektorie orthogonal auf die letzte Kante der am besten angepassten Kantenabfolge projiziert. Ergibt die orthogonale Projektion keine Position auf der Kante, wird, wie beim Auswahlkriterium aus Kapitel 6.10.1 ein Schritt nach dem Anfangsknoten auf der letzten Kante aufgetragen.

Abschließend werden alle Restriktionen des Map-Matchings aufgelistet, die es ermöglichen, adäquate Lösungen durch den Map-Matching Ansatz zu generieren:

- Referenztrajektorie muss länger als 15m sein und mindestens eine Abbiegung enthalten
- Begrenzte Anzahl an Endkonten der k-shortest path Routen durch einen zehn Meter Suchradius
- Routen beinhalten keine 180° Drehungen
- Es werden nur die fünf kürzesten Routen berechnet
- Die möglichen Routen dürfen nicht in die entgegengesetzte Richtung der Referenztrajektorie starten
- Routen dürfen nicht kürzer oder länger als zehn Meter zu der Referenztrajektorie sein
- Routen mit einem Kreuzkorrelationswert unter 0,3 werden ausgeschlossen
- Die möglichen Kantenabfolgen dürfen nicht über mehrere Stockwerke gehen

6.11 Stockwerkswechsel

Als zusätzliche Funktion soll der Algorithmus Stockwerkswechsel vollziehen können, sodass die Trajektorien nicht auf ein Geschoss eingeschränkt sind. Unterschiedliche Stockwerke

sind in den Testszenarien durch Stiegen und Lifte verbunden und müssen, wie in Kapitel 6.2.2 beschrieben, bei der Graphenerstellung speziell konstruiert/gekennzeichnet werden.

Zu Beginn wurde ein Ansatz entwickelt, der für den angestrebten Stockwerkswechsel keine Barometerdaten benötigt. Diesbezüglich wechselt der Algorithmus in ein darüberliegendes Geschoss wenn die aktuelle Position auf die Stiegenkante vor dem Knoten mit der Kennzeichnung "Ende des Stiegenaufgangs" fällt. Analoges gilt für einen Stockwerkswechsel in ein darunterliegendes Geschoss durch die Stiegenkante vor dem Knoten "Ende des Stiegenabgangs". Dieser simple Ansatz funktioniert nur dann, wenn der Graphen-basierte PDR-Algorithmus die Bewegung des Fußgängers während des Stockwerkswechsel korrekt wiedergibt. Würde beispielsweise durch Zustand 5 fälschlicherweise genau die Stiegenkante für einen Stockwerkswechsel ausgewählt werden, vollzieht der Algorithmus einen Wechsel in eine andere Etage obwohl dieser nicht getätigt wird. Zudem kann die Benutzung eines Liftes nicht mit diesem Ansatz nicht implementiert werden, da rein aus inertialen Daten das Zielstockwerk nicht robust ermittelt werden kann.

Ein weitaus robusterer Ansatz für diese Problematik wurde mit Hilfe eines Barometers umgesetzt. Durch diese zusätzliche Sensorik können, wie bereits in Kapitel 3.1.3 beschrieben, absolute Höhen bestimmt werden. Da diese Angaben jedoch stark von der Wetterlage abhängig sind, werden sie in Bezug auf eine Ausgangshöhe auf relative Höhen umgerechnet.

Die Abbildung 6.42 visualisiert die relativen Höhen aus Barometerdaten für die Testumgebung Steyrergasse 30. Als Ausgangshöhe dient dabei in diesem Ansatz die gemittelte absolute Höhe aus der ersten Stehphase. Bei bekannten Stockwerkshöhen der einzelnen Etagen können die relativen Höhen anschließend verschiedenen Stockwerken zugeordnet werden. Zusätzlich wurden die relativen Höhen durch einen alpha-beta Filter geglättet:

$$\hat{\mathbf{x}}_k = \begin{bmatrix} h \\ \Delta h \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \hat{\mathbf{x}}_{k-1} + \begin{bmatrix} \alpha \\ \frac{\beta}{\Delta t} \end{bmatrix} \left(h_k - \begin{bmatrix} 1 \\ 0 \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \hat{\mathbf{x}}_{k-1} \right) \tag{6.17}$$

Überschreiten die Barometerdaten während des Graphen-basierten PDR-Algorithmus einen bestimmten statistisch definierten Grenzwert eines darüber- oder darunterliegendes Stockwerkes, wird in dieses Geschoss gewechselt. Für die Bestimmung dieser Grenzwerte wurde in den rot markierten Stehphasen der Abbildung 6.42 die Standardabweichung ermittelt. Diese beläuft sich in diesen drei Fällen auf $\sim$ 20cm. Aufgrund der Geschosshöhen von minimal $3,65$m können dementsprechend Stockwerkswechsel durch Barometerdaten

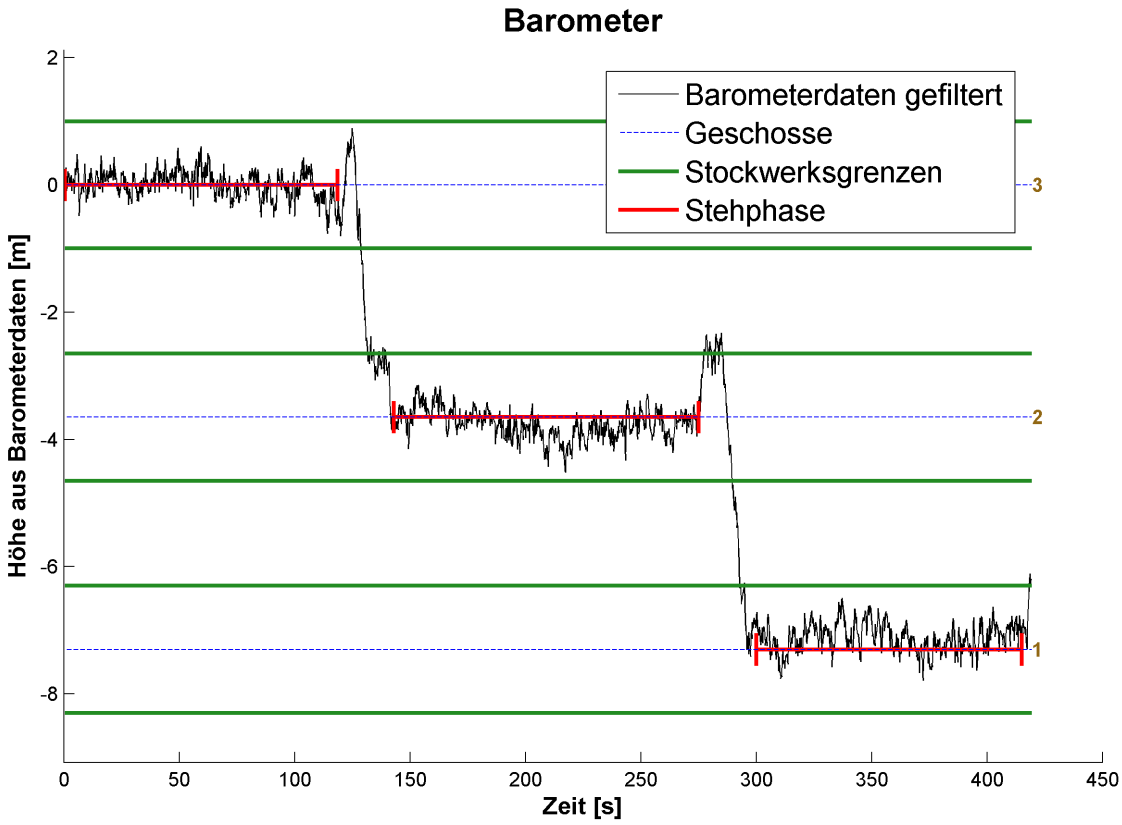


Abbildung 6.42: Relative Höhenänderungen aus Barometerdaten. Von Stockwerk 3 bis Stockwerk 1

eindeutig und unmissverständlich erkannt werden. Die Grenzwerte für ein Stockwerk wurden, wie in den grünen Linien aus Abbildung 6.42 ersichtlich, in Anlehnung an die Standardabweichung und zusätzliche empirische Tests auf ±1m festgelegt.

Erkennt der Algorithmus einen Stockwerkswechsel muss zuerst unterschieden werden, ob der Fußgänger dafür Stiegen oder einen Lift benützt. Da der Graphen-basierte PDR-Algorithmus auf einer relativen Positionierungsmethode beruht, würde eine falsch berechnete Position Auswirkung auf alle zukünftigen Positionen haben. Diesbezüglich soll der Stockwerkswechsel als eine Art Neuinitialisierung angesehen werden und unabhängig von den zuvor berechneten Positionen sein. Dementsprechend wird die Unterscheidung zwischen Lift und Stiege nicht aufgrund der Nähe zu Stiegenkanten oder Liftknoten getätigt, sondern unabhängig davon. Falls die Barometerdaten die Grenze eines benachbarten Stockwerks erreichen, benötigt der Algorithmus, wie in Abbildung 6.43 ersichtlich, als einzige Zusatzinformation für den Stockwerkswechsel die Anzahl der Schritte während der letzten 1,65m

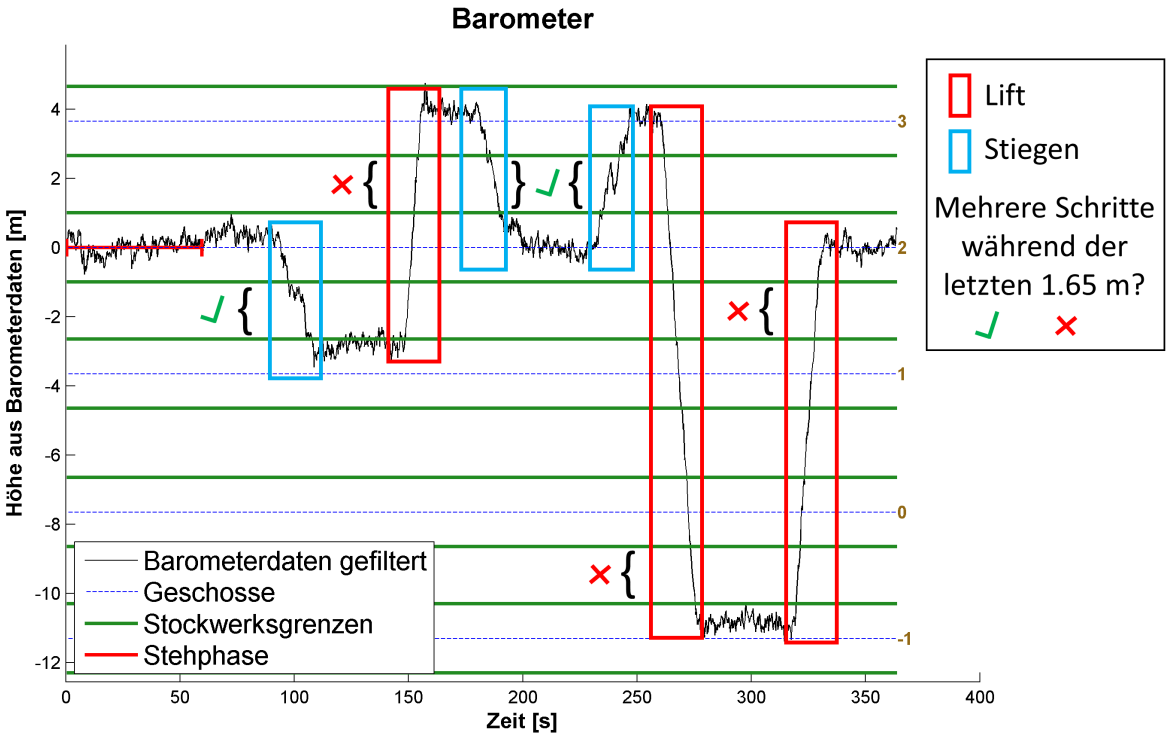


Abbildung 6.43: Unterscheidung zwischen Lift und Stiegen. Falls während der letzten 1,65m mehrere Schritte getätigt wurden

Höhenänderung. Diese Schranke beschreibt den Bereich zwischen zwei Stockwerksgrenzen (z.B. Steyrergasse 30: Stockwerkshöhe 3,65m - 1m Grenze oberes Geschoss - 1m Grenze unteres Geschoss = 1,65m). Tätigt der Fußgänger für diesen Zeitraum der Höhenänderung mehr als drei Schritte, kann davon ausgegangen werden, dass der User Stiegen steigt und sich nicht im Lift befindet. Da die Google Schritterkennung zusätzlich die ersten Schritte (ungefähr fünf) nach einer Stehphase nicht registriert, wird während der Zeit im Lift auch bei geringfügigen Positionsänderungen des Fußgängers kein Schritt detektiert. Diese Tatsache verstärkt das robuste Ergebnis für die Unterscheidung zwischen Lift und Stiegen.

Lift

Benützt der Fußgänger für einen Stockwerkswechsel einen Lift, ermittelt der Algorithmus, bei welchem Zielstockwerk der User den Lift verlässt. Diesbezüglich muss zu Beginn berechnet werden, wann der Lift nach einem Stockwerkswechsel wieder stehen bleibt. Hierbei wird während der Liftbenützung fortwährend die Standardabweichung der letzten 20 Barometermessungen berechnet. Dies entspricht bei einer Datenrate von fünf Hertz

einem Zeitraum von vier Sekunden. Fällt die Standardabweichung unter 40cm, entspricht $\sim 2\sigma$ der Ruhephase aus Abbildung 6.42, kann angenommen werden, dass der Lift sich nicht mehr in Bewegung befindet. Die Detektion des erreichten Stockwerks wird anhand des Höhenunterschiedes zum Ausgangsstockwerk berechnet. Tätigt der Fußgänger während der Stehphase des Liftes zusätzlich Schritte wurde das Zielstockwerk erreicht und der User verlässt den Lift. Andernfalls steigt eine zusätzliche Person in den Lift zu oder aus. Demgemäß verlässt der Fußgänger den Lift nicht und fährt weiter.

Die neue Startposition nach dem Stockwerkswechsel ist, wie beispielsweise in der Steyrergasse 30, die zugehörige Kante zu dem Knoten mit der semantischen Zusatzinformation `"Lift"`. Bei Liften mit zweiseitigen Ausgängen muss zusätzlich das Heading beim Verlassen des Liftes für die Kantendetektion beachtet werden.

Stiegen

Zu Beginn ist zu erwähnen, dass sich die Schrittlänge beim Treppensteigen verkürzt. Die standardisierte Länge von einzelnen Stufen beträgt 30cm. Demnach verringert sich die Schrittlänge des Fußgängers, wenn der User eine Stiegenkante erreicht. Diese kennt der Algorithmus durch die spezielle Kennzeichnung während der Graphenkonstruktion, siehe Tabelle 6.2.

Bei Stockwerkswechsel durch Stiegen ist die neue Startposition in einem benachbarten Stockwerk nicht so trivial wie bei einem Wechsel durch einen Lift. Für eine robuste Implementierung der neuen Startposition wird anhand von drei Fällen unterschieden:

1. Falls es eine Stiegenkante im nächsten Stockwerk gibt, auf welche die letzte Position des vorigen Stockwerks eine orthogonale Projektion mit einer orthogonalen Distanz unter 0,5m auf die Kante besitzt, führt der Algorithmus zuerst diesen Schritt aus, siehe Abbildung 6.44. Der Grenzwert von 0,5m bewerkstelligt, dass nur auf eine Kante projiziert wird, die in unmittelbarer Nähe zu der Kante der alten Position ist. Da die Stiegen in beiden Etagen modelliert werden, bedeutet diese Projektion nur, dass die gleiche Kante in einem anderen Stockwerk ausgewählt wird. Dadurch verändert sich die 2D Position nur minimalst. Da die Projektion ebenso auf die inverse Kante zutreffend ist, muss zusätzlich durch das aktuelle Heading die richtige Kante ausgewählt werden.

2. Liefert die orthogonale Projektion aus 1. kein Ergebnis, dann soll jene Stiegenkante herangezogen werden, für welche der Unterschied zwischen der Kantenausrichtung und

dem aktuellen Heading unter 35° liegt. Wenn mehrere Kanten zu treffen, wird die am nächsten liegende Kante zu dem Stieganabgangs- bzw. zu dem Stiegenaufgangsknoten gewählt.

3. Versagen beide Ansätze wird einfach die letzte Kante des Stiegenaufgangs oder -abgangs gewählt.

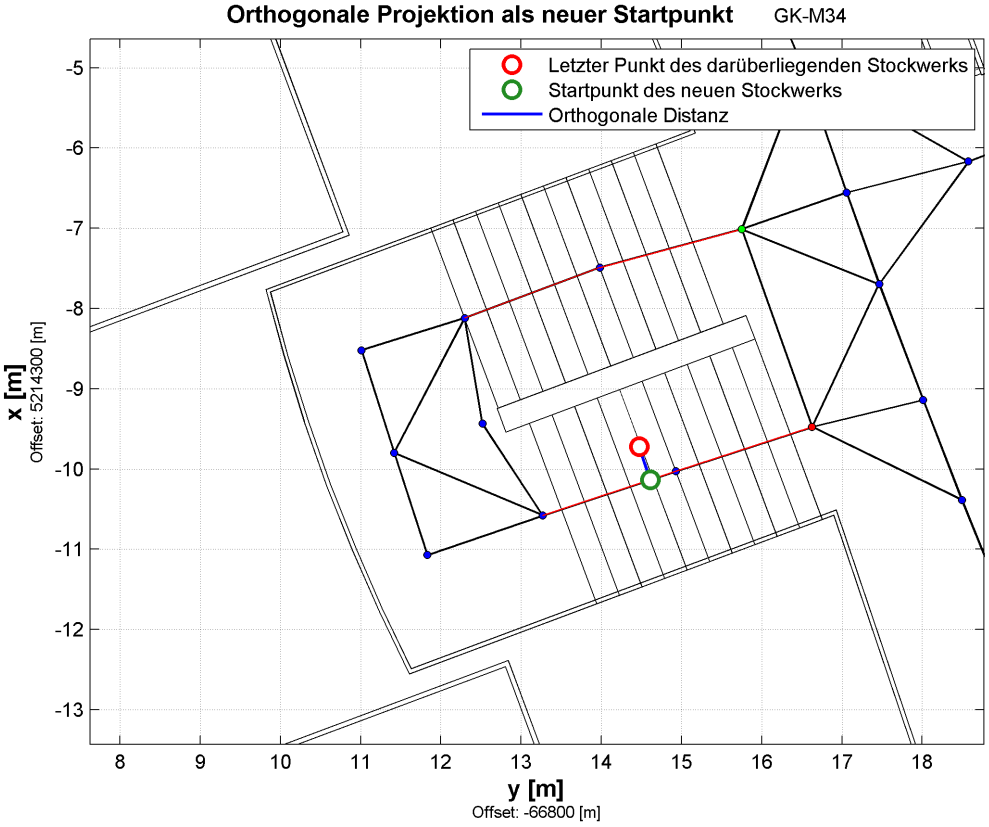


Abbildung 6.44: Neuer Startpunkt im nächsten Stockwerk durch eine orthogonale Projektion

Falls mehrere Lifte oder Stiegen in einem Ausgangsstockwerk existieren, muss aufgrund der Entfernung zu der letzten bekannten Position eine Entscheidung getroffen werden, welcher Lift oder welche Stiege benützt wurde. In diesem Fall ist der Algorithmus für ein richtiges Urteil von vorherigen Positionen abhängig. Dies kann jedoch im Hinblick auf die angestrebte Neuinitialisierung durch den Stockwerkswechsel als nicht störend beurteilt werden, da aufgrund der räumlich weitverteilten Stiegen in der Steyrergasse 30 ein geringfügiger Fehler in der Position trotzdem ein eindeutiges Ergebnis in den Testtrajektorien liefert. Anderenfalls müsste man auf zusätzliche absolute Positionsinformationen, wie z.B. BLE-Beacons, zurückgreifen.

Kapitel 7

Evaluierungen und Testtrajektorien

In diesem Kapitel werden die Auswirkung der Graphenkonstruktion sowie die Ergebnisse und speziellen Eigenschaften des Graphen-basierten PDR-Algorithmus hinsichtlich verschiedener Testtrajektorien in den Testumgebungen erörtert und visualisiert. Zusätzlich wird näher auf die Problemfelder des Map-Matching aus Kapitel 6.10.2 wie auch auf die Kalibrierung der Drehraten eingegangen. Abgeschlossen wird mit einem Vergleich zwischen dem entwickelten Algorithmus und einem von der Arbeitsgruppe Navigation bereitgestellten Partikelfilter.

7.1 Evaluierung der Graphenkonstruktion

Der Graph soll für eine bestmögliche Positionsschätzung des Algorithmus, alle möglichen Bewegungen des Fußgängers im Gebäude durch das Kanten- und Knotennetzwerk repräsentieren. In größeren Kreuzungsbereichen, bei denen der Fußgänger auf verschiedene Weisen ein und dieselbe Abbiegung tätigen kann, ist die Modellierung besonders wichtig. Beispielsweise kann die Abbiegung von Knoten eins zu Knoten fünf aus Abbildung 7.1 in zwei 45° oder zwei 90° Winkel vollzogen werden. Beinhaltet der Kreuzungsbereich, wie im linken Bild aus Abbildung 7.1 ersichtlich, nur Kanten im 90° Winkel, würde bei einer getätigten 45° Abbiegung eine lokale/globale Suche durch den Zustand 5 ausgelöst werden. Da keine Kante im Kreuzungsbereich für dieses Abbiegeverhalten vorhanden ist, würde auf eine falsche Kante innerhalb des Graphen referenziert werden. Deckt der Kreuzungsbereich jedoch beide Abbiegevarianten (rechtes Bild aus Abbildung 7.1) ab, kann die Trajektorie des Fußgängers durch den Graphen korrekt wiedergegeben werden. Dementsprechend sollen

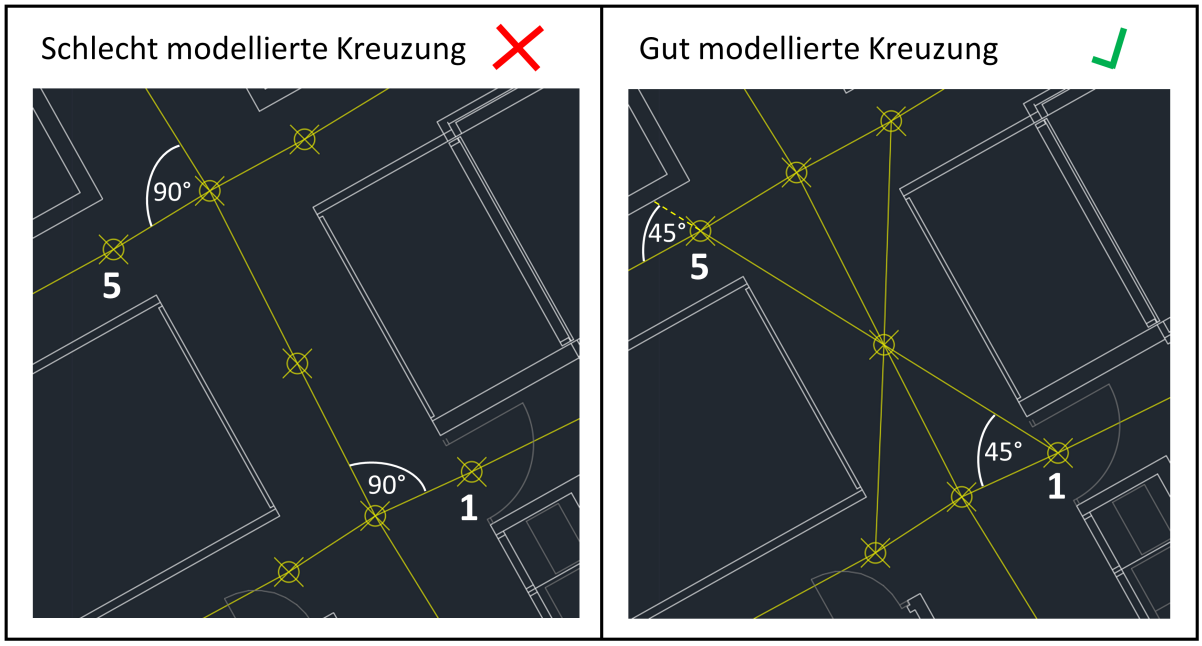


Abbildung 7.1: Modellierung von Kreuzungen

für eine adäquate Positionsschätzung durch den Algorithmus alle eventuellen Bewegungen des Fußgängers innerhalb großer Kreuzungsbereiche modelliert sein.

Neben der Modellierung der Kreuzungsbereiche hat die Länge der Kanten ebenfalls eine signifikante Auswirkung auf die Performance des Graphen-basierten PDR-Algorithmus. Sind die Kanten, so wie in Abbildung 7.2, zu kurz gewählt, beinhaltet der nächstgelegene Knoten keine zu der aktuellen Bewegungsrichtung passenden Kante. Da das Suchfenster fünf Schritte für eine Abbiegung betragen darf, könnte beispielsweise eine Abbiegung fünf Schritte ($\sim$ 3m) vor dem Kreuzungsbereich beginnen. Sind die Kanten kürzer als die angenommenen drei Meter würde der nächstgelegene Knoten nicht den Kreuzungsbereich der registrierten Abbiegung repräsentieren, sondern erst der übernächste Knoten. Dementsprechend würde für diesen beschriebenen Fall eine lokale Suche nach Zustand 5 gestartet werden.

Ein weiterer Nachteil von kurzen Kanten ist der höhere Speicherbedarf der Adjazenzliste. Für das Map-Matching-Verfahren aus Kapitel 6.10.2 haben die kurzen Kanten und die daraus resultierende höhere Anzahl an Knoten eine Auswirkung auf die Rechenzeit, da innerhalb des beschriebenen Suchradius mehr Knoten liegen, zu denen der k-shortest path Algorithmus durchgeführt werden muss.

Lange Kanten haben im Gegenzug die Problematik, dass ein Drift in den Gyroskopdaten

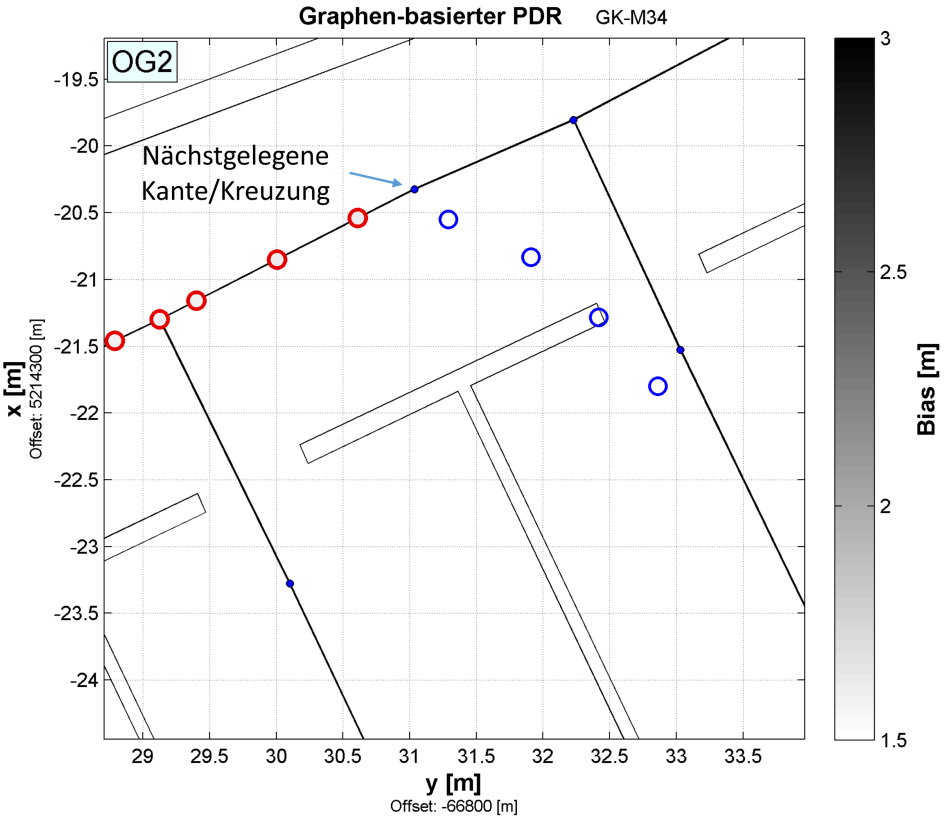


Abbildung 7.2: Problematik bei zu kurzen Kantenlängen - nächstgelegene Kreuzung beinhaltet keine Abbiegung

nicht ausreichend kompensiert werden kann, da die Kumulationszeiten der Drehraten von den Kantenlängen abhängen. Ein weiteres Problem bei langen Kanten ist der Anfangspunkt nach einer globalen/lokalen Suche aus Zustand 5. Dabei wird immer ein Schritt nach dem Anfangsknoten der neu gefundenen Kante als neuer Startpunkt gewählt. Wenn in Kreuzungsbereichen, wie in Abbildung 7.3, die Kantenlängen zu lange gewählt werden, ist der Startpunkt nach der globalen/lokalen Suche zu nah am Anfangsknoten bzw. am südlichen Gang und liegt nicht so wie der Ausgangspunktes der Suche zwischen den zwei Gängen. Im Optimalfall würde der neue Startpunkt in der Mitte der Kante liegen.

Damit dies realisiert werden kann, muss die Kante wie in Abbildung 7.4 in zwei kleinere Kanten geteilt werden. Diesbezüglich würde nach der Suche der neue Startpunkt annähernd in der Mitte liegen und die Position nach der Suche entspricht der tatsächlichen Position des Fußgängers wesentlich besser.

Dementsprechend muss je nach Kreuzungsbereichen ein Kompromiss zwischen zu kurzen und zu langen Kanten gefunden werden, damit die Trajektorie des Fußgängers zu jeder

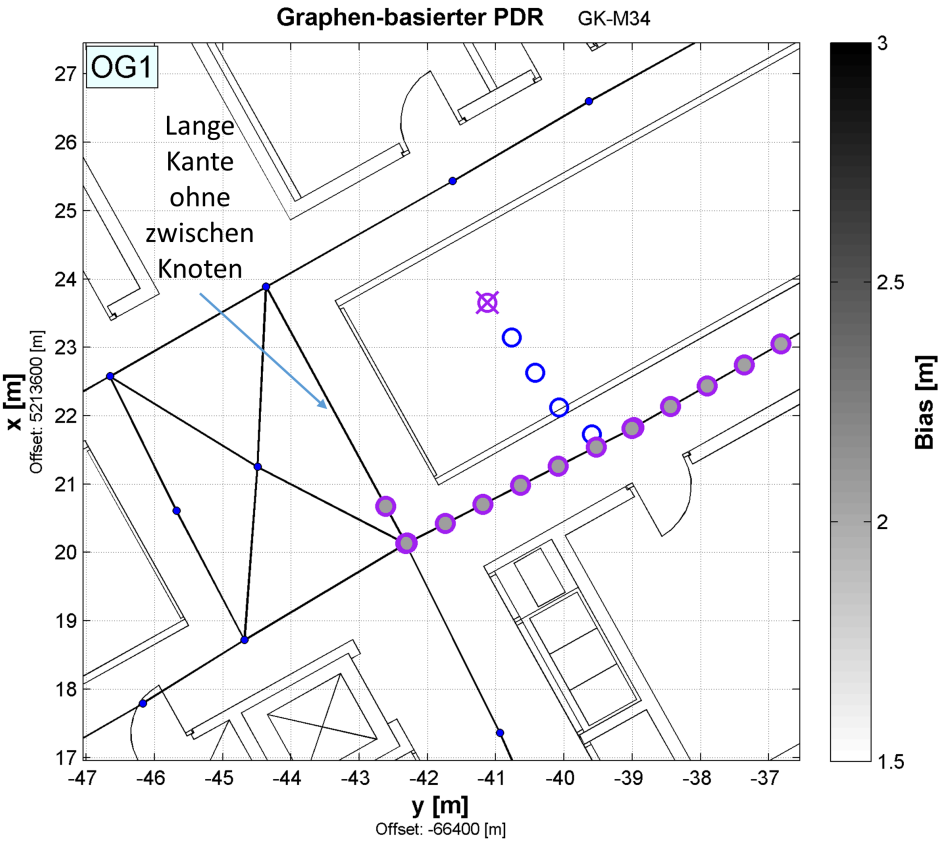


Abbildung 7.3: Zu lange Kantenlänge, der Startpunkt nach der globalen/lokalen Suche ist zu nah am Anfangskonten

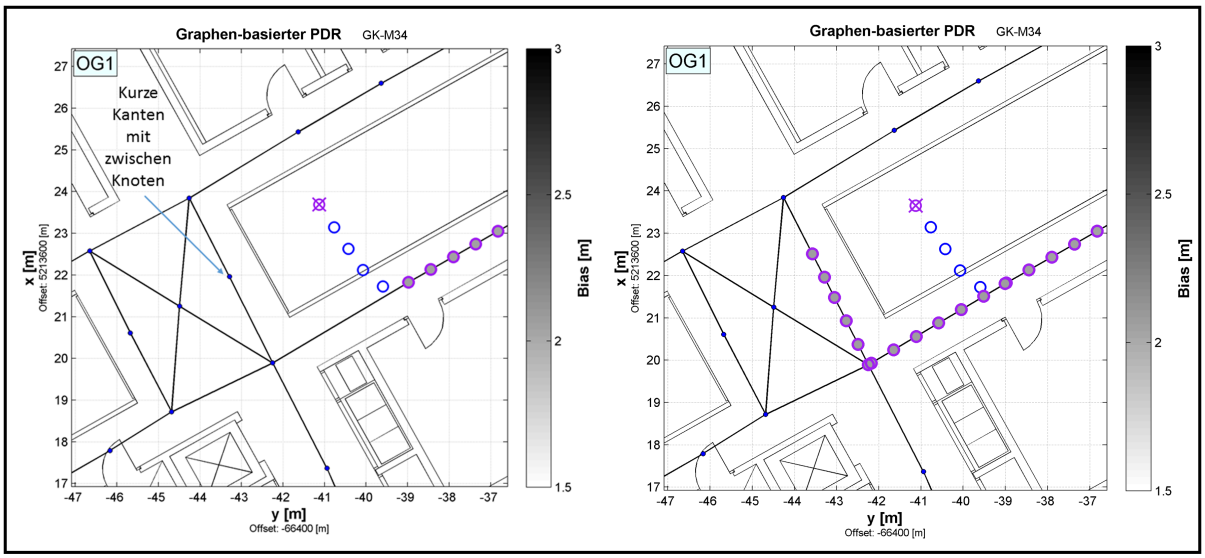


Abbildung 7.4: Richtige Kantenlängen bei Kreuzungen, für Startpunkt nach einer globalen/lokalen Suche

Zeit bestmöglich den gemessenen Daten entspricht. Da jedoch die globale/lokale Suche hauptsächlich bei Abbiegungen benötigt wird und somit Kanten auswählt, die in Kreuzungsbereichen liegen, soll bei der Konstruktion des Graphen speziell auf die Problematik aus Abbildung 7.3 und 7.4 geachtet werden. Die nachfolgende Tabelle weist die durchschnittlichen Kanten, Knoten und Kantenlängen pro Stockwerk der zwei Testgebiete auf, mit denen die besten Ergebnisse erzielt werden konnten. Zudem beinhaltet die Tabelle 7.1 das Verhältnis aus Knoten zu Kanten.

Tabelle 7.1: Durchschnittliche Kanten, Knoten, Kantenlängen pro Stockwerk und das Verhältnis aus Kanten zu Knoten der zwei Testgebiete

	Steyrergasse 30 6 Stockwerken	**Inffeldgasse 16** nur OG 1
∅ **Kantenlänge [m]**	2,3	2,4
∅ **Knoten pro Stockwerk**	94	320
∅ **gerichtete Kanten pro Stockwerk**	201	730
∅ **Verhältnis Knoten/Kanten pro Stockwerk**	0,47	0,43

Tabelle 7.1 zufolge sollen für die Konstruktion der Graphen in den Testgebieten etwas mehr als doppelt so viele gerichtete Kanten als Knoten verwendet werden. Dementsprechend werden die besten Ergebnisse erzielt, wenn die Graphen Eigenschaften eines dünnen Graphen besitzen. Des Weiteren sind für optimale Resultate eher kurze Kanten mit einer Länge von ca. 2,3m von Vorteil. Die Längen der Kanten hängen jedoch stark von den Gebäudestrukturen der Testgebiete ab und können somit nicht gleichbleibend auf 2,3m festgelegt werden.

Die Graphen besitzen zudem keine Abbiegungen unter 20°. Der Grenzwert für ein Geradeausgehen und einer Abbiegung von 10° eignet sich somit, um alle Abbiegungen des Graphen zu erkennen. Zusätzlich muss für diese Grenzwertdefinition auch gelten, dass während der geradlinigen Bewegung des Fußgängers das Smartphone nicht mehr als 10° durch externe Störgrößen, die nicht mit einer Abbiegebewegung korrelieren, rotiert. Die Abbildung 6.13 aus Kapitel 6.5 visualisiert die Richtungsabweichungen von den Kanten während einer Trajektorie. Die Richtungsabweichung ist, abgesehen von den drei tatsächlichen Abbiegungen, über den gesamten Messzeitraum während des Geradeausgehens stets unter 10°. Dementsprechend eignet sich dieser Grenzwert für die Unterscheidung zwischen einem Abbiegeverhalten und einer geradlinigen Bewegung des Fußgängers.

7.2 Testtrajektorien

In diesem Kapitel werden zwei Testtrajektorien, jeweils in einer Testumgebung, vorgestellt. Zu Beginn wird eine Trajektorie aus der Inffeldgasse präsentiert. Hierbei wird nicht auf einzelne Details eingegangen, sondern die Abbildung 7.5 dient dazu, den Unterschied zwischen einer ungestützen Schritt-basierten PDR-Trajektorie und der Trajektorie aus dem Graphen-basierten PDR-Algorithmus zu visualisieren. Die gemessene Route beginnt und endet an dem rot markierten Punkt.

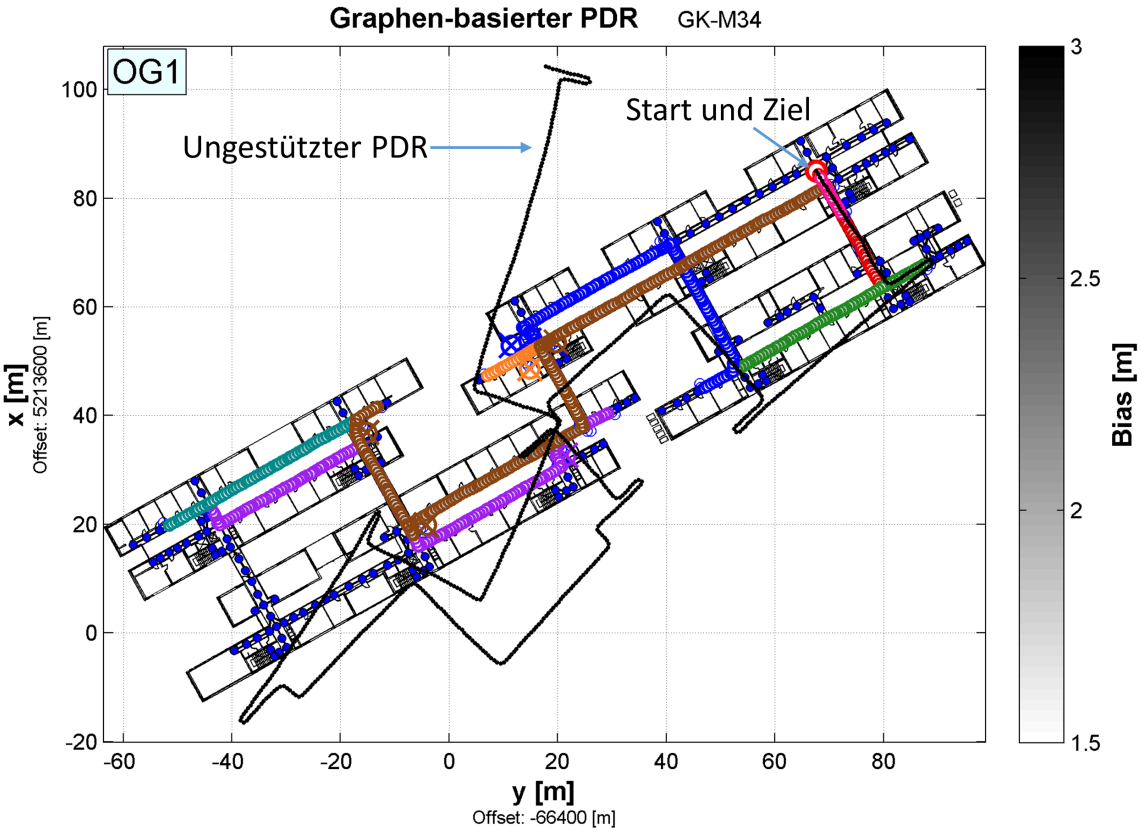


Abbildung 7.5: Testtrajektorie Inffeldgasse 16

Durch den entwickelten Algorithmus ist es möglich, die Trajektorie mit einer Länge von $\sim$ 528m und einer Dauer von ca. sieben Minuten auf den Graphen zu zwingen und somit innerhalb des Gebäudes zu halten. Die angestrebte Endkoordinate des Fußgängers konnte durch die Driftkompensierung und die erfolgreiche Zuordnung und Durchführung der fünf Zustände aus Kapitel 6 ebenso erfolgreich erreicht werden. Im Gegensatz zu den Positionen auf dem Graphen liefert die ungestützte Lösung kein plausibles Ergebnis, da

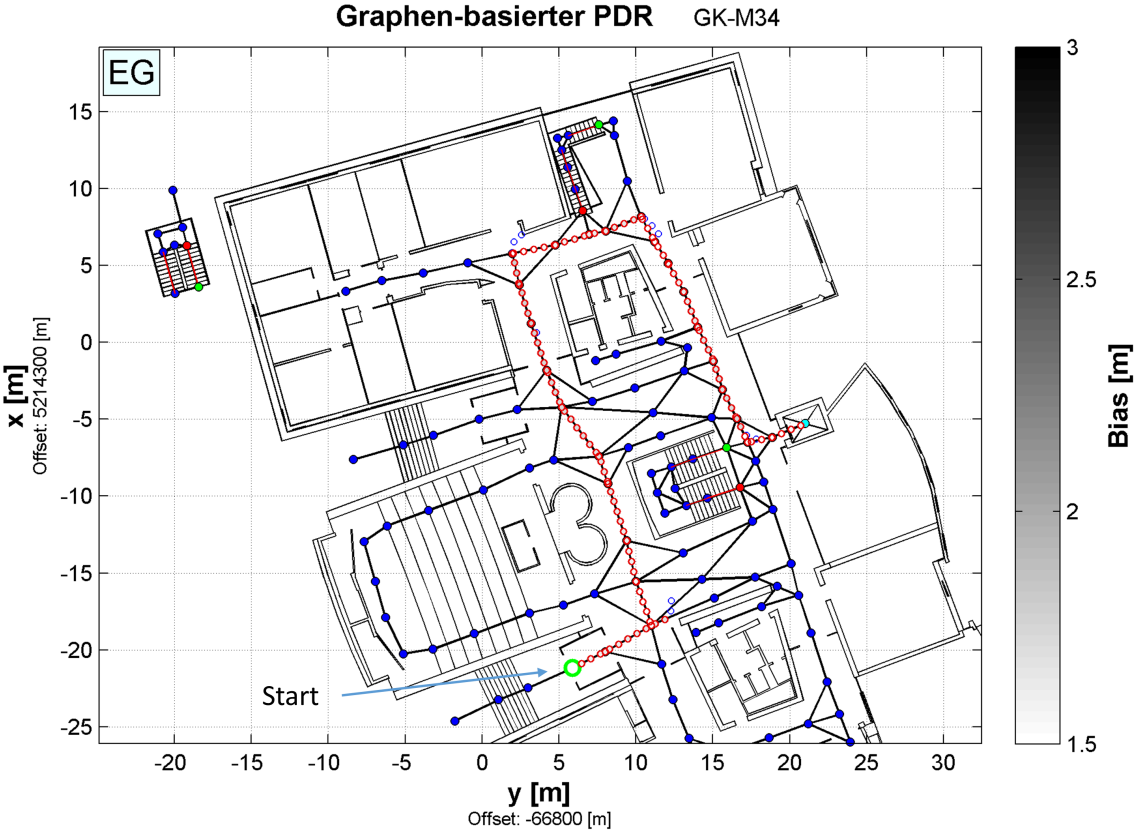


Abbildung 7.6: Testtrajektorie in der Startetage (EG) aus der Steyrergasse 30

sie die Gebäudestrukturen missachtet und aus dem Gebäude hinausführt. Vor allem der Drift, der durch die Kumulation der Drehraten entsteht, kann deutlich in der ungestützten PDR-Trajektorie erkannt werden.

Die zweite Testmessung von ca. vier Minuten wurde in der Steyrergasse 30 durchgeführt und beschreibt eine finale Trajektorie, die das gesamte Funktionsspektrum des entwickelten Graphen-basierten PDR-Algorithmus abdeckt. Dabei startet der Fußgänger außerhalb des Gebäudes. Der Startpunkt des Graphen wird demnach, wie in Abbildung 7.6 (grüner Punkt) ersichtlich, durch den Bluetooth-Beacon 40, siehe Abbildung 6.6, am südlichsten Eingang des Erdgeschosses registriert. Die Trajektorie von dem Startpunkt bis zu dem cyan markierten Lift-Knoten beinhaltet die Zustände 1 und 4 für geradlinige Bewegungen über eine Kante oder über den Endknoten hinaus. Zudem konnten alle vier Abbiegungen innerhalb des Suchraumes von fünf Schritten durch den Zustand 3 erfolgreich erkannt werden.

Als nächstes vollzieht der User einen Stockwerkswechsel mit dem Fahrstuhl in den ersten

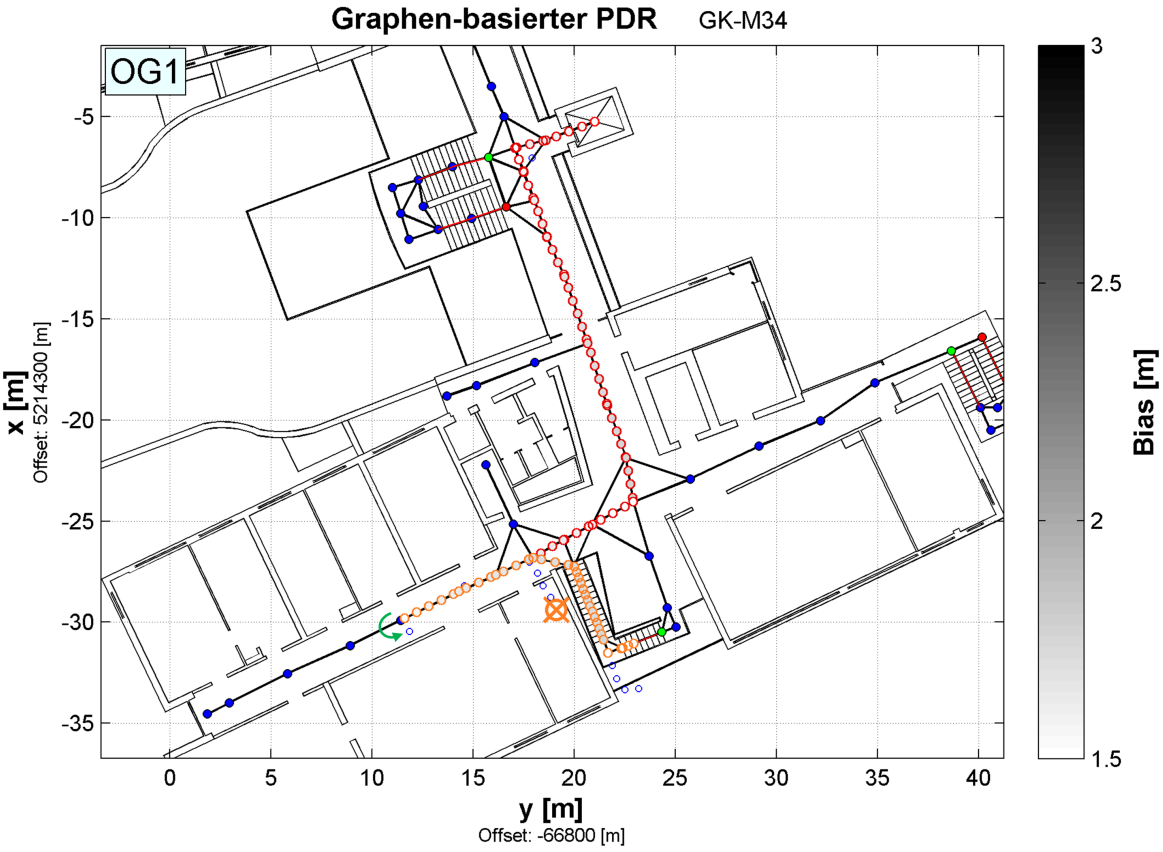


Abbildung 7.7: Testtrajektorie des OG1 der Steyrergasse 30

Stock. Der Geschosswechsel durch den Lift wird anhand des in Kapitel 6.11 beschriebenen Schrittzählers während der Höhenänderung der letzten 1,65m erkannt. Die Abbildung 7.7 visualisiert die anschließende Trajektorie im ersten Obergeschoss (OG 1). Ausgehend von dem Liftknoten biegt die Trajektorie des Fußgängers zweimal korrekt mit Hilfe des Zustandes 3 in den südwestlichen Gang ab. Anschließend registriert das Smartphone eine 180° Drehung und die Trajektorie vollzieht eine Wende. Dieser Vorgang wird visuell durch einen Wechsel der Farben von rot auf orange in Abbildung 7.7 illustriert.

Bei der darauffolgenden Abbiegung existiert bei dem nächstgelegenen Knoten keine inzidente Kante, welche die aktuelle Bewegungsrichtung widerspiegelt, da die angestrebte Abbiegung erst beim übernächsten Knoten konstruiert wurde. Diesbezüglich kann der Algorithmus nicht binnen fünf Schritten eine Kante finden. Dies ist auf die Generalisierung durch den Graphen, einer falschen Schritterkennung oder einer nichtzutreffenden Schrittlänge zurückzuführen. Dementsprechend wird der Fall 5 gewählt und eine lokale Suche gestartet. Das Resultat beinhaltet die zur Bewegungsrichtung passende Stiegenkante.

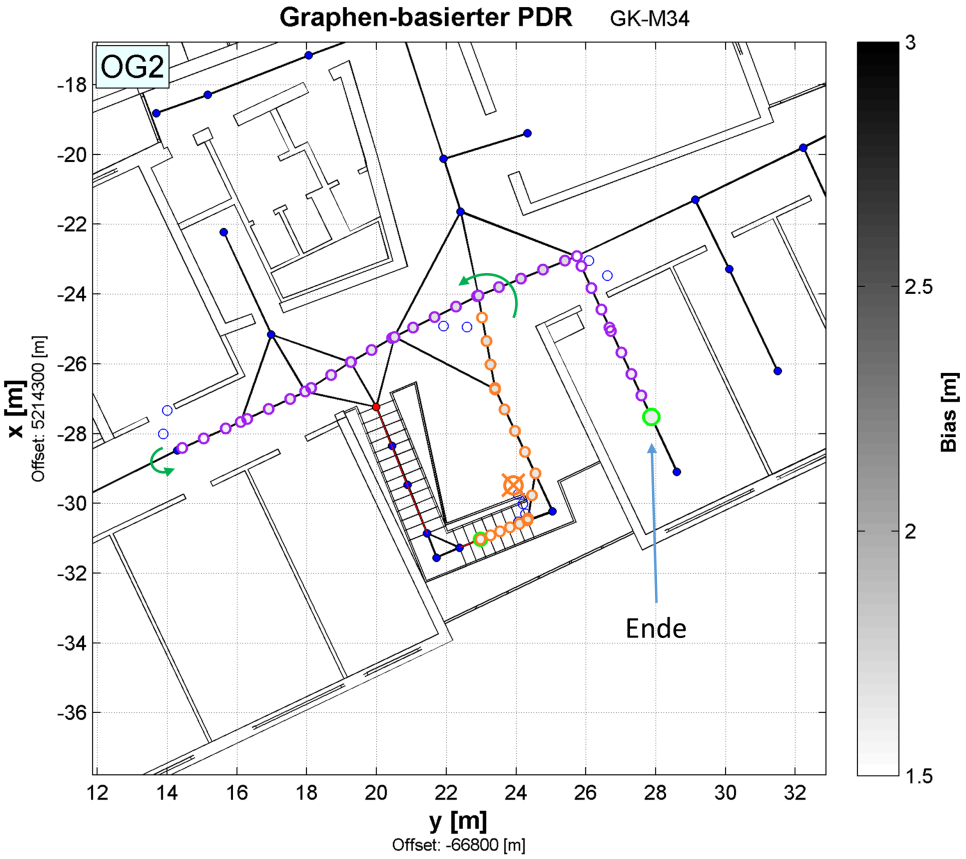


Abbildung 7.8: Testtrajektorie der Zieletage (OG2) der Steyrergasse 30

Fünf Schritte vor dem Ende der letzten Stiegenkante des Stiegenaufgangs registriert der Algorithmus durch die Barometerdaten einen Stockwerkswechsel über Stiegen. Da in dieser Situation alle Voraussetzungen für einen Stockwerkswechsel über Stiegen nach Schritt 1 des Kapitels 6.11 erfüllt sind, wird die Startposition im OG2 über eine orthogonale Projektion der zuletzt bekannten Position des OG1 gelöst. In diesem erreichten Zielstockwerk vollzieht der Fußgänger drei Abbiegungen und eine 180° Wende. Wobei die Abbiegung nach der Stiege nur mit Hilfe des Zustandes 5 vollzogen werden kann.

Abschließend ist anzumerken, dass der Algorithmus nur zwei von 13 Abbiegungen der gesamten Trajektorie von EG zu OG2 nicht korrekt erkannt hat. Die zwei lokalen Suchvorgänge referenzierten in dieser Testtrajektorie auf die richtigen Kanten, sodass über den gesamten Messzeitraum die Positionslösungen auf den Kanten den tatsächlichen Verlauf des Fußgängers repräsentieren können. Die Startposition bzw. der richtige Eingang konnte ebenso eindeutig anhand des Bluetooth-Beacons erkannt werden.

Die abschließende Tabelle beinhaltet eine Analyse anhand gemittelter repräsentativer Kenngrößen aus 26 Testtrajektorien der Steyrergasse 30 und aus neun der Inffeldgasse 16:

Tabelle 7.2: Analysen des Graphen-basierten PDR-Algorithmus aus insgesamt 35 Testtrajektorien

	Steyrergasse 30 ∅ aus 26 Trajektorien	**Inffeldgasse 16** ∅ aus 9 Trajektorien
Zeit [min]	4	10
Länge [m]	182	659
Anzahl Abbiegungen	16	37
Gewählte Kanten	82	270
Globale/Lokale Suchen	5	13
∅ **Bias [m]**	1,56	1,61
Suchvorgänge zu Kanten [%]	6	5
Suchvorgänge zu Abbiegungen [%]	31	36
Richtige Stockwerkswechsel [%]	96	-

Das Testgebäude in der Steyrergasse 30 ist mit seinen fünf Stockwerken ca. 20m hoch. Die horizontale Ausdehnung der einzelnen Etagen ist im Gegensatz zu dem Stockwerk OG1 aus der Inffeldgasse 16 wesentlich kleiner. Dementsprechend wurde die Steyrergasse 30 für kürzere ($\sim$ 4min) und dafür stockwerksübergreifende Trajektorien genutzt. Das Gebäude in der Inffeldgasse 16 dient vor allem für lange Testtrajektorien ($\sim$ 10min). Die Kantenauswahl für die Positionslösungen in beiden Testgebieten wurden zu 94-95% durch die Zustände 1 bis 4 festgelegt. Diesbezüglich konnten lediglich in fünf bis sechs Prozent der Fälle die neuen Kanten nicht innerhalb der fünf Suchschritte eindeutig erkannt werden und mussten durch eine globale oder lokale Suche gefunden werden.

Betrachtet man das Verhältnis zwischen Suchvorgänge und Abbiegungen, konnte in der Steyrergasse 30 ein besseres Ergebnis erzielt werden. In der Inffeldgasse 16 konnten 36% der Abbiegungen nur durch den Zustand 5 bewältigt werden. Dies ist darauf zurück zu führen, dass die Kanten speziell bei Abbiegungen relativ kurz gewählt werden mussten, um alle eventuellen Abbiegungsmöglichkeiten an einer Kreuzung abdecken zu können, siehe Abbildung 7.1.

Ein Wechsel in eine andere Etage konnte durch die Barometerdaten in 96% der Fälle korrekt vollzogen werden. Lediglich bei zwei von 56 getesteten Stockwerkswechseln konnte

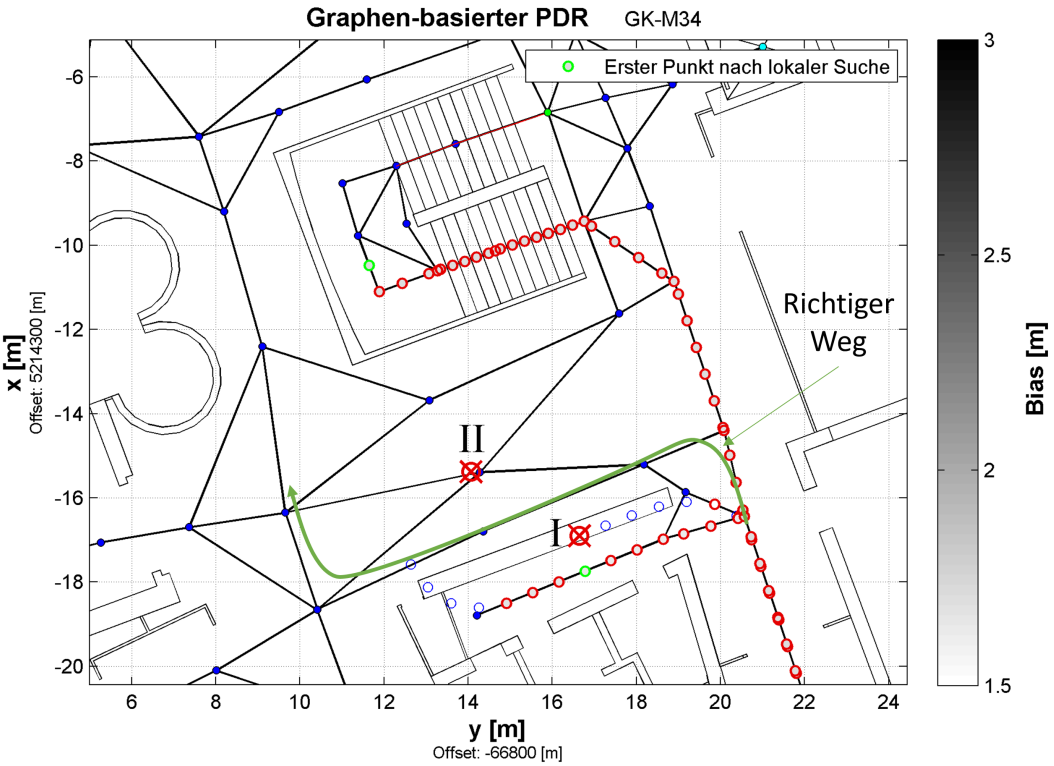


Abbildung 7.9: Falsche Wahl von Kanten aufgrund großer freier Flächen und einem dichten Graphen

der Algorithmus nach einem Wechsel die Trajektorie auf dem Graphen nicht korrekt wiedergeben. Dies lag daran, dass die neue Startposition fälschlicherweise auf eine falsche Stiegenkante gesetzt wurde. Ein Stockwerkswechsel mit Hilfe eines Fahrstuhles lieferte bei allen Testmessungen robuste und richtige Lösungen.

Von insgesamt 40 gemessenen Testtrajektorien konnten einzig fünf nicht die Endposition des Fußgängers durch die resultierende Trajektorie auf dem Graphen erreichen. Der Algorithmus versagt beispielsweise, wenn der Fußgänger eine 180° Drehung direkt an einem Kreuzungsbereich vollzieht. Des Weiteren erreicht der Algorithmus seine Grenzen, wenn der Fußgänger sich auf großen und freien Flächen bewegt, die durch einen dichten Graphen repräsentiert werden müssen. Die Abbildung 7.9 visualisiert ein fehlerhaftes Verhalten des Algorithmus in dem EG der Steyrergasse 30.

Bei der ersten Suche (I) wird aufgrund zweier paralleler Wege, die hinsichtlich der Gebäudestruktur lediglich ∼2m voneinander entfernt sind und eine gleiche Kantenausrichtung besitzen, fälschlicherweise die südlichere Option gewählt. Der Grund für diese Fehlentschei-

dung ist die Ausgangsposition der Suche, welche in diesem Fall minimal näher an dem südlichen Weg liegt. Diese Problematik würde bei weniger dichten Graphen nicht bestehen, da Entscheidungen aufgrund der Distanz zu einem Kantenkandidaten eindeutiger getroffen werden können. Die darauffolgende Suche (II) muss bereits nach einer Kante aufgrund des falschen Weges nach der ersten Suche (I) wieder aufgerufen werden. Deswegen basiert der neue Ausgangspunkt der Suche (II), wie in Kapitel 6.10.1 beschrieben, auf der alten plus den neuen Suchschritten. Das Ergebnis des zweiten Suchvorganges wählt eine Kante auf der Stiege. Diese Positionslösung ist jedoch in Bezug auf die eigentliche Trajektorie des Fußgängers gänzlich falsch.

7.3 Evaluierung Map-Matching

Zu Beginn dieses Kapitels wird die Problematik der Kreuzkorrelation für das Map-Matching-Verfahren in Abbildung 7.10 beschrieben. Die linke Trajektorie in Rot entspricht nicht dem Verlauf der Referenztrajektorie, trotzdem liefert die Kreuzkorrelation bei dieser Kantenabfolge den höchsten Peak aller möglichen Kantenabfolgen. Dies ist darauf zurückzuführen, dass bei einem Verschieben des roten Krümmungsverlaufs um fünf Schritte nach links, die Krümmungen dieser Trajektorie bei der Berechnung des Korrelationspeaks stärker ausfallen als die Krümmungen bei dem maximalen Kreuzkorrelationsergebnis der grünen Trajektorie, Bild rechts. Dementsprechend werden bei der Kreuzkorrelation des roten Krümmungsverlaufs höhere Werte mit den Krümmungen der Referenztrajektorie multipliziert. Dies führt zu einem größeren Korrelationspeak.

Für eine robustere und eindeutigere Aussage bezüglich der am besten zutreffenden Kantenabfolge wird im Anschluss an die Kreuzkorrelation eine Helmert-Transformation durchgeführt. In den meisten Fällen liefert diese Methode ein zweifelsfreies Ergebnis. Da jedoch die konstruierten Graphen in den Gebäuden im Verhältnis zu beispielsweise einem Straßennetz viel dichter und engmaschiger sind, liegen verschiedene Kreuzungsbereiche mit gleichen Abbiegerichtungen viel näher aneinander. Dementsprechend kann, wie in Abbildung 7.11 ersichtlich, die Helmert-Transformation die richtige Kantenabfolge (grün) nicht eindeutig identifizieren. Die Form der drei Routen aus Abbildung 7.11 ist annähernd ident, einzig in der Länge unterscheiden sie sich. Der geschätzte Maßstabsfaktor in der Helmert-Transformation dient eigentlich dazu, die Fehler, welche durch die Generalisierung der Kantenabfolgen oder durch falsche Schrittlängen entstehen, zu kompensieren. In diesem Fall gleicht er jedoch die unterschiedlichen Routenlängen dahingehend aus, dass

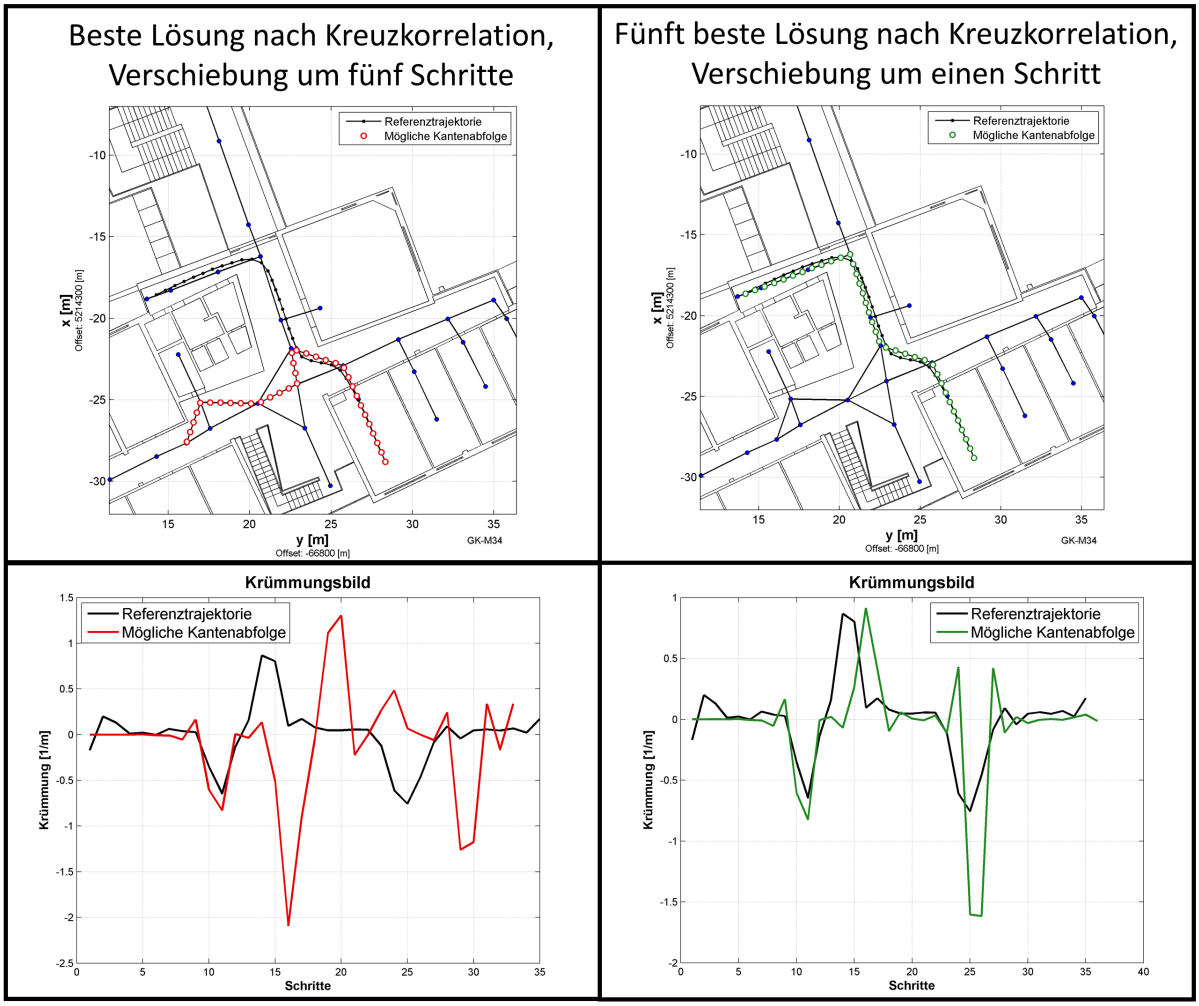


Abbildung 7.10: Problem Kreuzkorrelation - stärkere Krümmungen erzielen höheren Peak

alle drei Routen nach der Helmert-Transformation nahezu gleiche Quadratsummen der Verbesserungen aufweisen. Wären die drei Abbiegungen weiter voneinander entfernt, wie z.B. bei Straßennetzen, könnten die rote und die blaue Route, durch die in Kapitel 6.10.2 vorgestellten Plausibilitätsabfragen, ausgeschlossen werden.

Die Ergebnisse des Map-Matching-Verfahrens für den Zustand 5 wurde hauptsächlich in der Inffeldgasse 16 getestet, da in der Steyrergasse 30 überwiegend stockwerksübergreifende Trajektorien gemessen wurden. Dementsprechend kann dieses Verfahren nur selten in der Steyrergasse 30 genutzt werden, da keine k-shortest-path Routen als mögliche Kantenabfolgen für das Map-Matching über mehrere Etagen möglich sind. Unter den am Ende in Kapitel 6.10.2 vorgestellten Bedingungen lassen sich durch das Map-Matching-Verfahren sechs von neun Testtrajektorien in der Inffeldgasse 16 erfolgreich auf

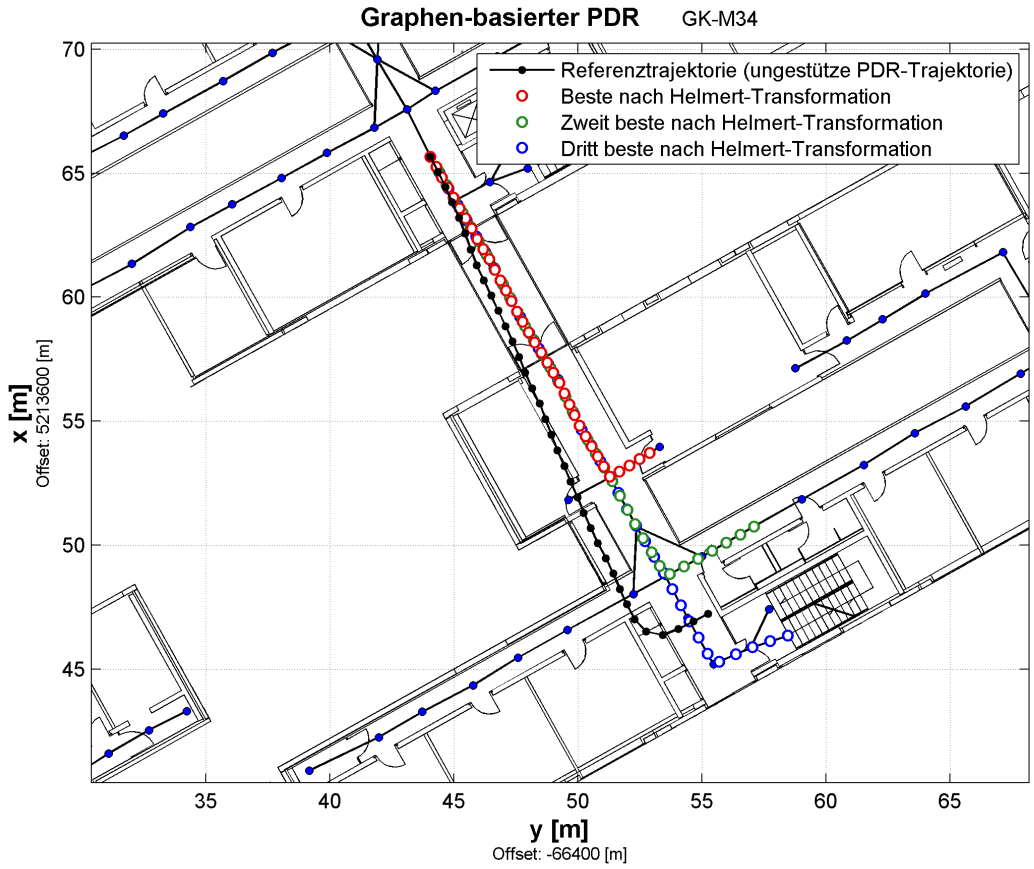


Abbildung 7.11: Mehrdeutiges Ergebnis der Routenauswahl durch eine Helmert-Transformation

den Graphen zwingen, sodass die angestrebte Zielposition des Fußgängers erreicht werden kann. Die produzierten Trajektorien auf dem Graphen entsprechen bis auf einzelne wenige unterschiedliche Kanten jenen aus dem Standardsuchverfahren aus Kapitel 6.10.1. Kann eine Bedingung für ein robustes Map-Matching nicht erfüllt werden, greift der Algorithmus auf das Standardverfahren zurück.

Trotz der unter diesen Umständen robusten Ergebnissen durch das Map-Matching wird primär der Zustand 5 über das Auswahlkriterium Richtungsabweichung δ und Distanz umgesetzt. Diese Methode ist wesentlich simpler, besitzt keine Einschränkungen und ist weniger rechenintensiv. Falls jedoch dieser Ansatz, z.B. aufgrund der in Abbildung 7.9 vorgestellten Problematik, keine sinnvolle Trajektorie mehr repräsentiert, kann auf das Map-Matching-Verfahren als Backup zurückgegriffen werden.

7.4 Drehraten - Kalibrierung

Neben der Schritterkennung ist vor allem die Richtungsabweichung δ aus den Drehratenmessungen für die Auswahl der Kanten verantwortlich. Dementsprechend untersucht dieses Kapitel, welche Auswirkung kalibrierte oder unkalibrierte Drehraten auf die resultierende Graphen-basierte Trajektorie haben. Die Abbildung 7.12 visualisierte Kanten in Grün wenn der Algorithmus die Kante innerhalb der fünf Schritte identifiziert hat. Musste eine globale/lokale Suche mit dem Zustand 5 gestartet werden, um eine Kante der aktuellen Bewegungsrichtung zuzuordnen, wurde die Kante rot eingefärbt.

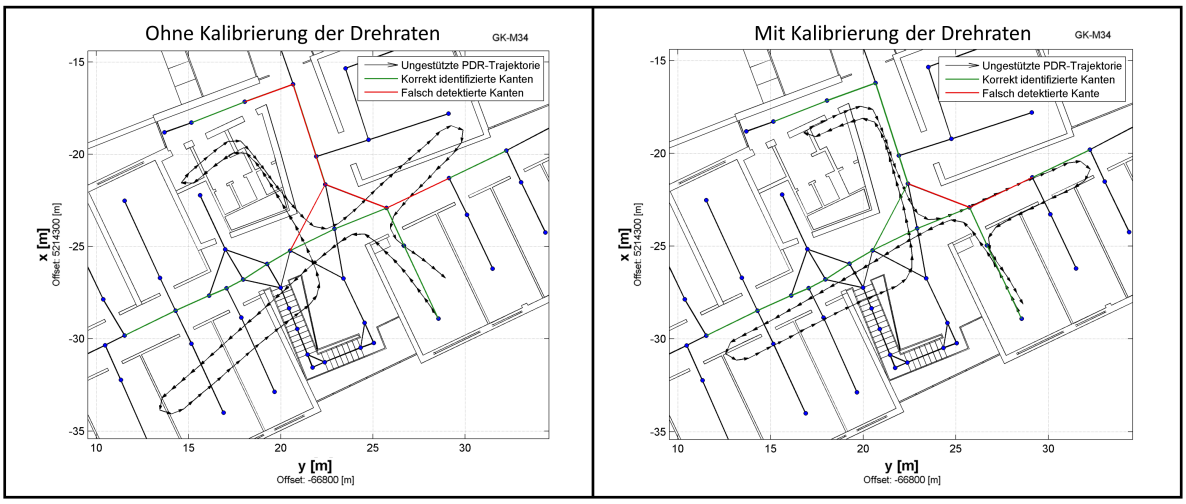


Abbildung 7.12: Vergleich der Trajektorien mit kalibrierten und unkalibrierten Drehraten

Die Trajektorie auf dem Graphen welche, durch unkalibrierte Drehraten berechnet wurde, kann sechs von 23 Kanten (26,1%) nicht innerhalb des Suchfensters identifizieren. Dem entgegengesetzt liefern kalibrierte Drehraten ein wesentlich besseres Ergebnis. Dabei konnten lediglich zwei von 23 Kanten (9,5%) nicht innerhalb der fünf Suchschritte zugeordnet werden. Eine weitere Analyse von insgesamt 15 Trajektorien aus den beiden Testumgebungen ist in Tabelle 7.3 festgehalten.

Die durchschnittliche Länge der 15 Testtrajektorien erstrecken sich über 326m, was ungefähr einem Messzeitraum von fünf Minuten entspricht. Dabei wurden im Mittel 128 Kanten ausgewählt und 13 Abbiegungen getätigt. Insgesamt mussten bei unkalibrierten Drehraten durchschnittlich zweimal öfter die globale/lokale Suche gestartet werden. Im Verhältnis zu allen gewählten Kanten können somit um zwei Prozent mehr Kanten innerhalb der fünf

Tabelle 7.3: Vergleich zwischen Trajektorien mit und ohne kalibrierten Drehraten anhand von 15 Testtrajektorien

∅ aus 15 Trajektorien	
Zeit [min]	5
Länge [m]	326
Kanten	128
Anzahl Abbiegungen	13
Globale/Lokale Suchen mit Kalibrierung	5
Globale/Lokale Suchen ohne Kalibrierung	7

Suchschritte eindeutig zugeordnet werden. Dementsprechend trägt eine Kalibrierung der Drehraten zu einer verbesserten Lösung des Algorithmus bei.

7.5 Vergleich Partikelfilter und Graphen-basierter PDR

Der abschließende Vergleich in Abbildung 7.13 beschreibt, abgesehen von der ungestützten PDR Lösung, eine Smartphone-basierte Trajektorie in der Steyrergasse 30 auf Basis zweier unterschiedliche Berechnungsmethoden. Einerseits visualisieren die roten und grünen Kreise die Positionslösungen des entwickelten Graphen-basierten PDR-Algorithmus und andererseits wurde die Trajektorie mit Hilfe eines Partikelfilters ausgewertet (violette Linie). Dieser Filter benützt als Basis, gleich wie der Graphen-basierte Ansatz, einen Schritt-basierten PDR. Die Berechnungssoftware für die Trajektorie aus dem Partikelfilter wurde von der Arbeitsgruppe Navigation am Institut für Geodäsie bereitgestellt (Hafner [8]).

Beide Berechnungsmethoden können Gebäudestrukturen in der Positionsschätzung mit einfließen lassen. Die Stützung durch Karteninformationen bei einem Partikelfilter basieren auf Rasterdaten. Diesbezüglich kann jedes georeferenzierte Pixel der Rasterkarte als Partikel definiert werden und besitzt somit eine Information über die Lage innerhalb des Gebäudes. Dadurch können die Informationen von Eingängen, Wänden, Korridoren oder Räumen in die Schätzung der Position mit einfließen. Die Wirkungsweise des Partikelfilters kann am nördlichsten Punkt der Trajektorie im Raum (a) erkannt werden. Die Messdaten

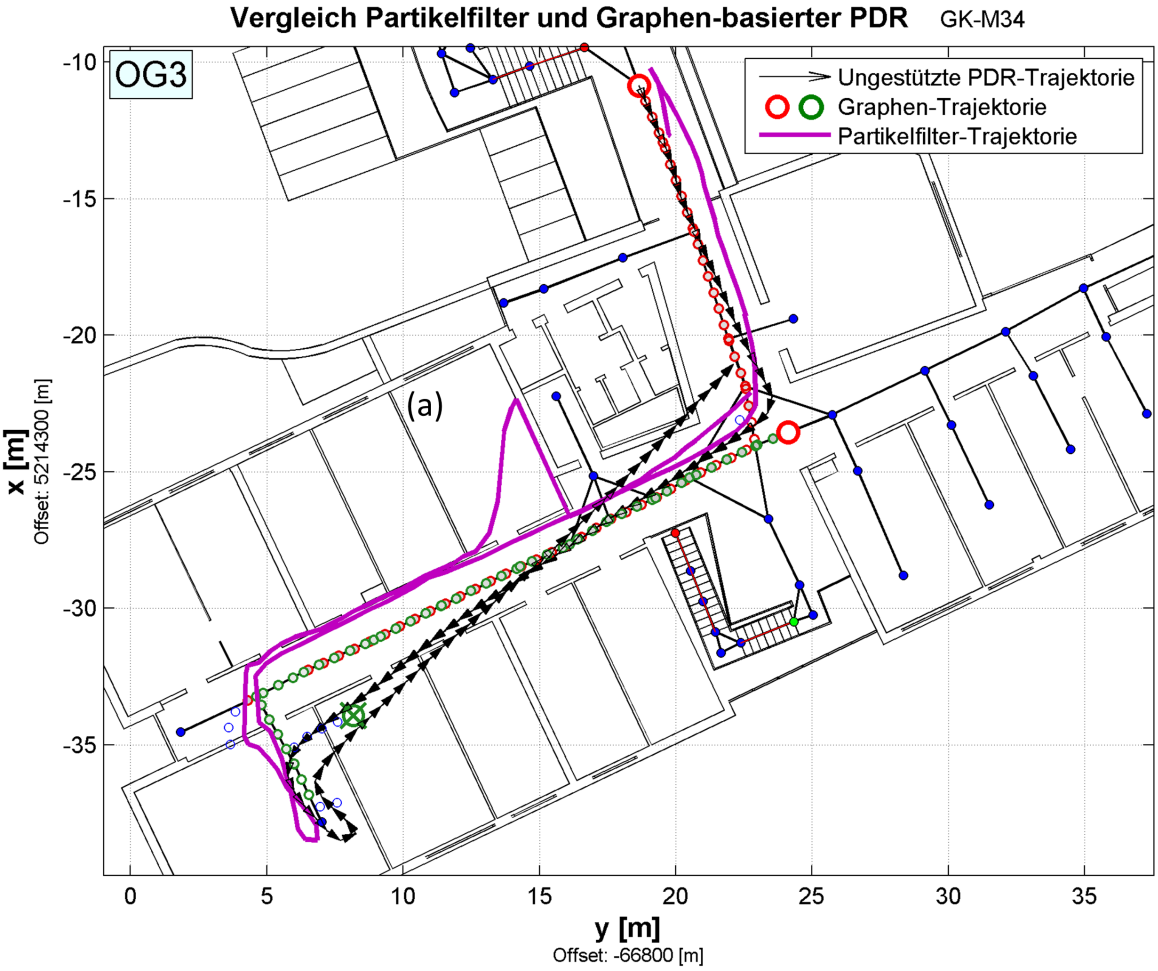


Abbildung 7.13: Vergleich der Trajektorie aus einem Parikelfilter und aus dem Graphen-basierten PDR Ansatz

würden eine Bewegung durch die rechte Wand beschreiben. Aufgrund der wandgestützten Positionsschätzung kann die Trajektorie jedoch nicht in diese Richtung fortgeführt werden. Die einzige Möglichkeit, dieser Bewegung eine Position zuzuordnen, ist somit wieder am Gang. Widersprüchliche Trajektorien außerhalb des Gebäudes oder durch Wände, wie in der ungestützen PDR-Lösung aus Abbildung 7.13, können auf diese Weise vermieden werden. Wenn die Partikelfilter-Trajektorie aus der Abbildung die Gebäudestruktur kreuzt, ist dies lediglich darauf zurückzuführen, dass die einzelnen Positionslösungen mit einer Linie verbunden wurden. Wurde eine Position, wie jene im Raum (a) falsch geschätzt und zum nächsten Zeitpunkt wieder durch eine Position am Gang korrigiert, führt zwangsläufig die violette Linie durch die Wände.

Bei dem Graphen-basierten Ansatz fließt die Karteninformation indirekt durch die Kon-

struktion des Graphen mit ein, da dieser aufbauend auf einem Gebäudeplan erstellt wird. Der Vorteil des entwickelten Algorithmus besteht darin, dass aufgrund der Restriktion auf die vordefinierten Knoten und Kanten keine sinnwidrigen Trajektorien durch die Gebäudestruktur und keine Ausreißer in der Position möglich sind. Diese Eigenschaft ist jedoch nur bei kleinräumigen und schmalen Gebäudestrukturen von Vorteil. Bei großen und freien Flächen, in denen der Fußgänger stark von dem vordefinierten Weg des Graphen abweichen kann, erreicht der Algorithmus seine Grenzen. Ein weiterer großer Vorteil von Graphenbasierten Algorithmen besteht in der Kombination mit Routing und Guidance Operationen. Diese beiden elementaren Methoden der Navigation benötigen für ihre Funktionsweise einen zugrundeliegenden Graphen. Dadurch können beispielsweise eine Manöverliste aus dem Routing und topologische Informationen für das Guidance abgeleitet werden. Die direkte Position auf diesem Graphen durch den entwickelten Ansatz ersetzt somit das konventionelle Map-Matching. Des Weiteren kann beispielsweise das Treppensteigen des Fußgängers einfach über die semantische Information von den zuvor definierten Stiegenkanten abgefragt werden und muss nicht über eine Schrittmustererkennung oder ähnliche Verfahren zusätzlich überprüft werden.

Die Positionen aus einem Partikel lassen sich nicht direkt für Routing- und Guidanceaufgaben verwenden, da sie nicht mit einem Graphen gekoppelt sind. Dementsprechend kann ein Partikelfilter alle Positionen innerhalb eins Gebäudes einnehmen. Dies ist aufgrund der starken Generalisierung des Graphen in Bezug auf die Bewegungsfreiheit des Fußgängers von Vorteil, jedoch kann es bei ungünstigen Konstellationen wie beispielsweise im Raum (a), siehe Abbildung 7.13, in Bezug zu dem Gang zu falschen Positionslösungen kommen. Im Gegensatz zu der komplexen Implementierung und dem hohen Rechenaufwand liegt der Vorteil des Partikelfilters darin begründet, dass er widersprüchliche Positionslösungen aufgrund ihrer geringen Wahrscheinlichkeit in Bezug auf die Messdaten und des Bewegungsmodells im weiteren Verlauf erkennt. Aus der Vielzahl an gewichteten Partikeln können somit zu jeder Zeit alternative Routen beibehalten und hinsichtlich ihrer Relevanz auf die aktuelle Position des Fußgängers überprüft werden. Der Graphen-basierte Ansatz besitzt keine Informationen über Alternativrouten oder über Wahrscheinlichkeiten alternativer Positionen. Liefert dieser Algorithmus somit keine richtigen Lösungen in Bezug auf die tatsächliche Position des Fußgängers, kann der Algorithmus sich nicht mehr ohne zusätzliche Hilfsmittel korrigieren.

Kapitel 8

Zusammenfassung und Ausblick

Zusammenfassung
Smartphones besitzen eine Vielzahl an Sensoren die sich für eine Indoor-Positionslösung eignen. Neben infrastrukturabhängigen Indoor-Positionierungsmethoden, welche kosten- und wartungsintensiv sind, besitzen Smartphones die Möglichkeit durch Beschleunigungs- und Drehratensensoren autonome Positionsschätzungen durch einen Schritt-basierten PDR durchzuführen. Die möglichst kleinen und kostengünstigen Messeinheiten weisen jedoch nur eine geringe Präzision und Sensitivität auf, da sie nicht für navigationsspezifische Zwecke konzipiert worden sind. Dies gibt dazu Anlass, die PDR-Lösungen durch zusätzliche Verfahren zu stützen.

In dieser Arbeit wurde zur Verbesserung der PDR-Lösung ein Graphen-basierter PDR-Algorithmus entwickelt. Dieser benützt mit Hilfe eines auf einem Gebäudeplan entworfenen Graphen Karteninformationen, um die Positionslösungen zu verbessern. Die resultierende Trajektorie wird über fünf Zustandsabfragen, welche auf der momentanen Richtungsabweichung in Bezug auf die Kantenausrichtung und der aktuellen Position auf der Kante beruhen, auf den Graphen gezwungen.

Durch diese Restriktion konnten widersprüchliche Trajektorien, die durch Wände oder aus dem Gebäude führen, ausgeschlossen werden. Eine weitere wesentliche Verbesserung in Bezug zur ungestützten Schritt-basierten PDR-Lösungen wurde durch die Driftkompensierung der Drehraten erzielt, da diese nicht über den gesamten Messezeitraum aufkumuliert werden, sondern nur über die Dauer einer Kante. Dementsprechend ist es durch den entwickelten Graphen-basierten PDR-Algorithmus möglich, die Trajektorie des Fußgängers trotz der relativen Positionierungsmethode über einen langen Zeitraum korrekt wiederzugeben. Die längste gemessene Testtrajektorie erstreckte sich dabei über 16min. Die tatsächliche

Endposition des Fußgängers konnte dabei, wie auch bei weiteren 34 von 40 Messungen, durch den Algorithmus erfolgreich erreicht werden.

Eine weitere Erkenntnis dieser Arbeit liegt darin begründet, dass sich Stockwerkswechsel eindeutig und ohne Ausreißer durch Barometermessungen bestimmen lassen. Lediglich die Festlegung der Startposition in einem neuen Stockwerk kann unter bestimmten Umständen, wie z.B. einem Stockwerkswechsel während einer Abbiegung, zu Problemen führen.

Generell kann schlussgefolgert werden, dass sich der entwickelte Algorithmus sehr gut für Testumgebungen eignet, bei denen die Bewegungsfreiheit des Fußgängers beispielsweise auf Grund schmaler Gänge oder bei taktilen Bodenleitsystemen in der Blindennavigation eingeschränkt ist. Bei großen freien Flächen, in denen die Bewegung des Fußgängers deutlich von den generalisierten Laufwegen des Graphen abweichen kann, erreicht der Algorithmus seine Grenzen. Ebenso hängt die Navigationsperformance stark von den konstruierten Kanten (Länge und Ausrichtung) und Kreuzungsbereichen ab. Dementsprechend ist ein solider definierter Graph essentiell für den Algorithmus.

Ausblick

Der Algorithmus konnte bislang noch nicht anhand einer Referenztrajektorie bezüglich seiner Genauigkeit überprüft werden. Dementsprechend wäre ein Vergleich der einzelnen Schrittlösungen zu der wahren Position erstrebenswert, um weitere Analysen möglich zu machen. Vertiefende Untersuchungen könnten auch im Übergangsbereich zwischen Outdoor und Indoor durchgeführt werden. Verbesserte GNSS-Positionen ermöglichen somit eine zusammenhängende Trajektorie durch verschiedene Gebäude und Freiluftbereiche. Da bis jetzt in dieser Arbeit eine *hand-held* Tragweise vorausgesetzt wird, könnten Verfahren implementiert werden, die andere Smartphone-Tragweisen ermöglichen würden. Weitere Untersuchungen könnten ebenso in der Kombination des Algorithmus mit anderen Positionierungsmethoden wie z.B. einem Bluetooth/WLAN Fingerprinting oder einem Partikelfilter angestrebt werden. Eine entsprechende Filterung kann dann aus der Vielzahl an Positionsinformationen die bestmögliche Position des Fußgängers schätzen. Somit behält man die Vorteile des Graphen-basierten Ansatzes, jedoch ist man nicht mehr auf die Knoten und Kanten des Graphen eingeschränkt. Dadurch wären freie Positionslösungen innerhalb des Gebäudes möglich. Ebenso könnte bei einer Kombination mit absoluten Positionsinformationen fehlerhafte Lösungen des Graphen-basierten PDR-Algorithmus kompensiert werden.

Literaturverzeichnis

[1] S. Banville und F. V. Diggelen. *Innovation: Precise positioning using raw GPS measurements from Android smartphones.* GPS World, Juli 2016. URL: http://gpsworld.com/innovation-precise-positioning-using-raw-gps-measurements-from-android-smartphones/ (letzter Zugriffszeitpunkt am: 16.05.2017).

[2] E. Bauer. *Indoor-Positionierung mit Inertialsensoren.* Masterarbeit, Technische Universität Graz, Juli 2014.

[3] A. Brajdic und R. Harle. *Walk Detection and Step Counting on Unconstrained Smartphones.* UbiComp, Zürich, September 2013.

[4] R. Chen und R. E. Guinness. *Geospatial Computing in Mobile Devices.* Artech House, 2014.

[5] R. Czommer. *Leistungsfähigkeit fahrzeugautonomer Ortungsverfahren auf der Basis von Map-Matching-Techniken.* Doktorarbeit, Universität Stuttgart, 2000.

[6] P. D. Groves. *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems.* Artech House, 2008.

[7] S. K. Gupta, S. Box, und R.E.Wilson. *Low cost infrastructure free form of indoor positioning.* International Conference on Indoor Positioning and Indoor Navigation, Oktober 2014.

[8] P. Hafner. *Development of a Positioning Filter for the Smartphone-based Navigation of Visually Impaired People.* Doktorarbeit, Technische Universität Graz, 2015.

[9] B. Hofmann-Wellenhof, K. Legat, und M. Wieser. *Navigation - principles of positioning and guidance.* Springer-Verlag Wien, 2003.

[10] B. Hofmann-Wellenhof, H. Lichtenegger, und E. Wasle. *GNSS - Global Navigation Satellite Systems GPS, GLONASS, GALIEO, and more.* Springer-Verlag Wien, 2008.

[11] B. Mayerhofer. *Map-Matching in der Fußgängernavigation für blinde Personen.* Masterarbeit, Technische Universität Graz, Juni 2005.

[12] T. Moder, C. Reitbauer, M. Dorn, und M. Wieser. *Calibration of Smartphone Sensor Data Usable for Pedestrian Dead Reckoning.* (unveröffentlicht), 2017.

[13] G. Navratil und M. Staudinger. *Ausgleichungsrechnung II.* Technische Universität Wien, Dezember 2006.

[14] J. Otter, N. Höggerl, E. Imrek, G. Stangl, und E. Zahn. *3-D Referenzsystem in Österreich.* BEV - Bundesamt für Eich- und Vermessungswesen, Wien, 2015.

[15] Teardown.com. *Teardown: Samsung crams a bevy of sensors in Galaxy S5.* globalsources, April 2014. URL: http://www.globalsources.com/NEWS/teardown-samsung-galaxy-S5.HTM (letzter Zugriffszeitpunkt 16.05.2017).

[16] S. Thrun, W. Burgard, und D. Fox. *PROBABILISTIC ROBOTICS.* MIT Press, 2006.

[17] D. Titterton und J. Weston. *Strapdown inertial navigation technology.* The Institution of Electrical Engineers, American Institut of Aeronautics and Astronautics, 2004.

[18] F. Wild-Pfeiffer und B. Schäfer. *MEMS-Sensoren, auch für die Geodäsie.* Zeitschrift für Vermessungswesen, Jänner 2011.

[19] R. Wilfinger. *Absolute positioning of vehicles in indoor environments using Wireless LAN and Bluetooth LE.* Masterarbeit, Technische Universität Graz, April 2015.

[20] T. Willemsen. *Fusionsalgorithmus zur autonomen Positionsschätzung im Gebäude, basierend auf MEMS-Inertialsensoren im Smartphone.* Doktorarbeit, HafenCity Universität, 2016.

[21] T. Willemsen, F. Keller, und H. Sternberg. *A Topological Approach with MEMS in Smartphones based on Routing-Graph.* International Conference on Indoor Positioning and Indoor Navigation, Oktober 2015.

[22] www.elektronik-kompendium.de. *MEMS - Micro-Electro-Mechanical Systems.* URL: https://www.elektronik-kompendium.de/sites/bau/1503041.htm (letzter Zugriffszeitpunkt 16.05.2017).

[23] www.howtomechatronics.com. *MEMS Accelerometer Gyroscope Magnetometer and Arduino*, November 2015. URL: http://howtomechatronics.com/how-it-works/electrical-engineering/mems-accelerometer-gyrocope-magnetometer-arduino/ (letzter Zugriffszeitpunkt 16.05.2017).

[24] www.ngdc.noaa.gov. *Magnetic Field Calcultors*. URL: https://www.ngdc.noaa.gov/geomag-web/ (letzter Zugriffszeitpunkt 16.05.2017).

[25] www.samsung.com. *Samsung Galaxy S4*. URL: http://www.samsung.com/at/consumer/mobile-devices/smartphones/galaxy-s/GT-I9505ZKAATO/ (letzter Zugriffszeitpunkt 16.05.2017).

[26] J. Y. Yen. *Finding the K Shortest Loopless Paths in a Network*. Management Science, 1971.

Abbildungsverzeichnis

Tabellenverzeichnis