Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Kai-Uwe von Deylen

Laserscanner basierte Offline-Kartierung zur Online-Lokalisierung im Straßenverkehr

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Kai-Uwe von Deylen

Laserscanner basierte Offline-Kartierung zur Online-Lokalisierung im Straßenverkehr

Masterarbeit eingereicht im Rahmen der Masterprüfung

im Studiengang Master of Science Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Korf
Zweitgutachter: Prof. Dr.-Ing. Meisel

Eingereicht am: 21.02.2017

Kai-Uwe von Deylen

Thema der Arbeit
Laserscanner basierte Offline-Kartierung zur Online-Lokalisierung im Straßenverkehr

Stichworte
Feature Detektion, Kartierung, SLAM, Lokalisierung, Partikelfilter, Laserscanner

Kurzzusammenfassung
Das hochautomatisierte Fahren erfordert die Wahrnehmung des Umfelds sowie die Bestimmung der Eigenpose in diesem Umfeld. In dieser Arbeit wird zunächst ein Verfahren zur Erkennung von Keypoints in 2D Laserscannerdaten vorgestellt. Diese Keypoints werden in einem GraphSLAM-basierten Verfahren zur Bestimmung der Fahrzeugtrajektorie verwendet, die dem Erstellen einer Gridkarte als Umgebungsabbild dient. Außerdem wird eine Lokalisierung mittels Partikelfilter auf ebendiesen Gridkarten vorgestellt. Diese Verfahren werden jeweils in typischen Szenarien des Straßenverkehrs evaluiert.

Kai-Uwe von Deylen

Title of the paper
Lidar based offline mapping and online localisation for road vehicles

Keywords
feature detection, mapping, SLAM, localization, particle filter, LIDAR

Abstract
The estimation of a vehicle's environment and its position in the environment are essential for highly automated driving applications. The contribution of this thesis consists of three elements: First, a keypoint detection method for 2D lidar data is proposed, that detects small freestanding objects in the environment. Second, these keypoints are then used in a GraphSLAM based algorithm to estimate the vehicle's trajectory, which in turn is used to build a gridmap based on the lidar data. Third, a particle filter is used to localize the vehicle in such gridmaps. The proposed methods are evaluated in typical road scenarios.

Inhaltsverzeichnis

1. Einleitung

Das hochautomatisierte Fahren ist sowohl im wissenschaftlichen als auch im industriellen Bereich ein aktuelles Forschungsgebiet, das sich über eine Vielzahl von Disziplinen und Themen erstreckt. Dazu gehört die Fähigkeit eines Fahrzeugs, das eigene Umfeld wahrzunehmen und zu interpretieren, das Fahrzeug in Relation zu dieser Umgebung zu setzen sowie auf Basis dieser Informationen weitere Aktionen zu planen und zu steuern.

Einen Teilbereich und zugleich Voraussetzung für viele weiterführende Anwendungen des automatisierten Fahrens bilden die Kartierung des Umfelds sowie die Lokalisierung des Fahrzeugs relativ zur Karte (vgl. [SNS11, ZGDH14]). Zur Umfelderfassung wird eine Vielfalt von Sensoren, wie bspw. Laserscanner, Radar oder Kameras, eingesetzt. Die so gewonnenen Informationen werden typischerweise in eine gemeinsame Umgebungskarte fusioniert, die den weiteren Teilsystemen, wie z. B. der Fahrplanung, dient (vgl. [BH08]). Darüber hinaus finden die Kartierung und die Lokalisierung in Bereichen wie energieeffizientes Fahren, Fahrerassistenzsysteme oder Referenzsysteme Anwendung.

Die Lokalisierung relativ zu einer bestehenden Karte erfolgt durch die Verfolgung der Fahrzeugpose über die Zeit. Hierzu werden i. d. R. Kalman-Filter oder Partikelfilter genutzt, deren Zustandsabbilder mittels Bewegungsmodell des Fahrzeugs prädiziert und durch Messungen der Sensorik aktualisiert werden.

Das Erzeugen der Umgebungskarte und die gleichzeitige Lokalisierung relativ zu dieser wird als SLAM (Simultanous Localization and Mapping) bezeichnet. Die Schwierigkeit besteht darin, dass die Fahrzeugpose relativ zur Karte bestimmt werden soll, während gleichzeitig die Karte anhand der ermittelten Posen zu aufzubauen ist. Das SLAM Problem gilt es u. a. dann zu Lösen, wenn eine Karte erzeugt werden soll, aber die

Fahrzeugposen unbekannt oder zu ungenau sind, wie es bspw. bei der Verwendung von GPS der Fall ist.

Zielsetzung

Die Ziele der vorliegenden Arbeit sind die Entwicklung eines Mappingverfahrens sowie eines Verfahrens zur Lokalisierung auf einer daraus resultierenden Karte. Abbildung 1.1 zeigt den Ablauf: Zunächst werden bei der sog. Mappingfahrt die Messdaten zu einer bestimmten Umgebung aufgezeichnet. Diese werden anschließend in dem Mappingverfahren verarbeitet, um so die Karte zu erzeugen. Bei sog. Lokalisierungsfahrten wird diese Karte verwendet, um die Eigenpose des Fahrzeugs relativ zu der Karte zu schätzen. Durch die Integration von GPS Messungen während der Kartenerzeugung lassen sich die Karte und damit auch die Eigenpose während der Lokalisierungsfahrt global referenzieren. Sowohl Mapping als auch Lokalisierung sollen im zweidimensionalen Raum erfolgen.

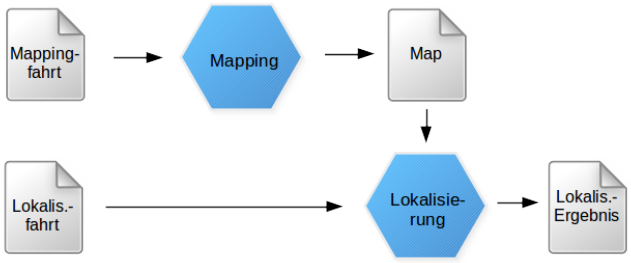


Abbildung 1.1.: Ablauf der Kartenerstellung und Lokalisierung.

Die Rahmenbedingungen für das Mapping sind Folgende: Es sollen die Daten der werkseitig vorhandenen Fahrzeugodometrie, eines Inertial Navigation Systems (INS) sowie der am Versuchsfahrzeug verbauten Laserscanner genutzt werden. Es sollen Eigenschaften (Features) in den Messdaten der Laserscanner erkannt werden, die anschließend im SLAM als Messungen der Umgebung dienen. Die resultierende Karte, die zur späteren Lokalisierung dient, soll anhand der Trajektorie erzeugt werden und die gesamte statische Umgebung unabhängig von den Features abbilden. Darüber hinaus ist es denkbar, dass später weitere Eigenschaften, wie bspw. Fahrspuren, in die Karte integriert werden. Um diese ohne erneute Lokalisierung richtig in der Karte zu

positionieren, ist es notwendig im Mappingverfahren die gesamte Trajektorie mit allen Fahrzeugposen zu schätzen.

Die spätere Lokalisierung in Lokalisierungsfahrten soll relativ zur erstellten Karte erfolgen, indem die Messungen der Laserscanner mit der Karte abgeglichen werden. Hier ist keine Bestimmung der gesamten Trajektorie erforderlich, sodass stets nur der aktuelle Zustand zu schätzen ist. Es können zusätzlich zu den Laserscannerdaten sowohl die Fahrzeugodometrie als auch das INS genutzt werden. Dies entspricht ebenfalls der Ausstattung der Versuchsfahrzeuge des *Projekt Harz* (siehe nachfolgender Abschnitt „Einbindung in Forschungsprojekte“).

Beide Verfahren sollen prototypisch innerhalb der bestehenden Ibeo Software Umgebung implementiert werden. Die Ergebnisse der Implementierung sollen evaluiert werden. Hierbei ist auch ein Vergleich zu anderen Lokalisierungmöglichkeiten wie GPS/IMU oder RTK-Systemen herzustellen.

Einbindung in Forschungsprojekte

Diese Arbeit wurde im Rahmen des *Projekt Harz* und des *HIGTHS* Projektes verfasst.

Das Projekt Harz [Ibe16] ist ein Projekt der Ibeo Automotive Systems GmbH, dessen Ziel die Entwicklung eines Prototypen zum hochautomatisierten Fahren[1] im Harzgebirge ist. Das Gebiet wurde bewusst als Zielumgebung gewählt, da es durch kurvenreiche Streckenverläufe, das Höhenprofil sowie teilweise schlechte oder nicht vorhandene Fahrbahnmarkierungen besondere Herausforderungen für die Teilsysteme der Fahrautomatisierung darstellt.

HIGHTS [HIG15] ist ein europäisch gefördertes Projekt mit dem Ziel ein Positionierungssystem mit einer Genauigkeit von mindestens 25cm zu entwickeln. Hintergrund dieses Ziels ist, dass vielen Anwendungen im Bereich intelligenter Transportsysteme die geografische Position der Fahrzeuge mit entsprechender Genauigkeit bekannt sein muss. Satellitengestützte Navigationslösungen wie GPS und Galileo können diese An-

[1] Begriffsdefinition der Bundesanstalt für Straßenwesen bzgl. Grad der Fahrautomatisierung siehe [GAA+12]

forderungen aber in vielen typischen Szenarien, z. B. Häuserschluchten, nicht erfüllen. An dieser Stelle setzt die vorliegende Arbeit an.

Gliederung

Diese Arbeit ist in folgende Kapitel gegliedert:

Kapitel 1 leitet die vorliegende Arbeit ein, definiert die Zielsetzung und beschreibt die Gliederung.

Kapitel 2 umfasst die benötigten **Grundlagen** zur vorliegenden Arbeit.

Kapitel 3 stellt **verwandte Arbeiten** vor. Es werden Arbeiten zu den Themen Featuredetektion in Laserscannerdaten sowie zu Kartierung und Lokalisierung mittels Laserscannern betrachtet.

Kapitel 4 beschreibt die **Keypoint Detektion**. Zunächst werden die Anforderungen aufgestellt und anschließend unterschiedliche Detektoren bezüglich der Anforderungen diskutiert. Daraufhin werden der in dieser Arbeit entwickelte Algorithmus und dessen Umsetzung vorgestellt. Das Kapitel schließt mit einer Evaluation des Algorithmus.

Kapitel 5 behandelt das Thema **Mapping**. Die Anforderungen an das SLAM-Verfahren der vorliegenden Arbeit werden definiert und das in dieser Arbeit umgesetzte SLAM-Verfahren sowie dessen Implementierung werden beschrieben. Außerdem wird die Erzeugung der Gridkarten, die der späteren Lokalisierung dienen, beschrieben und das SLAM-Verfahren evaluiert.

Kapitel 6 erläutert die **Lokalisierung**. Dazu wird auf den Partikelfilter zur Lokalisierung eingegangen und dessen Implementierung vorgestellt. Zuletzt wird dieser Lokalisierungsansatz ebenfalls evaluiert.

Kapitel 7 schließt die Arbeit mit einer **Zusammenfassung**, sich ergebenden **Schlussfolgerungen** sowie offen gebliebenen Punkten und einem **Ausblick** auf zukünftige Arbeiten ab.

2. Grundlagen

Dieses Kapitel behandelt die benötigten Grundlagen zur vorliegenden Arbeit. Zunächst werden in Abschnitt 2.1 die verwendeten Koordinatensysteme definiert. In Abschnitt 2.2 werden die Grundlagen zur Abbildung des Umfelds vermittelt. Dies umfasst Feature-basierte und Grid-basierte Karten. Anschließend geht Abschnitt 2.3 auf die Zustandsverfolgung mittels Kalman-Filter sowie Partikelfilter ein. Abschnitt 2.4 beschreibt das SLAM-Problem und den GraphSLAM-Lösungsansatz. Das verwendete Versuchsfahrzeug wird in Abschnitt 2.5 und die verbauten Laserscanner sowie das INS in den Abschnitten 2.6 und 2.7 vorgestellt.

2.1. Verwendete Koordinatensysteme

In der vorliegenden Arbeit werden vier Koordinatensysteme verwendet:

Weltkoordinaten Weltkoordinaten sind globale Koordinaten und werden im *WGS 84* [Nat00] Format angegeben.

Referenzkoordinaten Referenzkoordinaten sind in einem kartesischen Koordinatensystem gegeben, dessen Ursprung in Weltkoordinaten angegeben ist. Dieser wird in der vorliegenden Arbeit als Referenzpunkt bezeichnet. Die x-y-Ebene ist tangential zum WGS 84 Ellipsoid an der Stelle des Referenzpunktes, wobei die x-Achse nach Norden, die y-Achse nach Westen und die z-Achse senkrecht vom Ellipsoid abgehend ausgerichtet ist.

Fahrzeugkoordinaten Das Fahrzeugkoordinatensystem ist fix bezüglich der Fahrzeughinterachse. Der Ursprung ist definiert als senkrecht unter dem Mittelpunkt der

Fahrzeughinterachse auf Bodenhöhe. Die x-Achse ist zur Fahrzeugfront und die y-Achse senkrecht dazu in Fahrzeugblickrichtung links ausgerichtet. Der Ursprung wird stets als Pose in Referenzkoordinaten angegeben.

Sensorkoordinaten Sensorkoordinaten haben ihren Ursprung jeweils im Sensor, wobei die x-Achse in Sensorblickrichtung und die y-Achse senkrecht dazu in Sensorblickrichtung links liegt. Der Ursprung und die Sensorblickrichtung sind stets als Pose in Fahrzeugkoordinaten angegeben und werden nachfolgend als Anbauposition bezeichnet.

2.2. Umfeldmodellierung

Informationen über die Umgebung eines mobilen Roboters, in dieser Arbeit in Form eines KFZ, werden in digitalen Karten abgelegt. [TBF05] definiert eine Karte als abstraktes Modell des Umfelds, welches die Umgebung anhand einer Menge von Objekten oder Eigenschaften beschreibt:

$$m = \{m_1, m_1, \ldots, m_N\} \tag{2.1}$$

Ein solches Umgebungsmodell ist entweder Grid-basiert oder Feature-basiert. Grid-basierte Karten unterteilen den Raum in diskrete Abschnitte und speichern in jeder dieser Zellen die örtliche Karteninformation. Typischerweise werden die Zellen anhand der geografischen Position indiziert. In einer zweidimensionalen Karte stellt $m_{x,y} \in m$ die Zelle an der Position $\begin{pmatrix} x & y \end{pmatrix}^T$ dar. Feature-basierte Karten bilden die Umgebung hingegen nicht erschöpfend ab, sondern bestehen aus einer Menge von Merkmalen oder Eigenschaften mit Ortsangabe. Dabei stellt jedes $m_n \in m$ ein Feature mit dessen Eigenschaften und Position dar.

Darüber hinaus gibt es topologische Karten, die nicht die geometrischen Strukturen sondern deren Beziehungen zueinander modellieren. Beispielsweise ist ein Graph, dessen Knoten unterschiedliche Orte abbilden und dessen Kanten befahrbare Verbindungen zwischen den Orten darstellen, eine topologische Karte (vgl. [KB88]). Topologische Karten werden in der vorliegenden Arbeit nicht näher betrachtet. Nachfolgend wird auf

die Vor- und Nachteile von Feature- und Gridkarten eingegangen. Für darüber hinaus gehende Beschreibungen der Umfeldmodellierung wird auf [BH08] verwiesen.

Feature-basierte Karten

Es existieren diverse Arten von Features, die in einer Karte abgebildet werden können. Diese wurden im Projektbericht [vD16, Abschnitt 2.1] vorgestellt. Die Vorteile von Feature-basierten Karten sind ihre Speichereffizienz und die präzisen Positionsangaben der Features. Die hohe Präzision der Positionsangaben liegt darin begründet, dass zu jedem Feature die Koordinaten ohne Rasterung gespeichert werden. Dass nur relevante Informationen abgelegt werden und die Umgebung nicht erschöpfend abgebildet wird, hat die hohe Speichereffizienz zur Folge. Dies wirkt sich jedoch negativ auf die Zugriffseffizienz aus, da die Speicherposition nicht im Zusammenhang mit der Position des Features in der Umwelt steht.

Grid-basierte Karten

Grid-basierte Karten [ME85] unterscheiden sich untereinander primär in der Art der Information, die je Gridzelle abgelegt ist. So wird in einem Occupancy Grid jeder Zelle eine Belegtwahrscheinlichkeit zugeordnet, welche aussagt, ob die Zelle von einem Objekt belegt ist, wohingegen in einem Elevation Grid für jede Zelle die Höhe des Geländes angegeben ist. Die Vorteile von Gridkarten sind der effiziente Zugriff, da die Gridzellen über ihre geografische Position indiziert werden, sowie die erschöpfende Umgebungsabbildung, sodass für jede Position der Umgebung die vorhandenen Informationen oder auch die Absenz von Informationen gespeichert ist. Dies führt jedoch zu einem hohen Speicherverbrauch, da für jede Gridzelle im abgebildeten Bereich Speicher benötigt wird. Ein weiterer, grundlegender Vorteil von Gridkarten ist, dass im Voraus keine Annahmen bezüglich der Objektform bzw. der Form der Umgebung getroffen werden, sodass prinzipiell beliebige Objektformen abgebildet werden können.

Ein populärer Ansatz zur Reduzierung des Speicherbedarfs von Gridkarten ist die Verwendung einer adaptiven Zellauflösung. Dabei wird die Umgebung nicht in Zellen gleicher Größe aufgeteilt, sondern zusammenhängende Bereiche, für die dieselben

Eigenschaften gelten und somit der selbe Zelleninhalt gespeichert würde, werden zu einer größeren Zelle zusammengefasst. Es werden ausschließlich dort feinere Zellen verwendet, wo sich die Informationen lokal unterscheiden. Dies kann bspw. durch Baumstrukturen umgesetzt werden (vgl. [KKG+04]).

2.3. Zustandsverfolgung

Ein Hauptproblem bei der Zustandsverfolgung von dynamischen Systemen sind die Unsicherheiten, die durch Faktoren wie Messungenauigkeiten verursacht werden. Daher werden in der Robotik weitestgehend probabilistische Methoden eingesetzt, die die jeweiligen Unsicherheiten explizit mittels Wahrscheinlichkeitstheorie modellieren. Der zu verfolgende Systemzustand wird dabei als Wahrscheinlichkeitsverteilung p über den Zustandsraum repräsentiert.

In vielen Systemen ist der zukünftige Zustand ausschließlich vom aktuellen Zustand abhängig. Die Abfolge von Systemzuständen stellen in diesem Fall Markov-Ketten erster Ordnung dar, sodass für eine Verfolgung des Systemzustands stets nur der aktuelle Zustand geschätzt werden muss. Dazu wird in jedem Zeitschritt zunächst die derzeitige Schätzung anhand des Systemmodells aktualisiert (Prediction) und diese Schätzung anschließend anhand der aktuellen Messung(en) korrigiert (Update). Im nächsten Zeitschritt wiederholt sich der Ablauf. Dies wird auch als „Recursive State Estimate“ bezeichnet, da die Zustandsschätzung rekursiv über die Zeit entwickelt wird. Nachfolgend werden die zwei in dieser Arbeit verwendeten Ansätze, Kalman-Filter und Partikelfilter, erläutert.

2.3.1. Kalman-Filter

Das Kalman-Filter [Kal60] wird verwendet, um einen n-dimensionalen Systemzustand über die Zeit zu verfolgen. Hierbei wird der Zustand als multivariate gaußförmige Wahrscheinlichkeitsverteilung mit dem Mittelwert $x \in \mathbb{R}^n$ und der Kovarianz $P \in \mathbb{R}^{n \times n}$ repräsentiert. Es handelt sich demnach um ein parametrisches Filter, da die Art der

Verteilung vorgegeben ist und über Parameter bestimmt wird. Der Zustand wird in einem rekursiven Zyklus aus Prediction und Update verfolgt (vgl. Abb. 2.1).

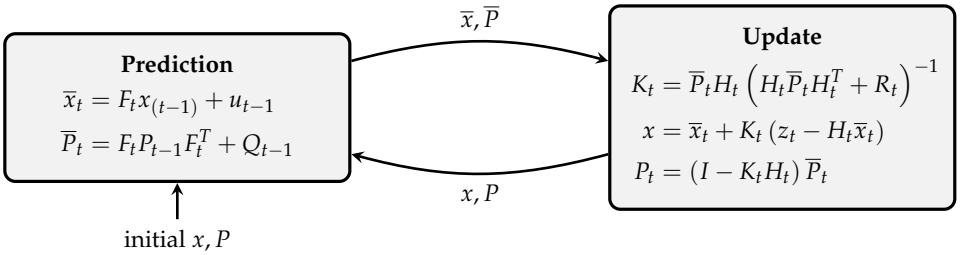


Abbildung 2.1.: Kalman-Filter Übersicht: Rekursive Zustandsabschätzung mittels Prädiktion und Update.

Im Prediction Schritt wird die a priori Verteilung $\overline{x}_t, \overline{P}_t$ zum Zeitpunkt t berechnet. Die Gleichungen 2.2 und 2.3 zeigen die Berechnung der neuen Verteilung. Dazu wird die vorige Zustandsschätzung, x_{t-1}, P_{t-1}, anhand des Systemmodells prädiziert. Dies geschieht durch Multiplikation des Zustands mit der *state transition matrix* F, die durch das Systemmodell definiert ist. Darüber hinaus kann eine Steuerung des Systems, falls vorhanden, über den optionalen *control vector* u_{t-1} mit der Unsicherheit Q_{t-1}in die Prediction einfließen. Q_{t-1} modelliert die Unsicherheit des Systemmodells und der Steuerung u_{t-1}.

$$\overline{x}_t = F_t x_{(t-1)} + u_{t-1} \tag{2.2}$$

$$\overline{P}_t = F_t P_{t-1} F_t^T + Q_{t-1} \tag{2.3}$$

Hierbei ist:		
	$\overline{x}_t, \overline{P}_t$	die a priori Verteilung zum Zeitpunkt t
	x_{t-1}, P_{t-1}	die a posteriori Verteilung zum Zeitpunkt $t-1$
	F_t	die state transition matrix zum Zeitpunkt t
	u_{t-1}	die Steuerung des Systems zum Zeitpunkt $t-1$
	Q_{t-1}	die Unsicherheit des Modells und der Steuerung zum Zeitpunkt $t-1$

Nachdem die Schätzung auf den Zeitpunkt der Messung prädiziert worden ist, wird sie anhand der Messung aktualisiert. Dies zeigen die Gleichungen 2.4 bis 2.8, die in den nächsten Absätzen erläutert werden. Bei diesem Update gehen die Mittelwerte

der a priori Verteilung $\overline{x}_t$ sowie der Messung z_t anhand ihrer Unsicherheiten $\overline{P}_t$ und R gewichtet in die a posteriori Verteilung ein.

$$y_t = z_t - H_t \overline{x}_t \tag{2.4}$$
$$S_t = H_t \overline{P}_t H_t^T + R_t \tag{2.5}$$
$$K_t = \overline{P}_t H_t^T S_t^{-1} \tag{2.6}$$
$$x_t = \overline{x}_t + K_t y_t \tag{2.7}$$
$$P_t = (I - K_t H_t)\, \overline{P}_t \tag{2.8}$$

Hierbei ist:

H_t	das Messmodell, das den Zustand auf die erwartete Messung abbildet
z_t	die Messung zum Zeitpunkt t
R_t	die Unsicherheit der Messung z_t als Kovarianzmatrix
y_t	die Innovation: Die Abweichung zwischen erwarteter und tatsächlicher Messung
S_t	die Innovation Kovarianz
K_t	das Kalman-Gain (Gewichtung der Innovation)
I	die Einheitsmatrix

Zunächst wird die *Innovation* der Messung berechnet. Sie ist die Differenz zwischen Messung und erwartetem Messwert (Gleichung 2.4). Hierbei ist H das Messmodell des Sensors, das den Zustand auf die entsprechende Messung abbildet. Somit ergibt $H_t \overline{x}_t$ den erwarteten Messwert basierend auf der a priori Schätzung. Es wird außerdem die zugehörige Kovarianz S_t als Summe der mittels des Messmodells rotierten Zustandskovarianz ($H_t \overline{P}_t H_t^T$) und der Messunsicherheit R_t gebildet (Gleichung 2.5).

In den Gleichungen 2.6 bis 2.8 wird die Innovation in die Zustandsschätzung eingebracht. Die Gewichtung, mit der die Innovation in die Zustandsschätzung einfließt, wird als *Kalman-Gain* K_t bezeichnet und auf Basis der Innovation Kovarianz S_t und der Kovarianz der a priori Schätzung $\overline{P}_t$ bestimmt (Gleichung 2.6). Der Einfluss der Messung wird umso höher, je höher die Unsicherheit der Zustandsschätzung und je geringer die Unsicherheit der Messung ist. Umgekehrt führt eine geringe Unsicherheit der a priori Schätzung oder eine hohe Messunsicherheit zu einem geringeren Kalman-

Gain. Anhand des Kalman-Gain und der Innovation wird anschließend die a posteriori Verteilung x_t, P_t berechnet (Gleichungen 2.7 und 2.8).

Da vor der ersten Messung kaum eine Aussage über den Zustand getroffen werden kann, sollte der Initialzustand einer Gleichverteilung möglichst nahekommen. Typischerweise wird der Zustand des Kalman-Filters daher mit einem zufälligen Mittelwert x und einer hohen Unsicherheit P initialisiert. Dies hat auch zur Folge, dass der ersten Messung ein hohes Gewicht beigemessen wird, da die Kovarianz der Prädiktion wesentlich größer als die Kovarianz der Messung ist ($\overline{P} \gg R$), was zu einem hohen Kalman-Gain K führt (vgl. Gleichung 2.6).

Für eine umfassendere Erläuterung und Herleitung des Kalman-Filters wird auf [TBF05, Kapitel 3] verwiesen.

2.3.2. Partikelfilter

Das Partikelfilter [Rub88, SG92] repräsentiert den Zustand ebenfalls als multivariate Wahrscheinlichkeitsverteilung über den Zustandsraum. Die Verteilung wird jedoch nicht wie beim Kalman-Filter parametrisch, sondern über eine Menge X_t von sog. Partikeln approximiert: $X_t := x_t^{[1]}, x_t^{[2]}, \ldots, x_t^{[M]}$. Jeder Partikel $x_t^{[m]}$ ist dabei eine konkrete Zustandshypothese. Die Wahrscheinlichkeitsverteilung ergibt sich dabei aus der Partikeldichte. Je höher die Partikeldichte an einer Stelle im Zustandsraum ist, desto Wahrscheinlicher entspricht diese Stelle dem tatsächlichen Zustand. Dementsprechend muss die Wahrscheinlichkeit, dass eine bestimmte Hypothese x_t in der Partikelmenge X_t enthalten ist, proportional zu der Wahrscheinlichkeit sein, dass diese Hypothese dem tatsächlichen Zustand entspricht. Die Repräsentation als Partikelmenge erlaubt ein wesentlich breiteres Spektrum an Verteilungen, insbesondere multimodale Verteilungen (vgl. Abbildung 2.2).

Die Verfolgung des Zustands über die Zeit findet auch beim Partikelfilter rekursiv statt. Listing 2.1 zeigt einen Rekursionsschritt als Pseudocode. Die Eingangsdaten sind die Partikel des letzten Zeitschrittes X_{t-1}, die aktuelle Messung z_t und ggf. der aktuelle control vector u_t. Aus diesen wird in jedem Zyklus mittels Prediction und Update die aktualisierte Zustandsrepräsentation X_t entwickelt:

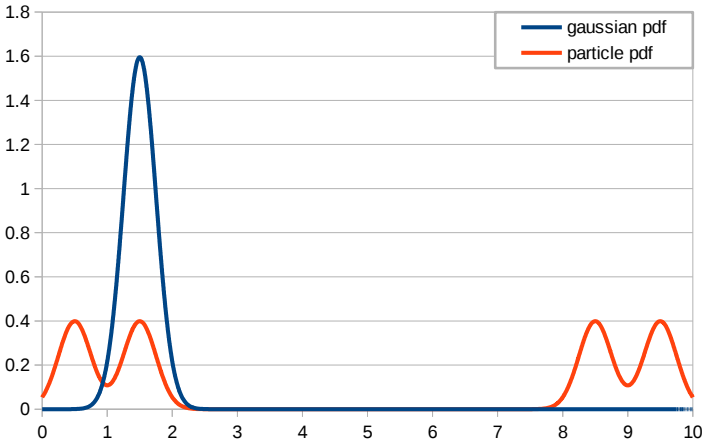


Abbildung 2.2.: Beispielhafte eindimensionale Zustandsabschätzungen mittels Gaußverteilung bzw. Partikeldichte. Während mittels der Gaußverteilung nur eine Mode dargestellt werden kann, erlaubt die Partikeldarstellung mehrere Moden.

Prediction: **Zeile 4** stellt den Prediction Schritt dar, der jeden Partikel mittels Systemmodell auf den Zeitpunkt t prädiziert. Zusätzlich werden die Partikel anhand eines Störungsmodells verrauscht, um die Unsicherheit des Systemmodells sowie des Steuerungsvektors zu modellieren und die Partikel nach dem Resampling (siehe Update) voneinander unterscheidbar zu machen. Die Menge der prädizierten Partikel stellt die a priori Verteilung des Partikelfilters dar.

Update: Das Update besteht aus zwei Schritten: Zunächst wird jedem Partikel ein Gewicht beigemessen, das aus dem Messmodell hervorgeht (**Zeile 5**) und somit der Wahrscheinlichkeit der aktuellen Messung basierend auf dem Zustand des jeweiligen Partikels entspricht. Dieser Schritt wird in der Literatur häufig als „measurement“ und das Gewicht als „importance factor“ bezeichnet. Anschließend folgt das Resampling (**Zeilen 7-10**). Dabei wird M mal einer der prädizierten Partikel ausgewählt und der Ergebnismenge X_t hinzugefügt. Die Wahrscheinlichkeit, dass ein bestimmter Partikel ausgewählt wird, ist dabei proportional zu dessen Gewicht. Auf diese Weise stellt die Ergebnismenge X_t die a posteriori Verteilung des Partikelfilters dar.

Analog zum Kalman-Filter sollte die initiale Verteilung einer Gleichverteilung möglichst nahekommen. Dies lässt sich theoretisch durch ein initiales Sampling mittels

```
1  particlefilter_step(X_{t-1}, u_t, z_t)
2      X̄_t = ∅
3      for m = 1 to M
4          x̄_t^[m] = predict(x_{t-1}^[m], u_t)        // using p(x_t|u_t, x_{t-1}^[m])
5          w_t^[m] = p(z_t|x̄_t^[m])
6          add (x̄_t^[m], w_t^[m]) to X̄_t
7      X_t = ∅
8      for m = 1 to M
9          select i with probability ∝ w_t^[i]
10         add x_t^[i] to X_t
11     return X_t
```

Listing 2.1: Allgemeiner Partikelfilter Algorithmus je Zyklus.

Gleichverteilung über den gesamten Zustandsraum erreichen, was aber je nach Größe des Zustandsraums nur bedingt praktikabel ist.

Abbildung 2.3 zeigt beispielhaft die Lokalisierung eines Roboters mit einem Partikelfilter. Im ersten Schritt (a) hat der Roboter bereits $5m$ zurückgelegt. Die Partikel sind noch weit über die Karte verteilt. Nachdem der Roboter sich zur Ecke oben links bewegt hat (b), haben sich Partikelcluster gebildet. Die Partikel sind bereits wesentlich weniger gestreut. Im letzten Abschnitt hat der Roboter sich zum Korridor bewegt (c). Die Partikelverteilung ist nahezu unimodal und spiegelt die tatsächliche Position gut wider.

Für eine umfassendere Erläuterung und Herleitung des Partikelfilters wird auf [TBF05, Kapitel 4] verwiesen.

2.4. SLAM-Problemdefinition und GraphSLAM

Thrun und Leonard [TL08] geben eine Übersicht über das Simultanous Localization and Mapping (SLAM) Problem und beleuchten verschiedene Lösungsansätze. Das SLAM-Problem besteht darin, eine Karte der Fahrzeugumgebung basierend auf der jeweils aktuellen Pose aus den Umfeldmessungen zu erstellen bzw. zu erweitern und

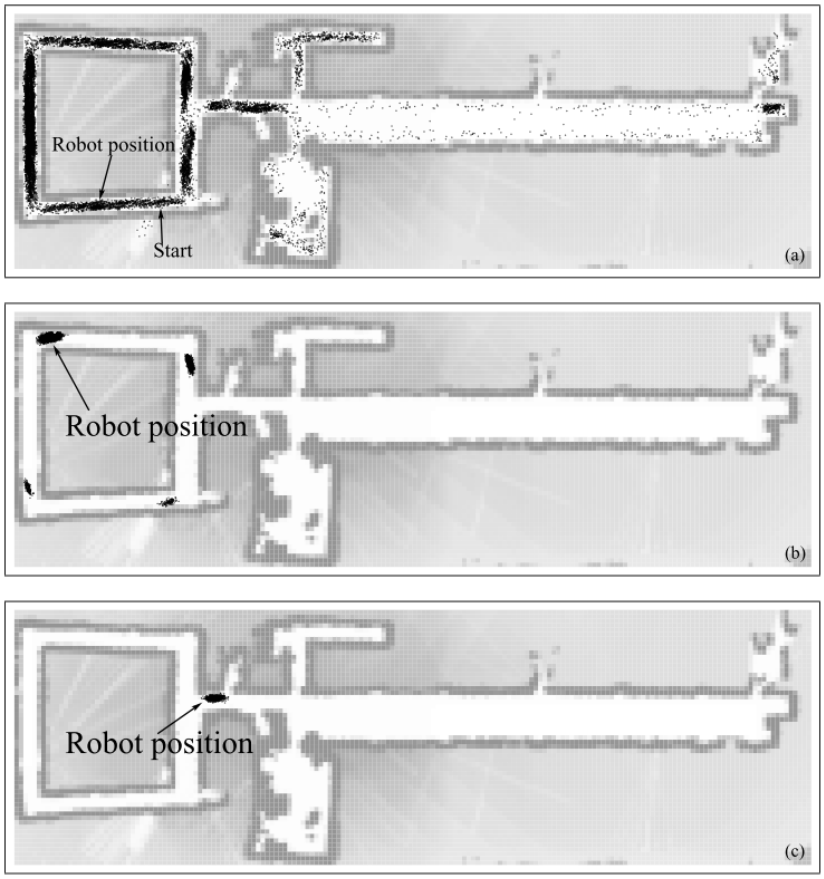


Abbildung 2.3.: Beispielhafte Roboterlokalisierung mittels Partikelfilter. Bildquelle: [TBF05].

gleichzeitig die Fahrzeugpose relativ zu dieser Karte zu bestimmen. Dabei sind sowohl die Fahrzeugbewegung als auch die Umfeldmessungen mit Unsicherheiten behaftet.

Es wird unterschieden zwischen *full SLAM* und *online SLAM*. Full-SLAM bezeichnet die Schätzung der gesamten Fahrzeugtrajektorie und der Karte, wohingegen im online-SLAM neben der Karte lediglich die aktuelle Fahrzeugpose ohne Trajektorie geschätzt wird:

$$\text{full SLAM:} \quad p(x_{0:t}, m | z_{0:t}, u_{0:t}) \tag{2.9}$$

$$\text{online SLAM:} \quad p(x_t, m | z_{0:t}, u_{0:t}) \tag{2.10}$$

Hierbei ist:
- x_t die Fahrzeugpose zum Zeitpunkt t
- m das Umfeld
- z_t die Umfeldmessung zum Zeitpunkt t, in der vorliegenden Arbeit der Scan zum Zeitpunkt t
- u_t die Fahrzeugodometrie zum Zeitpunkt t
- p die zu bestimmende Wahrscheinlichkeitsdichtefunktion

Die Schätzung mittels eines (Extended) Kalman-Filters, Partikelfilter oder eines Graph-basierten Algorithmus sind hierbei die drei grundlegenden Lösungsansätze. Ansätze die auf einem Kalman-Filter oder einem Partikelfilter basieren werden typischerweise für online-SLAM-Anwendungen verwendet, da die Zustandsschätzung iterativ in jedem Zeitschritt erfolgt (vgl. Abschnitt 2.3). Graph-basierte Ansätze eignen sich hingegen für full-SLAM, da zunächst alle Messungen gesammelt und diese dann gemeinsam zur Bestimmung der gesamten Trajektorie und der Karte genutzt werden. Nachfolgend wird der in dieser Arbeit genutzte GraphSLAM-Ansatz beschrieben.

Der GraphSLAM-Algorithmus wird von Thrun, Burgard und Fox in [TBF05] vorgestellt und geht auf verschiedene frühere Arbeiten zurück: [LM97, FH01, DK05]. Dieser Ansatz bildet einen Graphen über die Fahrzeugposen und Landmarkenpositionen, aus dem sich nichtlineare Bedingungen, nachfolgend als Constraints bezeichnet, ergeben. Die Optimierung dieser Constraints führt zu einer Maximum-Likelihood-Schätzung der Trajektorie und der Landmarkenpositionen. Abbildung 2.4 zeigt einen solchen Graphen mit fünf Fahrzeugposen $x_{0:4}$ und zwei Landmarken $m_{1:2}$. Die Kanten im Graphen stellen die Constraints dar, wobei die durchgezogenen Linien Constraints sind, die aus dem Bewegungsmodell hervorgehen, und gestrichelte Linien Constraints aus den Landmarkenmessungen sind.

Die Constraints sind nichtlineare Funktionen, welche das jeweilige Fehlerquadrat in Abhängigkeit der beiden damit verbundenen Knoten angeben. Das Constraint zwischen zwei Posen x_t, x_{t+1} definiert auf diese Weise das Fehlerquadrat der Abweichung der Pose x_{t+1} von der Erwartung aufgrund des Bewegungsmodells $g(u_t, x_t)$ bezogen auf die Unsicherheit des Bewegungsmodells R (Gleichung 2.11). Das Fehlerquadrat steigt folglich mit steigender Abweichung, wobei die Unsicherheit des Bewegungsmodells als Gewichtung des Constraints aufgefasst werden kann. Analog definiert ein Constraint zwischen einer Pose und einer Landmarke das Fehlerquadrat der Abweichung zwischen der tatsächlichen Messung z und der anhand des Messmodells erwarte-

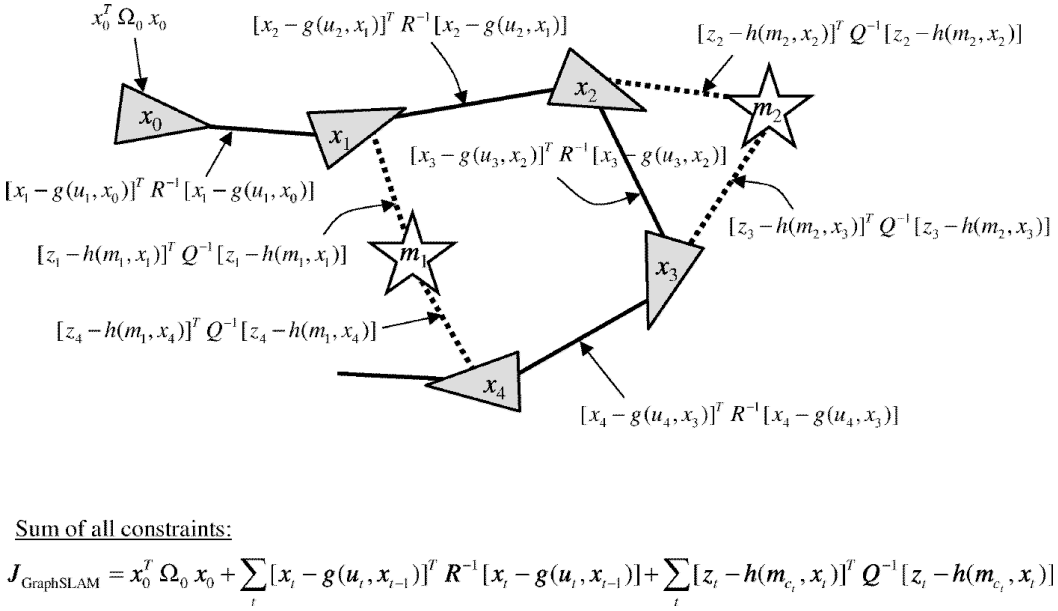


Abbildung 2.4.: Graph mit Posen, Landmarken und Constraints. Bildquelle: [TBF05].

ten Messung $h(m_i, x_t)$, wobei hier die Messunsicherheit Q als Gewichtung angesehen werden kann (Gleichung 2.12).

$$(x_{t+1} - g(u_t, x_t))^T R^{-1} (x_{t+1} - g(u_t, x_t)) \tag{2.11}$$

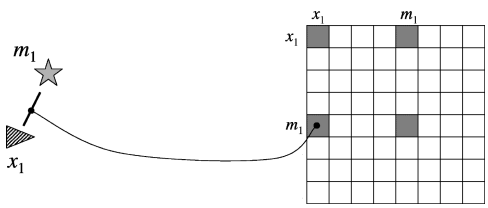
$$(z_{t,i} - h(m_i, x_t))^T Q^{-1} (z_{t,i} - h(m_i, x_t)) \tag{2.12}$$

Hierbei ist:

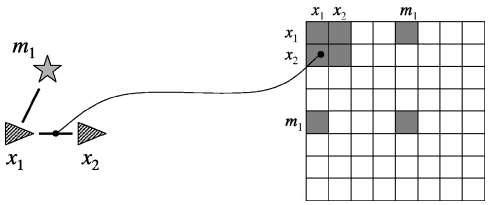
x_t	die Fahrzeugpose zum Zeitpunkt t
m_i	die i-te Landmarke
$z_{t,i}$	die Messung der Landmarke m_i zum Zeitpunkt t
g	das Bewegungsmodell, das den Zustand x_t und den control vector u_t auf den erwarteten Folgezustand $\overline{x}_{t+1}$ abbildet.
R	die Unsicherheit des Bewegungsmodells als Kovarianz
h	das Messmodell
Q	die Messunsicherheit als Kovarianz

Die Summe der Constraints ($J_{GraphSLAM}$ in Abbildung 2.4) bildet somit ein nichtlineares least-squares Problem. Zur Lösung werden die Constraints linearisiert und in ein lineares Gleichungssystem überführt. Abbildung 2.5 stellt das Erzeugen des linearen

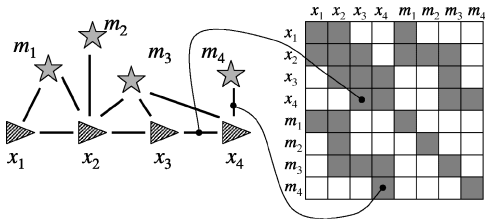
Gleichungssystems in Matrixschreibweise abstrakt dar. Die ersten Zeilen und Spalten repräsentieren jeweils die Posen des Graphen, die darauf Folgenden die Landmarken. In jeder Zelle der Matrix ist die Beziehung der Pose (bzw. Landmarke) der Zeile zur Pose (bzw. Landmarke) der Spalte eingetragen. Dies wird in Abbildung 2.5a verdeutlicht. Sie zeigt die Einträge für die Messung von m_1, welche aus einem Fehlerterm für m_1 in Abhängigkeit von x_1 und umgekehrt (an der diagonalen Achse gespiegelt) bestehen. Die Beziehung, die zwischen den Posen x_1 und x_2 mittels des Bewegungsmodells beschrieben wird, wird analog eingetragen und ist in Abbildung 2.5b dargestellt. Dieses Vorgehen wird fortgesetzt, bis alle Constraints im Gleichungssystem enthalten sind (Abbildung 2.5c). Da die Fehlerquadrate in $J_{GraphSLAM}$ addiert werden, ist das Hinzufügen von Constraints in das lineare Gleichungssystem ebenfalls additiv. Das bedeutet, dass zunächst alle Felder mit 0 initialisiert sind und die linearisierten Fehler in den einzelnen Schritten den entsprechenden Zellen hinzuaddiert werden.



(a) Messung d. Landmarke m_1



(b) Constraint aus Bewegungsmodell



(c) Vollständiges Gleichungssystem

Abbildung 2.5.: Überführung der Constraints in ein lineares Gleichungssystem in Matrixdarstellung. Bildquelle: [TBF05].

Die Linearisierung der Constraints erfolgt über das Taylorpolynom ersten Grades. Dazu wird die Stelle benötigt, an der die Fehlerfunktion linearisiert werden soll. Deshalb muss bereits vor der Linearisierung der Constraints eine erste Schätzung der Trajektorie und der Karte existieren, welche diese Linearisierungsstellen vorgibt. Der GraphSLAM-Algorithmus setzt eine solche initiale Schätzung voraus. Außerdem wird hier zur Erklärung des GraphSLAM-Algorithmus angenommen, dass die Zuordnung

einer Messung $z_{t,i}$ zu der entsprechenden Landmarke m_i bekannt ist (known correspondence)[1].

Nachdem alle Constraints eingetragen sind, führt das Lösen des linearen Gleichungssystems zu einer neuen Schätzung der Trajektorie sowie der Landmarkenpositionen. Die Constraints werden an den Stellen der neuen Schätzung erneut linearisiert und in ein Gleichungssystem überführt, dessen Lösung wiederum zu einer verbesserten Schätzung führt. Auf diese Weise wird im GraphSLAM-Algorithmus die Least Squares Schätzung iterativ approximiert.

Ein Teilproblem im Bereich SLAM stellt das loop-closing dar, welches im Folgenden an einem Beispiel verdeutlicht wird. Das Fahrzeug legt eine weite Strecke zurück und fährt anschließend durch einen zu Beginn bereits befahrenen Abschnitt. Aufgrund der eingangs erwähnten Unsicherheiten in den Messungen und der Fahrzeugbewegung liegen die aktuellen Posen und die Posen zu Beginn der Trajektorie geometrisch jedoch so weit auseinander, dass die Umfeldmessungen nicht mit den zu Beginn der Fahrt erzeugten Messungen assoziiert werden. Dies führt zu einer inkonsistenten Trajektorien- und Kartenschätzung.

Als loop-closing wird das Erkennen dieser Beziehungen zwischen nicht aufeinanderfolgenden Posen bezeichnet. Im known correspondence Fall erfolgt das loop-closing implizit, da aus den entsprechenden Posen dieselben Landmarken gemessen wurden. Im Graphen entsteht damit eine indirekte Beziehung zwischen diesen Posen über die gemeinsame Landmarke, sodass die Beziehung zwischen diesen Posen über die gemeinsam gemessene Landmarke modelliert ist und als Fehlerquadrat in die Abschätzung einfließt. Im unknown correspondence Fall liegt die Schwierigkeit des loop-closings im Erkennen der korrekten Assoziation der Messungen.

[1] Es wird zwischen „known correspondence“ und „unknown correspondence“ unterschieden, wobei Ersteres angibt, dass die Zuordnung von Messung zu Landmarke bekannt oder trivial ist, und Letzteres angibt, dass diese Assoziation nicht bekannt und somit Teil des zu lösenden Problems ist.

2.5. Versuchsfahrzeug

Als Versuchsfahrzeug dient ein BMW 530i Touring. Abbildung 2.6 zeigt den Systemaufbau bestehend aus Seriensensorik sowie zusätzlich verbauten Laserscannern, Inertial Navigation System (INS) und Fahrzeugrechner.

Die vom ABS- und ESP-System des Fahrzeugs ermittelten Odometriedaten werden mittels CAN-Bus kommuniziert. Der Fahrzeugrechner empfängt jeweils die aktuelle Geschwindigkeit sowie die Gierrate: $u_t = \begin{pmatrix} v_t & \omega_t \end{pmatrix}^T$.

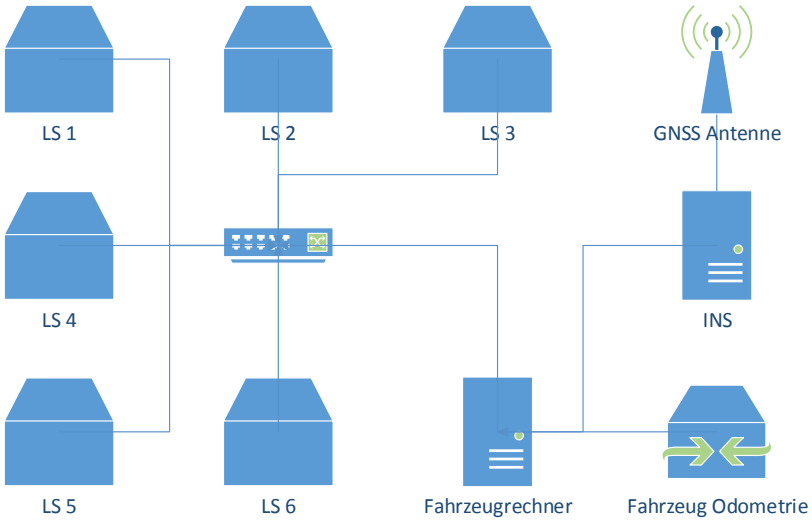


Abbildung 2.6.: Übersicht der Systeme im Versuchsfahrzeug.

Bewegungsmodell

Das Bewegungsmodell definiert die Zustandsübergänge bei der Verfolgung der Fahrzeugpose als Wahrscheinlichkeitsdichtefunktion p über den aktuellen Zustand x_t in Abhängigkeit von der Fahrzeugbewegung u_t und der vorherigen Fahrzeugpose x_{t-1}:

$$p(x_t|u_t, x_{t-1}) \tag{2.13}$$

In dieser Arbeit wird, den Ausgabewerten der Fahrzeugodometrie entsprechend, das „Velocity Motion Model“ (siehe [TBF05, S 121ff.]) angewandt. Die Bewegung u zum Zeitpunkt t besteht dabei aus der Längsgeschwindigkeit v und der Gierrate ω:

$$u_t = \begin{pmatrix} v_t \\ \omega_t \end{pmatrix} \tag{2.14}$$

Diese wird für die Zeitspanne zwischen den Messzeitpunkten $(t-1)$ und t als konstant angenommen, sodass sich eine Bewegung auf einem Kreisbogen ergibt. Der Zustand x_t in Abhängigkeit von x_{t-1} und u_t ist demnach:

$$\begin{pmatrix} x_t \\ y_t \\ \theta_t \end{pmatrix} = \begin{cases} \begin{pmatrix} x_{t-1} - \frac{v_t}{\omega_t} sin\theta_{t-1} + \frac{v_t}{\omega_t} sin\left(\theta_{t-1} + \omega_t \Delta t\right) \\ y_{t-1} + \frac{v_t}{\omega_t} cos\theta_{t-1} - \frac{v_t}{\omega_t} cos\left(\theta_{t-1} + \omega_t \Delta t\right) \\ \theta_{t-1} + \omega_t \Delta t \end{pmatrix} & \text{, falls } \omega_t \neq 0 \\ \begin{pmatrix} x_{t-1} + \Delta t \cdot v_t \cdot cos\theta_{t-1} \\ y_{t-1} + \Delta t \cdot v_t \cdot sin\theta_{t-1} \\ \theta_{t-1} \end{pmatrix} & \text{, sonst} \end{cases} \tag{2.15}$$

2.6. Laserscanner

In der vorliegenden Arbeit werden *Ibeo LUX* Sensoren eingesetzt, welche die Entfernung zur Umgebung mittels Laserstrahlen und der Time-of-Flight Methode messen. Der Sichtbereich reicht horizontal von $-60°$ bis $+50°$ und vertikal von $-1,6°$ bis $+1,6°$, wobei die Schrittweite horizontal $0,25°$ bzw. $0,5°$ (s. u.) und vertikal $0,8°$ beträgt. Die Abtastrate ist $25Hz$. Dabei wird die Umgebung horizontal spaltenweise mit absteigendem Winkel (von links nach rechts) gescannt. Das vom Sensor erzeugte Umgebungsabbild wird nachfolgend als Scan bezeichnet und besteht aus einem Zeitstempel sowie den einzelnen Messwerten der Winkelschritte, nachfolgend Scanpunkt genannt.

Abbildung 2.7 zeigt die genannten Eigenschaften im Detail. Links ist eine Seitenansicht dargestellt, mittig die Draufsicht und auf der rechten Seite ein vergrößerter Ausschnitt der Draufsicht. Die unteren Layer *1* und *2* sind blau dargestellt. Sie decken vertikal den Bereich $-1,6°$ bis $0,0°$ und horizontal $-50,0°$ bis $50,0°$ ab. Die oberen Layer *3* und *4* sind gelb dargestellt und decken $0,0°$ bis $1,6°$ vertikal bzw. $-60°$ bis $35°$ horizontal ab. Der grün gekennzeichnete Bereich wird durch alle Layer abgedeckt. Der Ausschnitt auf

der rechten Seite zeigt diesen Bereich. Es werden abwechselnd die Layer 1-2 und die Layer 3-4 gemessen, sodass sich im grünen Bereich eine vertikale Auflösung von $0,25°$ und im blauen sowie gelben Bereich von $0,5°$ ergibt. Aufgrund des geringen vertikalen Sichtbereichs wird in dieser Arbeit der vertikale Winkel vernachlässigt. Dazu werden alle Messungen auf die x-y-Ebene projiziert, sodass stets ein zweidimensionales Abbild der Umwelt entsteht.

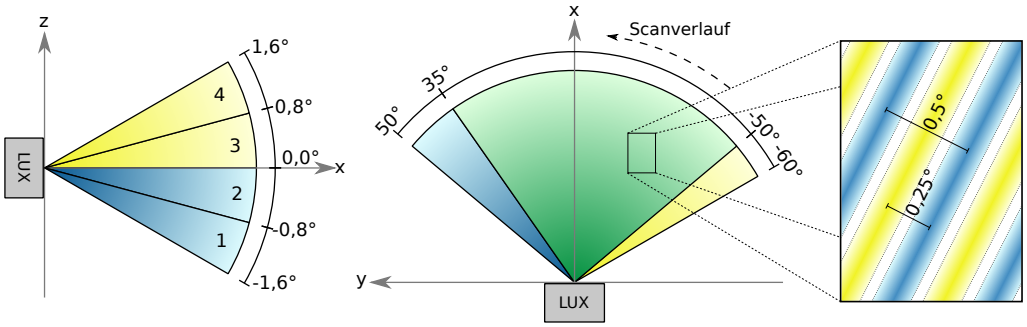


Abbildung 2.7.: Field of View d. ibeo LUX 2010.

Die maximale Entfernung zur Detektion eines Objektes hängt stark von dem Objekt selbst, insbesondere dessen Reflektivität, sowie dem Auftreffwinkel des Laserstrahls ab. Messungen mit einer Entfernung über $200m$ sind jedoch (abgesehen von Reflektoren[2]) kaum möglich.

Neben dem Sichtbereich und der Schrittweite ist auch die horizontale Strahlendivergenz der ausgesandten Laser-Impulse für diese Arbeit relevant, da auch Objekte am Rand eines Laserstrahls messbare Reflektionen verursachen können. Diese ist in Abbildung 2.8 anhand von zwei benachbarten Strahlen in Draufsicht dargestellt. Der Winkelbereich, in dem messbare Reflektionen auftreten, beträgt $+/-0,08°$ um die Schussrichtung. Somit ergibt sich zwischen den Messwinkeln jeweils ein toter Winkel von $0,09°$ (bzw. $0,34°$ im Randbereich mit $0,5°$ Schrittweite).

Sensormodell

Jede Messung, also jeder Scanpunkt, ist mit einer Messunsicherheit behaftet. Diese geht aus verschiedenen Faktoren hervor, wie beispielsweise aus der Strahlendivergenz oder

[2] Bspw. Rückstrahler, Verkehrsschilder

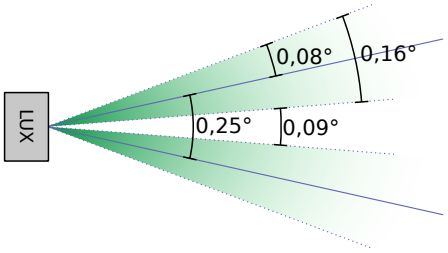
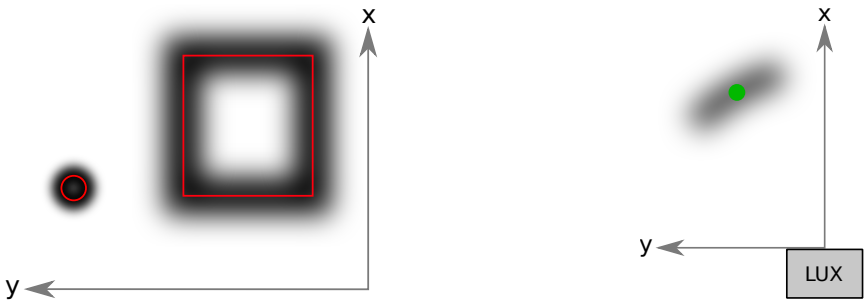


Abbildung 2.8.: Strahlendivergenz d. Laserscanners ibeo LUX (Draufsicht).

der Ungenauigkeit der Zeitmessung zwischen Senden des Laserimpulses und Empfang der Reflektion. Diese Unsicherheit wird als Gaußverteilung jeweils für den Winkel sowie für die Distanz angegeben. Bei probabilistischen Ansätzen wird diese Unsicherheit explizit als Wahrscheinlichkeitsverteilung über Messungen in Abhängigkeit von der Sensor- bzw. Fahrzeugpose und der Umgebung, $p(z_t|x_t, m)$, modelliert. Diese Verteilung wird als „measurement model“ oder Sensormodell bezeichnet. Abbildung 2.9a zeigt die Wahrscheinlichkeitsdichte beispielhaft.



(a) Sensormodell (b) Inverses Sensormodell

Abbildung 2.9.: Wahrscheinlichkeitsverteilung für Messungen. Je dunkler eine Stelle, desto höher ist die Wahrscheinlichkeit. Rot ist die tatsächliche Umgebung. Grün die Messung.

Bei der Detektion von Features in einem Scan oder der Kartenerzeugung ist das tatsächliche Umfeld hingegen unbekannt und es soll ein Umgebungsabbild aus der Messung abgeleitet werden. Dazu wird das inverse Sensormodell (inverse measurement model), $p(m|x_t, z_t)$, benötigt. Diese Dichtefunktion gibt die Wahrscheinlichkeit an, mit der sich ein Objekt, welches die Messung z_m verursacht hat, an der entsprechenden Stelle befindet (Abbildung 2.9b). In der vorliegenden Arbeit wird das durch die Gleichungen 2.16 bis 2.18 definierte inverse Sensormodell angewandt.

$$p(m|x_t, z_t) = p(m_\varphi|x_t, z_{t,\varphi}) \cdot p(m_{dist}|x_t, z_{t,dist}) \tag{2.16}$$

$$p(m_\varphi|x_t, z_{t,\varphi}) = \frac{1}{\sqrt{2\pi\sigma_\varphi^2}} \cdot exp\left(-\frac{(m_\varphi - z_{t,\varphi})^2}{2\sigma_\varphi^2}\right) \tag{2.17}$$

$$p(m_{dist}|x_t, z_{t,dist}) = \frac{1}{\sqrt{2\pi\sigma_{dist}^2}} \cdot exp\left(-\frac{(m_{dist} - z_{t,dist})^2}{2\sigma_{dist}^2}\right) \tag{2.18}$$

Hierbei ist:

m	die Umgebung in polaren Fahrzeugkoordinaten mit $m = \begin{pmatrix} m_\varphi & m_{dist} \end{pmatrix}^T$
x_t	die Fahrzeugpose
z_t	die Messung mit $z_t = \begin{pmatrix} z_{t,\varphi} & z_{t,dist} \end{pmatrix}^T$
σ_φ	die Winkelunsicherheit der Messung
σ_{dist}	die Distanzunsicherheit der Messung

Anbaupositionen am Versuchsfahrzeug

Am Versuchsfahrzeug befinden sich sechs Laserscanner, von denen jeweils zwei links und rechts, einer an der Fahrzeugfront sowie einer am Heck angebracht sind (siehe Abbildung 2.10). Die exakten Anbaupositionen bezüglich Fahrzeugkoordinaten sind in Tabelle 2.1 aufgeführt.

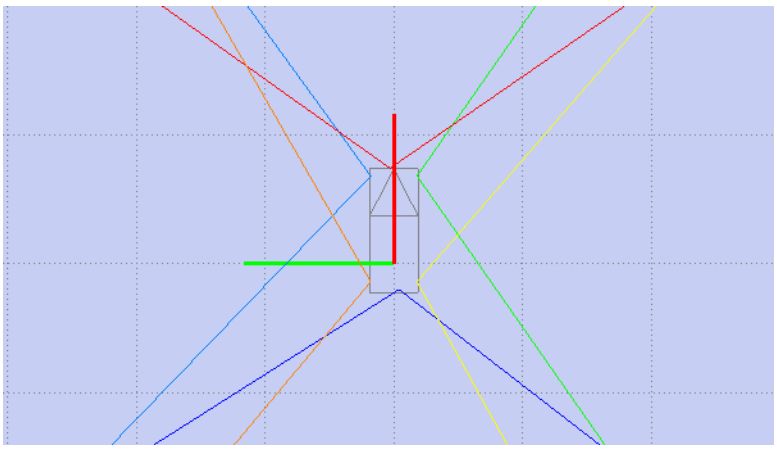


Abbildung 2.10.: Sichtbereich der Laserscanner entsprechend ihrer Anbaupositionen.

DeviceID	x	y	z	Gier	Nick	Roll
1	3,4m	0,9m	0,3m	86°	0°	0°
2	3,4m	-0,9m	0,3m	-85°	0°	0°
3	-0,7m	0,9m	0,45m	90°	0°	0°
4	-0,7m	-0,9m	0,45m	-91°	0°	0°
5	-1,0m	-0,2m	0,35m	-177,7°	0°	0°
6	3,7m	0,15m	0,3m	4,6°	0°	0°

Tabelle 2.1.: Anbaupositionen der Laserscanner am Fahrzeug

2.7. Inertial Navigation System

Neben der Seriensensorik und den Laserscannern ist ein Inertial Navigation Systen (INS), XSens MTi-G-710, verbaut. Dieses misst per GNSS[3] die Fahrzeugposition sowie über Beschleunigungssensoren und Gyroskope die Beschleunigung in Längs-, Quer- und Aufwärts-Richtung und die Beschleunigung der Gier-, Nick- und Rollwinkel. Zur Bestimmung von Gier und Position werden außerdem Magnetometer und Barometer eingesetzt. Ein integriertes Filter, dessen Bewegungsmodell entsprechend der Anwendung anhand vorgegebener Szenarien auszuwählen ist, verfolgt die Fahrzeugpose und -bewegung. In allen Anwendungen der vorliegenden Arbeit ist das Szenario „Automotive“ ausgewählt. Der Fahrzeugrechner empfängt jeweils die geschätzte Fahrzeugpose und -bewegung in Weltkoordinaten (vgl. Abschnitt 2.1), die Geschwindigkeit in Fahrzeugkoordinaten sowie die Gierrate.

[3] Globales Navigationssatellitensystem

3. Aktueller Stand der Forschung

Die Wahrnehmung der Umgebung und des eigenen Zustands in Bezug zur Umgebung umfasst ein weitreichendes Themenfeld, das seit einiger Zeit Gegenstand der Forschung ist. Die vorliegende Arbeit ist in die drei Themen „Keypoint Detektion“, „Kartenerzeugung“ und „Lokalisierung“ strukturiert. Dieses Kapitel zeigt den aktuellen Stand der Forschung auf und orientiert sich dabei strukturell an den drei genannten Themen. **Abschnitt 3.1** behandelt Arbeiten im Bereich Keypoint Detektion in Laserscannerdaten. Arbeiten zum Thema Kartenerzeugung und SLAM werden in **Abschnitt 3.2** vorgestellt. **Abschnitt 3.3** stellt Arbeiten zur Lokalisierung von Fahrzeugen, vor.

3.1. Keypoints

Um die Scans der Laserscanner in einer SLAM-Anwendung oder zur Lokalisierung nutzen zu können, müssen diese untereinander bzw. zu einer Karte vergleichbar und so die euklidische Transformation bestimmbar sein. Das Vergleichen der Scans nennt sich Scan Matching bzw. deren Einpassen Scan Registration, wobei zwischen Rohdaten-basierten Ansätzen, Grid-basierten Ansätzen und Feature-basierten Ansätzen unterschieden werden kann.

Als Rohdaten-basierter Ansatz sei hier stellvertretend das von Besl und McKay eingeführte Verfahren „Iterative Closest Point“ (ICP) [BM92, CM92] genannt. Dazu werden die Scans anhand einer initialen Schätzung übereinandergelegt und mittels Nearest-Neighbor-Verfahren die Scanpunkte des einen Scans jeweils einem Punkt des anderen Scans zugeordnet. Im Anschluss werden die Rotation und die Verschiebung zwischen den Scans ermittelt, die zur minimalen Distanz zwischen diesen zugeordneten Punkten führen. Das Verfahren wird so lange mit der verbesserten Schätzung wiederholt, bis das

Ergebnis konvergiert. In eindeutigen Umgebungen sind gute Ergebnisse zu erwarten, wohingegen bei Mehrdeutigkeiten, wie sie bspw. in einem Tunnel auftreten können, die Qualität der Ergebnisse unsicher ist. Darüber hinaus ist das Verfahren anfällig gegenüber schlechten Startwerten, da die Gefahr besteht nur ein lokales Minimum der Distanzfunktion in der Umgebung des Startwertes anstatt das Minimum an der Stelle der korrekten Transformation zu finden.

Feature-basierte Ansätze abstrahieren zunächst den Scan, indem Stellen mit besonderen Eigenschaften in den Scans erkannt und diese als „Features“ beschrieben werden. Die Features unterschiedlicher Scans werden dann einander zugeordnet und so für das Matching bzw. die Registration genutzt. Der Vorteil gegenüber der Assoziation von Scanpunkten zueinander (ICP) ist, dass durch einen Abgleich der Feature-Eigenschaften oftmals eine robustere Zuordnung möglich ist. Typischerweise werden geometrische Eigenschaften extrahiert und als Features repräsentiert. Cox [Cox91] verwendet als Umfeldrepräsentation eine vordefinierte Karte aus Linien und assoziiert die Scanpunkte der jeweils aktuellen Messung zu den Linien der Karte. Arras und Vestli beschreiben in [AV98] einen Ansatz zur Erkennung von Linien in Laserscannerdaten und nutzen diese zur Lokalisierung. [Wei11] verwendet u. a. Polygonzüge anstelle von einzelnen Linien, um Objekte der Umgebung zu repräsentieren.

Grid-basierte Ansätze umgehen das Zuordnungsproblem der Rohdaten- und Feature-basierten Ansätze. Anstatt eine Zuordnung zwischen Scanpunkten oder Features herzustellen und daraus die Transformation zu bestimmen, werden vorgegebene Transformationen lediglich durch Überlagern von Messung und Karte bewertet. Dazu wird die Transformation auf den Scan angewendet und die Bewertung dieser Transformation ergibt sich aus den von Scanpunkten getroffenen Zellen. Die einzelnen Gridzellen bilden dabei typischerweise die Umgebung entsprechend des Sensormodells ab (vgl. Abschnitt 2.6). Um mittels dieser Bewertungsfunktion eine rigide Transformation zwischen zwei Scans abzuleiten, tastet beispielsweise das von Olson in [Ols09] vorgestellte Verfahren einen Bereich des Zustandsraums ab und bewertet jede dieser möglichen Transformationen. Einen weiteren Vorteil stellt neben dem Umgehen der Zuordnung von Scanpunkten in diesem Fall das Abtasten des Zustandsraums dar, woraus sich die Kovarianzmatrix zur ermittelten Transformation bestimmen lässt. Zugleich resultiert jedoch aus ebendieser Abtastung ein hoher Rechenaufwand.

In der jüngeren Vergangenheit wurden zunehmend Feature Detektoren aus der Bildverarbeitung adaptiert, die Punktfeatures, auch als Keypoints bezeichnet, in Laserscannerdaten erkennen. Diese Keypoints stellen Eigenschaften der Umgebung dar, die für exakt einen Punkt zutreffen, z. B. dass es sich um eine Ecke handelt. In der vorliegenden Arbeit werden ebenfalls Keypoints detektiert und im SLAM verwendet. Unterschiedliche Ansätze dazu werden in Abschnitt 4.2 mit Bezug auf die in Abschnitt 4.1 definierten Anforderungen betrachtet.

3.2. SLAM

Thrun und Leonard [TL08] geben eine Übersicht über das SLAM-Problem und über probabilistische Lösungsansätze. Sie geben an, dass robuste Methoden für statische strukturierte Umgebungen mit beschränkter Größe existieren, Verfahren für unstrukturierte oder dynamische Umgebungen sowie die Anwendung der Ansätze auf sehr große Umgebungen dagegen offene Probleme darstellen. Es gibt drei grundlegende Ansätze zur Lösung des SLAM-Problems: Die Schätzung mittels Kalman-Filter, mittels Partikelfilter und Graph-basiert.

Smith und Cheeseman stellen in [SC86, SSC90] den Ansatz vor, den eigenen Zustand sowie die Features der Karte gemeinsam im Zustandsvektor eines Kalman-Filters zu schätzen. Dies ermöglicht die Modellierung von Unsicherheiten bei der Fahrzeugsteuerung bzw. -odometrie sowie von Messungenauigkeiten. Wie in Abschnitt 2.3.1 beschrieben, wird der Zustand durch einen Mittelwert und Kovarianz als Gaußverteilung dargestellt. Werden über die Zeit neue Features beobachtet, so wachsen dementsprechend der Zustandsvektor und die Kovarianzmatrix. Eine Schwierigkeit ist dabei die Assoziation von Messungen zu den Features in der Kartendarstellung. Darüber hinaus ist das Kalman-Filter-basierte SLAM in dieser Form auf eine Feature-basierte Umfelddarstellung angewiesen.

Partikelfilter-basierte SLAM-Ansätze repräsentieren die Pose als Partikelverteilung wie in Abschnitt 2.3.2 beschrieben. Für jeden Partikel wird zusätzlich eine Karte geschätzt, wobei jedes Feature in einem separaten Kalman-Filter geschätzt wird. Bei N Partikeln und M Features werden also $N \cdot M$ Kalman-Filter benötigt. Die Kombination von Partikel- und Kalman-Filter geht auf Rao [Rao45] und Blackwell [Bla47] zurück und

wird daher häufig als Rao-Blackwellized Partikelfilter bezeichnet. Die grundlegenden Arbeiten zur Verwendung von Rao-Blackwellized Partikelfiltern im Bereich SLAM stammen von Doucet u. a. [DdFMR00] sowie von Montemerlo u. a. [MTKW02].

Graph-basierte Ansätze, wie der in Abschnitt 2.4 beschriebene GraphSLAM, gehen zurück auf die Arbeiten [SC86] und [DW88], in denen relative Constraints beschrieben werden. Lu und Milios stellen in [LM97] einen Ansatz vor, der eine globale Optimierung der Zustandsschätzung anhand der Menge aller Constraints durchführt. Graph-basierte SLAM-Ansätze eignen sich insbesondere für full-SLAM Anwendungen, wohingegen die Abschätzung der gesamten Trajektorie, die den Graph-basierten Ansätzen inhärent ist, für online-SLAM von Nachteil ist.

Basierend auf den drei beschriebenen Paradigmen existieren diverse weiterführende Arbeiten, welche an dieser Stelle nicht weiter ausgeführt werden. Stattdessen sei auf die Arbeiten „Simultaneous localization and mapping (SLAM): Part I The Essential Algorithms“ [DWB06] sowie „Simultaneous localization and mapping (SLAM): Part II“ [BDW06] von Durrant-Whyte und Bailey verwiesen, die eine gute Übersicht über die Thematik geben.

Im Übrigen unterscheiden sich SLAM-Algorithmen in der Art ihrer Kartendarstellung, wobei insbesondere zwischen topologischen Ansätzen und geometrischen Ansätzen zu unterscheiden ist (vgl. Abschnitt 2.2). Auf topologische Ansätze wird in der vorliegenden Arbeit nicht näher eingegangen, wohingegen unterschiedliche Ansätze zur geometrischen Repräsentation in den Abschnitten 2.2 und 3.1 behandelt werden.

3.3. Lokalisierung

Die Lokalisierung kann als Teilproblem des SLAM-Problems aufgefasst werden, da weiterhin eine Lokalisierung anhand einer Karte erfolgt, jedoch ohne gleichzeitiges Schätzen der Karte. Prinzipiell sind demnach auch hier die im vorigen Abschnitt 3.2 genannten Arbeiten relevant. Aus diesem Grund werden nachfolgend nur einige beispielhafte Arbeiten vorgestellt, um einen Eindruck über den aktuellen Forschungsstand zu vermitteln.

Weiss hat sich im Rahmen seiner Doktorarbeit [Wei11] mit dem Erzeugen digitaler Umgebungskarten für Fahrerassistenzsysteme und der Posebestimmung auf ebendiesen beschäftigt. Die Grundlage der Lokalisierung ist dabei eine Gridkarte, welche je Zelle eine Belegtwahrscheinlichkeit angibt. In der vorliegenden Arbeit bildet ebenfalls eine Gridkarte die Basis der Lokalisierung. Weiss extrahiert jedoch Features aus der Gridkarte und assoziiert diese mit Features der aktuellen Messung, wohingegen in der vorliegenden Arbeit die Lokalisierung direkt auf der Gridkarte erfolgt. Anhand der Zuordnung der Features zur Karte berechnet Weiss die jeweilige Fahrzeugpose, welche jedoch nicht mittels Kalman- oder Partikelfilter verfolgt wird. Er gibt an, dass auf diese Weise eine hohe Genauigkeit erreicht wird, das zusätzliche Verfolgen mittels Filter jedoch die Konsistenz aufeinander folgender Posen erhöhen und Fehlassoziationen von Features verringern würde.

Lu u. a. [LSRR14] verfolgen die Fahrzeugpose mittels Partikelfilter. Dazu werden Fahrspurmarkierungen in Kamerabildern detektiert, diese anschließend den Fahrspurmarkierungen in OpenStreetMap[1] Karten zugeordnet und zusammen mit GPS-Daten im Filter genutzt. Die Autoren geben die Standardabweichung der ermittelten lateralen Position von der tatsächlichen Position mit ca. $1,2m$ an.

Zindler u. a. [ZGDH14] beschreiben eine Extended-Kalman-Filter-basierte Lokalisierung anhand von Inertialsensoren sowie Laserscannern, wobei Features aus den Laserscannerdaten extrahiert und mit den Features einer vordefinierten Karte abgeglichen werden. Besonderes Augenmerk liegt dabei auf einem erweiterten Bewegungsmodell, welches explizit die nichtlinearen Bewegungseigenschaften aufgrund der Reifen bei hochdynamischen Fahrmanövern beachtet. Als Ergebnis wird eine Genauigkeit der Lokalisierung von $20cm$ angegeben. Diese Genauigkeit kann aufgrund künstlich erzeugter Landmarken mit eindeutiger Zuordnung zur Karte erreicht werden. Wie in Kapitel 2 aufgezeigt wird, erlaubt die Verwendung eines Kalman-Filters jedoch keine Mehrdeutigkeiten, sodass eine fehlerhafte Assoziation der Features zur Karte einen enormen Fehler in der Zustandsschätzung zur Folge hätte.

Im Zusammenhang mit dem Thema „Straßenumfeld“ ist auch die technische Arbeit [MAF16] relevant. Sie beschreibt die Positionsbestimmung mittels Kalman-Filter basierend auf GNSS, Fahrzeugodometrie und Gyroskopen, wobei neben der Fahrzeugpose

[1] OpenStreetMap (OSM) ist ein Open Source Projekt, welches eine freie Datenbank mit Kartendaten zur Verfügung stellt (http://www.openstreetmap.org).

und -bewegung auch Unsicherheitsparameter und Kalibrierungswerte für die Sensorik im geschätzten Zustand enthalten sind. Hiermit wird das Ziel verfolgt, grob fehlerhafte Messungen zu erkennen und auszusortieren sowie für die verwendeten Messungen zutreffende und nicht rein modellbasierte Unsicherheitsschätzungen zu erhalten.

4. Keypoint Detektion

In der vorliegenden Arbeit wird ein Feature-basiertes SLAM-Verfahren (Kapitel 5) umgesetzt, um die Fahrzeugtrajektorie zu schätzen. Der Begriff Feature ist nicht exakt definiert, kann aber als eine wiedererkennbare Eigenschaft in einem Umgebungsabbild aufgefasst werden. Dies können z. B. bestimmte Farbverläufe in einem Bild oder bestimmte strukturelle Eigenschaften in einem Laserscan sein. Für das in dieser Arbeit umgesetzte SLAM-Verfahren ist es notwendig, dass die Position eines solchen Features als Punktkoordinate angegeben werden kann.

In diesem Kapitel liegt der Fokus auf der Detektion von Keypoints in zweidimensionalen Scans. Als Keypoint werden hier jene Features bezeichnet, die als Punkt dargestellt werden können und eine fest definierte Eigenschaft der Umgebung an diesem Punkt darstellen. Aufgrund des unstrukturierten Umfelds im Straßenverkehr und des SLAM-Ansatzes ergeben sich spezifische Anforderungen. Diese werden in Abschnitt 4.1 definiert. Anschließend werden in Abschnitt 4.2 verschiedene Detektionsansätze bezüglich dieser Anforderungen betrachtet. In Abschnitt 4.3 ist der in dieser Arbeit entwickelte Algorithmus und in Abschnitt 4.4 dessen Umsetzung beschrieben. Den Abschluss bildet die Bewertung des Verfahrens in Abschnitt 4.5.

4.1. Anforderungen

An die detektierten Features bestehen folgende Anforderungen:

Recall Als Recall wird die Wiedererkennungsrate bezeichnet. Features zur Verwendung im SLAM müssen in verschiedenen Scans wiedererkannt werden. Dabei ist es nicht zwingend notwendig, dass diese in direkt aufeinander folgenden

Scans erneut detektiert werden. Zudem müssen detektierte Features den zuvor detektierten Features zugeordnet (assoziiert) werden können.

Genauigkeit Die Genauigkeit der Fahrzeugpose bzw. -trajektorie hängt direkt von der Positionsgenauigkeit der detektierten Keypoints ab.

Unsicherheitsschätzung Um die Trajektorie anhand der Keypoints zu schätzen ist eine realistische Unsicherheitsschätzung der Keypoint Position zwingend erforderlich.

Statisch Es dürfen ausschließlich Keypoints genutzt werden, die die statische Umgebung abbilden, d. h. nicht von bewegten Objekten verursacht sind.

Unstrukturierte Umgebung Die durch einen Keypoint repräsentierte Eigenschaft muss für typische Umgebungen im Straßenumfeld geeignet sein. Es sind also solche Keypoints zu erkennen, die sowohl in strukturierter Umgebung wie Innenstädten als auch in unstrukturierter Umgebung (z. B. Landstraße) auftreten, sodass regelmäßig Keypoints detektiert werden und somit eine ausreichende Abdeckung der Umgebung gegeben ist.

Punktfeatures Die Position eines Features muss als Punkt beschreibbar sein. Diese Anforderung ergibt sich aus dem SLAM-Verfahren und ist bei den als Keypoint bezeichneten Features inhärent gegeben.

4.2. Betrachtung existierender Detektoren bzgl. Anforderungen

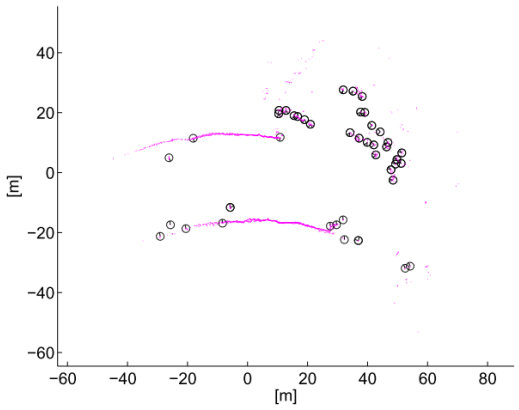
In der Bildverarbeitung werden Keypoints zur Wiedererkennung von Umgebungen oder Objekten bereits erfolgreich verwendet. Als Beispiele seien hier SIFT [Low99] oder SURF [BTV06] genannt. In der jüngeren Vergangenheit wird diese Idee im Bereich von Laserscannern adaptiert und einige Arbeiten zur Detektion von Keypoints wurden veröffentlicht. Nachfolgend werden die Ansätze dieser Arbeiten beschrieben und bezüglich der zuvor aufgestellten Anforderungen bewertet. Es sei darauf hingewiesen, dass es sich stets um zweidimensionale Scans handelt und angenommen wird, dass diese ausschließlich statische Umgebungsinformationen enthalten.

Zlot und Bosse [ZB09] fassen einen Scan als Funktion über den Scanwinkel auf, wobei der Wert an jeder Stelle durch die jeweilige gemessene Distanz gegeben ist. Über diese Funktion wird per numerischer Differentiation die zweite Ableitung gebildet (Laplace Operator). Lokale Maxima, in denen die zweite Ableitung positiv ist, werden als Keypoints identifiziert. Auf diese Weise werden Kanten, die zum Laserscanner hin deuten, in der Umgebung detektiert (vgl. Abbildung 4.1a). Lokale Maxima mit negativem Wert, d. h. vom Laserscanner weg deutend, werden verworfen, da diese häufig von einer Verdeckung durch Objekte im Vordergrund verursacht werden.

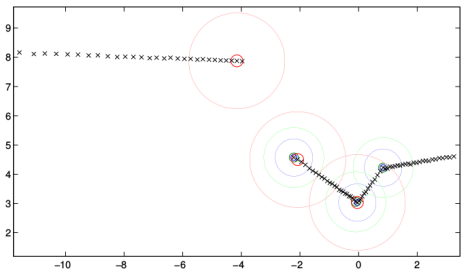
Tipaldi und Arras [TA10] stellen drei Ansätze zur Keypoint Detektion vor. Im ersten Verfahren wird der Scan ebenfalls als Funktion der Distanz je Scanwinkel aufgefasst und die Keypoints werden mittels Laplace Operator detektiert. Dabei wird jedoch auf einer Scale-Space Repräsentation verfahren. Das bedeutet, dass die Funktion mittels gaußschem Filter unterschiedlicher Größe (scale) geglättet wird, sodass auf unterschiedlich stark geglätteten Funktionen verfahren wird. Anhand von Extrema nach Durchführung des Laplace-Operators werden Kanten auf unterschiedlichen scales detektiert. Das Verfahren entspricht somit der Definition eines Laplacian of Gaussian (LoG) auf einem Laserscan. Der zweite Ansatz nutzt als Werte der Eingangsfunktion nicht die gemessene Distanz, sondern den Normalenvektor jeder Messung, der basierend auf den benachbarten Messungen berechnet wird. Auf dieser Funktion können nun Kanten anhand von Extrema in der ersten Ableitung oder Blobs anhand des zuvor beschriebenen LoG detektiert werden. Der dritte beschriebene Ansatz geht auf [UH08] zurück. Hierbei wird die Differenz zwischen der ungeglätteten Eingangsfunktion und den geglätteten Funktionen der scale-space-Repräsentation betrachtet. Die lokalen Extrema stellen Kanten im Scan dar und werden somit als Keypoints detektiert (vgl. Abbildung 4.1b).

Li und Olsen [LO10] erzeugen aus den Scandaten ein Bild in Draufsicht (siehe Abb. 4.1c), sodass im Anschluss prinzipiell sämtliche Vorgehen aus der Bildverarbeitung angewendet werden können. Von diesem Bild werden Scale-Space Repräsentationen mit verschiedenen Scales erzeugt und mittels Kanade-Tomasi corner detector [TK91] Ecken detektiert. Die detektierten Ecken in diesem Bild sollen vertikalen Kanten in der Umgebung entsprechen.

Kallasi, Rizzini und Caselli [KRC16] stellen den „FALKO“ Detektor vor, der ebenfalls Kanten erkennt (vgl. Abbildung 4.1d). Dazu wird um jeden Scanpunkt ein polares Grid erzeugt und die benachbarten Scanpunkte in diesem Grid eingetragen. Ein Scanpunkt

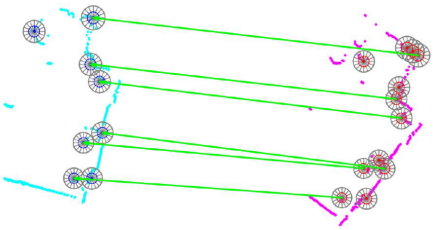


(a) Zlot und Bosse (Bildquelle: [ZB09])



(b) Tipaldi und Arras (Bildquelle: [TA10])

(c) Li und Olsen (Bildquelle: [LO10])



(d) Kallasi, Rizzini und Caselli (Bildquelle: [KRC16])

Abbildung 4.1.: Vergleich verschiedener Keypoint Detektoren. Kennzeichnung der Keypoints durch (a) schwarze Kreise (b) rote Kreise (c) blaue sowie rote Punkte (d) schwarze Kreise. Abbildung (d) zeigt zwei Scans, die jeweils detektierten Keypoints sowie deren Assoziation.

wird als Keypoint identifiziert, wenn zwei Kriterien erfüllt sind: (1) Die Nachbarn zu beiden Seiten des Scanpunktes dürfen nur wenige radiale Abschnitte des Grids belegen. (2) Die Winkeldifferenz zwischen den belegten Abschnitten der linken und der rechten Nachbarn muss innerhalb eines definierten Bereiches liegen (vgl. Abbildung 4.2).

Tabelle 4.1 zeigt die Bewertung der Ansätze bezüglich der zuvor aufgestellten Anforderungen. Dazu sind einige Feststellungen hervorzuheben, die nachfolgend aufgezeigt werden.

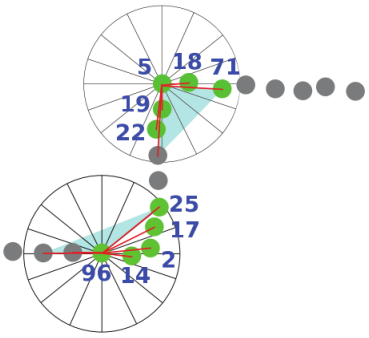


Abbildung 4.2.: FALKO Detektor. Der obere beispielhaft ausgewählte Scanpoint wird als Keypoint identifiziert, wohingegen der untere verworfen wird. (Bildquelle: [KRC16])

Ansatz	Recall	Genauigkeit	Unsicherheit	Statisch	Umgebung	Punkt
[ZB09]	*	+	-	*	-	+
[TA10]	*	+	-	*	-	+
[LO10]	*	+	+	*	-	+
[KRC16]	+	+	-	*	-	+

Tabelle 4.1.: Bewertung der Ansätze bezüglich der aufgestellten Anforderungen.
+ „erfüllt“ bzw. „gut geeignet“
0 „ausreichend“
* „keine Bewertung“ (Aufgrund fehlender Informationen)
- „nicht erfüllt“ bzw. „nicht geeignet“

Alle beschriebenen Ansätze detektieren vertikale Kanten, welche im horizontalen Scan als Punktfeature gelten und somit für das SLAM-Verfahren tauglich sind. Die einzelnen Ansätze sind jedoch eher für strukturierte Umgebungen geeignet: [KRC16] detektiert explizit solche Stellen, an der sich zwei gerade Abschnitte treffen. Die Ansätze [ZB09], [TA10] und [LO10] setzen diese Strukturiertheit nicht explizit voraus. Dennoch ist deren Anwendung in unstrukturierter Umgebung problematisch, da die Betrachtung der Funktion von Scanwinkel zu Distanz bzw. die Betrachtung der Funktionseigenschaften ein gewisses Maß an Strukturiertheit bedingt. Dies ist in Abbildung 4.1a sichtbar, welche im oberen rechten Abschnitt übermäßig viele Keypoints im unstrukturierten Abschnitt der Umgebung zeigt.

Bezüglich der Anforderungen „Recall“ und „Statisch“ kann an dieser Stelle keine vergleichende Aussage getroffen werden. Die Genauigkeit der Keypoints hängt bei allen

Ansätzen stark von der Messgenauigkeit der Laserscanner sowie der Scanpunktdichte ab und wird darüber hinaus von der Umgebung beeinflusst. Dies ist z. B. dann der Fall, wenn Keypoints an abgerundeten Ecken detektiert werden. Eine Schätzung der Positionsunsicherheit je Keypoint erfolgt nur bei Li und Olsen. Die Featuredichte wird basierend auf den Abbildungen 4.1 für alle Ansätze als ausreichend angesehen und es handelt sich bei allen Ansätzen um Punktfeatures.

Insgesamt erfüllt keiner der Ansätze alle aufgestellten Anforderungen und wäre somit für die Verwendung im SLAM-Verfahren der vorliegenden Arbeit geeignet. Dies liegt an erster Stelle daran, dass die Ansätze jeweils Features detektieren, die zwar in stark strukturierter Umgebung häufig auftreten oder dort gut wiedererkennbar sind, in unstrukturierter Umgebung jedoch kaum zu finden sind. Daher wurde in der vorliegenden Arbeit ein neuer Algorithmus zur Detektion von Keypoints im Straßenumfeld entwickelt. Dieser wird im nachfolgenden Abschnitt vorgestellt.

4.3. Algorithmus zur Keypoint Detektion

Der hier beschriebene Algorithmus detektiert schmale, freistehende Objekte, wie Laternenpfähle oder schmale Bäume. Es werden dazu Scanpunkte als Kandidaten identifiziert, deren Nachbarn entweder in ausreichender Entfernung liegen, sodass sie ein anderes Objekt gemessen haben, oder so nah liegen, dass sie das selbe Objekt gemessen haben, ohne dass dessen Ausdehnung einen Schwellwert überschreitet. Um Fehldetektionen zu reduzieren werden außerdem die Echo-Puls-Breite[1] sowie die Anzahl der Scanpunkte auf dem potentiellen Keypoint-Objekt berücksichtigt. Anschließend werden diese Kandidaten segmentiert und die Kandidaten je Segment zusammengefasst, um so Häufungen von Kandidaten zu einem Keypoint zu kombinieren. Dies ist beispielhaft in Abbildung 4.3 dargestellt. Nachfolgend werden zunächst die geometrischen Kriterien definiert, die zur Identifizierung der Kandidaten dienen. Anschließend wird der gesamte Algorithmus zur Keypoint Detektion anhand von Pseudocode beschrieben.

[1] Die Echopulsbreite (EPW) ergibt sich aufgrund des Messverfahrens im Sensor aus der Helligkeit sowie der Dauer des reflektierten Laserpulses. Je höher die Reflektivität des Objektes und somit je heller die Reflektion oder je länger die Pulsdauer, desto höher ist die gemessene EPW. Falsch positive Messungen haben i. d. R. eine geringe EPW.

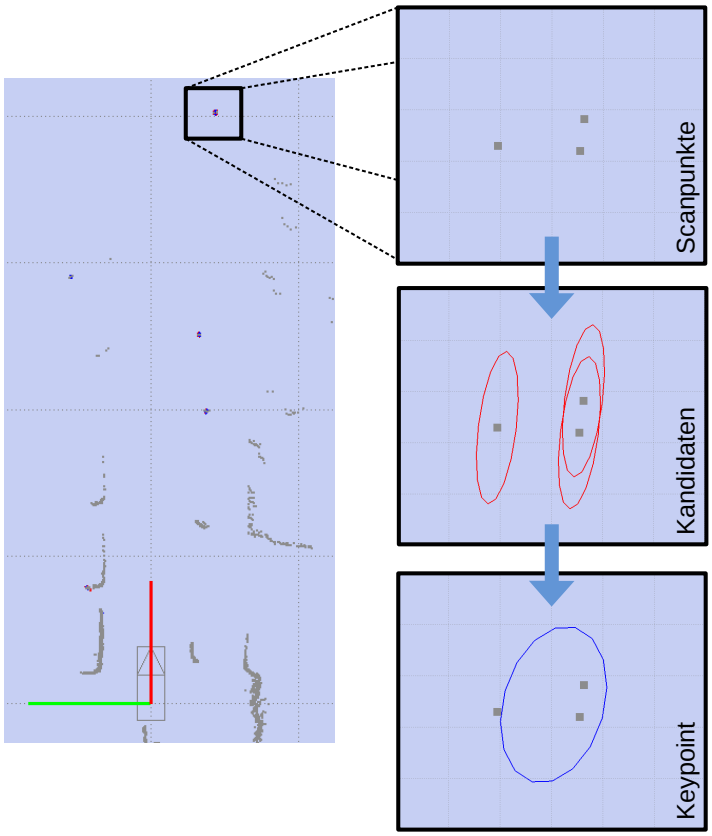


Abbildung 4.3.: Erkennung von Keypoints. Links: Übersicht über das Szenario. Rechts: Nahaufname des Ablaufes mit der Identifikation von Kandidaten (rot) und der anschließenden Kombination der Kandidaten zu einem Keypoint (blau). Es sind jeweils die Unsicherheitsellipsen (1σ-Umgebung) dargestellt.

Den Kern der Keypoint Erkennung bildet die Überprüfung der geometrischen Kriterien, welche die besagten Kandidaten identifiziert. Diese sind als Übersicht in Abbildung 4.4 dargestellt, wobei der Punkt p_i als Kandidat gilt, wenn im grau hinterlegten Bereich keine Scanpunkte existieren. Die Überprüfung findet anhand der folgenden Grenzwerte statt:

$G_{roiDist}$ ist der Radius, in dem keine weiteren Objekte als der Keypoint gemessen werden dürfen. Dieser ist abhängig von der Entfernung des zu überprüfenden Kandidaten, da die Distanz zwischen Scanpunkten benachbarter Messwinkel linear mit der Entfernung steigt (vgl. Abschnitt 2.6).

$G_{maxDist}$ definiert die maximale Ausdehnung eines Keypoints. Dies ist der Radius, in dem weitere Scanpunkte zulässig sind, da sie dem selben Objekt zugeschrieben werden, ohne dass das Objekt die maximale Ausdehnung überschreitet.

G_{cover} ist die Entfernung, bis zu der andere Scanpunkte als potenzielle Verdeckung des zu überprüfenden Keypoints gelten.

$G_{roiAngle}$ bestimmt den Winkelbereich, in dem keine Messungen mit geringerer Distanz als G_{cover} liegen dürfen, da diese auf eine Verdeckung schließen lassen würden.

G_{numPts} definiert, wie viele Scanpunkte mindestens auf dem Keypoint Objekt liegen müssen. Aufgrund der mit der Entfernung steigenden Distanz zwischen benachbarten Scanpunkten ist dieser Wert entgegengesetzt linear zur Entfernung des Scanpunkts.

G_{epw} definiert die minimale Echopulsbreite, die ein Scanpunkt aufweisen muss, um als Kandidat zu gelten.

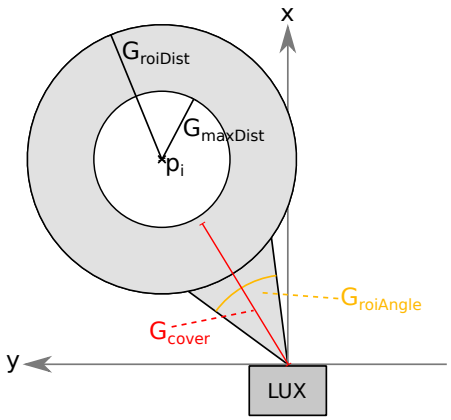


Abbildung 4.4.: Übersicht der geometrischen Keypoint Kriterien. p_i kann nur als Keypoint Kandidat identifiziert werden, wenn im grau gekennzeichneten Bereich keine Scanpunkte existieren.

Listing 4.1 zeigt den Algorithmus zur Keypoint Detektion als Pseudocode. Zeile 1 ist der Methodenkopf mit dem Scan als Eingangsparameter. In den Zeilen 2-14 erfolgt die Identifizierung der Keypoint Kandidaten aus den einzelnen Scanpunkten (rote Ellipsen in Abbildung 4.3). Die Zeilen 3-4 überprüfen die Echopulsbreite des Scanpunktes, die den Schwellwert $G_{epw} = 1m$ überschreiten muss. Anhand dieses Kriteriums sollen Fehldetektionen, die nicht von einem tatsächlichen Objekt verursacht wurden, reduziert werden. Anschließend wird der Scanpunkt auf die zuvor erläuterten geometrischen Kriterien hin überprüft. Dazu werden zunächst die Schwellwerte der geometrischen

```
1  detectKeypoints(scan)
2      for each p_i ∈ scan
3          if (p_i.epw < G_epw)
4              continue
5          G_roiDist = Param_roiDist,abs + ||p_i|| · Param_roiDist,rel
6          G_roiAngle = Param_roiAngle,abs
7          G_maxDist = Param_maxDist,abs
8          G_numPts = Param_numPts,abs + ||p_i|| · Param_numPts,rel
9          G_cover = ||p_i|| · Param_cover,rel
10         success = checkNeighbors(scan, p_i, G_roiDist, G_roiAngle,
11                                  G_maxDist, G_numPts)
12         if (!success)
13             continue
14         candidates.push_back(p_i)
15
16     for i = 0;i < candidates.size(); i++
17         segment.clear()
18         c_i = candidates[i]
19         segment.push_back(c_i)
20         G_maxDist = Param_maxDist,abs + ||p_i|| · Param_maxDist,rel
21         for j = i + 1; j < candidates.size()
22             c_j = candidates[j]
23             if (||c_j − c_i|| < G_maxDist)
24                 segment.push_back(c_j)
25                 candidates.erase(c_j)
26             else
27                 j++
28         keypoint = combineCandidates(segment)
29         result.push_back(keypoint)
30     return result
```

Listing 4.1: Detektion von Keypoints.

Kriterien bestimmt (Zeilen 5-9). Diese sind konstant oder werden durch lineare Funktionen über die Messdistanz definiert. Der Ablauf der anschließenden Überprüfung (Zeilen 10-11) ist in Listing 4.2 aufgeführt und wird im weiteren Verlauf erläutert. Ist diese Prüfung erfolgreichm, wird der aktuelle Scanpunkt in die Menge der Keypoint Kandidaten, `candidates`, aufgenommen.

Nach Identifikation aller Kandidaten werden diese in den Zeilen 16-28 segmentiert und die Segmente zu einem Keypoint zusammengefasst (blaue Ellipse in Abbildung 4.3). Dabei werden alle Kandidaten, deren Entfernung unter $G_{maxDist}$ liegt, demselben Segment zugeordnet und zu einem Keypoint kombiniert (Zeilen 18-29). Dazu wird, mit dem ersten beginnend, jeweils der nächste Kandidat als Ausgangspunkt für ein Segment ausgewählt (Zeilen 17-19) und alle verbleibenden Kandidaten werden auf ihre Zugehörigkeit zum Segment überprüft (Zeilen 20-27). Jeder Kandidat, der dem Segment zugeordnet wird, wird aus der Menge der Kandidaten entfernt, um dessen Zuordnung zu mehreren Segmenten zu verhindern (Zeilen 23-25). Das Segment ist vollständig, sobald alle verbleibenden Kandidaten auf ihre Zugehörigkeit hin überprüft worden sind. Es wird ein Keypoint aus dem vollständigen Segment erstellt und der Ergebnismenge `result` hinzugefügt (Zeilen 28-29). Der Keypoint ist dann gegeben durch μ, den Mittelwert der Kandidaten, und Σ, der Unsicherheit des Keypoints als Summe der Kovarianz aus der Verteilung der Kandidaten und der Messunsicherheit (vgl. Listing 4.3). Das Verfahren wird mit den verbleibenden Kandidaten wiederholt, bis keine Kandidaten mehr übrig sind.

In Listing 4.2 ist die Überprüfung eines möglichen Keypoint-Kandidaten anhand der benachbarten Punkte als Pseudocode aufgeführt. Ein Scanpunkt soll nur dann als Kandidat detektiert werden, wenn es sich um ein freistehendes Objekt mit geringer horizontaler Ausdehnung handelt. Scanpunkte, die nicht vom selben Objekt herrühren, dürfen dementsprechend nur in ausreichender euklididischer Distanz auftreten. D. h. es dürfen sich in der Region of Interest (ROI) mit dem Radius $G_{roiDist}$ um den zu prüfenden Punkt p_i keine weiteren Objekte befinden (Zeilen 4-5). p_i kann nur unter der Bedingung als Keypoint-Kandidat gelten, dass alle Scanpunkte innerhalb der ROI nicht weiter als $G_{maxDist}$ von p_i entfernt liegen, da ansonsten von einem weiteren Objekt in der Nähe auszugehen ist oder das Objekt, das die Messung p_i verursacht hat, eine zu große Ausdehnung hat (Zeilen 6-7).

```
1  checkNeighbors(scan, p_i, G_roiDist, G_roiAngle, G_maxDist, G_numPts, G_cover)
2      numPts = 0
3      for each p_k ∈ scan : k > i − G_roiNum, k < i + G_roiNum, k ≠ i
4          l = ||p_k − p_i||
5          if (l < G_roiDist)
6              if (l > G_maxDist)
7                  return false
8              numPts++
9          else if (|p_k.angle − p_i.angle| < G_roiAngle && ||p_k|| < G_cover)
10             return false
11     if (numPts < G_numPts)
12         return false
13     return true
```

Listing 4.2: Prüfung der Keypoint-Kriterien für benachbarte Scanpunkte.

Die ROI umfasst darüber hinaus alle Scanpunkte, deren Scanwinkel innerhalb des $G_{roiAngle}$-Intervalls um den Winkel von p_i liegen. Diese werden dahingehend überprüft, ob sie das gemessene Objekt verdecken, wodurch dessen gesamte Ausdehnung nicht erfasst werden konnte (Zeilen 9-10). Da die meisten Punkte $p_k \in$ `scan` außerhalb dieser Regionen liegen, werden ausschließlich G_{roiNum} Punkte links sowie rechts von p_i betrachtet. Dies setzt voraus, dass die Scanpunkte nach Winkel sortiert sind. Als weiteres Kriterium wird angenommen, dass eine Mindestanzahl von Scanpunkten G_{numPts} das freistehende Objekt gemessen haben. Dies wird mittels der Zeilen 2, 8 und 11-12 sichergestellt.

Listing 4.3 zeigt das Erzeugen eines Keypoints aus mehreren Messungen. Die Eingangsdaten sind mehrere Messungen, hier als Array `segment` angegeben und der Rückgabewert ist der resultierende Keypoint bestehend aus Mittelwert `mean` und Kovarianzmatrix `cov`. Zunächst wird in den Zeilen 2 - 5 der Mittelwert der Messungen berechnet und anschließend die Positionsunsicherheit bestimmt (Zeilen 7 - 12). Die Unsicherheit ergibt sich hier als Summe aus der Verteilung der Messungen und der Messunsicherheit für den Keypoint entsprechend des Sensormodells (vgl. Abschnitt 2.6), wobei als Winkelunsicherheit $\sigma_\varphi = 0,35$ und als Distanzunsicherheit $\sigma_{dist} = 0,15$ angenommen wird. Diese sind höher als die eigentliche Messunsicherheit der Sensoren, um den Widerspruch zwischen der Annahme, dass Landmarken keine Ausdehnung besitzen, und der tatsächlichen Ausdehnung von Objekten zu modellieren.

```
1  combineCandidates(segment)
2      mean = ( 0 )
               ( 0 )
3      for each p_i ∈ segment
4          mean += p_i
5      mean /= segment.size(); // Mittelwert
6
7      cov = ( 0  0 )
             ( 0  0 )
8      for each p_i ∈ segment
9          tmp = p_i − mean
10         cov += tmp · tmp^T
11     cov /= (segment.size()-1); // empirische Varianz
12     cov += measUncertainty
13
14     return mean, cov
```

Listing 4.3: Kombination von mehreren Messungen zu einem Keypoint.

4.4. Umsetzung der Keypoint Detektion

Der zuvor beschriebene Algorithmus zur Keypoint Detektion wurde in einem Framework der Ibeo Automotive Systems umgesetzt. Dieses Framework arbeitet mit fusionierten Scans, die sich aus jeweils einem Scan pro verbautem Sensor zusammensetzen, sodass sich eine Rundumsicht ergibt (vgl. Abschnitt 2.6). Die Keypoints werden hingegen auf Einzelscans detektiert, also jeweils auf einem Scan eines einzelnen Sensors, da nur bei Betrachtung in den jeweiligen Sensorkoordinaten eine mögliche Verdeckung von Objekten korrekt erkannt werden kann. Hieraus ergibt sich der in Abbildung 4.5 als Flussdiagram dargestellte Ablauf.

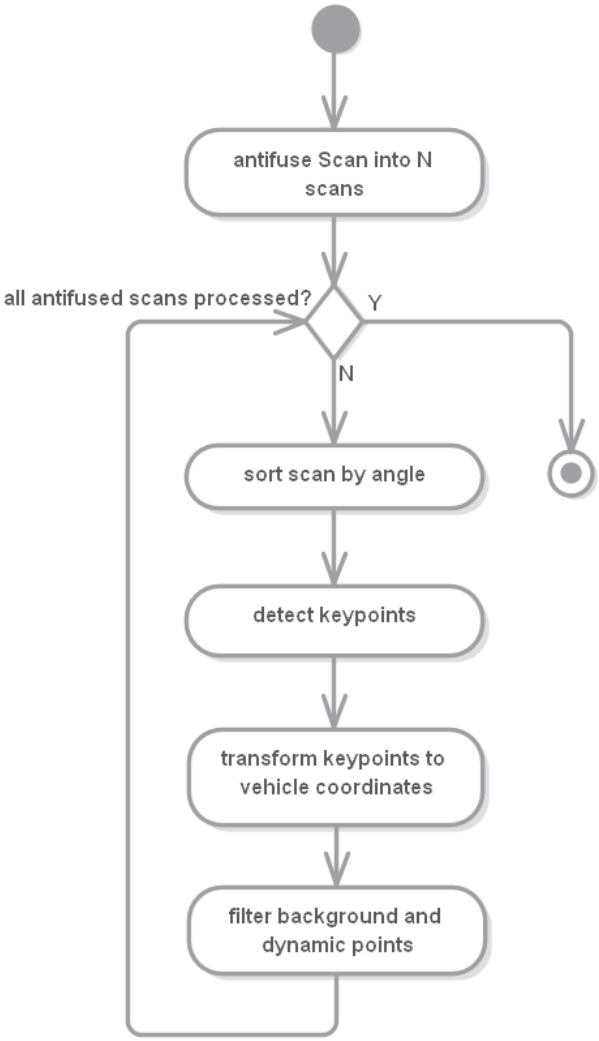


Abbildung 4.5.: Umsetzung der Keypoint Detektion für fusionierte Scans.

Zunächst wird der fusionierte Scan in die Einzelscans aufgeteilt und diese vom Fahrzeugkoordinatensystem in Sensorkoordinaten überführt. Dazu werden die Einzelscans anhand der Anbauparameter jedes Scanners transformiert (Gleichung 4.1)

$$\begin{aligned} p_S &= \begin{pmatrix} p_{S,x} \\ p_{S,y} \end{pmatrix} \\ &= \begin{pmatrix} cos(-M_\varphi) & -sin(-M_\varphi) \\ sin(-M_\varphi) & cos(-M_\varphi) \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & -M_x \\ 0 & 1 & -M_y \end{pmatrix} \cdot \begin{pmatrix} p_{V,x} \\ p_{V,y} \\ 1 \end{pmatrix} \end{aligned} \tag{4.1}$$

$$p_V = \begin{pmatrix} p_{V,x} \\ p_{V,y} \end{pmatrix} = \begin{pmatrix} cos(M_\varphi) & -sin(M_\varphi) & M_x \\ sin(M_\varphi) & cos(M_\varphi) & M_y \end{pmatrix} \cdot \begin{pmatrix} p_{S,x} \\ p_{S,y} \\ 1 \end{pmatrix} \tag{4.2}$$

Hierbei ist:
- p_V der Punkt in Fahrzeugkoordinaten
- p_S der Punkt in Sensorkoordinaten
- M Die Anbaupose des Sensors bestehend aus Position $(M_x \;\; M_y)^T$ und Gierwinkel M_φ

Anschließend wird über diese Einzelscans iteriert. Die Scanpunkte eines Einzelscans werden zunächst nach Scanwinkel sortiert. Es folgt die eigentliche Detektion der Keypoints, wobei folgende Werte für die Parameter des Algorithmus genutzt werden:

$$\begin{aligned} Param_{roiDist,abs} &= 1m \\ Param_{roiDist,rel} &= sin(4 \cdot 0.25°) \approx 0,0175 \\ Param_{roiAngle,abs} &= 1.5° \\ Param_{maxDist,abs} &= 0.25m \\ Param_{numPts,abs} &= 2 \\ Param_{numPts,rel} &= 0 \\ Param_{cover,rel} &= 1 \end{aligned}$$

Daraufhin werden die Keypoints in das Fahrzeugkoordinatensystem transformiert. Gleichung 4.2 zeigt die Transformation. Abschließend werden alle Keypoints, die von dynamischen Objekten verursacht wurden, entfernt. Als dynamisch werden alle Keypoints angesehen, die innerhalb der Bounding Box eines vom Sensor getrackten, dynamischen Objekts liegen. Alle verbleibenden Keypoints bilden das Ergebnis.

4.5. Bewertung des Verfahrens

In diesem Unterkapitel wird das zuvor beschriebene Verfahren zur Detektion von Keypoints bezüglich der in Abschnitt 4.1 aufgestellten Anforderungen bewertet. Da einige Kriterien keine quantitative Bewertung erlauben, erfolgt zunächst eine verbal-argumentative Bewertung bezüglich aller definierten Anforderungen. Dem folgt eine quantitative Bewertung der Kriterien „Recall“ und „Unstrukturierte Umgebung“.

Recall Wenn sich in der jeweiligen Fahrzeugumgebung ausreichend kleine, freistehende Objekte befinden, die vom Laserscanner erfasst werden, ist von einer hohen Wiedererkennungsrate der Keypoints auszugehen. In wie weit dies zutrifft, zeigt die quantitative Bewertung (s. u.).

Genauigkeit Die Genauigkeit der detektierten Keypoints wird hauptsächlich durch zwei Einflüsse bestimmt: Zum einen hängt die Positionsgenauigkeit direkt von der Messgenauigkeit der Laserscanner ab, zum anderen nimmt die Art der hier detektierten Keypoints Einfluss. Das Modell nimmt an, dass das gemessene Objekt keine Ausdehnung besitzt und daher alle Scanpunkte des Laserscanners, die das Objekt treffen, den selben Punkt messen. Tatsächlich haben die gemessenen Objekte jedoch eine Ausdehnung, sodass die Scanpunkte nicht vom selben physikalischen Punkt stammen, was dazu führt, dass die Positionsgenauigkeit der Keypoints sinkt. Diese Ungenauigkeit ist begrenzt durch die maximale Ausdehnung eines als Keypoint detektierten Objektes (vgl. Abbildung 4.4: $G_{maxDist}$). In der vorliegenden Arbeit wird versucht, diese Ungenauigkeit in der Positionskovarianz eines Keypoints zu berücksichtigen, indem mehrere Scanpunkte des selben Objekts zu einem Keypoint zusammengefasst werden und deren Verteilung in die Unsicherheitsschätzung einfließt. Im Gegensatz hierzu detektieren die in Abschnitt 4.2 vorgestellten Ansätze Eigenschaften, die tatsächlich einem bestimmten Punkt ohne Ausdehnung zuzuordnen sind. Dennoch wird an dieser Stelle angenommen, dass die Genauigkeit der in diesem Kapitel beschriebenen Keypoints eine gute Trajektorienschätzung erlaubt und auf die Evaluation der Kartierung (Abschnitt 5.8) verwiesen.

Unsicherheitsschätzung Eine realistische Unsicherheitsschätzung der Keypoint Position ist zwingend erforderlich und erfolgt in der vorliegenden Arbeit, wie im

vorigen Abschnitt beschrieben, anhand der Messunsicherheiten der verwendeten Laserscanner sowie der Verteilung der zugehörigen Scanpunkte und somit der Ausdehnung des gemessenen Objekts. Die geschätzten Unsicherheiten fallen demnach aufgrund der Modellannahme, dass die gemessenen Objekte keine Ausdehnung haben, größer aus als es den reinen Messunsicherheiten entspricht. Dennoch ermöglichen diese Unsicherheiten die Verwendung der Keypoints im SLAM-Verfahren. An dieser Stelle sei ebenfalls auf die Bewertung der Trajektorieschätzung in Abschnitt 5.8 verwiesen, da die Unsicherheiten direkten Einfluss auf diese haben.

Statisch Das Ausgangsinterface der Laserscanner liefert neben den Scanpunkten auch dynamische Objekte. Diese werden in der beschriebenen Umsetzung verwendet, um Keypoints, die von dynamischer Umgebung verursacht wurden, auszusortieren. Sofern alle dynamischen Objekte in den Messdaten als solche erkannt werden, beschreiben die Keypoints somit ausschließlich statische Objekte. Dasselbe Vorgehen lässt sich ebenso auf die in Abschnitt 4.2 beschriebenen Detektoren anwenden.

Unstrukturierte Umgebung Die Idee der Detektion von kleinen freistehenden Objekten soll explizit der Detektion von Keypoints in unstrukturierter Umgebung, insbesondere im Straßenumfeld, dienen. Dies können bspw. Zaunpfähle, Leitpfosten, Laternenmasten, Schilder oder dünne Bäume sein. In welchem Maß dies gelingt zeigt die quantitative Bewertung (s. u.).

Punktfeatures Die detektierten Keypoints werden durch ihre Position und der Positionskovarianz beschrieben. Die durch das SLAM-Verfahren gegebene Anforderung, dass die Position eines Features als Punkt beschreibbar sein muss, ist somit erfüllt.

Die argumentative Bewertung der einzelnen Anforderungen ergibt, dass alle Anforderungen durch das Detektionsverfahren bzw. dessen Umsetzung berücksichtigt sind. Dabei bleibt jedoch offen, wie geeignet das Verfahren bezüglich „Unstrukturierte Umgebung“ und „Recall“, d. h. wie hoch die Erkennungs- und Wiedererkennungsrate in typischen Straßenumgebungen, tatsächlich ist. Es ist zu erwarten, dass die Ergebnisse je nach Umgebung variieren. Daher wurden zur quantitativen Bewertung die folgenden fünf Szenarien ausgewählt: Fahrt durch Autobahn, Fahrt durch Tunnel,

Fahrt durch Industriegebiet, Fahrt durch Innenstadt und Fahrt durch Landstraße. Die Streckenverläufe sind in Anhang B aufgeführt.

Zur Bewertung wurden zunächst Daten mit dem Versuchsfahrzeug aufgezeichnet und diese anschließend prozessiert. Dabei wurden in jedem Scan die Keypoints detektiert, diese anhand der Fahrzeugodometrie in ein gemeinsames Koordinatensystem überführt und zuletzt die Keypoints unterschiedlicher Scans mittels Nearest-Neighbor-Verfahren einander zugeordnet. Bei der Zuordnung wurde eine maximale Distanz von $1m$ zugelassen. Um fehlerhafte Zuordnungen aufgrund schlechter Odometrie und somit eine Überschätzung der Wiedererkennungsrate zu vermeiden, wurden außerdem nur Assoziationen zwischen Keypoints erzeugt, deren Messungen maximal 10 Sekunden auseinander liegen.

Abbildung 4.6 zeigt die durchschnittliche Anzahl an Keypoints pro Scan in rot und die durchschnittliche Anzahl der wiedererkannten Keypoints in grün. Als wiedererkannt gilt ein Keypoint, wenn er zu einem zuvor detektierten Keypoint zugeordnet wird; eine Zuordnung eines späteren Keypoints zum aktuell betrachteten wird dabei nicht berücksichtigt. Um die Poseänderung des Fahrzeugs allein anhand der Keypoints zu bestimmen, wären demnach je Scan mindestens zwei wiedererkannte Keypoints nötig.

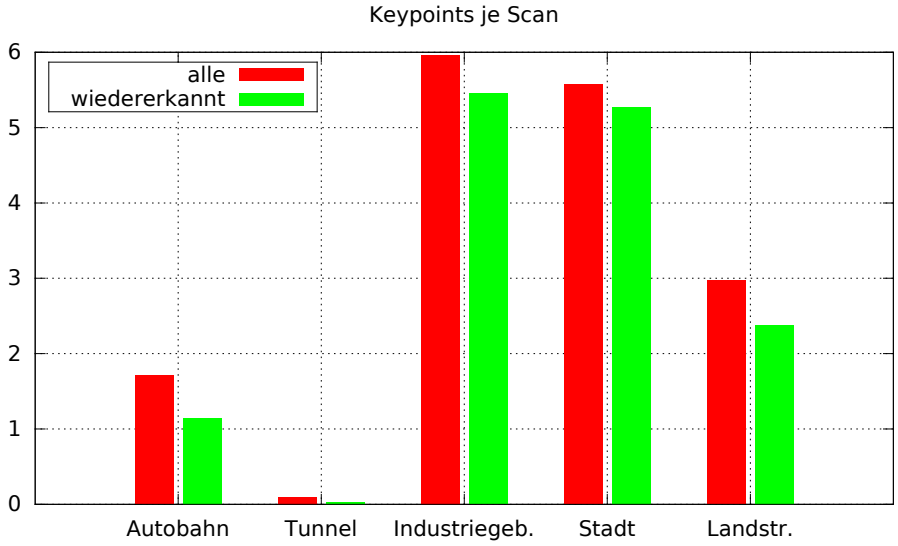


Abbildung 4.6.: Detektions- und Wiedererkennungsrate der Keypoints je Zeitschritt

In den Szenarien Industriegebiet, Innenstadt und Landstraße wird diese Grenze überschritten, wobei insbesondere im Industriegebiet sowie innerstädtisch eine wesentlich höhere Rate erreicht wird. Auf der Autobahn werden hingegen durchschnittlich keine zwei Keypoints pro Scan erkannt. Es ist dennoch zu erwarten, dass die Trajektorieschätzung auf der Autobahn mit Keypoints, Fahrzeugodometrie und INS bessere Ergebnisse liefert als ohne Keypoints. Im Tunnel werden nahezu keine Keypoints detektiert.

Abbildung 4.7 zeigt, wie häufig eine Landmarke im Durchschnitt als Keypoint detektiert wurde. In rot ist der Durchschnitt der Detektionen pro Landmarke dargestellt, wobei alle Landmarken berücksichtigt werden. In grün ist ebenfalls der Durchschnitt der Detektionen je Landmarke dargestellt, wobei jene Landmarken nicht berücksichtigt werden, die nur einmal detektiert wurden. Zur Bestimmung der Trajektorie ist es von Vorteil, wenn die gleichen Landmarken in möglichst vielen verschiedenen Scans detektiert wurden. Dies führt dazu, dass zeitlich weiter auseinander liegende Posen direkt miteinander in Verbindung gebracht werden können, um so das Akkumulieren von Fehlern über die Zeit zu reduzieren. Auch in dieser Betrachtung wurden in den Szenarien „Industriegebiet“ und „Stadt“ die besten Ergebnisse erzielt. Es ist jedoch zu beachten, dass eine direkte Abhängigkeit von der Fahrzeuggeschwindigkeit besteht, da sich eine Landmarke bei geringerer Geschwindigkeit länger im Sichtbereich befindet.

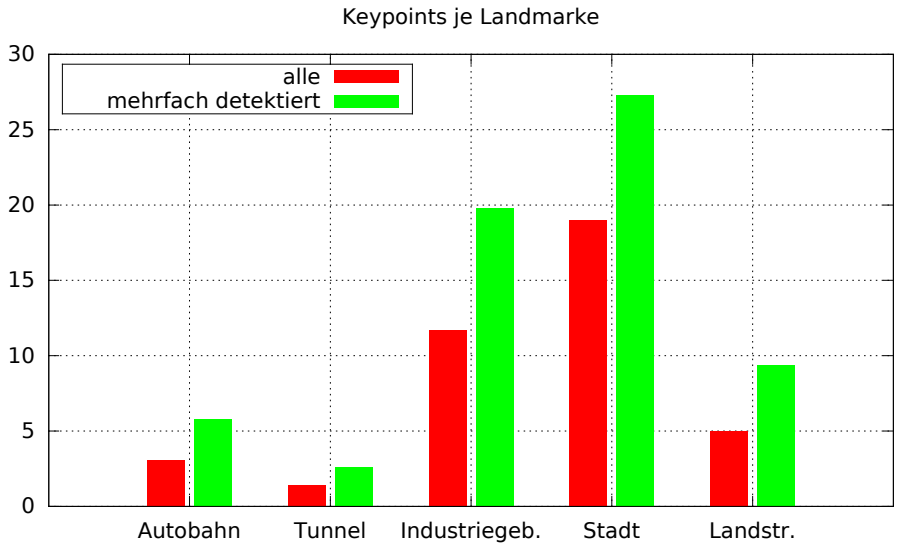


Abbildung 4.7.: Detektions- und Wiedererkennungsrate der Keypoints je Landmarke

Zusammenfassend ergeben sich für die Szenarien „Industriegebiet“, „Innenstadt“ und „Landstraße“ Ergebnisse, die eine gute Trajektorienschätzung erwarten lassen. Die Verwendung der hier vorgestellten Keypoints im Tunnelszenario erweist sich hingegen als nicht sinnvoll. Im Szenario „Autobahn“ werden zwar nur wenige Keypoints detektiert, jedoch ist durch die mehrfache Detektion der Landmarken in durchschnittlich über fünf Scans eine gute Trajektorieschätzung in Verbindung mit Fahrzeugodometrie und INS nicht ausgeschlossen.

5. Kartenerzeugung

In der vorliegenden Arbeit wird ein Verfahren zur Erzeugung von Gridkarten entwickelt. Diese dienen zur Lokalisierung des Fahrzeugs bei späteren Fahrten. Dazu wird zunächst die Fahrzeugtrajektorie der Mappingfahrt bestimmt und basierend auf dieser sowie den Laserscannerdaten die Gridkarte erzeugt.

Es wird ein GraphSLAM-basiertes Verfahren zur Lösung des SLAM-Problems eingesetzt (vgl. Abschnitt 2.4). Dabei entsprechen die Fahrzeugposen sowie die im vorigen Kapitel beschriebenen Keypoints den Knoten des Graphen. Die Constraints, die Kanten im Graphen, sind durch die Fahrzeugodometrie, die INS-Messungen sowie die Keypoints gegeben. Nachfolgend werden die Begriffe Keypoints und Landmarken folgendermaßen verwendet: Als Keypoints werden die in den Scans detektierten Punktfeatures bezeichnet. Landmarken sind die eigentlichen Objekte der Umgebung, sodass ein Keypoint eine Messung der Landmarke ist. Keypoints in unterschiedlichen Scans, die vom selben physikalischen Objekt herrühren, sind demnach unterschiedliche Messungen der selben Landmarke.

Die Schätzung der Trajektorie und der Landmarken erfolgt in drei Abschnitten (siehe Abbildung 5.1). Der erste Abschnitt des Verfahrens ist die Initialisierung, bei der die initiale Trajektorie mittels Kalman-Filter geschätzt wird, die Keypoints der einzelnen Scans detektiert werden und die gesamte Trajektorie in Abschnitte (Cluster) von $10m$ Länge unterteilt wird. Laut [TBF05, S. 369] ist ein solches Aufteilen der Trajektorie in Cluster üblich, um die Skalierbarkeit und Effizienz des Verfahrens zu erhöhen. Die Trajektorien und Karten innerhalb der einzelnen Cluster werden dazu zunächst separat durch ein SLAM-Verfahren geschätzt. Anschließend wird in einem weiteren SLAM-Verfahren die Cluster-übergreifende Trajektorie geschätzt, wobei jeder Cluster als eine Pose interpretiert und so die Anzahl der Posen in der gesamten Trajektorie reduziert wird. In der vorliegenden Arbeit sprechen darüber hinaus weitere Gründe für die

Aufteilung in Cluster: Zum Einen soll die Aufteilung in Cluster die Umsetzung weiterführender Anwendungen erleichtern. Solche Anwendungen können beispielsweise das gemeinsame Mapping mit mehreren Fahrzeugen oder das Mapping mehrerer, voneinander getrennter Fahrten (vgl. [SBH+15]) sein. Denkbar ist auch ein späteres Aktualisieren der Karten bei geändertem Umfeld; ein trivialer Ansatz wäre das Ersetzen der entsprechenden Cluster-Karten. Neben den Vorteilen für weitere Anwendungen soll die Schätzung der Landmarkenpositionen innerhalb der Cluster zu einem verbesserten loop-closing bei der anschließenden Cluster-übergreifenden Schätzung führen. Im zweiten Abschnitt des Verfahrens werden die Trajektorien und Landmarken innerhalb der einzelnen Cluster mittels GraphSLAM geschätzt. Der letzte Schritt des SLAM-Verfahrens ist die Schätzung der gesamten Trajektorie basierend auf den einzelnen Clustern.

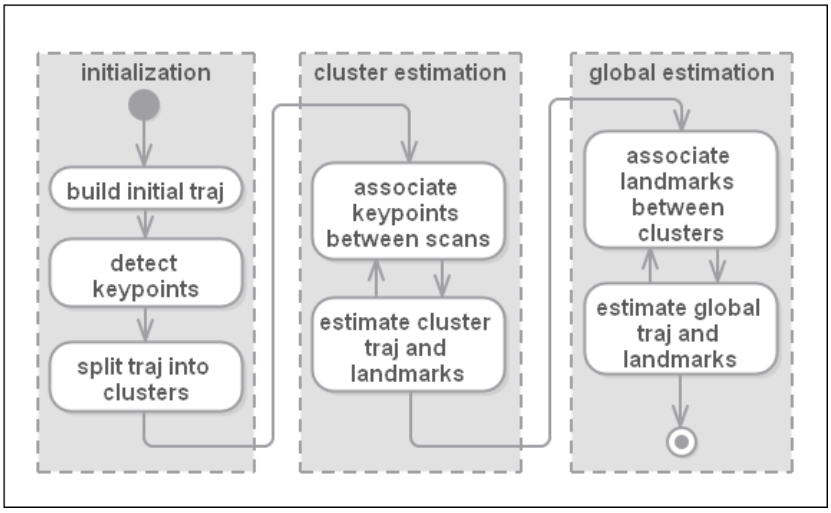


Abbildung 5.1.: Übersicht des SLAM-Verfahrens.

Nachfolgend werden zunächst die Anforderungen an das SLAM in Abschnitt 5.1 definiert. Anschließend wird das SLAM-Verfahren erläutert. Abschnitt 5.2 geht dazu auf die Initialisierung ein. Dem folgend wird die Assoziationsstrategie zur Zuordnung von Keypoints aus unterschiedlichen Scans in Abschnitt 5.3 behandelt. Abschnitt 5.4 beschreibt die Schätzung der Trajektorien und Landmarken innerhalb der Cluster und Abschnitt 5.5 die anschließende Schätzung der Gesamttrajektorie. Die Umsetzung des SLAM-Verfahrens ist in Abschnitt 5.6 und das anschließende Erzeugen der Gridkarte in Abschnitt 5.7 geschildert. Das Kapitel schließt mit der Evaluation in Abschnitt 5.8.

5.1. Anforderungen

In der vorliegenden Arbeit wird ausschließlich die Fahrzeugtrajektorie als Ergebnis des SLAM benötigt, da die zu erzeugende Karte anschließend anhand der Trajektorie sowie der Scans erzeugt wird. Die gleichzeitige Schätzung der Landmarken verfolgt dabei das Ziel, die Genauigkeit der geschätzten Trajektorie mittels der Laserscannerdaten zu verbessern. Das Verfahren soll die Fahrzeugodometrie, das INS sowie die Laserscannerdaten nutzen und die geschätzte Trajektorie soll lokal konsistent sein. Damit ist gemeint, dass zwischen kurz aufeinander folgenden Posen lediglich ein so geringer Fehler in der relativen Verschiebung und Rotation zueinander besteht, dass keine Sprünge in der Trajektorie entstehen. Darüber hinaus soll die Trajektorie bezüglich der Messdaten und der zugehörigen Messunsicherheiten plausibel sein, d. h. die durch die Messdaten gegebene tatsächliche Wahrscheinlichkeitsverteilung möglichst genau abbilden. Loops sind zu detektieren, sodass das doppelte Befahren einer Straße nicht zu versetzten Landmarken in der Karte führt.

5.2. Initialisierung

Für GraphSLAM wird eine initiale Trajektorie zur Linearisierung der Constraints benötigt. In der vorliegenden Arbeit wird diese mit Hilfe eines Kalman-Filters erzeugt, indem einzelne Posen für jeden Zeitschritt aus dem a posteriori Zustand zu einer Trajektorie zusammengefasst werden. Für das Kalman-Filter werden Fahrzeugodometrie und INS verwendet als Messungen verwendet. Der Zustand besteht dabei aus der Pose, der Geschwindigkeit und der Gierrate, wobei die Pose in Referenzkoordinaten angegeben ist. Als Ursprung der Referenzkoordinaten in Weltkoordinaten dient die Position der

ersten INS-Messung. Der Kalman-Filter wird entsprechend der Gleichungen 5.1 und 5.2 initialisiert.

$$\mu = \begin{pmatrix} x \\ y \\ \varphi \\ v \\ \omega \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{5.1}$$

$$\Sigma = \begin{pmatrix} 9999 & 0 & 0 & 0 & 0 \\ 0 & 9999 & 0 & 0 & 0 \\ 0 & 0 & 9999 & 0 & 0 \\ 0 & 0 & 0 & 9999 & 0 \\ 0 & 0 & 0 & 0 & 9999 \end{pmatrix} \tag{5.2}$$

Anschließend wird der Zustand mittels der INS- und Fahrzeugodometriemessungen aktualisiert (jeweils Prediction und Update) und der jeweilige a posteriori Zustand als Pose der initialen Trajektorie gespeichert. Das zur Prediction verwendete Bewegungsmodell ist im Grundlagenkapitel (Abschnitt 2.5) definiert, wobei die Geschwindigkeit und Gierrate aus dem aktuellen Zustand und nicht aus der Fahrzeugodometrie verwendet werden. Da im Bewegungsmodell eine konstante Geschwindigkeit und Gierrate angenommen werden, ändern sich diese Zustandsparameter während der Prediction nicht. Das Bewegungsmodell wird je Zeitschritt an der Stelle des aktuellen Zustands linearisiert. Hieraus ergibt sich der Prediction-Schritt mit der State Transition Matrix F (Gleichungen 5.3 bis 5.5). Die in Gleichung 5.3 angegebenen partiellen Ableitungen sind der Übersicht halber im Anhang A.1 aufgeführt.

$$F = \begin{pmatrix} \frac{\partial g_x}{\partial x} & \frac{\partial g_x}{\partial y} & \frac{\partial g_x}{\partial \varphi} & \frac{\partial g_x}{\partial v} & \frac{\partial g_x}{\partial \omega} \\ \frac{\partial g_y}{\partial x} & \frac{\partial g_y}{\partial y} & \frac{\partial g_y}{\partial \varphi} & \frac{\partial g_y}{\partial v} & \frac{\partial g_y}{\partial \omega} \\ \frac{\partial g_\varphi}{\partial x} & \frac{\partial g_\varphi}{\partial y} & \frac{\partial g_\varphi}{\partial \varphi} & \frac{\partial g_\varphi}{\partial v} & \frac{\partial g_\varphi}{\partial \omega} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.3}$$

$$\overline{\mu} = F\,\mu \tag{5.4}$$

$$\overline{\Sigma} = F\,\Sigma\,F^T \tag{5.5}$$

Hierbei ist:

μ, Σ	die a posteriori Verteilung des vorigen Zeitschritts
$\overline{\mu}, \overline{\Sigma}$	die a priori Verteilung des aktuellen Zeitschritts
F	die state transition matrix des aktuellen Zeitschritts
$\frac{\partial g_a}{\partial b}$	Die Ableitung nach b des Bewegungsmodells für den Zustandsparameter a.

Die Measurement Updates für INS-Messungen und Fahrzeugodometriemessungen sind durch die allgemeinen Kalman-Filter-Update-Gleichungen 2.4 bis 2.8 gegeben. Die Gleichungen 5.6 bis 5.8 zeigen die dabei jeweils verwendete INS Messung z_{INS} mit der Unsicherheit R_{INS}[1] und dem INS Messmodell H_{INS}.

$$z_{INS} = \begin{pmatrix} x_{INS} & y_{INS} & \varphi_{INS} & v_{INS} & \omega_{INS} \end{pmatrix}^T \tag{5.6}$$

$$R_{INS} = \begin{pmatrix} \sigma_x\sigma_x & 0 & 0 & 0 & 0 \\ 0 & \sigma_y\sigma_y & 0 & 0 & 0 \\ 0 & 0 & \sigma_\varphi\sigma_\varphi & 0 & 0 \\ 0 & 0 & 0 & \sigma_v\sigma_v & 0 \\ 0 & 0 & 0 & 0 & \sigma_\omega\sigma_\omega \end{pmatrix} \tag{5.7}$$

$$H_{INS} = I \tag{5.8}$$

Hierbei ist:

z_{INS}	die INS Messung mit der Pose $(x_{INS}\ \ y_{INS}\ \ \varphi_{INS})^T$ und der Bewegung $(v_{INS}\ \ \omega_{INS})^T$
R_{INS}	die zugehörige Messunsicherheit
H_{INS}	das zugehörige Messmodell

[1] Das INS gibt keine Kovarianz, sondern Standardabweichungen der einzelnen Parameter an.

Die Odometrie Messung z_{Odo} mit Odometrie Messunsicherheit R_{Odo} und das Odometrie Messmodell H_{Odo} sind in den Gleichungen 5.9 bis 5.11 angegeben.

$$z_{Odo} = \begin{pmatrix} v_{Odo} & \omega_{Odo} \end{pmatrix}^T \tag{5.9}$$

$$R_{Odo} = \begin{pmatrix} \sigma_v \sigma_v & 0 \\ 0 & \sigma_\omega \sigma_\omega \end{pmatrix} \tag{5.10}$$

$$H_{Odo} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.11}$$

Hierbei ist:

z_{Odo}	die Fahrzeugodometrie Messung mit Bewegung $(v_{Odo} \;\; \omega_{Odo})^T$
R_{Odo}	die zugehörige Messunsicherheit
H_{Odo}	das zugehörige Messmodell

Nachdem alle Messungen verarbeitet wurden und je Zeitschritt die Pose $\mu_{x,y,\varphi}$ gespeichert wurde, wird die so entstandene Trajektorie in Cluster von $10m$ Ausdehnung unterteilt. Außerdem wird für alle Laserscans die Detektion der Keypoints mit dem in Kapitel 4 beschriebenen Algorithmus durchgeführt.

Sowohl zur Schätzung der Trajektorie und der Landmarken innerhalb der einzelnen Cluster als auch zur anschließenden Schätzung der gesamten Trajektorie und Landmarken wird eine Assoziationsstrategie benötigt, um die Keypoints der unterschiedlichen Scans einander zuzuordnen. Dies ist im nachfolgenden Abschnitt 5.3 erläutert.

5.3. Assoziation von Features

Im Grundlagenabschnitt über GraphSLAM (Abschnitt 2.4) wird known correspondence angenommen. D. h., dass die Zugehörigkeit von Messung zur Landmarke, bzw. in diesem Fall zwischen den Keypoints unterschiedlicher Scans, bekannt ist. Dies ist in der vorliegenden Arbeit jedoch nicht der Fall, sodass ein Algorithmus benötigt wird, um diese Zuordnungen herzustellen. In [vD16] werden verschiedene Assoziationsstrategien vorgestellt und bezüglich Punktlandmarken untersucht. Die Ergebnisanalyse zeigt, dass je nach Szenario unterschiedliche Ansätze geeignet sind. So sind ortsbasierte Verfahren, wie z. B. Nearest-Neighbor, insbesondere für die Assoziation von Features über kurze Zeiträume geeignet, da der relative Fehler der geschätzten Eigenpose zwi-

schen den Messzeitpunkten meist gering ist. Für die Zuordnung zwischen zeitlich weit auseinander liegenden Keypoint-Messungen sind hingegen eher Verfahren geeignet, welche die Konstellation der gemessenen Keypoints pro Scan betrachten.

Aufgrund der kurzen Abschnitte von $10m$ kann innerhalb der Cluster bereits bei der initialen Trajektorie von geringen relativen Fehlern der Posen zueinander ausgegangen werden, sodass sich hier ein Assoziationsansatz anbietet, der auf die räumliche Nähe von Features abzielt. Hierbei soll die Positionsunsicherheit der Keypoints in die Assoziation einfließen. Daher werden die Wahrscheinlichkeitsdichtefunktionen der Keypoint-Positionen miteinander verglichen, um Assoziationen innerhalb der Cluster sowie zwischen den Landmarken benachbarter Cluster zu erzeugen. Die Assoziation von Landmarken zwischen unterschiedlichen Clustern, die nicht benachbart sind, erfolgt per sogenannter Triangle Association [WKD05], da bei diesen der relative Fehler zwischen den Posen größer ausfallen kann und die Position einzelner Landmarken allein nicht als Kriterium genügt. Insbesondere zur Erkennung von Loops genügt keine Zuordnung auf Basis der Position einzelner Keypoints.

Nachfolgend wird zunächst eine Bhattacharyya-Distanz-basierte Strategie, die zur Assoziation innerhalb der Cluster und zwischen direkt aufeinanderfolgenden Clustern dient, und anschließend die Triangle Association zur Zuordnung der Landmarken unterschiedlicher Cluster erläutert.

Zuordnung mittels Bhattacharyya-Distanz

Die Bhattacharyya-Distanz [Bha46] ist ein Distanzmaß zwischen zwei Verteilungen und wird oftmals zum Vergleich von Histogrammen oder Gaußverteilungen verwendet. Da die Position eines Keypoints stets als gaußförmige Wahrscheinlichkeitsverteilung mit Mittelwert und Kovarianz angegeben wird, kann die Bhattacharyya-Distanz als Distanzmaß zwischen zwei Keypoints verwendet werden. Für zwei Gaußverteilungen

$n$ter Dimension mit den Mittelwerten $\mu_a, \mu_b \in \mathbb{R}^n$ und den Kovarianzen $\Sigma_a, \Sigma_b \in \mathbb{R}^{n \times n}$ wird die Distanz wie folgt berechnet:

$$\Sigma = \frac{\Sigma_a + \Sigma_b}{2} \tag{5.12}$$

$$D_B = \frac{1}{8}(\mu_a - \mu_b)^T \Sigma^{-1}(\mu_a - \mu_b) + \frac{1}{2} ln \left(\frac{det\Sigma}{\sqrt{det\Sigma_a det\Sigma_b}} \right) \tag{5.13}$$

Die Bhattacharyya-Distanz wird in dieser Arbeit verwendet, um Keypoints verschiedener Scans einander zuzuordnen. Dabei sollen alle Keypoints der selben Landmarke zugeordnet werden, deren Distanz zueinander einen Schwellwert ϵ nicht überschreitet. Außerdem sollen all jene Keypoints der selben Landmarke zugeordnet werden, deren Distanz zu einem gemeinsamen dritten Keypoint den Schwellwert unterschreiten, auch wenn die Distanz zwischen diesen Keypoints selbst die Schwelle überschreitet. Listing 5.1 zeigt den verwendeten Algorithmus als Pseudocode und wird nachfolgend erläutert.

Die Eingangsdaten sind die Keypoints $z_{0:N}$ der Scans 0 bis N mit ihren Messunsicherheiten $Q_{0:N}$ sowie die entsprechenden aktuellen Poseschätzungen $x_{0:N}$ mit ihren Unsicherheiten $P_{0:N}$. Diese werden auf die Kartenschätzung m und die Korrespondenzvariable $c_{0:N}$ abgebildet. Dazu werden zunächst die Keypoints in Referenzkoordinaten überführt (Zeilen 2 bis 5) und in z_{all} und Q_{all} gespeichert. Die Zuordnung der Keypoints zueinander erfolgt anschließend in den Zeilen 7 bis 20 auf diesen Mengen, indem allen zur selben Landmarke gehörenden Keypoints dieselbe Segment-ID zugeordnet wird. `segIdCnt` ist ein Counter, welcher die jeweils nächste unbenutzte Segment-ID angibt und wird in Zeile 7 initialisiert. Die Korrespondenzvariable c_{tmp} gibt für jeden Keypoint die Segment-ID an und wird in den Zeilen 8 und 9 initialisiert.

In den Zeilen 11 bis 20 erfolgt die eigentliche Assoziation, bei der über alle Keypoints $z_{all,l} \in z_{all}$ iteriert wird. Dazu wird dem Keypoint $z_{all,l}$ zunächst die nächste unbenutzte Segment-ID zugeordnet, falls dieser noch zu keinem Segment gehört (Zeilen 12 und 13). Die innere Schleife iteriert über alle nachfolgenden Keypoints $z_k \in z_{all} : k > l$ (Zeilen 14 bis 20), wobei die Bhattacharyya-Distanz zwischen z_k und $z_{all,l}$ überprüft wird (Zeile 15). Liegt diese unter dem Schwellwert ϵ, dann wird der entsprechende Keypoint z_k bzw. das gesamte Segment, dem er angehört, in das Segment des aktuellen Keypoints $z_{all,l}$ aufgenommen (Zeilen 16 bis 20).

```
associateBhattacharyya(z_{0:N}, Q_{0:N}, x_{0:N}, P_{0:N})
    for each z_i ∈ z_{0:N}
        z_{i,Ref}, Q_{i,Ref} = toRefFrame(z_i, Q_i, x_i, P_i)
        add all z_{i,Ref,k} ∈ z_{i,Ref} to z_all
        add all Q_{i,Ref,k} ∈ Q_{i,Ref} to Q_all

    segIdCnt = 0
    for each z_{all,l} ∈ z_all
        c_{tmp,l} = −1

    for each z_{all,l} ∈ z_all
        if (c_{tmp,l} == −1)
            c_{tmp,l} = segIdCnt++
        for each z_k ∈ z_all : k > l
            if (bhattacharrya(z_{all,l}, Q_{all,l}, z_{all,k}, Q_{all,k}) < ε)
                if (c_{tmp,k} == −1)
                    c_{tmp,k} = c_{tmp,l}
                else
                    for each c_{tmp,i} ∈ c_tmp : c_{tmp,i} == c_{tmp,k}
                        c_{tmp,i} = c_{tmp,l}

    segments = empty map<int, set<Keypoints> >
    for each z_{all,l} ∈ z_all
        add z_{all,l} to segments[c_{tmp,l}]
    m = combine(segments)
    idx = 0
    for each z_i ∈ z_{0:N}
        for each z_{i,k} ∈ z_i
            c_{i,k} = c_{tmp,idx}
            idx++

    return m, c_{0:N}
```

Listing 5.1: Assoziation von Messungen unterschiedlicher Scans zueinander.

Nachdem die Assoziation abgeschlossen ist, wird in den Zeilen 22 bis 30 die Karte m erstellt und das Zwischenergebnis der Assoziation in die Korrespondenzvariable $c_{0:N}$ überführt. Dazu werden alle Keypoints, die zum selben Segment mit der ID $c_{tmp,l}$ gehören, jeweils der Menge `segments`$[c_{tmp,l}]$ hinzugefügt und jedes dieser Segmente zu einer Landmarke kombiniert (Zeilen 22 bis 25). Die Kombination mehrerer Messungen zu einer Landmarke erfolgt analog zu dem Verfahren in der Keypoint Detektion, in der ebenfalls mehrere Messungen zu einer Landmarke kombiniert werden (vgl. Abschnitt 4.3), wobei hier keine zusätzliche Messunsicherheit in die Kovarianzmatrix einfließt. Das Füllen der Korrespondenzvariable $c_{0:N}$ erfolgt in den Zeilen 26 bis 30. c_{tmp} enthält bereits alle Zuordnungen, gibt diese jedoch für die Menge aller Keypoints in Referenzkoordinaten , z_{all}, an. Dementsprechend ist je Keypoint $z_{i,k}$ lediglich die Zuordnung $c_{tmp,idx}$ zu übernehmen, die dem selben Keypoint in Referenzkoordinaten zugeordnet ist.

Zuordnung mittels Triangle Association

T. Weiss hat in seinen Publikationen das Verfahren „Triangle Association“ (TrAss) vorgestellt [WKD05, Wei11]. Der Algorithmus dient in den genannten Arbeiten dem Assoziieren von Messungen zu Landmarken in einer Karte. Da hierbei die aktuelle Poseschätzung stark fehlerbehaftet oder sehr unsicher sein kann, genügt die Position einzelner Messungen nicht als Kriterium.

Der TrAss-Algorithmus vergleicht aus diesem Grund die Konstellation von Messungen in einem Zeitschritt mit der Konstellation von Landmarken in der Karte. Abbildung 5.2 zeigt dies an einem Beispielszenario. Je Landmarke $L_i \in m$, die im Sichtbereich des Sensors liegt, werden anhand einer Region of Interest die Kandidaten $K_j \in z$ aus der Messung identifiziert. Es werden Dreiecke aus allen Dreierkombinationen der Landmarken (in der Abbildung blau dargestellt) und aus allen Dreierkombinationen der Kandidaten (in der Abbildung rot dargestellt) gebildet.

Das Verfahren basiert auf dem Prinzip, dass ein Dreieck vollständig durch dessen drei Seitenlängen beschrieben werden kann. Alle Dreiecke werden als Punkte in einem dreidimensionalen Koordinatensystem aufgetragen, wobei die Koordinaten jeweils den Seitenlängen des Dreiecks entsprechen, wie es im unteren Koordinatensystem der

Abbildung 5.2 dargestellt ist. Die Dreiecke der Kandidaten werden als Kandidatenpunkte und das Dreieck der Landmarken als Modellpunkt bezeichnet. Die jeweilige Distanz eines Kandidatenpunktes zum Modellpunkt entspricht der Abweichung der Konstellation der drei Kandidaten von der Konstellation der drei Landmarken (oberes Koordinatensystem der Abbildung). Die Kandidatenkombinationen, deren Abweichung unter dem Schwellwert ϵ_Δ liegt, gelten als gültige Dreiecke. Die Kandidaten eines gültigen Dreiecks werden den entsprechenden Landmarken des Landmarkendreiecks zugeordnet.

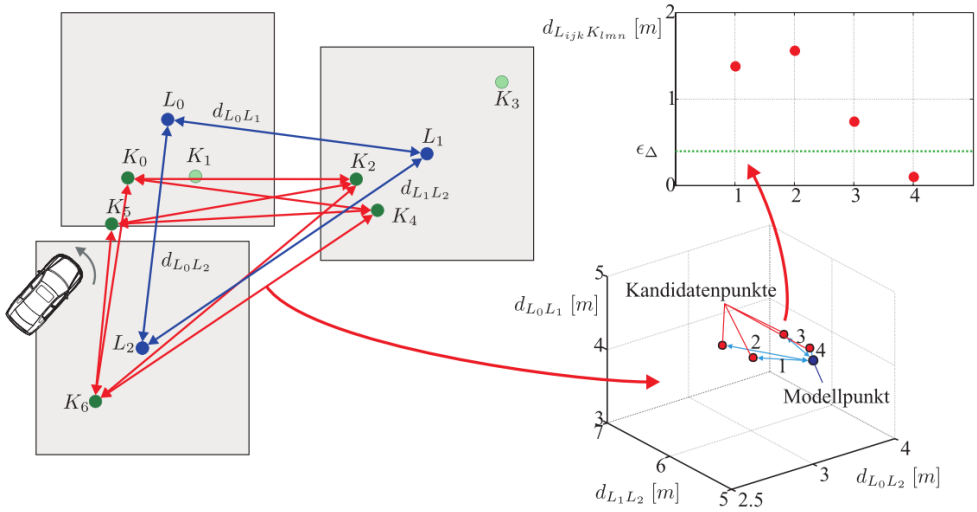


Abbildung 5.2.: Beispielszenario zur Triangle Association. Die Landmarken der Karte sowie deren Dreieckskombination sind in blau, die ROI der Landmarken in grau dargestellt (hier quadratisch). Die Messungen sind in grün und deren Dreieckskombinationen in rot dargestellt. Bildquelle: [Wei11].

In der vorliegenden Arbeit wird die Assoziation von Landmarken unterschiedlicher Cluster mittels angepasster Triangle Association durchgeführt. Listing 5.2 zeigt den Algorithmus als Pseudocode und wird nachfolgend erläutert.

```
1 associateTrass(m, z)
2     candidates = empty map<landmark, list <measurement> >
3
4     for each L_i ∈ m, F_i ∈ z
5         if (||L_i − F_i|| < ε_roi)
6             append F_i to candidates[L_i]
```

```
7   for each L_i ∈ candidates
8       for each F_i ∈ candidates[L_i]
9           candidateValid = false;
10          for each L_j ∈ candidates : L_j ≠ L_i
11              for each F_j ∈ candidates[L_j] : F_j ≠ F_i
12                  if (abs(||L_i − L_j|| − ||F_i − F_j||) < ε_dist)
13                      candidateValid = true
14                      break 2
15          if (¬ candidateValid)
16              delete F_i from candidates[L_i]
17      if (candidates[L_i] == ∅)
18          delete L_i from candidates
19
20  for each L_i ∈ candidates
21      for each F_i ∈ candidates[L_i]
22          hasValidTriangle = false
23          for each L_j ∈ candidates : L_j ≠ L_i
24              for each F_j ∈ candidates[L_j] : F_j ≠ F_i
25                  for each L_k ∈ candidates : L_k ≠ L_i, L_k ≠ L_j
26                      for each F_k ∈ candidates[L_k] : F_k ≠ F_i, F_k ≠ F_j
27                          e_{i,j} = (||F_i − F_j|| − ||L_i − L_j||)^2
28                          e_{j,k} = (||F_j − F_k|| − ||L_j − L_k||)^2
29                          e_{k,i} = (||F_k − F_i|| − ||L_k − L_i||)^2
30                          if (sqrt(e_{i,j}^2 + e_{j,k}^2 + e_{k,i}^2) < ε_Δ)
31                              v_L = ( L_j − L_i ) × ( L_k − L_i )
                                      (   1   )     (   1   )
32                              v_F = ( F_j − F_i ) × ( F_k − F_i )
                                      (   1   )     (   1   )
33                              if (v_{L,z} · v_{F,z} > 0)
34                                  hasValidTriangle = true
35                                  break 4
36          if (¬ hasValidTriangle)
37              delete F_i from candidates[L_i]
38      if (candidates[L_i] == ∅)
39          delete L_i from candidates
40
41  c = empty map<measurement_idx, landmark_idx>
42  for each (L_n, F_m) ∈ candidates
43      c_m = n
```

```
44      for each F_i ∈ z : i ∉ c
45          c_i = −1
46
47      return c
```

Listing 5.2: Assoziation von Messungen zur Karte mittels TrAss.

Der in dieser Arbeit genutzte, angepasste TrAss-Algorithmus ist grob in vier Abschnitte strukturiert: In den Zeilen 2 bis 6 werden die Kandidaten zu den Landmarken anhand der ROI identifiziert, wobei `candidates[`L_i`]` die Menge der zur Landmarke L_i gehörenden Kandidaten ist. Anschließend wird in den Zeilen 7 bis 18 das „Abstandskriterium“ überprüft. Hierbei gilt, dass die Entfernung zwischen zwei Kandidaten in etwa mit der Entfernung der jeweiligen Landmarken übereinstimmen muss (ϵ_{dist}). In den Zeilen 20 bis 39 erfolgt die Prüfung des „Dreieckskriteriums“ sowie die „Dreieckverifikation“. Diese Schritte werden im nachfolgenden Absatz näher erläutert. Zuletzt wird in den Zeilen 41 bis 45 als Ergebnis die Korrespondenzvariable c, die die Zuordnung jeder Messung $F_m \in z$ zur Landmarke L_n ist, erstellt. Ein Ergebnis mit dem Wert -1 bedeutet an dieser Stelle, dass die Messung nicht zugeordnet werden konnte.

Das Dreieckskriterium (Zeile 27 bis 30) stellt die zuvor erläuterte und in Abbildung 5.2 dargestellte Überprüfung von Dreieckskombinationen dar. Hierbei wird jeweils für den Kandidaten F_i überprüft, ob eine gültige Dreieckskombination von Kandidaten existiert, indem in den Zeilen 27 bis 29 das Fehlerquadrat jeder Seite des Dreiecks berechnet und in Zeile 30 der gesamte Fehler anhand des Schwellwertes ϵ_Δ überprüft wird. Wurde eine gültige Kandidatenkombination gefunden, wird die Dreiecksverifikation durchgeführt. Dazu werden die Vektoren, die von der Landmarke L_i zu L_j und L_k zeigen, sowie die Vektoren von Kandidat F_i zu F_j und F_k um die z-Komponente mit $z = 1$ erweitert und jeweils das Kreuzprodukt der Landmarkenvektoren und der Kandidatenvektoren gebildet (Zeilen 31 und 32). Das Kreuzprodukt gibt den Vektor an, der senkrecht zu beiden Vektoren steht. Haben die z-Komponenten der beiden Kreuzprodukte das selbe Vorzeichen, so ist die Zuordnung valide. Hierdurch wird sichergestellt, dass ein Kandidatendreieck nicht gespiegelt zum Landmarkendreieck ist.

Das originale Triangle-Association-Verfahren enthält darüber hinaus zwei weitere Kriterien, die als Kombination der Dreiecke und als Gitterkriterium bezeichnet werden. Diese

Verifikationsschritte wurden hier ausgelassen, da empirische Überprüfungen gezeigt haben, dass diese bei den vorliegenden Daten kaum Auswirkungen auf das Ergebnis haben. Für eine Beschreibung des ursprünglichen Verfahrens sowie für eine tiefergehende Beschreibung der zuvor erläuterten Kriterien wird auf [Wei11, Abschnitt 4.4.3] verwiesen.

Das vorgestellte Verfahren ist in der Lage, zwei Mengen von Landmarken, z.B. die Landmarken zwischen zwei Clustern oder zwischen einer Karte und einem Cluster, zu assoziieren. Listing 5.3 zeigt das Verfahren, um die Landmarken von N Clustern zu assoziieren und, basierend auf den aktuellen Clusterposen $x_{0:N}$, eine Karte von Landmarken zu erzeugen. Die Landmarkenkarte dient der Linearisierung der Constraints im SLAM. Die Eingangswerte sind die Landmarkenkarten der Cluster $z_{0:N}$, welche auf die resultierende Karte sowie die Korrespondenzvariable abgebildet werden: $z_{0:N}, x_{0:N} \rightarrow m, c$.

```
1  associateClusterTrass(z_{0:N}, x_{0:N})
2      segments = empty map<segId, set<landmarks> >
3      segIdCnt = 0
4      for each z_i ∈ z_{0:N}
5          m_tmp = combine(segments)
6          z_{i,Ref} = toRefFrame(z_i, x_i)
7          c_i = associateTrass(m_tmp, z_{i,Ref})
8          for each z_{i,Ref,k} ∈ z_{i,Ref}
9              if (c_{i,k} < −1)
10                 c_{i,k} = segIdCnt++
11             add z_{i,Ref,k} to segments[c_{i,k}]
12     m = combine(segments)
13     return m, c
```

Listing 5.3: Assoziation von Landmarken zwischen Clustern

Die Landmarken der einzelnen Clusterkarten (Input) werden in den Zeilen 2 bis 11 nacheinander mittels Triangle Association einander zugeordnet und so in Gruppen (`segments`) eingeordnet. Jede Gruppe repräsentiert dabei eine Landmarke der resultierenden Karte m (Output). In den Zeilen 4 bis 11 wird über die einzelnen Cluster iteriert. In jedem Iterationsschritt wird zunächst eine temporäre Karte anhand der

bisher identifizierten Gruppen erzeugt, indem alle Landmarken einer Gruppe zu einer einzelnen Landmarke kombiniert werden (Zeile 5). Die Landmarken des aktuellen Clusters i werden dann zu den Landmarken der temporären Karte assoziiert (Zeilen 6 und 7).

Anschließend werden die Landmarken des aktuellen Clusters jeweils der Gruppe hinzugefügt, zu der sie assoziiert wurden (Zeilen 8 bis 11). Zeilen 9 und 10 decken dabei den Fall ab, dass die aktuelle Clusterlandmarke $z_{i,k}$ keiner Gruppe zugeordnet werden konnte und somit eine neue Gruppe hinzuzufügen ist. Dies ist zum Beispiel im ersten Iterationszyklus der Fall, da hier noch keine Gruppe existiert. In diesem Fall gilt zunächst nach der siebten Zeile $\forall c_{0,k} \in c_0 : c_{0,k} = -1$, sodass durch die Zeilen 9 und 10 für jede Landmarke des ersten Clusters eine neue Gruppe begonnen wird. Nachdem alle Cluster verarbeitet wurden, wird zuletzt die Ergebniskarte m aus den Gruppen erzeugt (Zeile 12).

Die Kombination mehrerer Messungen zu einer Landmarke in den Zeilen 5 und 12 erfolgt analog zu dem Verfahren in der Keypoint Detektion, in der ebenfalls mehrere Messungen zu einer Landmarke kombiniert werden (vgl. Abschnitt 4.3). An dieser Stelle fließt jedoch keine zusätzliche Messunsicherheit in die Kovarianzmatrix ein.

5.4. Schätzung von Clustertrajektorie und -landmarken

Wie zuvor beschrieben, wird bei der Initialisierung die gesamte Trajektorie in Cluster von $10m$ Ausdehnung unterteilt. Dieser Abschnitt erläutert die Schätzung der Trajektorie und der Landmarkenpositionen innerhalb dieser Cluster mittels GraphSLAM (siehe Abschnitt 2.4). Listing 5.4 führt das angewendete Verfahren als Pseudocode auf und wird nachfolgend erläutert. Dieses Verfahren wird dabei für jeden Cluster separat durchgeführt.

Die Eingangsdaten zur Schätzung der Trajektorie und der Landmarken sind: Die initiale Trajektorieschätzung $\mu_{0:N}$ mit der Unsicherheit je Pose $\Sigma_{0:N}$, die Fahrzeugodometrie $u_{0:N}$ mit den Unsicherheiten $R_{0:N}$ sowie die Keypoints $z_{0:N}$ mit den Unsicherheiten $Q_{0:N}$, wobei $z_i \in z_{0:N}$ jeweils die Menge der Keypoints in dem Zeitschritt i und Q_i die Menge der entsprechenden Kovarianzen ist. Im Grundlagenkapitel wurde beschrieben, dass

```
1  optimizeCluster(μ_{0:N}, Σ_{0:N}, u_{0:N}, R_{0:N}, z_{0:N}, Q_{0:N})
2      while μ_{0:N} not converged
3          ξ = 0          // information vector
4          Ω = 0          // information matrix
5          c, m = associateBhattacharyya(μ_{0:N}, Σ_{0:N}, z_{0:N}, Q_{0:N})
6          for i = 0...N
7              if i > 0
8                  x̄ = g(μ_{i-1}, u_i)
9                  Ω_{i,i-1} = Ω_{i,i-1} + ( -G_i^T ) R_i^{-1} ( -G_i  I )
                                           (  I    )
10                 ξ_{i,i-1} = ξ_{i,i-1} + ( -G_i^T ) R_i^{-1} (x̄ - G_i · μ_{i-1})
                                           (  I    )
11             for all keypoints z_{i,k} ∈ z_i
12                 l = c_{i,k}
13                 z̄ = h(μ_i, m_l)
14                 tmpState = ( μ_{i,x}  μ_{i,y}  μ_{i,φ}  m_{l,x}  m_{l,y} )^T
15                 Ω_{i,k} = Ω_{i,k} + H_{i,k}^T  Q_{i,k}^{-1}  H_{i,k}
16                 ξ_{i,i-1} = ξ_{i,i-1} + H_{i,k}^T  Q_{i,k}^{-1}  (z_{i,k} - z̄ + H_{i,k}·tmpState)
                                                                    ( 10^-4   0     0    )
17         Ω_{N/2,N/2} = Ω_{N/2,N/2} + (  0     10^-4   0    )^-1
                                       (  0      0     10^-4 )
18         ξ_{N/2,N/2} = ξ_{N/2,N/2} + ( 0  0  0 )^T
19         μ_{0:N+M} = solve(Ω, ξ)
20         Σ_{0:N+M} = Ω^{-1}
21     return μ_{0:N+M}, Σ_{0:N+M}
```

Listing 5.4: Schätzung der Trajektorie und der Landmarken innerhalb eines Clusters. Basierend auf [TBF05]

GraphSLAM ein iteratives Näherungsverfahren ist. Die Schleife in den Zeilen 2 bis 18 wird dementsprechend ausgeführt, bis die Trajektorieschätzung $\mu_{0:N}$ konvergiert.

In jedem Optimierungsschritt werden zunächst der information vector und die information matrix initialisiert (Zeilen 3 und 4). Diese Variablen stellen das linearisierte Gleichungssystem dar. Daraufhin werden die Keypoints der einzelnen Zeitschritte auf Basis der aktuellen Trajektorieschätzung einander zugeordnet (Zeile 5). Diese Zuordnung geschieht wie im Abschnitt 5.3 beschrieben mittels Bhattacharyya-Distanz und liefert die Korrespondenzvariable c sowie die aktuelle Schätzung der Landmarkenkarte, die zur Linearisierung der Keypointconstraints verwendet werden. Anschließend wird in den Zeilen 6 bis 16 über die Zeitschritte iteriert, um die Constraints je Zeitschritt zu linearisieren und in die information matrix und den information vector zu übernehmen.

In den Zeilen 7 bis 10 wird jeweils das Constraint aus der Fahrzeugodometrie zwischen den Posen μ_{i-1} und μ_i aufgestellt. Dazu wird zunächst der Erwartungswert $\overline{x}$ der $i$ten Pose basierend auf dem Bewegungsmodell $g(\mu_{i-1}, u_i)$ berechnet (Zeile 8). Der information gain wird aus dem linearisierten Bewegungsmodell G_i und der Kovarianz der Fahrzeugodometrie R_i errechnet und auf die information matrix addiert (Zeile 9). Analog wird die Information basierend auf dem linearisierten Bewegungsmodell G_i, der Kovarianz R_i, der aktuellen Schätzung der Pose μ_{i-1} sowie dem Erwartungswert $\overline{x}$ der Pose μ_i auf den information vector addiert (Zeile 10). Der Erwartungswert $\overline{x}$ wird dabei durch das nicht linearisierte Bewegungsmodell errechnet. Das Bewegungsmodell $g(\mu_{i-1}, u_i)$ und dessen Linearisierung G_i an der Stelle μ_{i-1}, u_i sind in den Gleichungen 5.14 und 5.15 angegeben.

$$g(\mu_{i-1}, u_i) = \begin{cases} \mu_{i-1} + \begin{pmatrix} -\frac{v_i}{\omega_i} sin(\mu_{i-1,\varphi}) + \frac{v_i}{\omega_i} sin\left(\mu_{i-1,\varphi} + \omega_i \Delta t\right) \\ \frac{v_i}{\omega_i} cos(\mu_{i-1,\varphi}) - \frac{v_i}{\omega_i} cos\left(\mu_{i-1,\varphi} + \omega_i \Delta t\right) \\ \omega_i \Delta t \end{pmatrix} & \text{, falls } \omega \neq 0 \\ \mu_{i-1} + \begin{pmatrix} \Delta t \cdot v_i \cdot cos(\mu_{i-1,\varphi}) \\ \Delta t \cdot v_i \cdot sin(\mu_{i-1,\varphi}) \\ 0 \end{pmatrix} & \text{, sonst} \end{cases} \tag{5.14}$$

$$G_i = \begin{cases} \begin{pmatrix} 1 & 0 & -\frac{v_i}{\omega_i} cos(\mu_{i-1,\varphi}) + \frac{v_i}{\omega_i} cos\left(\mu_{i-1,\varphi} + \omega_i \Delta t\right) \\ 0 & 1 & -\frac{v_i}{\omega_i} sin(\mu_{i-1,\varphi}) + \frac{v_i}{\omega_i} sin\left(\mu_{i-1,\varphi} + \omega_i \Delta t\right) \\ 0 & 0 & 1 \end{pmatrix} & \text{, falls } \omega \neq 0 \\ \begin{pmatrix} 1 & 0 & -\Delta t \cdot v_i \cdot sin(\mu_{i-1,\varphi}) \\ 0 & 1 & \Delta t \cdot v_i \cdot cos(\mu_{i-1,\varphi}) \\ 0 & 0 & 1 \end{pmatrix} & \text{, sonst} \end{cases} \tag{5.15}$$

Hierbei ist:		
	μ_i	die Pose im $i$ten Zeitschritt
	u_i	die Fahrzeugodometrie im $i$ten Zeitschritt
		mit Geschwindigkeit v_i und Gierrate ω_i
	$g(\mu_{i-1}, u_i)$	das Bewegungsmodell hier als Messmodell der Odometrie
	G_i	die Linearisierung von $g(\mu_{i-1}, u_i)$

In den Zeilen 11 bis 16 werden die Constraints der im $i$ten Zeitschritt gemessenen Keypoints dem Gleichungssystem hinzugefügt. Dazu wird über alle Keypoints des aktuellen Zeitschritts, $z_{i,k} \in z_i$, iteriert. Die Korrespondenzvariable $c_{i,k}$ gibt an, welche Landmarke der $k$te Keypoint des $i$ten Zeitschritts gemessen hat (Zeile 12). Mit dem Keypoint Messmodell h werden diese Landmarke und die aktuelle Schätzung der $i$ten Pose in Zeile 13 auf die erwartete Messung $\bar{z}$ abgebildet. In Zeile 14 werden die aktuelle Schätzung der Pose und die Position der Landmarke für nachfolgende Matrizenrechnungen in einem Spaltenvektor zusammengefasst. $H_{i,k}$ ist das an der Stelle μ_i, m_l linearisierte Keypoint Messmodell. In den Zeilen 15 und 16 wird das linearisierte Constraint zur information matrix und zum information vektor addiert. Das nichtlineare Messmodell $h(\mu_i, m_l)$ sowie dessen Linearisierung H_i sind in den Gleichungen 5.16 und 5.17 aufgeführt. Die im linearisierten Messmodell verwendeten partiellen Ableitungen sind der Übersicht halber in Anhang A.2 aufgeführt.

$$h(\mu_i, m_l) = \begin{pmatrix} -cos(\mu_{i,\varphi}) & -sin(\mu_{i,\varphi}) & 0 & cos(\mu_{i,\varphi}) & sin(\mu_{i,\varphi}) \\ sin(\mu_{i,\varphi}) & -cos(\mu_{i,\varphi}) & 0 & -sin(\mu_{i,\varphi}) & cos(\mu_{i,\varphi}) \end{pmatrix} \cdot \begin{pmatrix} \mu_{i,x} \\ \mu_{i,y} \\ \mu_{i,\varphi} \\ m_{l,x} \\ m_{l,y} \end{pmatrix} \tag{5.16}$$

$$H_{i,l} = \begin{pmatrix} \frac{\partial h_x}{\partial \mu_{i,x}} & \frac{\partial h_x}{\partial \mu_{i,y}} & \frac{\partial h_x}{\partial \mu_{i,\varphi}} & \frac{\partial h_x}{\partial m_{l,x}} & \frac{\partial h_x}{\partial m_{l,y}} \\ \frac{\partial h_y}{\partial \mu_{i,x}} & \frac{\partial h_y}{\partial \mu_{i,y}} & \frac{\partial h_y}{\partial \mu_{i,\varphi}} & \frac{\partial h_y}{\partial m_{l,x}} & \frac{\partial h_y}{\partial m_{l,y}} \end{pmatrix} \tag{5.17}$$

Hierbei ist:	μ_i	die Pose im $i$ten Zeitschritt
	m_l	die $l$te Landmarke
	$h(\mu_i, m_l)$	das Keypoint Messmodell
	$H_{i,l}$	die entsprechende Linearisierung von $h(\mu_i, m_l)$

Damit sind alle Constraints zwischen den Posen sowie zwischen Posen und Landmarken linearisiert und im Gleichungssystem enthalten. Diese sind jeweils relative Constraints zwischen Knoten im Graphen, sodass intuitiv betrachtet zwar die Form des Graphen bestimmt werden kann, jedoch nicht die Position des gesamten Graphen. Mathematisch bedeutet dies, dass das Gleichungssystem singulär ist und somit keine eindeutige Lösung existiert. Aus diesem Grund wird ein weiteres Constraint eingefügt, das besagt, dass die ½te Pose, d. h. die mittlere Pose, an der Koordinate $(0\ 0\ 0)^T$ liegt. Dies geschieht in den Zeilen 17 und 18, wobei Zeile 18 ausschließlich der Verdeutlichung im Pseudocode dient.

Nachdem in den Zeilen 6 bis 18 die Constraints linearisiert wurden, wird das Gleichungssystem in Zeile 19 gelöst, was zu einer verbesserten Schätzung der Trajektorie und der Landmarken führt. $\mu_{0:N+M}$ setzt sich hierbei zusammen aus der Trajektorie $\mu_{0:N}$ und der Abschätzung der M Landmarken $\mu_{N+1:N+M}$. Die Unsicherheitsschätzung als Kovarianz ergibt sich anschließend aus der Inversen der information matrix Ω (Zeile 20). Im nächsten Schleifendurchlauf werden die Constraints an der Stelle dieser verbesserten Trajektorienschätzung linearisiert. Da sich aufgrund der geänderten Trajektorie auch die Zuordnung von Keypoints ändern kann, wird auch die Zuordnung der Keypoints auf Basis der aktualisierten Trajektorie in jedem Zyklus wiederholt.

Die Optimierung ist abgeschlossen, wenn $\mu_{0:N}$ konvergiert ist, bzw. nur mehr minimalen Änderungen pro Zyklus unterliegt. Das Ergebnis (Zeile 21) besteht dann je Cluster aus der Schätzung der Trajektorie sowie der Landmarken und jeweils deren Unsicherheiten. Im nachfolgenden Abschnitt 5.5 wird die Schätzung der gesamten Trajektorie und aller Landmarken auf Basis dieser Clusterschätzungen beschrieben.

5.5. Schätzung der gesamten Trajektorie und der Landmarken

Zur Schätzung der gesamten Trajektorie wird jeder Cluster als ein Zustand angesehen, sodass je Cluster eine Pose im Graphen enthalten ist. Pro Cluster werden eine INS-Messung, eine Fahrzeugodometrie-Messung sowie die Landmarken der jeweiligen Cluster zur Erzeugung der Constraints genutzt. Listing 5.5 zeigt die Optimierung als Pseudocode und wird nachfolgend erläutert.

```
1  optimize(μ_{0:N}, Σ_{0:N}, u_{0:N}, R_{0:N}, z_{kp,0:N}, Q_{kp,0:N}, z_{INS,0:N}, Q_{INS,0:N})
2      while μ not converged
3          ξ = 0          // information vector
4          Ω = 0          // information matrix
5          c_adjacent, m_adjacent = associateBhattacharyya(μ_{0:N}, Σ_{0:N}, z_{kp,0:N}, Q_{kp,0:N})
6          c_loop, m_loop = associateClusterTrass(μ_{0:N}, z_{kp,0:N})
7          for i = 1...N
8              add odometry constraint to ξ, Ω
9              add adjacent keypoint constraints to ξ, Ω
10             add loop keypoint constraints to ξ, Ω
11             z̄ = h_INS(μ_i)
12             Ω_{i,i} = Ω_{i,i} + H_INS^T Q_{INS,i} H_INS
13             ξ_i = ξ_i + H_INS^T Q_{INS,i} (z_{INS,i} − z̄ + H_INS · μ_i)
14         μ_{0:N+M} = solve(Ω, ξ)
15         Σ_{0:N+M} = Ω^{-1}
16     return μ_{0:N+M}, Σ_{0:N+M}
```

Listing 5.5: Clusterübergreifende Schätzung der Trajektorie und der Landmarken.

Die Eingangsdaten (Zeile 1) sind die initiale Schätzung $\mu_{0:N}$, $\Sigma_{0:N}$, die Fahrzeugodometrie $u_{0:N}$, $R_{0:N}$, die Landmarken der Cluster $z_{kp,0:N}$, $Q_{kp,0:N}$ sowie eine INS Messung je Cluster $z_{INS,0:N}$, $Q_{INS,0:N}$. Das Verfahren wird, analog zum Schätzen der Trajektorie innerhalb der Cluster, iterativ wiederholt, bis diese konvergiert ist (Zeile 2). Dazu werden auch hier in jeder Iteration die Landmarken neu assoziiert und die Constraints aus der Fahrzeugodometrie sowie die Landmarken Constraints wie in Abschnitt 5.4 beschrieben linearisiert (Zeilen 3-10). Die Assoziation der Keypoints zwischen aufeinander folgenden Clustern erfolgt mittels Bhattacharyya-Distanz (Zeile 5 und 9), zwischen nicht benachbarten Clustern mittels Triangle Association (Zeile 6 und 10).

Zusätzlich wird pro Pose das Constraint der INS Messung ins Gleichungssystem aufgenommen (Zeilen 11 bis 13). Das hierbei verwendete Messmodell sowie dessen Linearisierung sind trivial, da es sich um direkte Messungen der Pose handelt. Sie sind in den Gleichungen 5.18 und 5.19 angegeben. Der letzte Schritt in jedem Schleifendurchlauf ist die Bestimmung der verbesserten Trajektorie, der Landmarken und jeweils deren Unsicherheiten (Zeilen 14 und 15).

$$h_{INS}(\mu_i) = \begin{pmatrix} \mu_{i,x} \\ \mu_{i,y} \\ \mu_{i,\varphi} \end{pmatrix} \tag{5.18}$$

$$H_{INS} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{5.19}$$

Hierbei ist:		
	μ_i	die Pose im $i$ten Zeitschritt
	$h_{INS}(\mu_i)$	das INS Messmodell
	H_{INS}	die Linearisierung des Messmodells

Das Ergebnis besteht aus der Trajektorie und den Landmarken (Zeile 16). Um hieraus die vollständige Trajektorie der Fahrt mit sämtlichen Posen zu erhalten, werden im Anschluss die zuvor innerhalb jeden Clusters separat geschätzten Posen anhand der optimierten Clusterpose in Referenzkoordinaten transformiert.

5.6. Umsetzung

Im Rahmen dieser Arbeit wurde das beschriebene SLAM-Verfahren prototypisch umgesetzt. Das iterative Schätzen der Trajektorie innerhalb der Cluster sowie anschließend der gesamten Trajektorie über alle Cluster erfolgt dabei mit einer festen Anzahl an Iterationsschritten. Die empirische Betrachtung hat gezeigt, dass sich die Trajektorie- und Landmarkenschätzung i. d. R. bereits nach wenigen Iterationen kaum weiter ändert, sodass auf eine explizite Überprüfung des Abbruchkriteriums „Konvergenz der Schätzung“ verzichtet wird.

Im Folgenden sind die Parameter aufgeführt, die für die zuvor beschriebenen Algorithmen verwendet werden:

Assoziation mittels Bhattacharyya Für die Zuordnung mittels Bhattacharrya Distanz innerhalb der Cluster wird im $i$ten Iterationsschritt der Schwellwert $\epsilon = 0,15 \cdot i + 0,1$ genutzt. Für die Zuordnung zwischen aufeinanderfolgenden Clustern wird im ersten Iterationsschritt der Schwellwert $\epsilon = 0,1$ und anschließend $\epsilon = 0,85$ verwendet.

Assoziation mittels TrAss Als Schwellwert der Region of Interest gilt $\epsilon_{roi} = 10m$. Als maximaler Distanzfehler zwischen zwei Landmarken und zwei Messungen gilt $\epsilon_{dist} = 0,1m$ und der maximal erlaubte Dreiecksfehler ist $\epsilon_{\Delta} = 0,15m$.

Schätzung der Cluster Es werden je Cluster stets fünf Iterationen durchgeführt, ohne eine Erkennung durchzuführen, ob die Schätzung konvergiert ist.

Schätzung der gesamten Trajektorie Es werden anstelle der Konvergenzüberprüfung stets zehn Iterationen durchgeführt. Da jeder Cluster hierbei als eine Pose angenommen wird, werden jeweils eine INS- und eine Fahrzeugodometriemessung pro Cluster genutzt. Als INS-Messung wird die Messung der mittleren Pose innerhalb des Clusters verwendet. Als Constraint der Odometrie wird die Messung verwendet, die die letzte Pose im vorigen Cluster mit der ersten Pose im aktuellen Cluster verbindet. Dies sind diejenigen Odometriemessungen, die nicht zur Schätzung innerhalb der Cluster herangezogen werden.

5.7. Erzeugung der evidenzbasierten Gridkarte

Das Ergebnis des SLAM-Verfahrens sind die gefahrene Trajektorie sowie eine Landmarkenkarte. Die Lokalisierung auf dieser Karte erfordert die Detektion der Keypoints während der Lokalisierungsfahrt sowie deren Assoziation zu den Landmarken der Karte. Werden bei der Lokalisierung andere Keypoints extrahiert als die Landmarken der Karte, so führt dies zu einer fehlerhaften Lokalisierung. Darüber hinaus zeigt [vD16], dass die Assoziationsverfahren jeweils in einigen typischen Umgebungen im Straßenverkehr gut geeignet, in anderen jedoch nicht ausreichend sind.

Die Rohdaten einer Messung, d. h. ein Scan in Form einer Punktwolke, bildet die Umgebung nicht-parametrisch ab und ermöglicht somit die Beschreibung der gesamten beobachteten aus beliebig komplexen Strukturen bestehenden Umgebung. Außerdem stellt die Menge aller Scans einer Fahrt die gesamte beobachtete Umgebung dar, sodass bei Umgebungsänderungen bis zu einem gewissen Grad weiterhin eine Lokalisierung anhand der übrigen Daten möglich ist. Als Beispiel sei hier ein parkendes Auto genannt, das bei späteren Fahrten dort nicht mehr steht. Bei einer rohdatenbasierten Lokalisierung besteht dadurch nicht die Problematik der Feature-Extraktion und -zuordnung.

Die Lokalisierung bei späteren Fahrten soll aus diesen Gründen rohdatenbasiert erfolgen. Hierzu wird eine Gridkarte erzeugt, die die während der Mappingfahrt beobachtete Umgebung abbildet. Wie in Kapitel 3 aufgezeigt wird, werden dazu typischerweise Occupancy Grids genutzt, die je Zelle die Belegtwahrscheinlichkeit angeben. In der vorliegenden Arbeit wird das in [JKK16] vorgestellte, auf der Dempster-Shafer Theorie basierende, Grid verwendet, das je Zelle unterschiedliche Aussagen ermöglicht: Ob die Zelle belegt ist, ob die Zelle freien Raum abbildet sowie ob sich anhand der Scans überhaupt eine Aussage über diese Eigenschaften treffen lässt.

Dieses Unterkapitel enthält nachfolgend eine kurze Übersicht über die Dempster-Shafer-Theorie und eine Übersicht der verwendeten Gridkarte und deren Schnittstellen. Weiterhin wird die Kartenerzeugung auf Basis der GraphSLAM-Trajektorie und der Scandaten beschrieben.

Dempster-Shafer Theorie

Die Evidenztheorie oder Dempster-Shafer-Theorie [Sha76] ist eine Erweiterung der Wahrscheinlichkeitstheorie. Sie dient dazu, Evidenzen aus verschiedenen Quellen unterschiedlicher Glaubwürdigkeit zusammenzuführen und so zu einem Glauben über den tatsächlichen Zustand zu gelangen. Dazu werden alle Möglichkeiten des Systemzustands durch sich gegenseitig ausschließende und den Zustandsraum vollständig abdeckende Hypothesen als Menge Ω definiert. Die Funktion $m : \mathcal{P}(\Omega) \rightarrow [0,1]$ bildet jedes Element der Potenzmenge von Ω auf eine sogenannte belief mass ab, die das Vertrauen darauf angibt, dass diese Teilmenge den tatsächlichen Zustand wiederspiegelt. Diese Funktion stellt die Zustandsrepräsentation dar.

Neben der mass function $m(A)$ sind zwei weitere Funktionen der Dempster-Shafer Theorie in dieser Arbeit von Bedeutung: Die belief function $bel(A)$ ist die Summe der Massen, die Teilmenge von A sind, und definiert somit die Untergrenze der Wahrscheinlichkeit von A (Gleichung 5.20). Die plausibility function $pl(A)$ definiert die Obergrenze der Wahrscheinlichkeit von A und gibt an, wie plausibel A ist. Dies ist definiert als die Summe der Massen von allen Teilmengen, deren Schnitt mit A nicht leer ist (Gleichung 5.21). $pl(A)$ ist somit das Maß der nicht-Widerlegung von A. Andersherum gibt $1 - pl(A)$ den Grad der Widerlegung von A durch die berücksichtigten Evidenzen an.

$$bel(A) = \sum_{B|B \subseteq A} m(B) \tag{5.20}$$

$$pl(A) = \sum_{B|B \cap A \neq \emptyset} m(B) \tag{5.21}$$

Dempster-Shafer Gridkarte

In dieser Arbeit wird die in [JKK16] vorgestellte Karte als Umgebungsabbildung verwendet. Es handelt sich dabei um eine Gridkarte mit adaptiver Zellauflösung, wobei der Zustand jeder Zelle als mass function repräsentiert wird. Die Menge der Propositionen ist definiert als $\Omega = \{o, f\}$, wobei o für belegt (occupied) und f für frei (free) steht. An dieser Stelle wird nur das Interface der Karte beschrieben, da die vorliegende Arbeit deren Entwicklung und Implementierung nicht umfasst, sondern diese als Teil des

verwendeten Software Frameworks der Ibeo Automotive Systems lediglich in dieser Arbeit genutzt wird.

Die Gridkarte wird iterativ aufgebaut, wobei das Eingangsinterface jeweils einen Scan in Sensorkoordinaten, die Anbauposition des Laserscanners am Fahrzeug sowie die aktuelle Fahrzeugpose in Referenzkoordinaten erwartet. Anhand dieser Daten wird die mass function der entsprechenden Zellen aktualisiert.

Für den Zugriff auf die Karteninhalte ist eine Zugriffsfunktion `search(`p`)` definiert, welche die Gridzelle zu einer gegebenen Koordinate p zurückgibt. Das Interface der Zelle stellt die Funktionen `getOccupiedProb()` und `getFreeProb()` bereit, welche die belief mass $m(\{o\})$ bzw. $m(\{f\})$ der Zelle zurückgeben. Es gibt exakt diese zwei Propositionen in der Menge Ω und es gilt stets $m(\emptyset) = 0$ und $\sum_{A \in \mathcal{P}(\Omega)} m(A) = 1$. Somit lässt sich die belief mass $m(\{o, f\})$ berechnen als $m(\{o, f\}) = 1 - m(\{o\}) - m(\{f\})$. Damit ist die gesamte Funktion m verfügbar und es lassen sich die belief function sowie die plausibility function berechnen.

Erzeugung der Dempster-Shafer Gridkarte

Die Gridkarte wird auf Basis der mittels des GraphSLAM-Verfahren (Kapitel 5) geschätzten Trajektorie, der Scandaten der Laserscanner sowie der von den Sensoren detektierten dynamischen Objekten erzeugt. Dazu wird über die einzelnen Fahrzeugposen iteriert und in jedem Zyklus jene Scanpunkte, die ein dynamisches Objekt gemessen haben, aussortiert und die Karte anhand der verbleibenden Scanpunkte und der Fahrzeugpose aktualisiert. Das Listing 5.6 zeigt den Ablauf als Pseudocode und wird nachfolgend beschrieben.

Die Eingangsdaten sind die Trajektorie $x_{0:N}$, die Scans je Zeitschritt $z_{0:N}$ und die zu den Scan gehörenden dynamischen Objekte $d_{0:N}$ (Zeile 1). In Zeile 2 wird die Karte initialisiert, sodass der Zustand aller Zellen als unbekannt angenommen wird. Damit gilt für alle Zellen: $m(\{o\}) = m(\{f\}) = 0$ und $m(\{o, f\}) = 1$. In den Zeilen 3 bis 8 wird über die Fahrzeugposen iteriert. In jedem Zyklus wird zunächst der aus allen Sensoren fusionierte Scan in die Scans der einzelnen Sensoren $z_{single,k}$ aufgeteilt (Zeile 4). Dann werden je Scan eines Sensors die Punkte in der Nähe von dynamischen Objekten aussortiert (Zeile 6), der Scan in Sensorkoordinaten transformiert (Zeile 7) und der

```
1 buildMap(x_{0:N}, z_{0:N}, d_{0:N})
2     m = initDSGrid()
3     for each x_t ∈ x_{0:N}
4         z_single = toSingleScans(z_t)
5         for each z_{single,k} ∈ z_single
6             z_tmp = filterDynamicPts(z_{single,k}, d_t)
7             z_tmp.transformToScannerCoordinates()
8             m.process(z_tmp, x_t)
9 return m
```

Listing 5.6: Erzeugen der Gridkarte anhand Scandaten, dynamischen Objekten und Fahrzeugtrajektorie.

Gridkarte hinzugefügt (Zeile 8). Das Ergebnis ist die vollständige Gridkarte (Zeile 9). Abbildung 5.3 zeigt beispielhaft eine solche Karte.

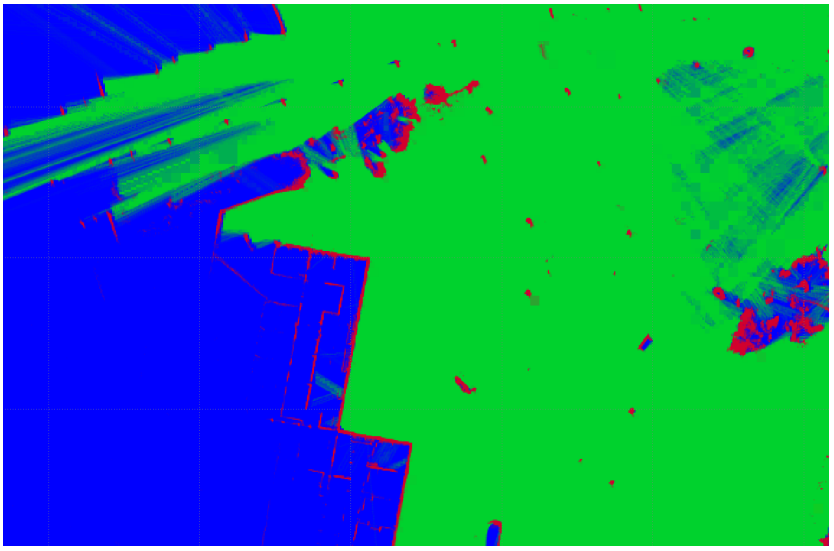


Abbildung 5.3.: Ausschnitt einer Dempster-Shafer Gridkarte. Blau entspricht $m(\{o, f\})$ (unbekannt), rot $m(\{o\})$ (belegt) und grün $m(\{f\})$ (frei).

5.8. Evaluation der Kartenerzeugung

Dieses Unterkapitel bewertet die Kartenerzeugung bezüglich ihrer Korrektheit und lässt die Positionsgenauigkeit außer Acht. Mit Positionsgenauigkeit ist die Abweichung der Karteninhalte von deren tatsächlichen Positionen in der realen Welt gemeint. Als „korrekt“ wird hier bezeichnet, dass die relative Position von Karteninhalten zueinander ihrer tatsächlichen relativen Position zueinander entspricht. Eine Karte, deren Inhalte die tatsächliche Umgebung exakt abbildet, aber in ihrer Gesamtheit um bspw. $1m$ verschoben ist, ist demnach korrekt, hat jedoch eine geringe Positionsgenauigkeit. Dagegen ist z. B. eine gestauchte Karte weder korrekt noch genau.

Genauigkeit und Korrektheit einer erzeugten Karte hängen an erster Stelle von der im SLAM-Verfahren geschätzten Trajektorie ab. Dementsprechend steht hier die Bewertung des SLAM-Verfahrens, insbesondere in Bezug auf die Korrektheit der Trajektorie, im Fokus. Die Anforderungen an das SLAM-Verfahren wurden in Abschnitt 5.1 aufgestellt und bestehen im Wesentlichen aus

1. Schätzung der Trajektorie anhand Fahrzeugodometrie, INS und Scans,
2. Annäherung der tatsächlichen Wahrscheinlichkeitsverteilung entsprechend der verwendeten Eingangsdaten,
3. geringer Fehler in der Poseänderung benachbarter Posen und
4. Erkennung von Loops.

Das vorgestellte Verfahren schätzt die Fahrzeugtrajektorie unter Verwendung der geforderten Eingangsdaten. Dabei ist das Ergebnis durch die Verwendung des GraphSLAM Ansatzes eine Näherung der tatsächlichen Wahrscheinlichkeitsverteilung. Die Erkennung von Loops erfolgt durch die Verwendung der Triangle Association. Damit sind die Punkte 1, 2 und 4 erfüllt. Zur genaueren Bewertung der Trajektorienschätzung (Listenpunkt 3) wurde als Referenzsystem zusätzlich ein RTK[2] System (Genesys ADMA) auf dem Versuchsfahrzeug verbaut, mit dessen Messungen die im GraphSLAM-Verfahren geschätzten Posen verglichen werden. Auf die Ergebnisse aus diesem Vergleich wird im Folgenden näher eingegangen.

[2]Ein Real Time Kinematics (RTK) System nutzt neben den empfangenen Sattelitendaten (bspw. GPS) zusätzlich Korrekturdaten von stationären Referenzstationen zur Bestimmung der Position und erreicht somit eine Genauigkeit von bis zu wenigen cm.

Die Globale Positionierung der Trajektorie hängt von den im Mapping verwendeten GPS-Daten ab, wohingegen die Fahrzeugodometrie sowie die Keypoints vorwiegend Einfluss auf die relative Poseänderungen haben. Vor diesem Hintergrund und zur Bewertung der Korrektheit wird entsprechend die Poseänderungen über die Zeit, d. h. die Form der Trajektorie, bewertet. Dazu wird jeweils die Poseänderung von der geschätzten Pose x_{t_1} zur geschätzten Pose x_{t_2} mit der Poseänderung von Referenzpose x_{ref,t_1} zu x_{ref,t_2} verglichen. Dabei wird stets die Poseänderung über 0.2 Sekunden betrachtet ($t_2 = t_1 + 0.2s$). Diese Bewertung wird auf denselben Strecken durchgeführt, die zur Bewertung der Keypoint Detektion (Abschnitt 4.5) ausgewählt wurden. Das Tunnelszenario wird nicht betrachtet, da die Keypoint Detektion hierbei keine befriedigenden Ergebnisse liefert.

Die Abbildungen 5.4 bis 5.7 zeigen die Ergebnisse der Bewertung anhand von Fehlerhistogrammen. Bei der Betrachtung der Ergebnisse ist zu beachten, dass im Rahmen der vorliegenden Arbeit keine explizite zeitliche Synchronisierung zwischen bewerteter Trajektorie und Referenztrajektorie erfolgte. Die in den Abbildungen dargestellten Abweichungen sind daher sowohl auf den zu bewertenden Fehler in der Trajektorie als auch auf den Zeitversatz der Trajektorien zurückzuführen.

Abbildung 5.4 zeigt das Ergebnis im Szenario „Autobahn“. Die Fehler befinden sich in Fahrtrichtung im Bereich weniger Zentimeter, wobei die jeweils zurückgelegte Distanz in der Zeit $t_2 - t_1 = 0.2s$ bei einer Geschwindigkeit von $120^{km}/_{h}$ ca. $6,66m$ beträgt. Die Fehler in Querrichtung betragen ebenfalls wenige Zentimeter und die Rotationsfehler sind im Bereich einiger zehntel Grad. Die visuelle Inspektion der Trajektorie zeigt zwei Auffälligkeiten. Die Erkennung von Loops führt am Trajektorienende zu Fehlern in der Trajektorie (siehe Abb. C.1 in Anhang C), welche jedoch die Lokalisierung anhand der Karte nicht offensichtlich verhindern. Außerdem ist die Referenztrajektorie für einen Streckenabschnitt stark gestört (siehe Abb. C.2).

Abbildung 5.5 zeigt das Ergebnis im Szenario „Industriegebiet“. Trotz der jeweils in den betrachteten $0,2s$ geringeren zurückgelegten Strecke wurden höhere Fehler gemessen als im Autobahnszenario. Sie befinden sich aber auch hier im Bereich weniger Zentimeter. Auffällig ist der hohe Anteil von Rotationsfehlern über $1°$. In der Nahaufnahme der Ergebnistrajektorie und der Referenztrajektorie (Abb. C.3) ist erkennbar, dass die Referenztrajektorie verrauscht ist, was diese Fehler begründen könnte. Abbildung C.4

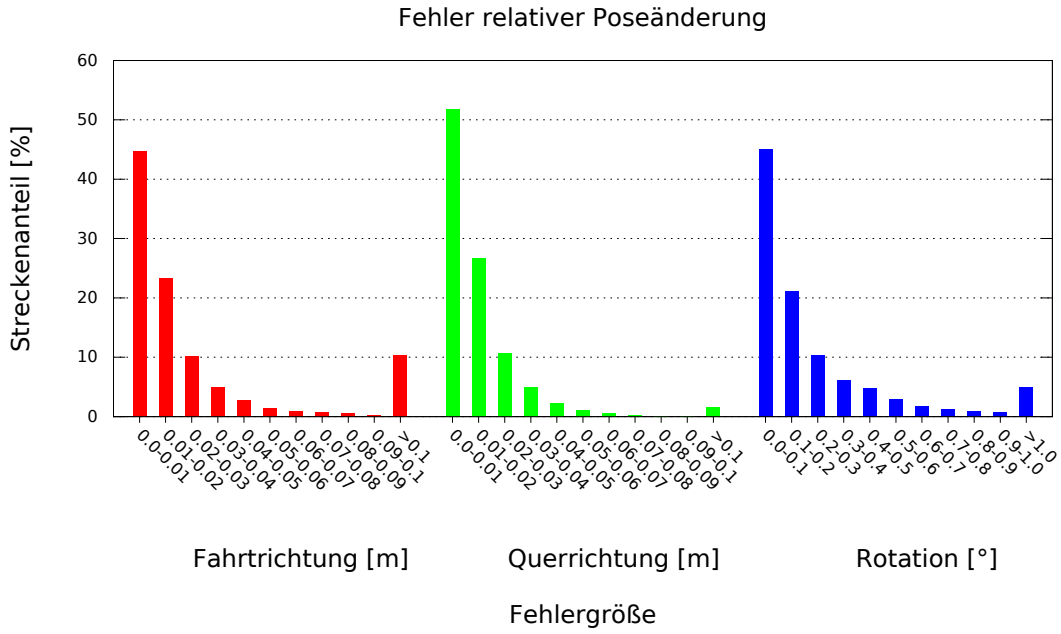


Abbildung 5.4.: Fehlerhistogramm der Trajektorieänderungen auf der Strecke „Autobahn“

zeigt die Trajektorien in der Übersicht. Die visuelle Inspektion der Ergebnistrajektorie sowie der resultierenden Karte ergab keine Auffälligkeiten.

Abbildung 5.6 zeigt das Ergebnis im Szenario „Stadt“. Der Hauptteil der Fehlergrößen bewegt sich hier ebenfalls im Bereich weniger Zentimeter bei der Position bzw. zehntel Grad in der Rotation. Bei der visuellen Inspektion der Trajektorie fiel auf, dass an einigen Stellen der Positions- und Rotationsfehler zwischen verschiedenen Clustern gravierend ist. Da an diesen Stellen keine Landmarken aus den betroffenen Clustern mit Landmarken anderer Cluster assoziiert wurden, ist anzunehmen, dass die Clusterposen primär aus den GPS Constraints hervorgehen und die Fehler nicht wie im Autobahnszenario durch fehlerhafte Zuordnung von Landmarken verursacht ist. Vielmehr ermöglicht die Absenz von zugeordneten Landmarken an diesen Stellen das hohe Gewicht der GPS-Daten für die Pose der entsprechenden Cluster. Darüber hinaus sind infolge dessen auch die Abschnitte der Trajektorie, in der die Clusterposen zueinander korrekt sind, bezüglich Weltkoordinaten verschoben und rotiert. Die erzeugte Karte ist zur späteren Lokalisierung darauf nicht geeignet. Die Abbildungen C.5 bis C.7 zeigen die Fehler.

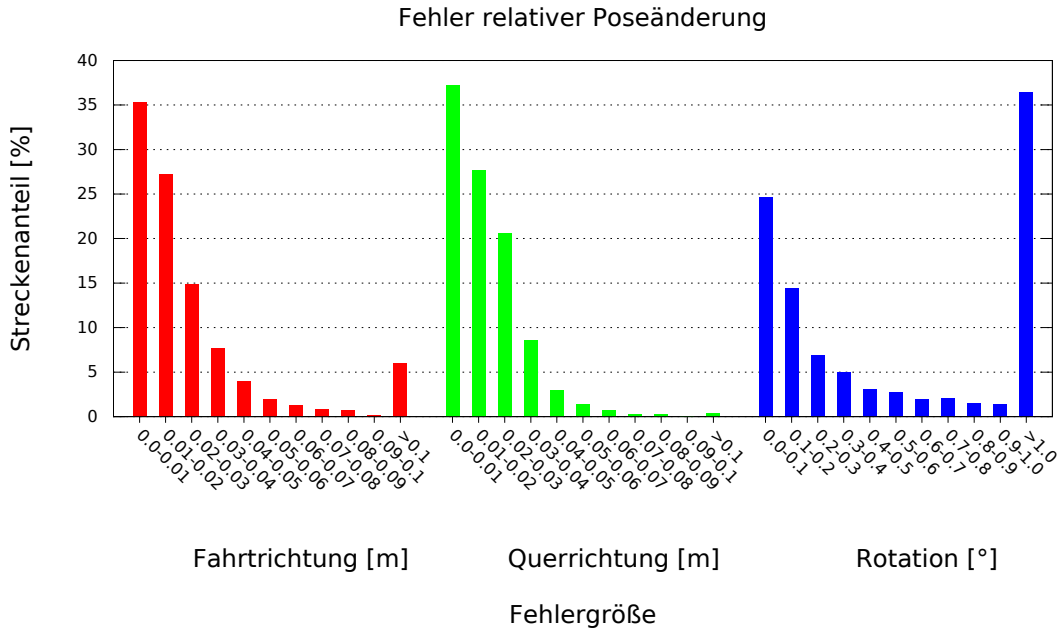


Abbildung 5.5.: Fehlerhistogramm der Trajektorieänderungen auf der Strecke „Industriegebiet“

Abbildung 5.7 zeigt das Ergebnis im Szenario „Landstraße“. Die Fehler befinden sich auch hier im Bereich weniger Zentimeter. Die visuelle Inspektion der Ergebnistrajektorie sowie der resultierenden Karte zeigte keine Auffälligkeiten. Die Referenztrajektorie ist auch bei dieser Strecke verrauscht (vgl. Abb. C.8).

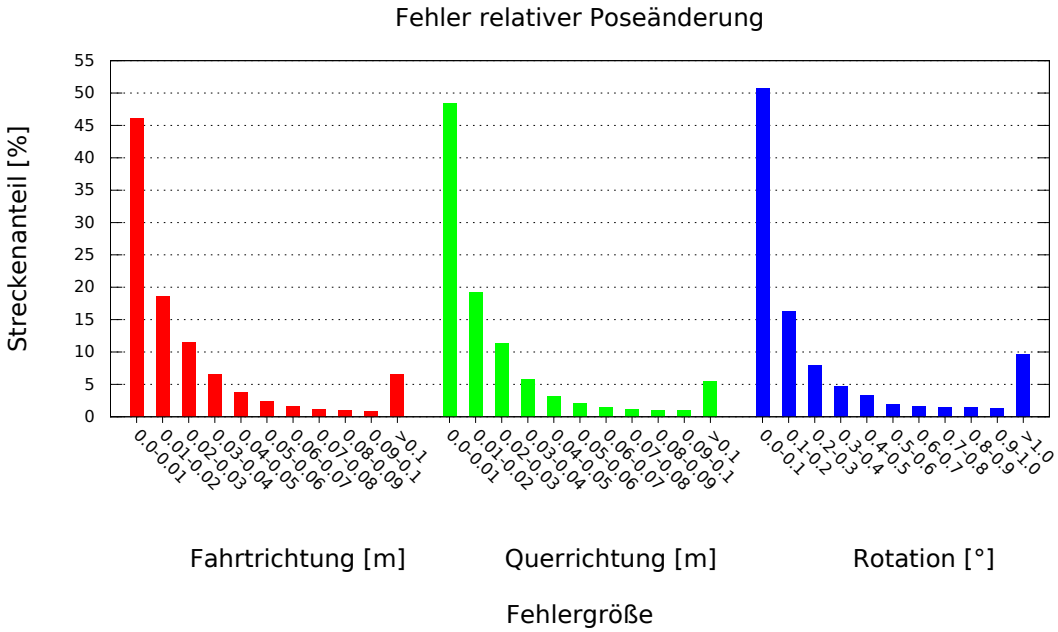


Abbildung 5.6.: Fehlerhistogramm der Trajektorieänderungen auf der Strecke „Stadt“

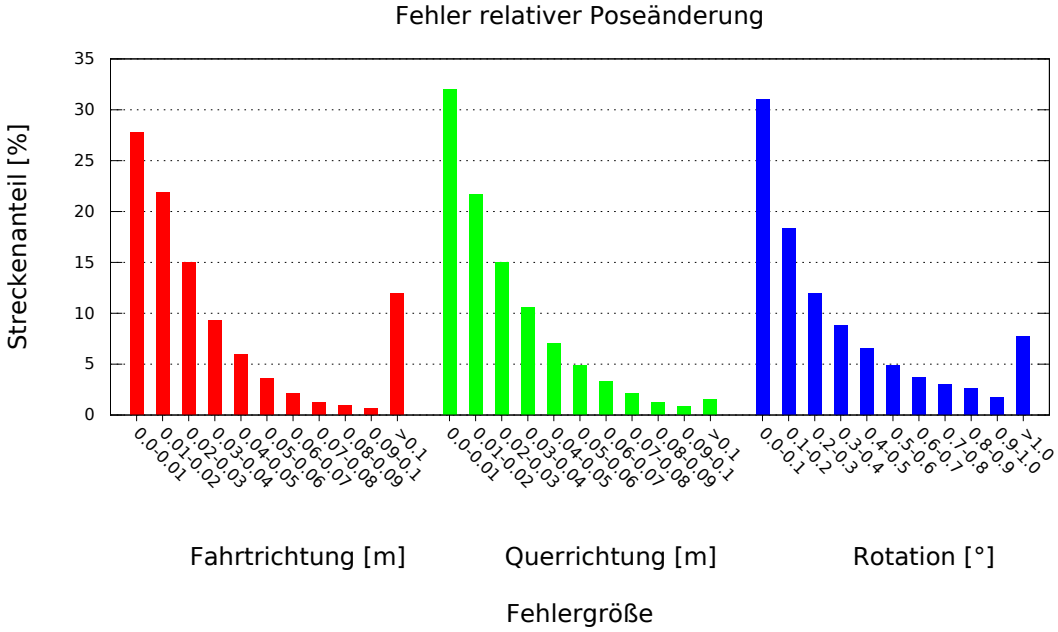


Abbildung 5.7.: Fehlerhistogramm der Trajektorieänderungen auf der Strecke „Landstraße“

6. Lokalisierung mittels Partikelfilter

Zur Lokalisierung auf der zuvor beschriebenen Gridkarte wird ein Partikelfilter (vgl. Abschnitt 2.3.2) genutzt, um so die Verfolgung des Zustands möglichst robust gegen Mehrdeutigkeiten beim Vergleich von Scandaten mit der Karte zu machen. Der verfolgte Zustand x ist die Fahrzeugpose und besteht aus Position und Gierwinkel in Referenzkoordinaten: $x = \begin{pmatrix} x_x & x_y & x_\varphi \end{pmatrix}^T$, sodass jeder Partikel x_i der Partikelmenge X eine konkrete Hypothese darstellt: $x_i = \begin{pmatrix} x_{i,x} & x_{i,y} & x_{i,\varphi} \end{pmatrix}^T$. Wie im Grundlagenkapitel beschrieben, besteht ein Zyklus im Partikelfilter aus den Schritten Prediction und Update, wobei das Update sich in die Gewichtung der Partikel und das anschließende Resampling unterteilt.

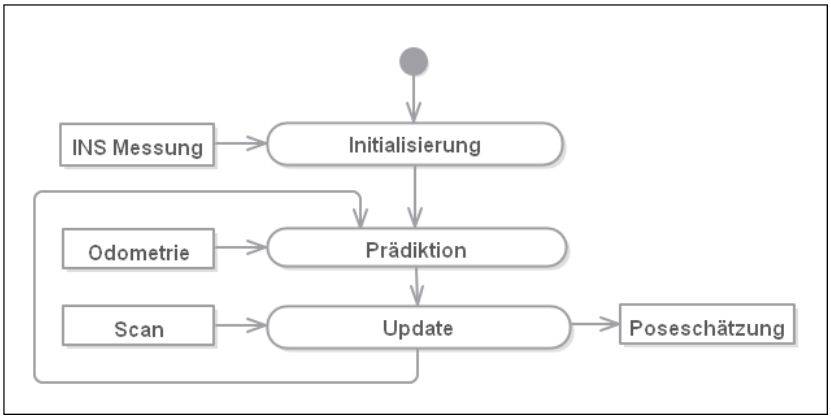


Abbildung 6.1.: Flussdiagramm des Partikelfilters zur Lokalisierung

In Abbildung 6.1 sind die einzelnen Schritte als Flussdiagramm dargestellt. Sie werden in den Abschnitten dieses Kapitels erläutert. Die Initialisierung des Partikelfilters anhand einer INS Messung wird in Abschnitt 6.1 behandelt. Abschnitt 6.2 beschreibt die Prediction mittels Odometrie und Bewegungsmodell. Anschließend wird das Update erläutert, wobei die Partikelgewichtung basierend auf der Gridkarte in Abschnitt 6.3 und das darauf folgende Resampling in Abschnitt 6.4 beschrieben wird. Die Abschnit-

te 6.5 und 6.6 umfassen als Abschluss des Kapitels die prototypische Umsetzung sowie die Evaluation des Lokalisierungsverfahrens.

6.1. Initialisierung des Filters

Ein Partikelfilter wird in der Theorie als Gleichverteilung initialisiert, indem alle Partikel gleichmäßig auf dem gesamten Zustandsraum verteilt werden. Dies ist aufgrund der Größe des Zustandsraumes hier jedoch nicht möglich. Die Initialisierung des Partikelfilters erfolgt daher auf Basis der ersten INS Messung. Dazu werden die Partikel entsprechend einer Gaußverteilung mit der INS Messung als Mittelwert erzeugt. Die Initialisierung ist als Pseudocode in Listing 6.1 aufgeführt. Die Eingangsdaten sind die Pose der INS Messung x_{INS}, deren Kovarianzmatrix Σ_{INS} sowie die Anzahl der zu erzeugenden Partikel N (Zeile 1). X ist die Menge der Partikel. In den Zeilen 3 bis 5 werden die Partikel erzeugt und der Menge der Partikel hinzugefügt. Die Funktion `sampleGaussian(`μ`,`Σ`)` erzeugt einen Partikel entsprechend der übergebenen Gaußverteilung.

```
1 pf_init(x_INS, Σ_INS, N)
2     X = ∅
3     for i = 1..N
4         x_i = sampleGaussian(x_INS, Σ_INS)
5         add x_i to X
6     return X
```

Listing 6.1: Initialisierung des Partikelfilters mittels INS.

6.2. Prediction

Bei der Prediction wird jeder Partikel basierend auf der Fahrzeugodometrie und dessen Unsicherheit mittels Bewegungsmodell prädiziert. Listing 6.2 zeigt den entsprechenden Pseudocode. Dabei wird jeweils die zur Prädiktion verwendete Odometrie $\overline{u}_t$ als

Zufallswert aus der Gaußverteilung u_t, R_t gezogen (Zeile 4). Dies führt zum einen dazu, dass die Unsicherheit der Fahrzeugodometrie korrekt abgebildet wird, und zum anderen werden somit die Partikel unterscheidbar, die beim Resampling vom selben Ursprungspartikel stammen. Die Prädiktion des jeweiligen Partikels erfolgt dann durch das Bewegungsmodell unter Verwendung der Odometrie $\overline{u}_t$ (Zeile 5).

```
1 pf_predict(X_{t-1}, u_t, R_t)
2     X̄_t = ∅
3     for each x_{t-1,i} ∈ X_{t-1}
4         ū_t = sampleGaussian(u_t, R_t)
5         x̄_i = g(x_{t-1,i}, ū_t)
6         add x̄_i to X̄_t
7     return X̄_t
```

Listing 6.2: Prediction des Partikelfilters mittels Fahrzeugodometrie.

6.3. Gewichtung der Partikel mittels Gridkarte

Das Partikelgewicht ist ein Maß für die Wahrscheinlich der Hypothese, die der Partikel repräsentiert, im Vergleich zu den übrigen Partikeln. Es wird im Allgemeinen durch das Sensormodell $p(z|x,m)$ bestimmt. Das Partikelgewicht gibt demnach an, wie wahrscheinlich die Messung z laut Karte ist, wenn sich das Fahrzeug an der jeweiligen Partikelpose befindet (vgl. Abschnitt 2.3.2). In dieser Arbeit wird die Gewichtung durch den Abgleich des aktuellen Scans mit der Dempster-Shafer Gridkarte bestimmt. Dazu wird der Scan je Partikelpose in das Koordinatensystem der Karte transformiert und das Gewicht durch die von Scanpunkten getroffenen Gridzellen berechnet. Aufgrund des Straßenumfelds als Umgebung ergeben sich vier Szenarien, welche durch die Gewichtungsfunktion zu berücksichtigen sind:

Unverändert Die Umgebung ist unverändert, sodass der Scan sehr gut mit der Karte übereinstimmt, wenn er an der korrekten Pose über die Karte gelegt wird.

Neue Objekte Es befinden sich Objekte im aktuellen Umfeld, die während der Mappingfahrt nicht vorhanden waren. Dies führt zu Scanpunkten an Positionen, an denen der Karte nach Freiraum zu erwarten wäre.

Fehlende Objekte Es fehlen Objekte im aktuellen Umfeld, die während der Mappingfahrt beobachtet wurden, wie z. B. parkende Fahrzeuge. Dies führt dazu, dass über belegte Zellen hinweg gemessen wird, also Scanpunkte hinter dem besagten Objekt in der Karte liegen.

Verschobene Objekte Wenn beispielsweise ein parkendes Fahrzeug zwischen Mappingfahrt und Lokalisierung etwas umgeparkt wurde, dann befinden sich Scanpunkte im aktuellen Scan, die nicht zur Karte passen, aber durch ihre Nähe zum Belegtraum in der Karte u. U. dazu führen, dass diese dem Belegtraum zugeordnet werden.

In der vorliegenden Arbeit erfolgt die Partikelgewichtung in drei Schritten (siehe Abbildung 6.2). Die Outlier Rejection verwirft Scanpunkte aus dem aktuellen Scan, die, für die Gesamtheit der Partikel betrachtet, auf Zellen mit hoher Freiraumwahrscheinlichkeit treffen. Auf diese Weise soll die Robustheit gegen *neue Objekte* erhöht werden. Es folgt die eigentliche Berechnung des Gewichts jedes Partikels. Um auch gegen *fehlende Objekte* möglichst robust zu sein, werden bei der Berechnung des Gewichts ausschließlich die Zellen der Gridkarte betrachtet, welche durch Scanpunkte gemessen wurden. Alternativ ist eine Betrachtung aller Zellen, die der Laserstrahl traversiert hat (ray tracing), denkbar. Ray tracing hat jedoch im Fall von *fehlenden Objekten* den Nachteil, dass ein Partikel trotz guter Zustandshypothese schlechter bewertet wird. Der Fall von *unveränderter* Umgebung ist implizit durch die Bewertungsfunktion abgedeckt. *Verschobene Objekte* stellen die größte Herausforderung dar, da je nach Verhältnis des Anteils von verschobener Umgebung zum Anteil unveränderter Umgebung u. U. nicht erkennbar ist, welcher Teil der Umgebung sich verändert hat. Sofern die Partikelverteilung sich stark an einem Punkt konzentriert, wird diese Problematik durch die Outlier Rejection begrenzt, da jene Scanpunkte aussortiert werden, die neben dem Objekt der Karte liegen. Nach der Berechnung der jeweiligen Partikelgewichte folgt die Normalisierung der Gewichte, sodass die Summe aller Gewichte 1 ergibt ($\sum_{w \in W} = 1$). In den nächsten Absätzen werden die Outlier Rejection und die Bewertungsfunktion detailliert erläutert.

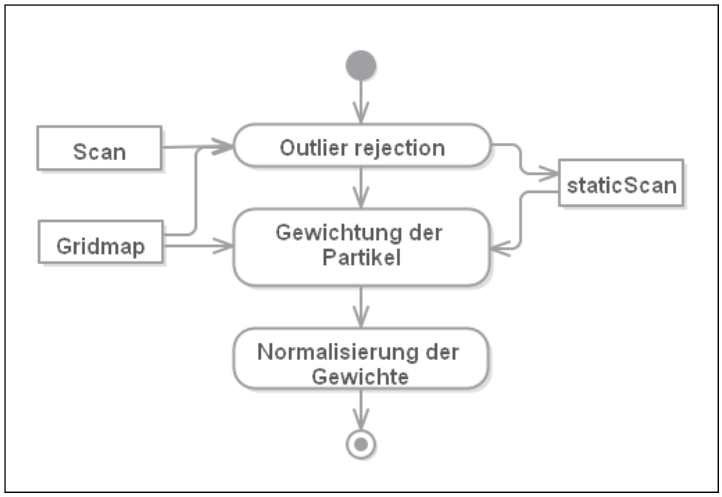


Abbildung 6.2.: Flussdiagramm der Partikelgewichtung

Outlier Rejection

Das Ziel der Outlier Rejection ist das Aussortieren von jenen Scanpunkten, die von Objekten stammen, die nicht in der Karte abgebildet sind (Szenario *neues Objekt*). Thrun, Burgard und Fox [TBF05] beschreiben dazu einen Algorithmus, der jene Scanpunkte aussortiert, die vom Sensor aus gesehen vor den Objekten in der Karte liegen. Dazu wird mittels Partikelverteilung und Karte die Wahrscheinlichkeit $p(short)$ berechnet, dass der Scanpunkt von einem *neuen Objekt* verursacht wurde. Aufgrund der von Thrun, Burgard und Fox verwendeten Kartenart und des dort verwendeten Sensormodells ist dazu jedoch ein Raytracing notwendig, um das in Messrichtung am nächsten liegende Objekt in der Karte zu identifizieren.

Die Verwendung des Dempster-Shafer-Grids erlaubt aufgrund der expliziten Modellierung von Freiraum ein daran angelehntes Verfahren, welches das rechenintensive Raytracing umgeht. Hier ist die grundlegende Idee, dass Scanpunkte an Orten mit hoher Freiraumwahrscheinlichkeit in der Karte von *neuen Objekten* verursacht wurden und daher nicht in die Partikelgewichtung eingehen sollten. Anstellte der Wahrscheinlichkeit $p(short)$ wird in dieser Arbeit das Verhältnis zwischen der Freiraummasse $m(\{f\})$ und der Unbekanntmasse $m(\{f,o\})$ für die Gesamtheit der Partikel betrachtet.

Listing 6.3 zeigt das modifizierte Verfahren als Pseudocode. Die Eingangsdaten sind die a priori Verteilung der Partikel $\overline{X}_t$, der Scan z_t sowie die Gridkarte m (Zeile 1). z_{static} ist die Ergebnisvariable und wird mit den Scanpunkten befüllt, welche nicht aussortiert werden. In den Zeilen 3 bis 10 wird über die einzelnen Scanpunkte $z_{t,i}$ iteriert. Der Scanpunkt wird anhand jedes Partikels in Referenzkoordinaten transformiert und die belief mass für *frei* (Zeile 7) sowie für *belegt* (Zeile 8) der entsprechenden Gridzelle summiert (Zeilen 4-8). p ist nun die Summe der belief mass für Freiraum über alle Partikel und q die Summe der belief mass für *belegt*. Der Scanpunkt wird als valide angenommen, wenn das Verhältnis $p/p+q$ einen Schwellwert ε nicht überschreitet (Zeilen 9 und 10). $p+q$ ist das Gegenteil der aufsummierten belief mass für unbekannt $bel(\{o,f\})$. Das Kriterium lässt sich demnach als Verhältnis von Freiraummasse zum Grad der Bekanntheit der Umgebung bezeichnen, wobei diese Werte bezüglich der a priori Partikelverteilung bestimmt werden.

```
1  pf_rejectOutlier(X̄_t, z_t, m)
2      z_static = ∅
3      for each z_{t,i} ∈ z_t
4          p = q = 0
5          for each x̄_{t,k} ∈ X̄_t
6              z_ref = transform(z_{t,i}, x̄_{t,k})
7              p = p + bel_{m_z_ref}({f})
8              q = q + bel_{m_z_ref}({o})
9          if (q > 0 ∧ p/(p+q) < ε)
10             add z_{t,i} to z_static
11     return z_static
```

Listing 6.3: Outlier rejection auf Basis der Partikelverteilung und der Karte.

Gewichtungsfunktion

Nach der Outlier Rejection werden die Partikel anhand der verbleibenden Scanpunkte gewichtet. Das Gewicht ergibt sich aus den Gridzellen, auf denen die gemessenen Scanpunkte liegen, nachdem sie jeweils an der Pose des $i$ten Partikels in Referenzkoordinaten transformiert wurden. Wie in Abschnitt 2.3.2 beschrieben, muss das Gewicht

proportional zur Wahrscheinlichkeit der Messung an der Pose des jeweiligen Partikels sein: $w_i \propto p(z|x_i, m)$. Diese Wahrscheinlichkeit ist das Produkt der Wahrscheinlichkeiten für die einzelnen Scanpunkte: $p(z|x_i, m) = \prod_{z_k \in z} p(z_k|x_i, m)$. Durch die hohe Anzahl an Scanpunkten je Scan zusammen mit der Wertdiskretisierung durch digitale Datentypen würde dies jedoch dazu führen, dass nahezu alle Partikel das Gewicht $w_i = 0$ erhalten. Daher wird in der vorliegenden Arbeit eine Distanzfunktion zwischen Messung (Scanpunkt) und Karte, $d(z_{i,k}, m)$, und darauf basierend das Gewicht des $i$ten Partikel, w_i, definiert. Dies ist in den Gleichungen 6.1 und 6.2 aufgeführt und wird nachfolgend erläutert.

$$d(z_{i,k}, m) := 3 \cdot \left(1 - \mathrm{bel}_{m_{z_{i,k}}}(\mathrm{o}) \cdot \mathrm{pl}_{m_{z_{i,k}}}(\mathrm{o})\right) \tag{6.1}$$

$$w_i := \frac{1}{N} \sum_{z_{i,k} \in z_i} e^{-d(z_{i,k}, m)} \tag{6.2}$$

Hierbei ist:

z_i	die Menge aller Scanpunkte; diese sind anhand der Pose des $i$ten Partikel transformiert
$z_{i,k}$	der $k$te Scanpunkt in z_i
N	die Anzahl aller Scanpunkte in z_i
m	die Gridkarte
$m_{z_{i,k}}$	die Gridzelle, auf der der Scanpunkt $z_{i,k}$ liegt
$d(z_{i,k}, m)$	das Distanzmaß zwischen Karte und Scanpoint
$\mathrm{bel}_{m_{z_{i,k}}}(\mathrm{o})$	die belief mass für „belegt“ in der Gridzelle $m_{z_{i,k}}$
$\mathrm{pl}_{m_{z_{i,k}}}(\mathrm{o})$	die plausibility für „belegt“ der Gridzelle $m_{z_{i,k}}$
w_i	das resultierende Gewicht des $i$ten Partikel

Das Gewicht eines Partikels ergibt sich hier als Mittelwert der Gewichte pro Scanpunkt. Das Gewicht eines Scanpunktes ist dabei definiert als $e^{-d(z_{i,k}, m)}$, sodass das Gewicht in exponentieller Abhängigkeit von der Abweichung zwischen Messung und Karte abfällt. Die Distanz zur Karte wird definiert als $3 \cdot \left(1 - \mathrm{bel}_{m_{z_{i,k}}}(o) \cdot \mathrm{pl}_{m_{z_{i,k}}}(o)\right)$. Daraus ergibt sich das nachfolgend beschriebene Verhalten für unterschiedliche belief masses in den Zellen:

- Eine hohe belief mass für „belegt“ und geringe belief mass für „frei“ führt zu einem hohen Gewicht, da sowohl *bel(o)* als auch *pl(o)* groß sind.

- Bei mittlerer belief mass für „belegt“ und ebenfalls mittlerer belief mass für “frei“ fällt *pl(o)* geringer aus, sodass auch das Gewicht geringer ausfällt. Dies kann auftreten, wenn aufgrund von dynamischen Objekten bei der Mappingfahrt eine Zelle zeitweise als belegt und zeitweise als frei gemessen wurde.
- Bei geringer belief mass für „belegt“ ist das Gewicht gering. In diesem Fall ist entweder die belief mass für „frei“ hoch, sodass sowohl *bel(o)* als auch *pl(o)* klein sind, oder die belief mass für „belegt oder frei“ (also Unbekannt) ist hoch, was zu einer hohen Plausibilität *pl(o)* bei weiterhin geringem *bel(o)* führt.
- Der Faktor 3 in der Distanzfunktion sorgt dafür, dass der Wertebereich von $e^{-d(z_{i,k},m)}$ bei $e^{-3} \approx 0,05$ anstatt bei $e^{-1} \approx 0,37$ beginnt. Dies führt zu einer entsprechend größeren Gewichtsdifferenz zwischen schlecht und gut zur Karte passenden Partikeln.

Bezüglich der vier eingehend genannten Szenarien verhält sich die Gewichtung folgendermaßen: Bei Messungen einer „**unveränderten**“ Umgebung werden nahezu keine Scanpunkte durch die Outlier Rejection aussortiert und den Partikeln mit einer guten Hypothese wird ein hohes Gewicht zugeordnet, da die Scanpunkte auf Zellen mit hoher Masse für „belegt“ und geringer Masse für „frei“ liegen. Stark von der Realität abweichende Partikelposen werden hingegen schlechter bewertet, da die Scanpunkte auf freien oder unbekannten Zellen liegen.

Bei einer guten Poseschätzung, d. h. bei geringer Verteilung der Partikel, wird der Fall „**Neue Objekte**“ durch die Outlier Rejection abgedeckt, sodass aufgrund der verbleibenden Scanpunkte ein Verhalten ähnlich dem Fall „unverändert“ zu erwarten ist. Bei großer Streuung der Partikel können die entsprechenden Scanpunkte jedoch nicht zuverlässig aussortiert werden. Das Mengenverhältnis von unveränderter Umgebung zu neuen Objekten hat in diesem Fall großen Einfluss auf das Gewicht der einzelnen Partikel.

Im Szenario „**Fehlende Objekte**“ ist zwischen zwei Fällen zu unterscheiden: Wenn von der Fahrzeugposition aus gesehen hinter dem fehlenden Objekt ebenfalls Informationen in der Karte vorhanden sind, dann ist eine Gewichtung wie bei unveränderter Umgebung anzunehmen. Sind diese Informationen jedoch nicht bekannt, sondern die Zellen hinter dem fehlenden Objekt haben eine sehr hohe Unbekanntmasse, dann hängt die

Gewichtung wiederum von Verhältnis der veränderten zur unveränderten Umgebung ab.

Das Szenario „**Verschobene Objekte**“ stellt die größte Herausforderung der vorgestellten Szenarien dar. Die Gewichtung hängt hier stets vom Verhältnis der von verschobenen Objekten verursachten Scanpunkte zu denen aus unveränderter Umgebung ab.

Im Allgemeinen lässt sich festhalten, dass eine stark veränderte Umgebung eine Systemgrenze für die korrekte Bewertung der Partikel und damit für die Lokalisierung darstellt.

6.4. Resampling

Der abschließende Schritt in jedem Zyklus des Partikelfilters ist das Resampling: das Erzeugen einer neuen Partikelmenge aus der bestehenden und somit die Aktualisierung der Wahrscheinlichkeitsverteilung (vgl. Abschnitt 2.3.2). In dieser Arbeit wird das in [Thr12] beschriebene, *Resampling Wheel* gennante, Verfahren angewendet. Dieses wird nachfolgend anhand des Pseudocodes in Listing 6.4 sowie des zugehörigen Beispiels in Abbildung 6.3 erläutert.

Das Gesamtgewicht aller Partikel wird als Rad / Kreis aufgefasst, wobei jedem Partikel ein Abschnitt des Kreises zugeordnet wird, dessen Größe dem jeweiligen Partikelgewicht entspricht (siehe Abbildung 6.3). Zu Beginn des Algorithmus werden die Anzahl auszuwählender Partikel N sowie das höchste Partikelgewicht w_{max} bestimmt (Zeilen 2-6). In den Zeilen 8 und 9 werden die Variable `idx`, die auf den jeweiligen Partikel X_{idx} und dessen Gewicht w_{idx} verweist, und die Hilfsvariable β initialisiert. Daraufhin wird in jedem Schleifendurchlauf in den Zeilen 10 bis 15 ein Partikel wie folgt ausgewählt.

In jedem Iterationsschritt wird ein zufälliger Wert aus dem Bereich $[0, 2 \cdot w_{max}]$ auf die Hilfsvariable β addiert (Zeile 11). `idx` $+\beta$ zeigt dann auf den Kreisabschnitt des zu wählenden Partikels (Abb. 6.3a). Um den entsprechenden Partikel auszuwählen, muss β kleiner als das Partikelgewicht sein, auf das `idx` zeigt. Dazu wird solange das aktuelle Partikelgewicht von β abgezogen und der Index `idx` inkrementiert, bis diese

Bedingung erfüllt ist (Zeilen 12 bis 14). Wie in Abbildung 6.3b dargestellt ist, gibt `idx` nun den auszuwählenden Partikel an. Dieser wird der Ergebnismenge X hinzugefügt (Zeile 15) und der nächste Schleifendurchlauf begonnen.

Die Abbildungen 6.3c und 6.3d zeigen beispielhaft zwei weitere Schleifendurchläufe. Es wird deutlich, dass durch die Aufteilung des Rads anhand der Partikelgewichte die Wahrscheinlichkeit, dass ein bestimmter Partikel ausgewählt wird, proportional zu dessen Gewicht ist. Ein Vorteil dieses Verfahrens ist, dass es nicht voraussetzt, dass die Summe der Gewichte 1 ergibt. Auf diese Weise ist keine Normalisierung der Partikelgewichte für das Resampling erforderlich.

```
1  resample(X̄, w)
2      N = |X̄|
3      w_max = 0
4
5      for each w_i ∈ w
6          w_max = max(w_max, w_i)
7
8      β = 0
9      int idx = uniform(0, N − 1)
10     for i = 0..(N − 1)
11         β = β + uniform(0, 2 · w_max)
12         while (β > w_idx)
13             β = β − w_idx
14             idx = (idx + 1) mod N
15         add X̄_idx to X
16
17     return X
```

Listing 6.4: Resampling Wheel Algorithmus [Thr12].

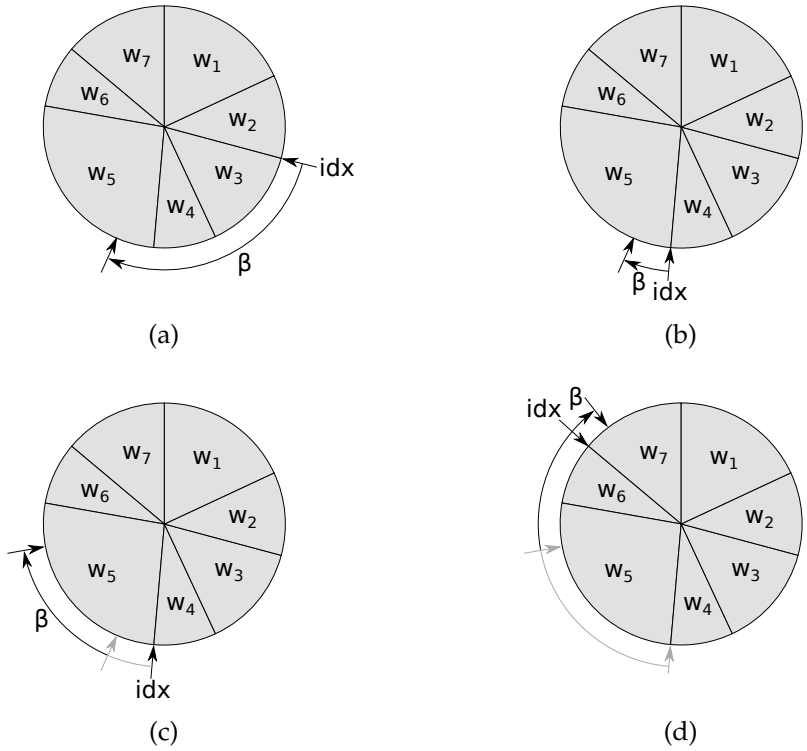


Abbildung 6.3.: Beispiel zum Resampling Wheel Algorithmus. (a) und (b) zeigen den ersten Auswahlzyklus, (c) und (d) jeweils einen Weiteren.

6.5. Umsetzung des Partikelfilters

Das beschriebene Verfahren zur Lokalisierung mittels Partikelfilter wurde im Rahmen der vorliegenden Arbeit prototypisch implementiert, wobei folgende Punkte zu nennen sind:

Map Lookup Tables Bei der Bewertung von N Partikeln mittels M Scanpunkten erfolgen $N \cdot M$ Zugriffe auf die Gridkarte, die als Quadtree implementiert ist. Um das $N \cdot M$-fache Traversieren des Baumes zu umgehen, wird jeweils ein lokales Grid mit fester Zellgröße in Form eines Arrays für die Freimasse und eines für die Belegtmasse erzeugt. Beide lokale Gridkarten umfassen jeweils $200m \times 200m$ und werden an der aktuellen Positionsschätzung (Mittelwert der Partikel) er-

zeugt. Die Erzeugung dieser „Lookup Tables“ erfolgt in einem Thread parallel zur Poseschätzung.

Parallele Partikelgewichtung Die Bewertung der Partikel erfolgt parallelisiert in weiteren Threads, wobei in dieser Arbeit sieben Threads zur Bewertung eingesetzt werden. Diese Stelle bietet eine einfache Möglichkeit, die Laufzeit durch Erhöhung der Threads zu verbessern.

Output als Gaußverteilung Das Outputinterface ist für Anzeigezwecke als Gaußverteilung (Mittelwert und Kovarianz) umgesetzt, wobei die Verteilung jeweils aus der a posteriori Verteilung der Partikel berechnet wird. Bei annähernd Unimodaler Verteilung der Partikel nähert sich der Output der tatsächlichen Partikelverteilung an. Multimodale Partikelverteilungen werden so jedoch nur schlecht repräsentiert.

6.6. Evaluation der Lokalisierung

Die Bewertung der Lokalisierung erfolgt analog zur Bewertung des SLAM-Verfahrens anhand des RTK Referenz Systems, wobei wieder die Poseänderung im Lokalisierungsergebnis mit der Poseänderung der Referenz über den Zeitraum von jeweils $0.2s$ betrachtet wird. Diese Bewertung erfolgt jeweils anhand einer gesonderten Fahrt auf der Strecke des jeweiligen Szenarios (Lokalisierungsfahrt).

Außerdem ist bei der Lokalisierung insbesondere die Genauigkeit der Fahrzeugpose bezüglich der Karte von Interesse. Um diese Genauigkeit zu bewerten, wird eine Referenztrajektorie benötigt, die exakt zur Karte passt. Aus diesem Grund wird zusätzlich zur Lokalisierung einer Lokalisierungsfahrt auch jeweils eine Lokalisierung mit der Mappingfahrt, also mit den selben Eingangsdaten, die beim Erstellen der Karte genutzt wurden, durchgeführt. Diese Lokalisierungsergebnisse werden mit der im Mapping erzeugten Trajektorie verglichen, da diese exakt mit der Karte übereinstimmt.

Beide beschriebenen Evaluationsverfahren werden in den Szenarien „Autobahn“, „Industriegebiet“ und „Landstraße“ durchgeführt. Das Szenario „Tunnel“ wurde bereits beim Mapping ausgeschlossen und im Szenario „Stadt“ konnte keine zur Lokalisierung geeignete Karte erzeugt werden (vgl. Abschnitt 5.8). Die Ergebnisse sind in den Abbil-

dungen 6.4 bis 6.9 dargestellt und werden nachfolgend erläutert. Es ist zu beachten, dass für die Histogramme unterschiedliche Größen der Histogramklassen verwendet wurden, um die jeweilige Verteilung möglichst aussagekräftig darzustellen. Des Weiteren wurden die Posen der ersten Sekunden nicht in die Bewertung aufgenommen, um mögliche Auswirkungen aus der Initialisierungsphase, in der die Partikelwolke weit verteilt ist, auszuschließen.

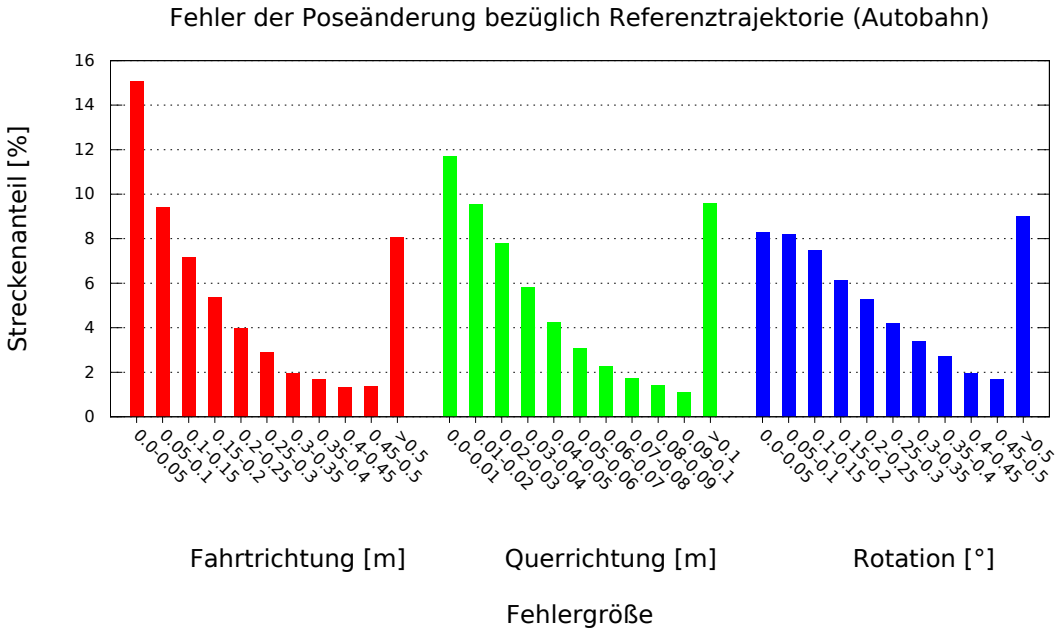


Abbildung 6.4.: Fehlerhistogramm der Poseänderung der Lokalisierung ggü. RTK System. Szenario „Autobahn“.

Abbildung 6.4 zeigt die Fehler der Poseänderung über $0.2s$ im Vergleich zum RTK System im Szenario „Autobahn“. Auffällig ist der hohe Fehler in Fahrtrichtung in der Größe einiger Dezimeter. Dieser ist in der Art der Umgebung begründet. Auf weiten Teilen der Strecke sind hauptsächlich Leitplanken gemessen worden, die aufgrund ihrer parallelen Ausrichtung kaum eine Abschätzung in Fahrtrichtung erlauben. Die Fehler in Querrichtung befinden sich in der Größenordnung einiger Zentimeter und die Rotationsfehler im Bereich einiger zehntel Grad.

Abbildung 6.5 zeigt die Fehler der Posen relativ zur Karte im Szenario „Autobahn“. Der Fehler in Fahrtrichtung ist mit Werten von einigen Dezimetern bis über einem Meter sehr hoch und wie zuvor beschrieben auf die Fahrzeugumgebung zurückzuführen.

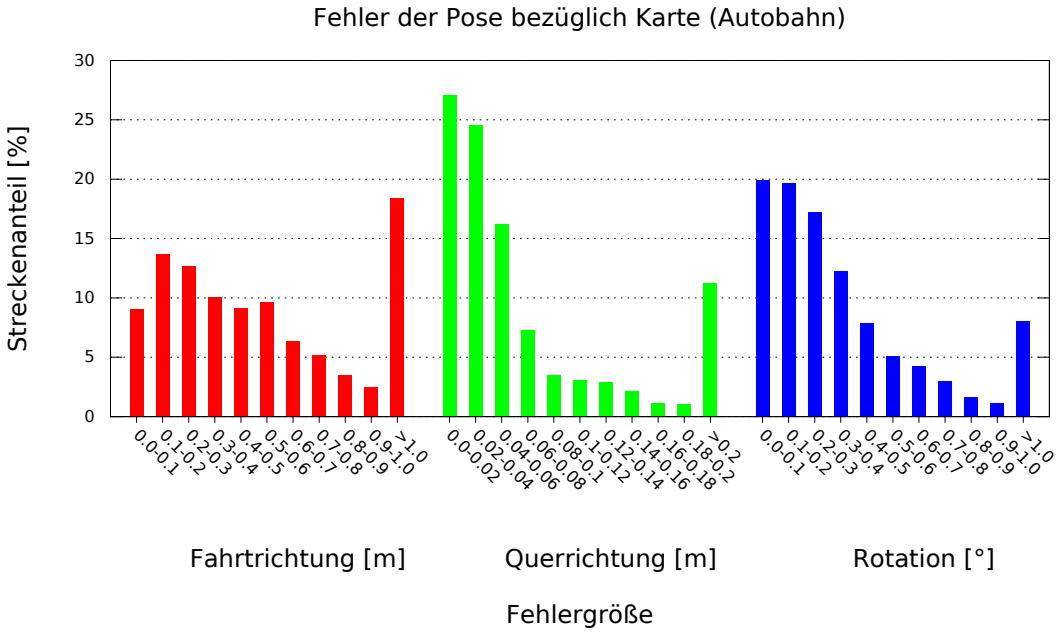


Abbildung 6.5.: Fehlerhistogramm der Pose aus der Lokalisierung ggü. Karte. Szenario „Autobahn“.

Der geringe Fehler in Querrichtung von wenigen Zentimetern in Verbindung mit dem recht geringen Rotationsfehler in der Größenordnung von zehntel Grad würde in den meisten Fällen eine präzise Zuordnung des eigenen sowie anderer Fahrzeuge zu Fahrspuren erlauben. Die geringen Fehler in Querrichtung und Rotation unterstützen die These, dass der hohe Fehler in Fahrtrichtung von der für Autobahnen typischen Umgebung verursacht ist. Die Karte im Szenario „Autobahn“ ist an einer Stelle recht stark fehlerbehaftet (vgl. Abschnitt 5.8 sowie Abbildung C.1). Abbildung C.9 zeigt die Lokalisierung an dieser Stelle als Bildsequenz und verdeutlicht die Robustheit des Partikelfilters.

Abbildung 6.6 zeigt die Fehler der Poseänderung im Vergleich zum RTK System im Szenario „Industriegebiet“. Die Fehler in Fahrt- sowie in Querrichtung betragen wenige Zentimeter und die Rotationsfehler einige hundertstel bis wenige zehntel Grad. Die Fehler sind durchweg geringer als im Autobahn Szenario, wobei jedoch die höhere Geschwindigkeit im Autobahn-Szenario zu berücksichtigen ist, die zu einer längeren zurückgelegten Strecke in den jeweils betrachteten $0.2s$ führt. Auch hier sind jedoch die Fehler in Fahrtrichtung deutlich höher als in Querrichtung. Dies lässt sich zum Einen

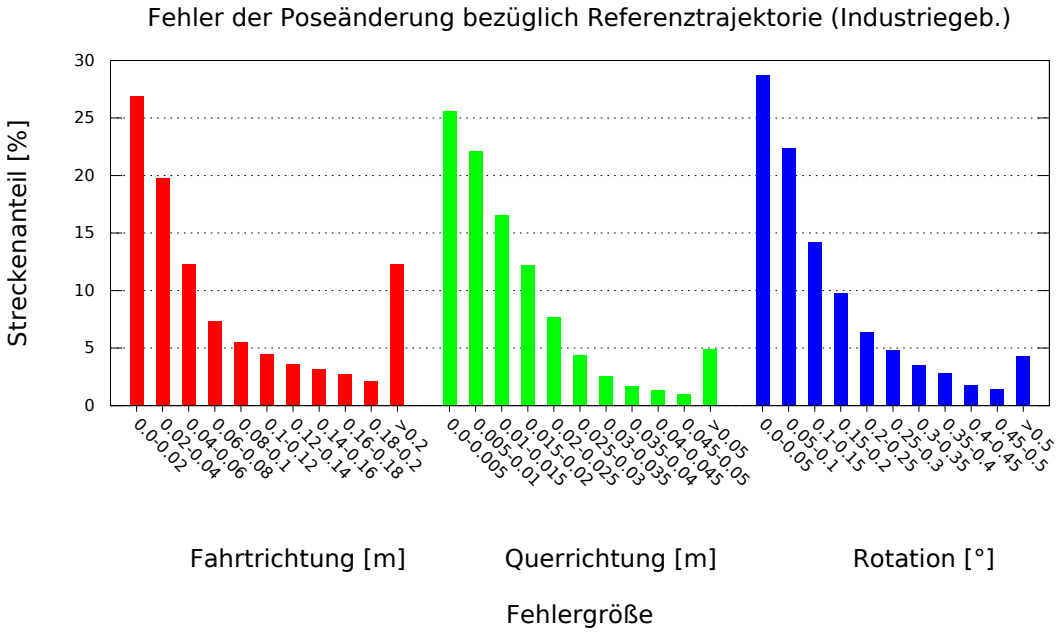


Abbildung 6.6.: Fehlerhistogramm der Poseänderung der Lokalisierung ggü. RTK System. Szenario „Industriegebiet“.

durch die eigentliche Fahrzeugbewegung erklären, die hauptsächlich in Fahrtrichtung stattfindet. Zum Anderen kann dies an einem Zeitversatz der Lokalisierung mit der Referenztrajektorie des RTK Systems liegen, da diese nicht genau zeitsynchron sind.

Abbildung 6.7 zeigt die Fehler der Posen relativ zur Karte im Szenario „Industriegebiet“. Es sind durchweg geringe Fehler zu beobachten. Querrichtungs- und Rotationsfehler sind mit den Ergebnissen im Szenario Autobahn vergleichbar. Die Fehler in Fahrtrichtung fallen im Vergleich zum Autobahn Szenario wesentlich geringer aus, da im Industriegebiet die Umgebung in Fahrtrichtung eindeutige Eigenschaften aufweist. Dass die Fehler in Fahrtrichtung gegenüber der Karte ebenfalls höher ausfallen als in Querrichtung spricht dafür, dass die beim Vergleich zum RTK System beobachteten Fehler nicht durch einen Zeitversatz hervorgerufen werden.

Abbildung 6.8 zeigt die Fehler der Poseänderung im Vergleich zum RTK System im Szenario „Landstraße“. Die Auswertung zeigt sehr geringe Fehler in Querrichtung und Rotation. Auch in Fahrtrichtung sind sehr geringe Fehler zu beobachten, wobei jedoch ein Ausschlag bei den Fehlern über $10cm$ liegt. Dieses Ergebnis beruht darauf, dass

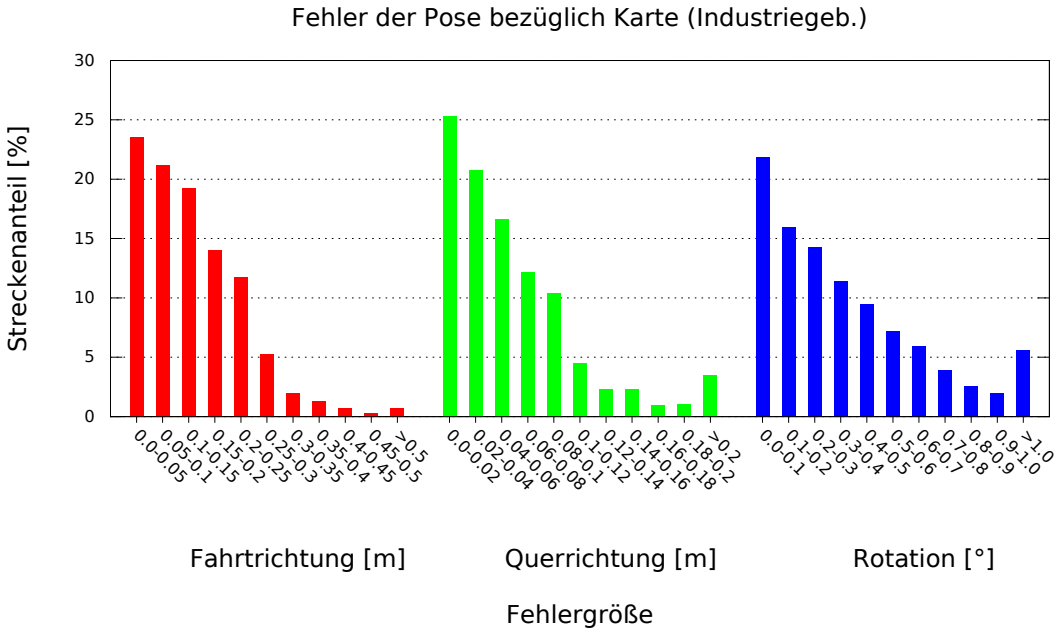


Abbildung 6.7.: Fehlerhistogramm der Pose aus der Lokalisierung ggü. Karte. Szenario „Industriegebiet“.

einige Streckenabschnitte ähnlich dem Autobahnszenario eine in Fahrtrichtung nicht eindeutige Umgebung haben.

Abbildung 6.9 zeigt die Fehler der Posen relativ zur Karte im Szenario „Landstraße“. Auch hier sind die Fehler in Querrichtung und Rotation gering, wobei sie jeweils einen Ausschlag von knapp 10% bei Fehlern von über 20cm in Querrichtung und über 1° Rotation aufweisen. Die Fehler in Fahrtrichtung fallen entsprechend der Art der Umgebung und der Fahrzeugbewegung höher aus und haben ein lokales Maximum bei einem Fehler von über 50cm. Es ist zu beachten, dass keine Korrelation der Fehler der unterschiedlichen Posekomponenten überprüft wurde. Demnach kann keine Aussage getroffen werden, ob die lokalen Maxima der unterschiedlichen Fehlerarten auf die selben Posen zurückzuführen sind oder an unterschiedlichen Stellen der Trajektorie auftreten.

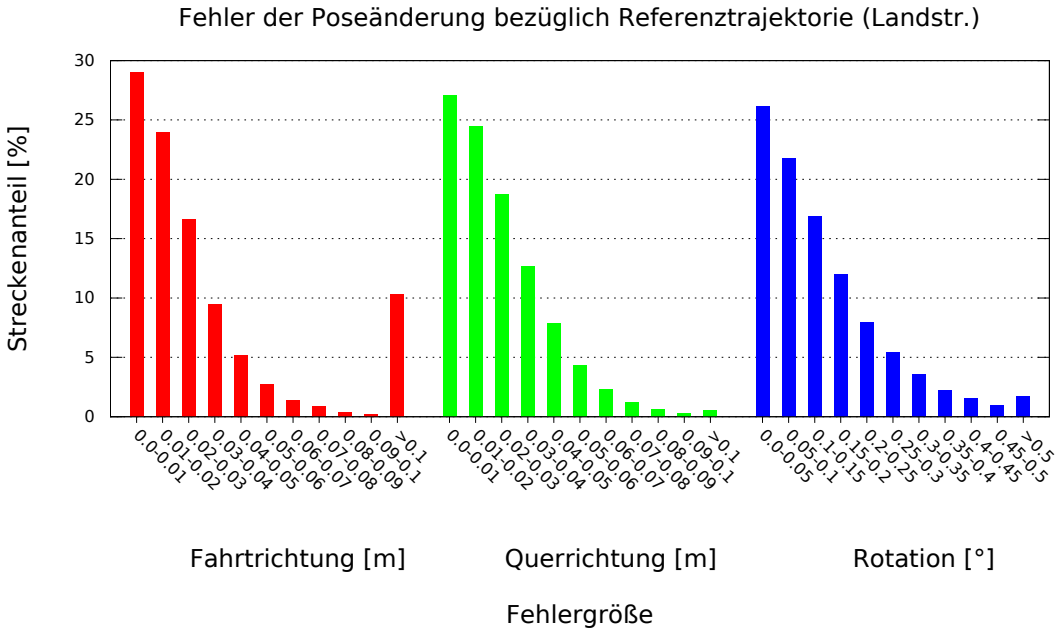


Abbildung 6.8.: Fehlerhistogramm der Poseänderung der Lokalisierung ggü. RTK System. Szenario „Landstraße“.

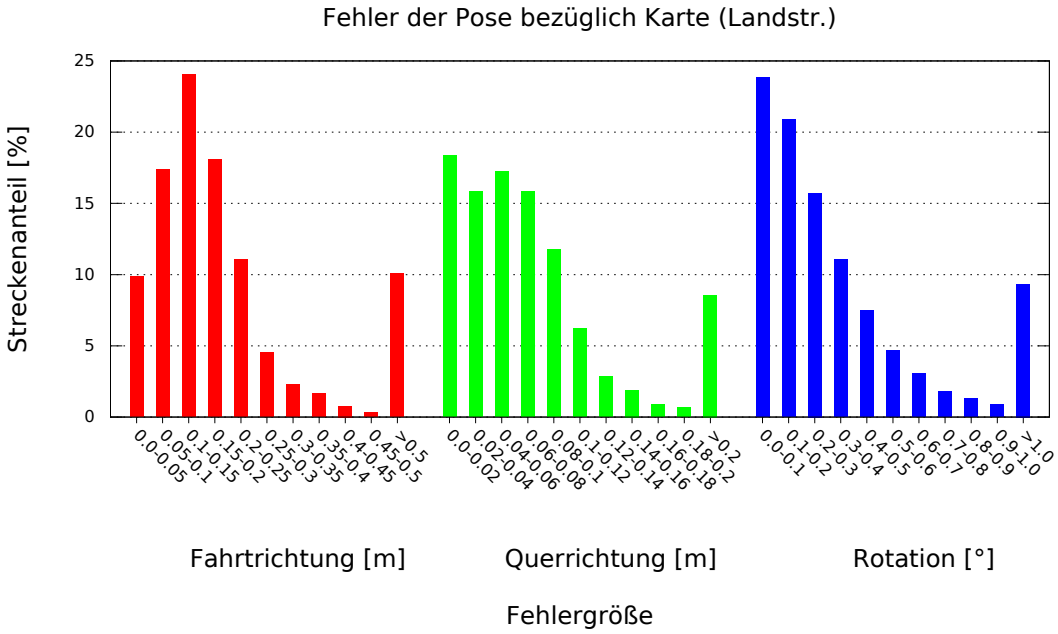


Abbildung 6.9.: Fehlerhistogramm der Pose aus der Lokalisierung ggü. Karte. Szenario „Landstraße“.

7. Fazit

In der vorliegenden Arbeit wurden zunächst die benötigten Grundlagen zur Umfeldmodellierung, Zustandsabschätzung sowie zu SLAM vermittelt. Es wurde ein Verfahren zur Erkennung von Landmarken entwickelt und umgesetzt. Basierend hierauf wurde ein Graph-basiertes full-SLAM-Verfahren umgesetzt, um anhand der Trajektorie eine Gridkarte zu erzeugen. Des Weiteren wurde ein Partikelfilter zur Lokalisierung auf solchen Gridkarte entwickelt. Dazu wurden eine Ausreißererkennung sowie eine Bewertungsfunktion für die Laserscannerdaten entwickelt. Die Detektion der Keypoints, die Kartierung sowie die Lokalisierung wurden prototypisch umgesetzt und anhand realer Messdaten in verschiedenen Szenarien evaluiert.

Die in Kapitel 1 beschriebenen Ziele – die Entwicklung und prototypische Umsetzung eines Mappingverfahrens sowie eines Lokalisierungsverfahrens – wurden demnach erreicht. Die Evaluation hat ergeben, dass die Genauigkeit des SLAM-Verfahrens sowie der Lokalisierung im Regelfall hoch ist, jedoch situationsbedingt nicht ausreicht, um bspw. die im HIGHTS Projekt angestrebte Positionsgenauigkeit von 25cm durchgehend zu erreichen. Es kann dennoch gefolgert werden, dass die vorgestellten Verfahren geeignete Kandidaten für die Ziele der beiden Forschungsprojekte darstellen und in diesem Zusammenhang als vielversprechende Ausgangspunkte für weiterführende Entwicklungen dienen können.

Zukünftige Verbesserungen im Mappingverfahren könnten eine genauere Schätzung der Landmarken (bspw. durch zusätzliche Eigenschaften wie Landmarkendurchmesser) für eine höhere Kartengenauigkeit oder die Verwendung zusätzlicher Features aus den Scandaten sein. Letzteres könnte zur Lösung der Problematik im Szenario „Stadt“ beitragen. In der Lokalisierung stellen die Fehler in Fahrtrichtung die größte Problematik dar, sodass an dieser Stelle Verbesserungsbedarf besteht. So könnte möglicherweise durch die zusätzliche Schätzung der Fahrzeugdynamik, also der Geschwindigkeit

und Gierrate, ein robusteres Verhalten erreicht werden. Da die Verfahren in dieser Arbeit prototypisch umgesetzt wurden, wurde die Laufzeit hier außer Acht gelassen. Das Partikelfilter bietet für zukünftige Arbeiten Möglichkeiten zur Verbesserung, wie z. B. eine dynamische, der jeweiligen Partikelverteilung angepasste, Partikelanzahl (KLD-Sampling [Fox02]), welche die Laufzeit reduzieren kann. Dies wäre insbesondere bei zusätzlicher Schätzung der Fahrzeugdynamik von Vorteil, da durch den größeren Zustandsraum der Rechenaufwand steigt.

Nicht zuletzt ist auch eine weitergehende und umfangreichere Evaluation der entwickelten Verfahren sinnvoll. Wichtig wäre zur Evaluation des Mappingverfahrens beispielsweise eine Betrachtung der Posen in Weltkoordinaten zusätzlich zur Form der Trajektorie. Für die Bewertung der Lokalisierung ist eine Strategie zur Bewertung der Pose bezüglich der Karte sinnvoll, welche nicht die Verwendung der selben Eingangsdaten für Mapping und Lokalisierung erfordert.

Abschließend lässt sich zusammenfassen, dass sowohl das Mappingverfahren als auch die Lokalisierung in den Szenarien „Autobahn“, „Industriegebiet“ und „Landstraße“ im Allgemeinen funktionieren und die weitere Verfolgung der Ansätze sinnvoll erscheint.

A. Linearisierungen Systemmodell und Messmodell

A.1. Linearisierung des Systemmodells

Das verwendete Bewegungsmodell ist definiert als A.1:

$$x_t = \begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{cases} \begin{pmatrix} x - \frac{v}{\omega} sin\theta + \frac{v}{\omega} sin\left(\theta + \omega\Delta t\right) \\ y + \frac{v}{\omega} cos\theta - \frac{v}{\omega} cos\left(\theta + \omega\Delta t\right) \\ \theta + \omega\Delta t \end{pmatrix} & \text{, falls } \omega \neq 0 \\ \begin{pmatrix} x + \Delta t \cdot v \cdot cos\theta \\ y + \Delta t \cdot v \cdot sin\theta \\ \theta \end{pmatrix} & \text{, sonst} \end{cases} \tag{A.1}$$

Die Linearisierung für die Kalman-Filter-Prediction in Gleichung 5.3 (Initiale Trajektorie des SLAM, Abschnitt 5.2) ist für $\omega \neq 0$ gegeben durch die Gleichungen A.2 bis A.17 und für $\omega = 0$ durch die Gleichungen A.2 und A.18 bis A.32.

$$F = \begin{pmatrix} \frac{\partial g_x}{\partial x} & \frac{\partial g_x}{\partial y} & \frac{\partial g_x}{\partial \varphi} & \frac{\partial g_x}{\partial v} & \frac{\partial g_x}{\partial \omega} \\ \frac{\partial g_y}{\partial x} & \frac{\partial g_y}{\partial y} & \frac{\partial g_y}{\partial \varphi} & \frac{\partial g_y}{\partial v} & \frac{\partial g_y}{\partial \omega} \\ \frac{\partial g_\varphi}{\partial x} & \frac{\partial g_\varphi}{\partial y} & \frac{\partial g_\varphi}{\partial \varphi} & \frac{\partial g_\varphi}{\partial v} & \frac{\partial g_\varphi}{\partial \omega} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.2}$$

$$\frac{\partial g_x}{\partial x} = 1 \tag{A.3}$$

$$\frac{\partial g_x}{\partial y} = 0 \tag{A.4}$$

$$\frac{\partial g_x}{\partial \varphi} = \frac{-v}{\omega}\cos\varphi + \frac{-v}{\omega}\cos(\varphi + \omega\Delta t) \tag{A.5}$$

$$\frac{\partial g_x}{\partial v} = \frac{-\sin\varphi}{\omega} + \frac{\sin(\varphi + \omega\Delta t)}{\omega} \tag{A.6}$$

$$\frac{\partial g_x}{\partial \omega} = \frac{v \cdot \sin\varphi}{\omega^2} + \frac{v \cdot \cos(\varphi + \omega\Delta t)\Delta t}{\omega} - \frac{v \cdot \sin(\varphi + \omega\Delta t)}{\omega^2} \tag{A.7}$$

$$\frac{\partial g_y}{\partial x} = 0 \tag{A.8}$$

$$\frac{\partial g_y}{\partial y} = 1 \tag{A.9}$$

$$\frac{\partial g_y}{\partial \varphi} = \frac{-v}{\omega}\sin\varphi + \frac{-v}{\omega}\sin(\varphi + \omega\Delta t) \tag{A.10}$$

$$\frac{\partial g_y}{\partial v} = \frac{\cos\varphi}{\omega} - \frac{\cos(\varphi + \omega\Delta t)}{\omega} \tag{A.11}$$

$$\frac{\partial g_y}{\partial \omega} = \frac{-v \cdot \cos\varphi}{\omega^2} + \frac{v \cdot \sin(\varphi + \omega\Delta t)\Delta t}{\omega} + \frac{v \cdot \cos(\varphi + \omega\Delta t)}{\omega^2} \tag{A.12}$$

$$\frac{\partial g_\varphi}{\partial x} = 0 \tag{A.13}$$

$$\frac{\partial g_\varphi}{\partial y} = 0 \tag{A.14}$$

$$\frac{\partial g_\varphi}{\partial \varphi} = 1 \tag{A.15}$$

$$\frac{\partial g_\varphi}{\partial v} = 0 \tag{A.16}$$

$$\frac{\partial g_\varphi}{\partial \omega} = \Delta t \tag{A.17}$$

$$\frac{\partial g_x}{\partial x} = 1 \tag{A.18}$$
$$\frac{\partial g_x}{\partial y} = 0 \tag{A.19}$$
$$\frac{\partial g_x}{\partial \varphi} = -\Delta t \cdot v \cdot \sin\varphi \tag{A.20}$$
$$\frac{\partial g_x}{\partial v} = \Delta t \cdot \cos\varphi \tag{A.21}$$
$$\frac{\partial g_x}{\partial \omega} = 0 \tag{A.22}$$
$$\frac{\partial g_y}{\partial x} = 0 \tag{A.23}$$
$$\frac{\partial g_y}{\partial y} = 1 \tag{A.24}$$
$$\frac{\partial g_y}{\partial \varphi} = \Delta t \cdot v \cdot \cos\varphi \tag{A.25}$$
$$\frac{\partial g_y}{\partial v} = \Delta t \cdot \sin\varphi \tag{A.26}$$
$$\frac{\partial g_y}{\partial \omega} = 0 \tag{A.27}$$
$$\frac{\partial g_\varphi}{\partial x} = 0 \tag{A.28}$$
$$\frac{\partial g_\varphi}{\partial y} = 0 \tag{A.29}$$
$$\frac{\partial g_\varphi}{\partial \varphi} = 1 \tag{A.30}$$
$$\frac{\partial g_\varphi}{\partial v} = 0 \tag{A.31}$$
$$\frac{\partial g_\varphi}{\partial \omega} = 0 \tag{A.32}$$

A.2. Linearisierung des Keypoint Messmodells

Das im GraphSLAM verwendete Keypoint-Messmodell ist definiert als A.33 (entspricht Gleichung 5.16 im Abschnitt 5.4 „Schätzung von Clustertrajektorie und -landmarken“).

$$h(\mu_i, m_l) = \begin{pmatrix} -cos(\mu_{i,\varphi}) & -sin(\mu_{i,\varphi}) & 0 & cos(\mu_{i,\varphi}) & sin(\mu_{i,\varphi}) \\ sin(\mu_{i,\varphi}) & -cos(\mu_{i,\varphi}) & 0 & -sin(\mu_{i,\varphi}) & cos(\mu_{i,\varphi}) \end{pmatrix} \cdot \begin{pmatrix} \mu_{i,x} \\ \mu_{i,y} \\ \mu_{i,\varphi} \\ m_{l,x} \\ m_{l,y} \end{pmatrix} \tag{A.33}$$

Die Linearisierung $H_{i,l}$ ist gegeben durch die Gleichungen A.34 bis A.44.

$$H_{i,l} = \begin{pmatrix} \frac{\partial h_x}{\partial \mu_{i,x}} & \frac{\partial h_x}{\partial \mu_{i,y}} & \frac{\partial h_x}{\partial \mu_{i,\varphi}} & \frac{\partial h_x}{\partial m_{l,x}} & \frac{\partial h_x}{\partial m_{l,y}} \\ \frac{\partial h_y}{\partial \mu_{i,x}} & \frac{\partial h_y}{\partial \mu_{i,y}} & \frac{\partial h_y}{\partial \mu_{i,\varphi}} & \frac{\partial h_y}{\partial m_{l,x}} & \frac{\partial h_y}{\partial m_{l,y}} \end{pmatrix} \tag{A.34}$$

$$\frac{\partial h_x}{\partial \mu_{i,x}} = -cos(\mu_{i,\varphi}) \tag{A.35}$$

$$\frac{\partial h_x}{\partial \mu_{i,y}} = -sin(\mu_{i,\varphi}) \tag{A.36}$$

$$\frac{\partial h_x}{\partial \mu_{i,\varphi}} = sin(\mu_{i,\varphi}) \cdot \mu_{i,x} - cos(\mu_{i,\varphi}) \cdot \mu_{i,y} - sin(\mu_{i,\varphi}) \cdot m_{l,x} + cos(\mu_{i,\varphi}) \cdot m_{l,y} \tag{A.37}$$

$$\frac{\partial h_x}{\partial m_{l,x}} = cos(\mu_{i,\varphi}) \tag{A.38}$$

$$\frac{\partial h_x}{\partial m_{l,y}} = sin(\mu_{i,\varphi}) \tag{A.39}$$

$$\frac{\partial h_y}{\partial \mu_{i,x}} = sin(\mu_{i,\varphi}) \tag{A.40}$$

$$\frac{\partial h_y}{\partial \mu_{i,y}} = -cos(\mu_{i,\varphi}) \tag{A.41}$$

$$\frac{\partial h_y}{\partial \mu_{i,\varphi}} = cos(\mu_{i,\varphi}) \cdot \mu_{i,x} + sin(\mu_{i,\varphi}) \cdot \mu_{i,y} - cos(\mu_{i,\varphi}) \cdot m_{l,x} - sin(\mu_{i,\varphi}) \cdot m_{l,y} \tag{A.42}$$

$$\frac{\partial h_y}{\partial m_{l,x}} = -sin(\mu_{i,\varphi}) \tag{A.43}$$

$$\frac{\partial h_y}{\partial m_{l,y}} = cos(\mu_{i,\varphi}) \tag{A.44}$$

B. Teststrecken

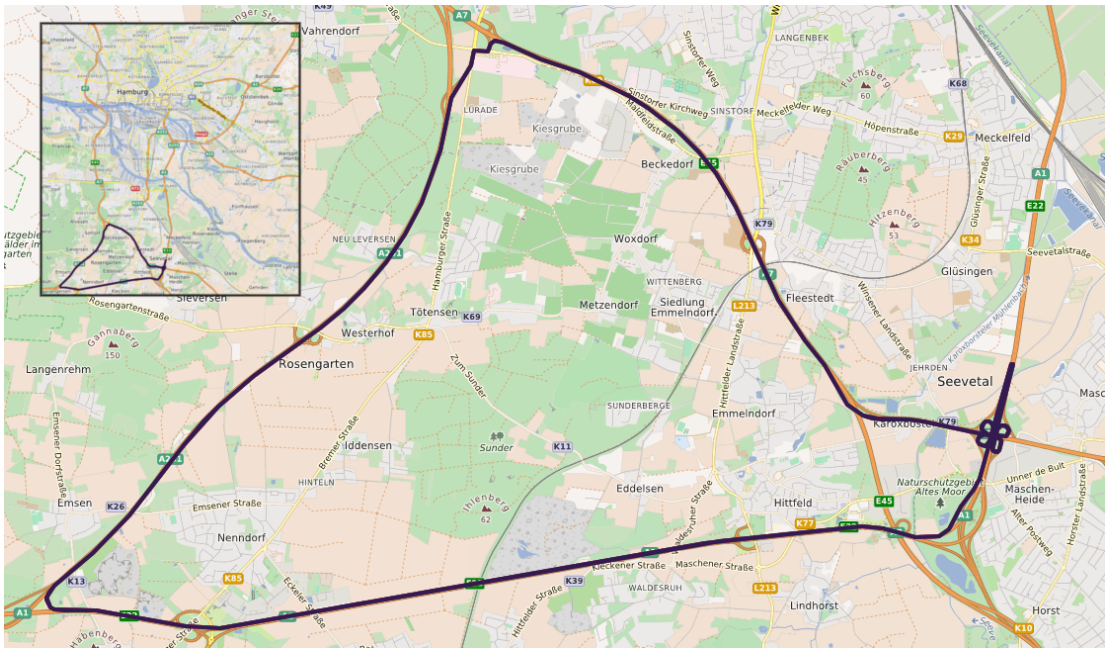


Abbildung B.1.: Streckenverlauf des Szenarios „Autobahn“. Bildquelle: [ors]

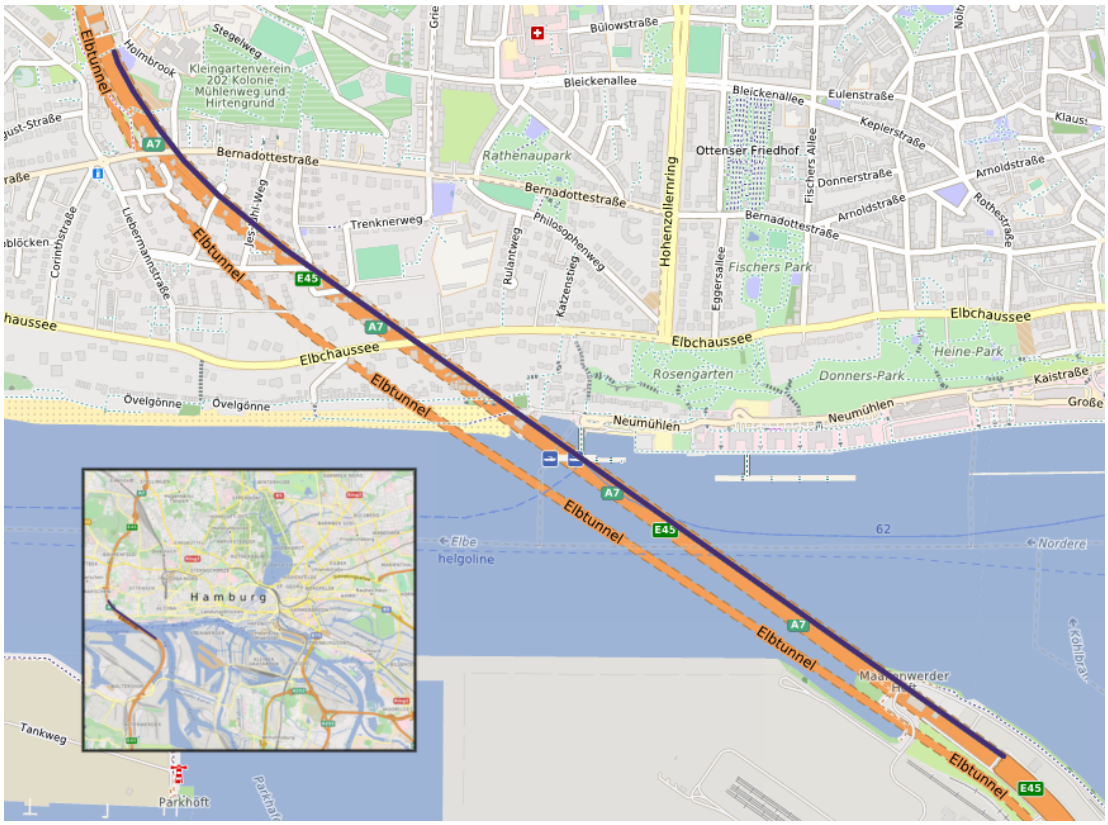


Abbildung B.2.: Streckenverlauf des Szenarios „Tunnel“. Bildquelle: [ors]

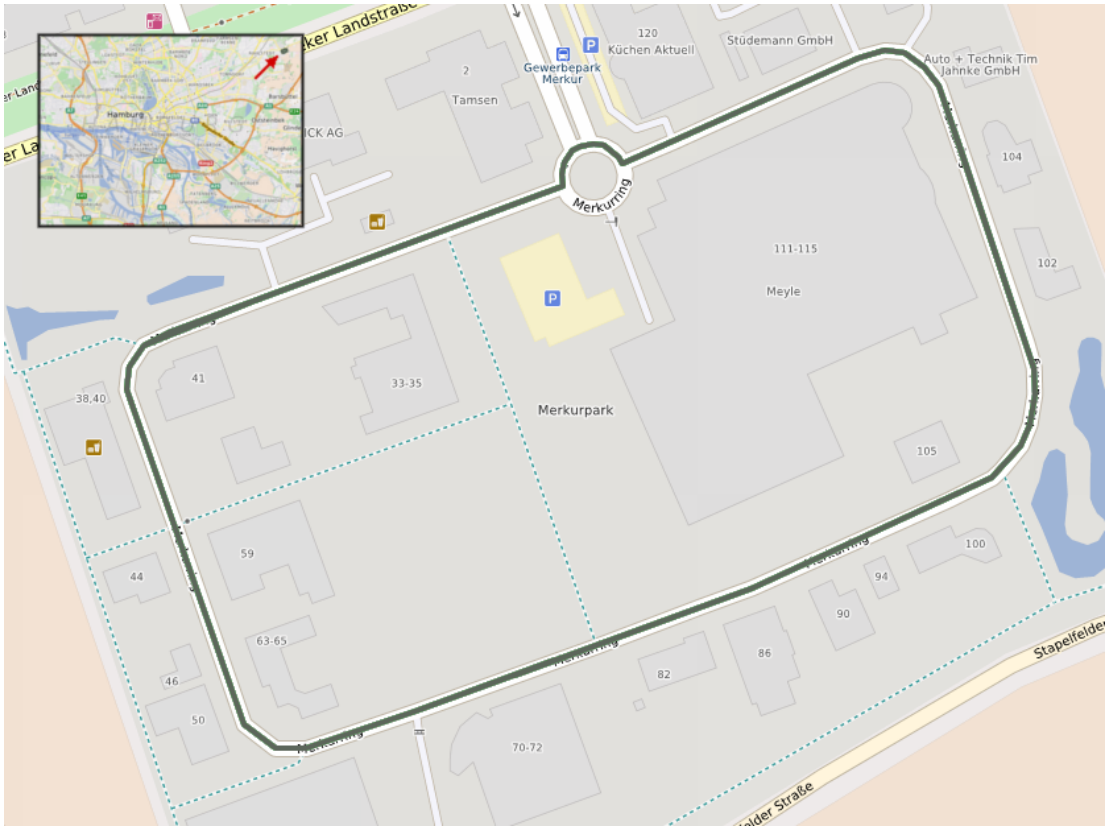


Abbildung B.3.: Streckenverlauf des Szenarios „Industriegebiet“. Bildquelle: [ors]

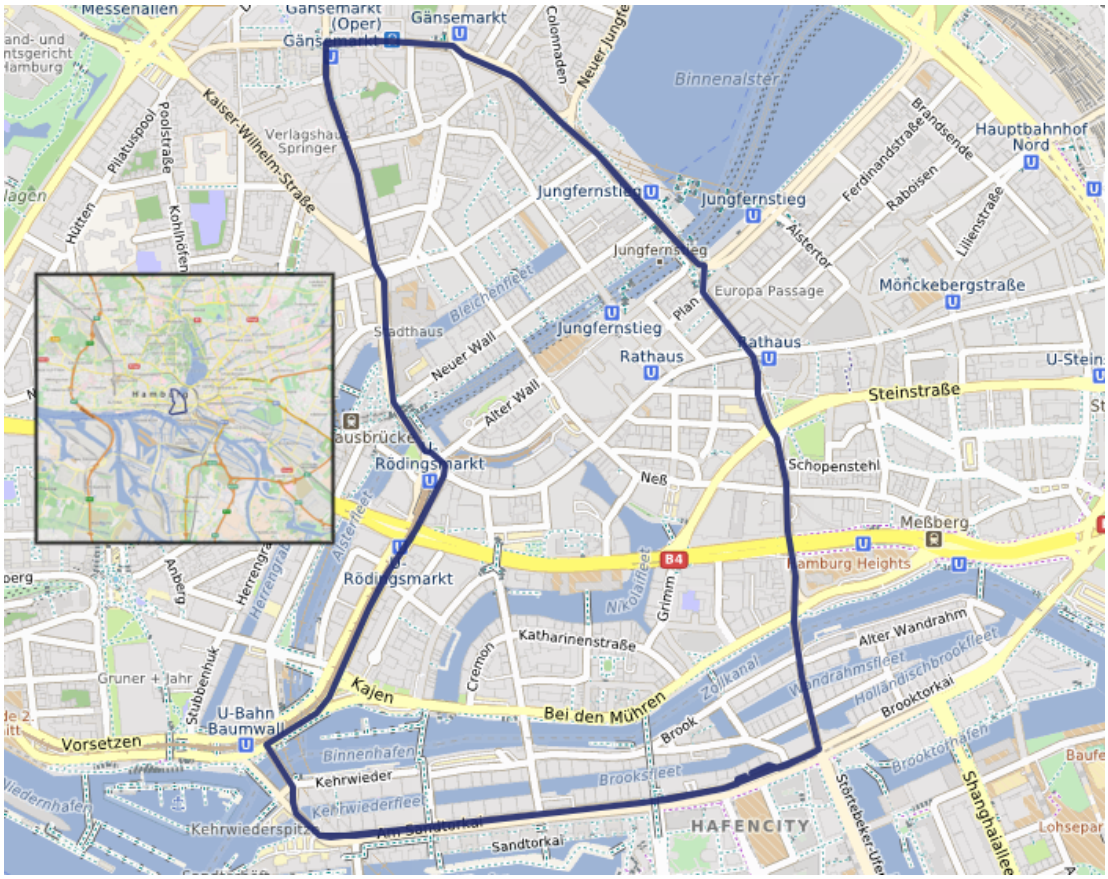


Abbildung B.4.: Streckenverlauf des Szenarios „Stadt“. Bildquelle: [ors]

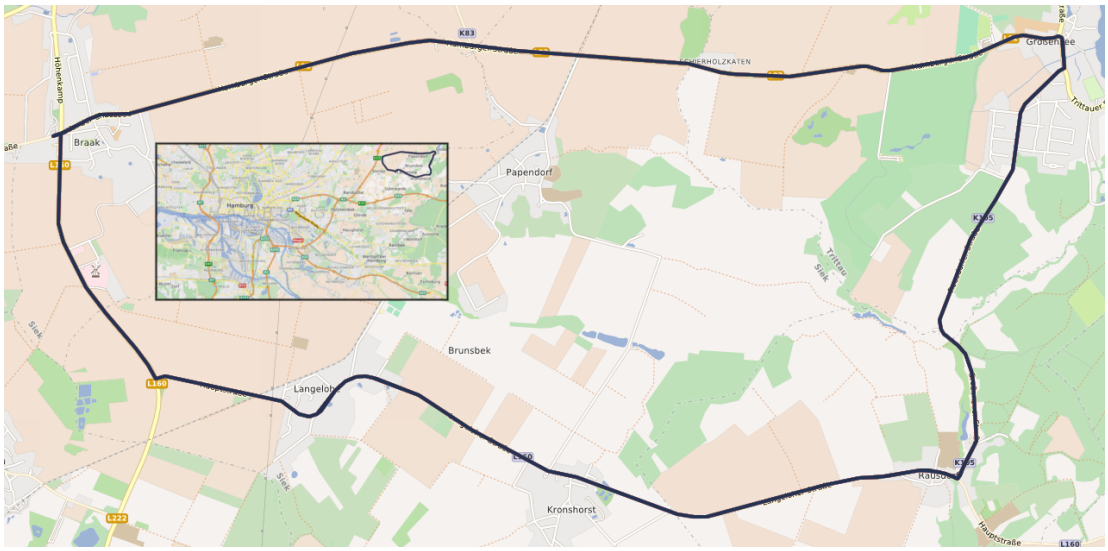


Abbildung B.5.: Streckenverlauf des Szenarios „Landstraße“. Bildquelle: [ors]

C. Evaluationsergebnisse

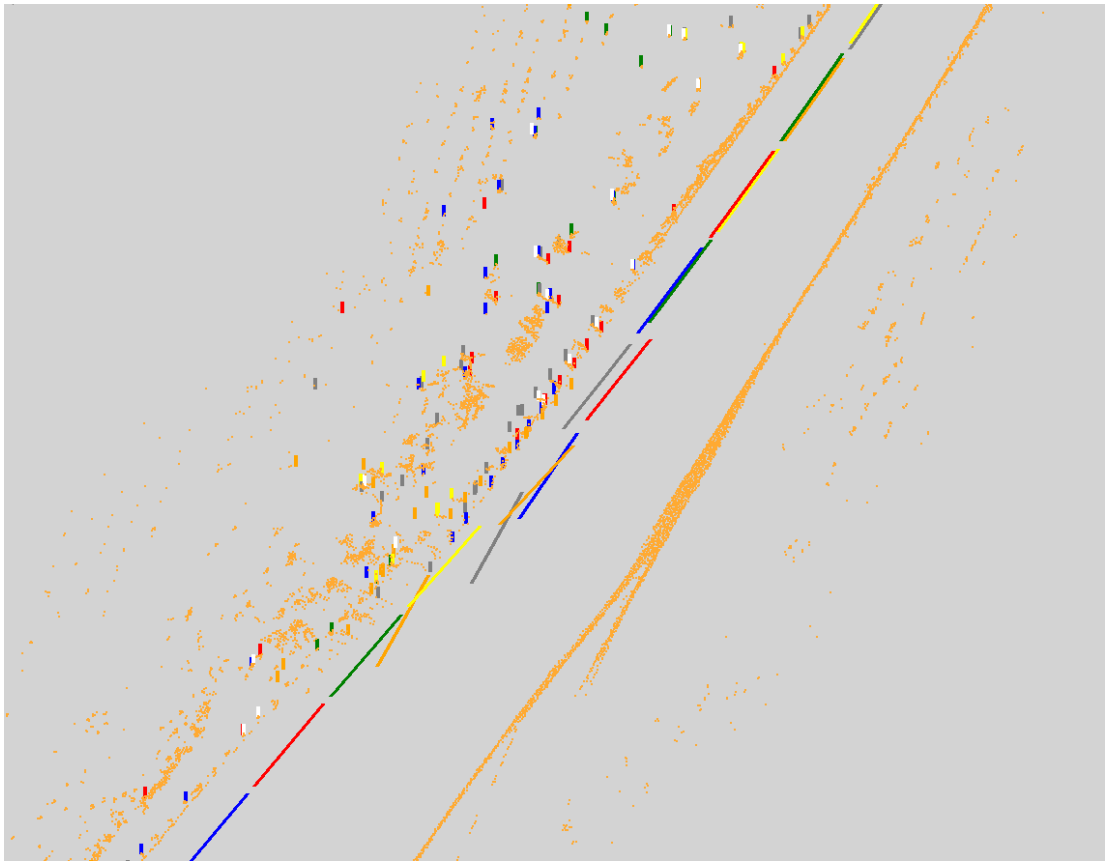


Abbildung C.1.: Fehlerhafte Loop Erkennung im Szenario „Autobahn“. Die einzelnen Cluster des GraphSLAM-Verfahrens sowie die zugehörigen Landmarken sind farblich gekennzeichnet. Farbige Landmarken, die einander zugeordnet wurden und zur Schätzung der Cluster-übergreifenden Trajektorie dienen, sind weiss. Die Scanpunkte der Laserscanner sind orange dargestellt.

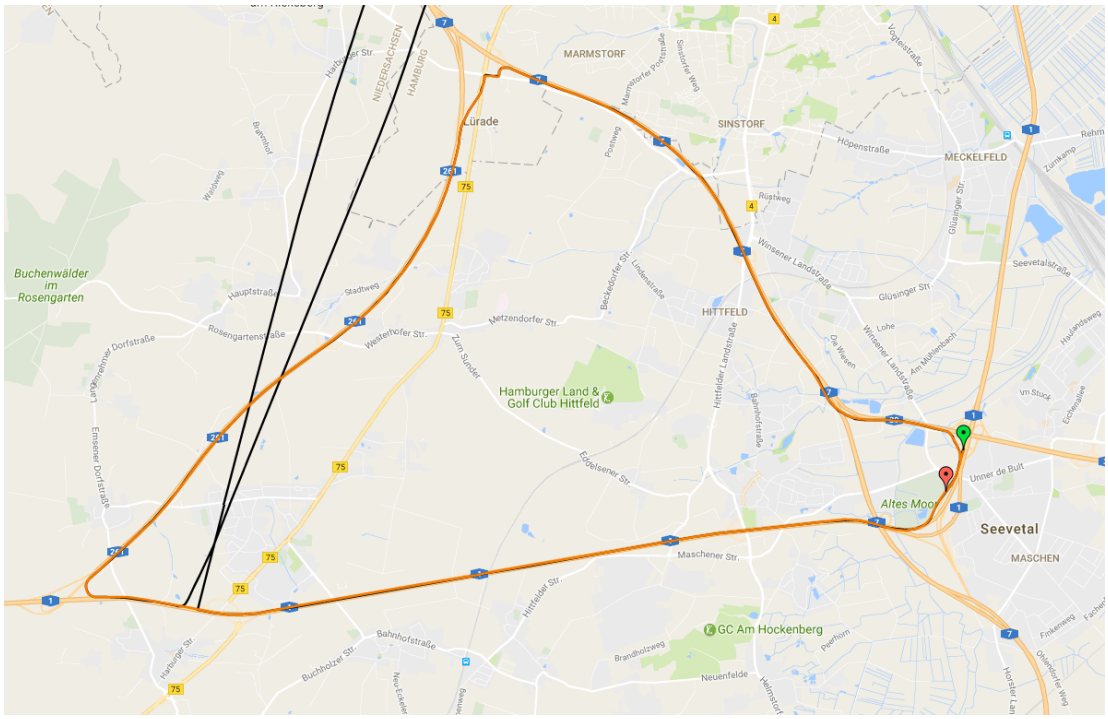


Abbildung C.2.: Ergebnistrajektorie (orange) und Referenztrajektorie (schwarz) im Szenario „Autobahn“. Bildquelle: [Goo]

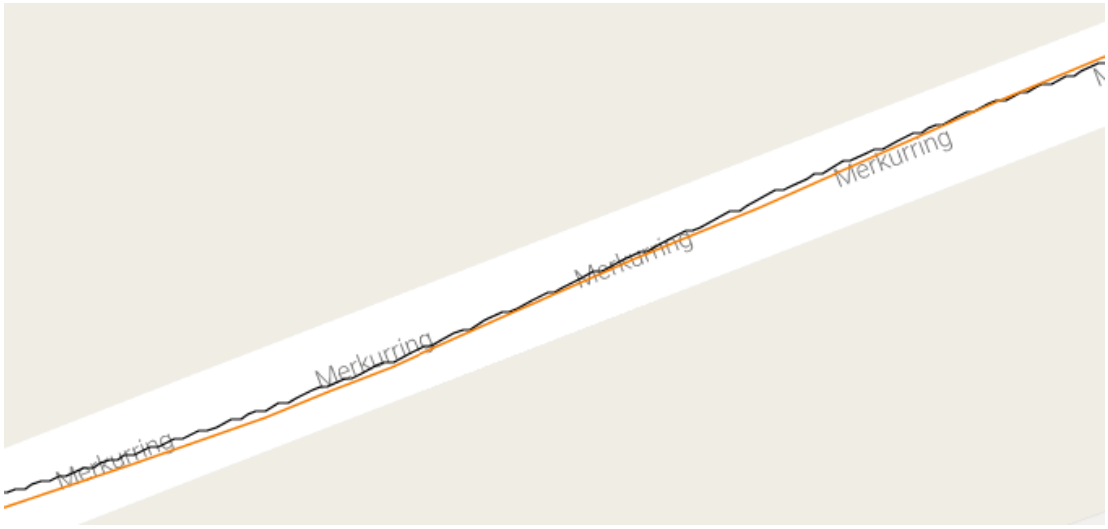


Abbildung C.3.: Nahaufnahme der Ergebnistrajektorie (orange) und der Referenztrajektorie (schwarz) im Szenario „Industriegebiet“. Bildquelle: [Goo]

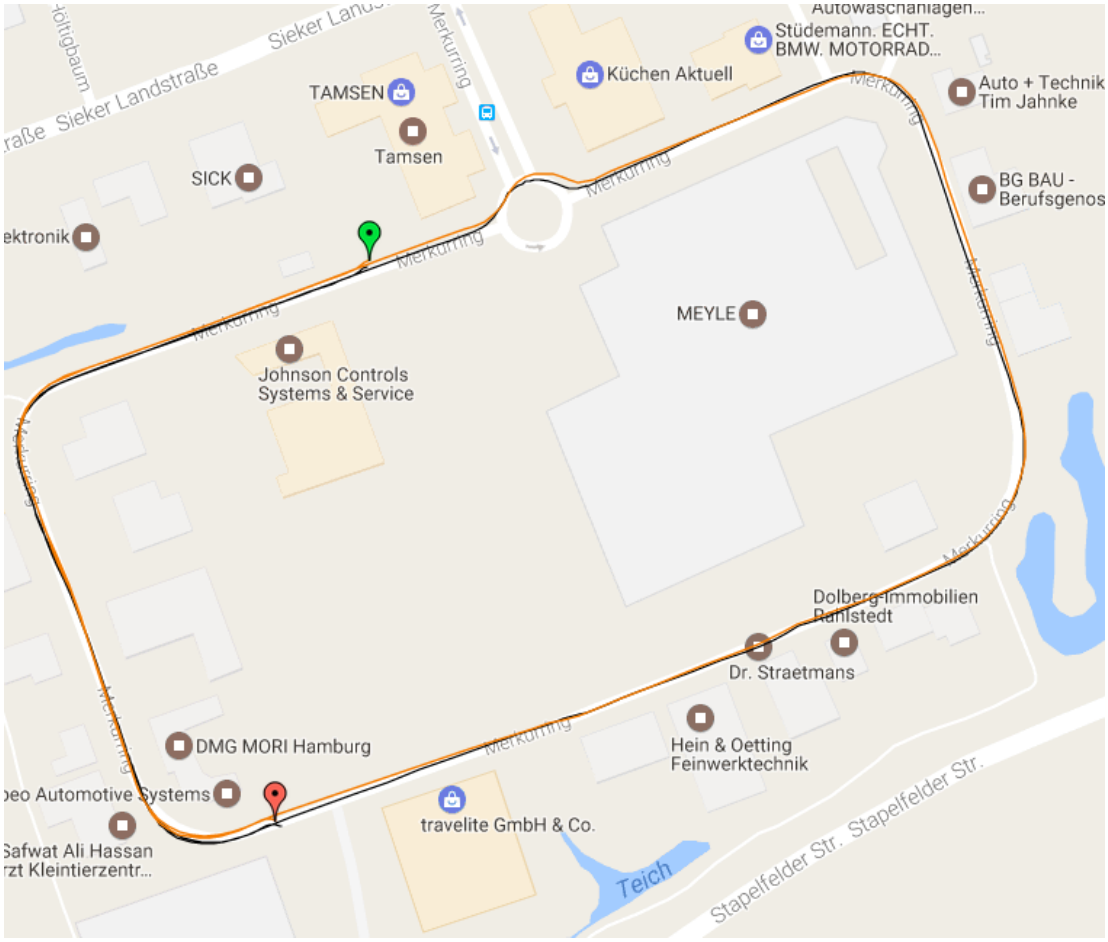


Abbildung C.4.: Ergebnistrajektorie (orange) und Referenztrajektorie (schwarz) im Szenario „Industriegebiet“. Bildquelle: [Goo]

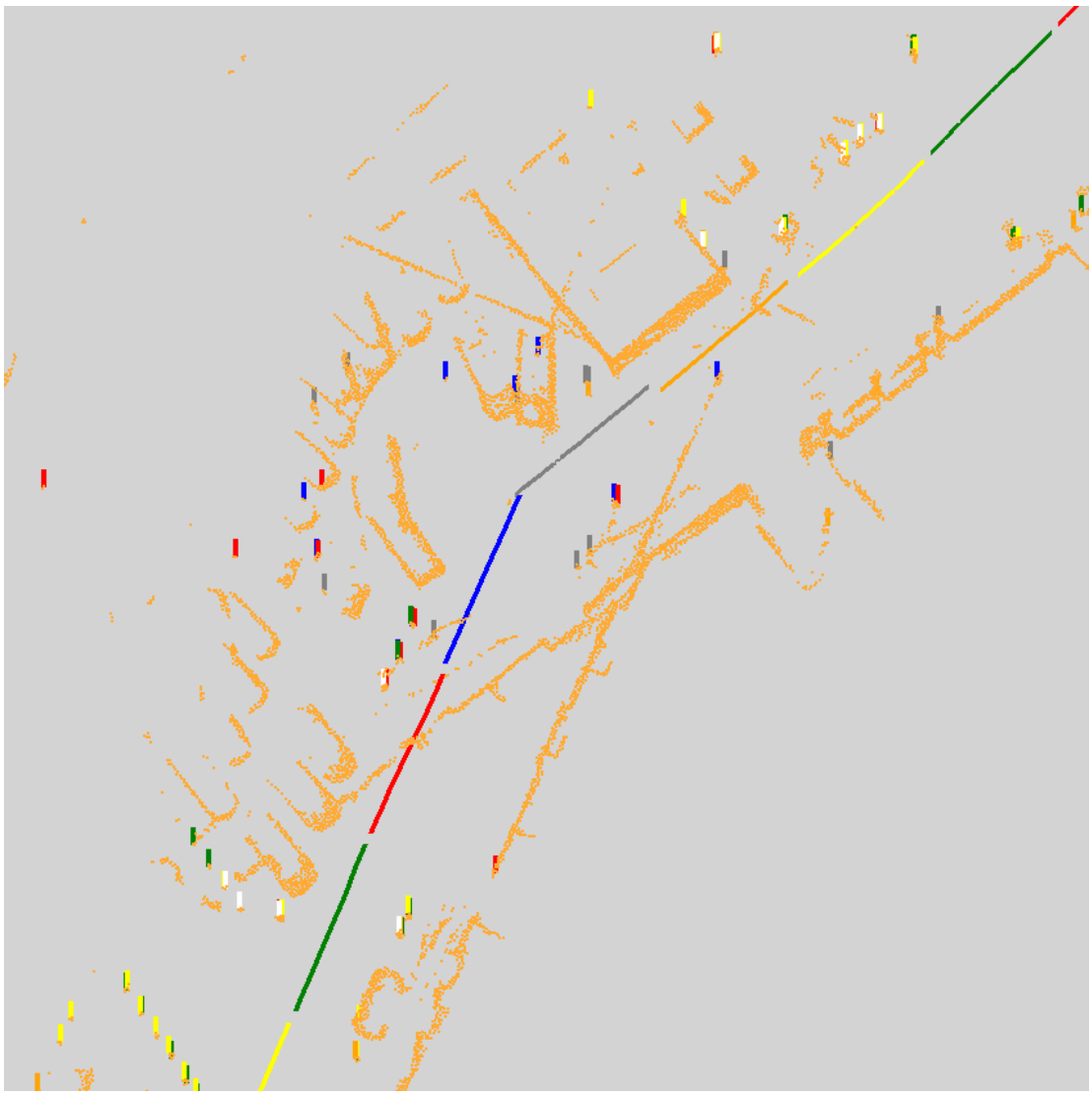


Abbildung C.5.: Erster fehlerhafter Trajektorienabschnitt im Szenario „Stadt“. Die einzelnen Cluster des GraphSLAM-Verfahrens sowie die zugehörigen Landmarken sind farblich gekennzeichnet. Farbige Landmarken, die einander zugeordnet wurden und zur Schätzung der Clusterübergreifenden Trajektorie dienen, sind weiss. Die Scanpunkte der Laserscanner sind orange dargestellt.

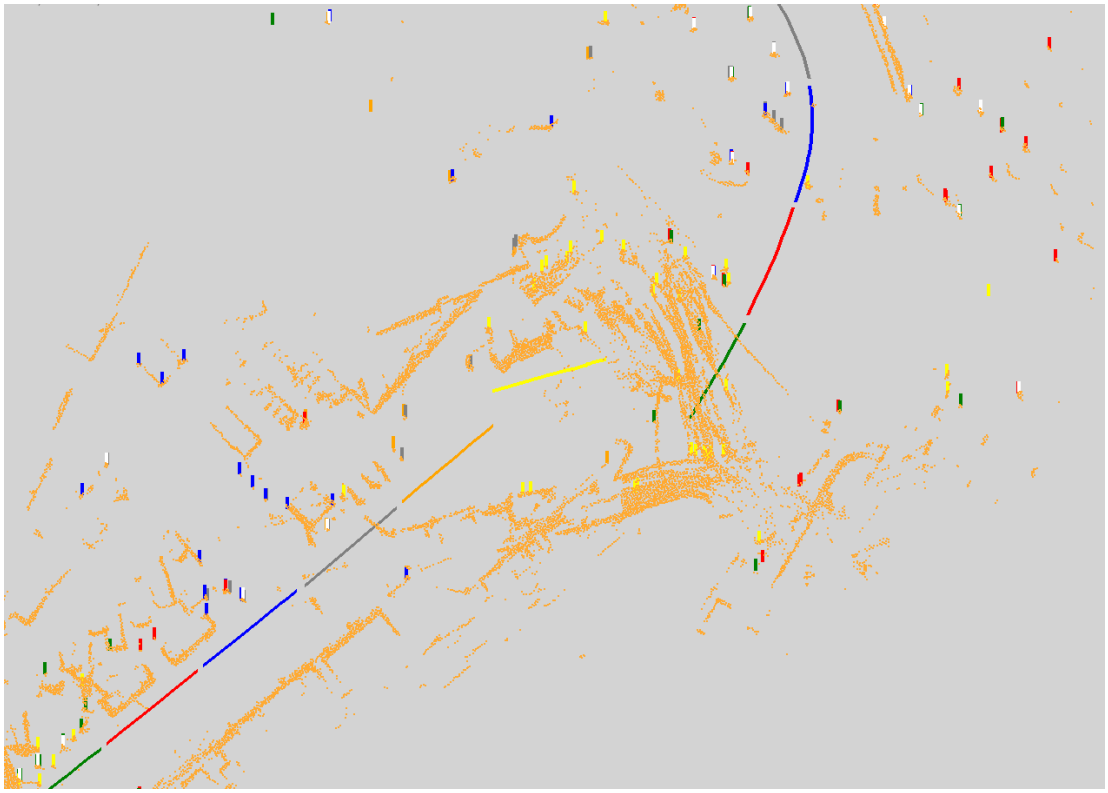


Abbildung C.6.: Zweiter fehlerhafter Trajektorienabschnitt im Szenario „Stadt“. Die einzelnen Cluster des GraphSLAM-Verfahrens sowie die zugehörigen Landmarken sind farblich gekennzeichnet. Farbige Landmarken, die einander zugeordnet wurden und zur Schätzung der Clusterübergreifenden Trajektorie dienen, sind weiss. Die Scanpunkte der Laserscanner sind orange dargestellt.

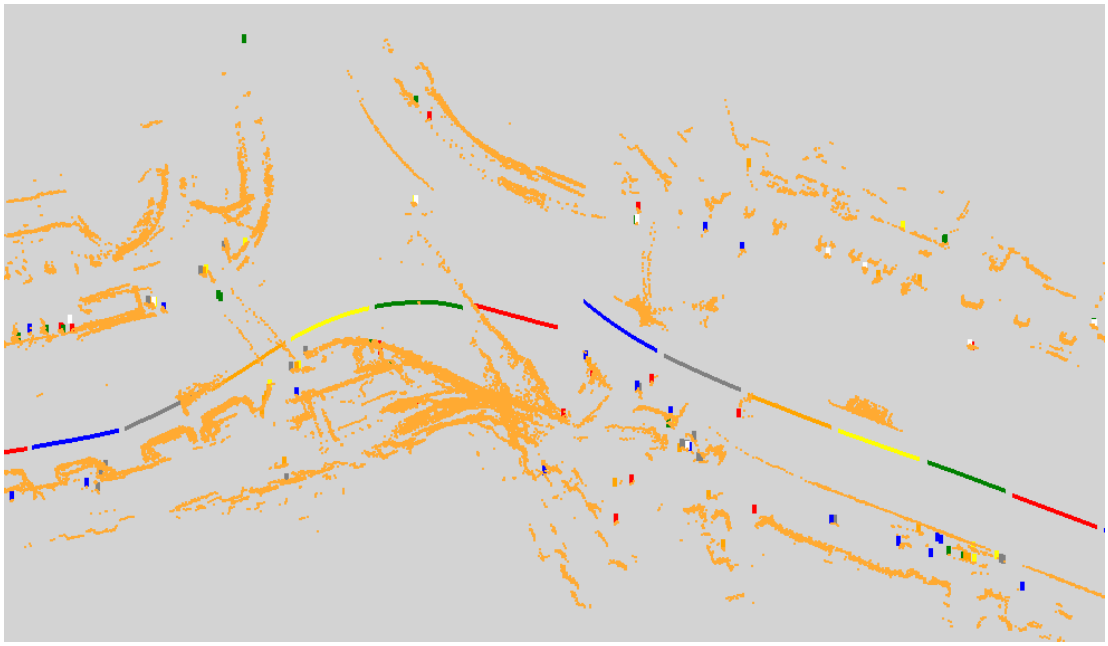


Abbildung C.7.: Dritter fehlerhafter Trajektorienabschnitt im Szenario „Stadt“. Die einzelnen Cluster des GraphSLAM-Verfahrens sowie die zugehörigen Landmarken sind farblich gekennzeichnet. Farbige Landmarken, die einander zugeordnet wurden und zur Schätzung der Clusterübergreifenden Trajektorie dienen, sind weiss. Die Scanpunkte der Laserscanner sind orange dargestellt.

Abbildung C.8.: Nahaufnahme der Ergebnistrajektorie (orange) und der Referenztrajektorie (schwarz) im Szenario „Landstraße“. Bildquelle: [Goo]

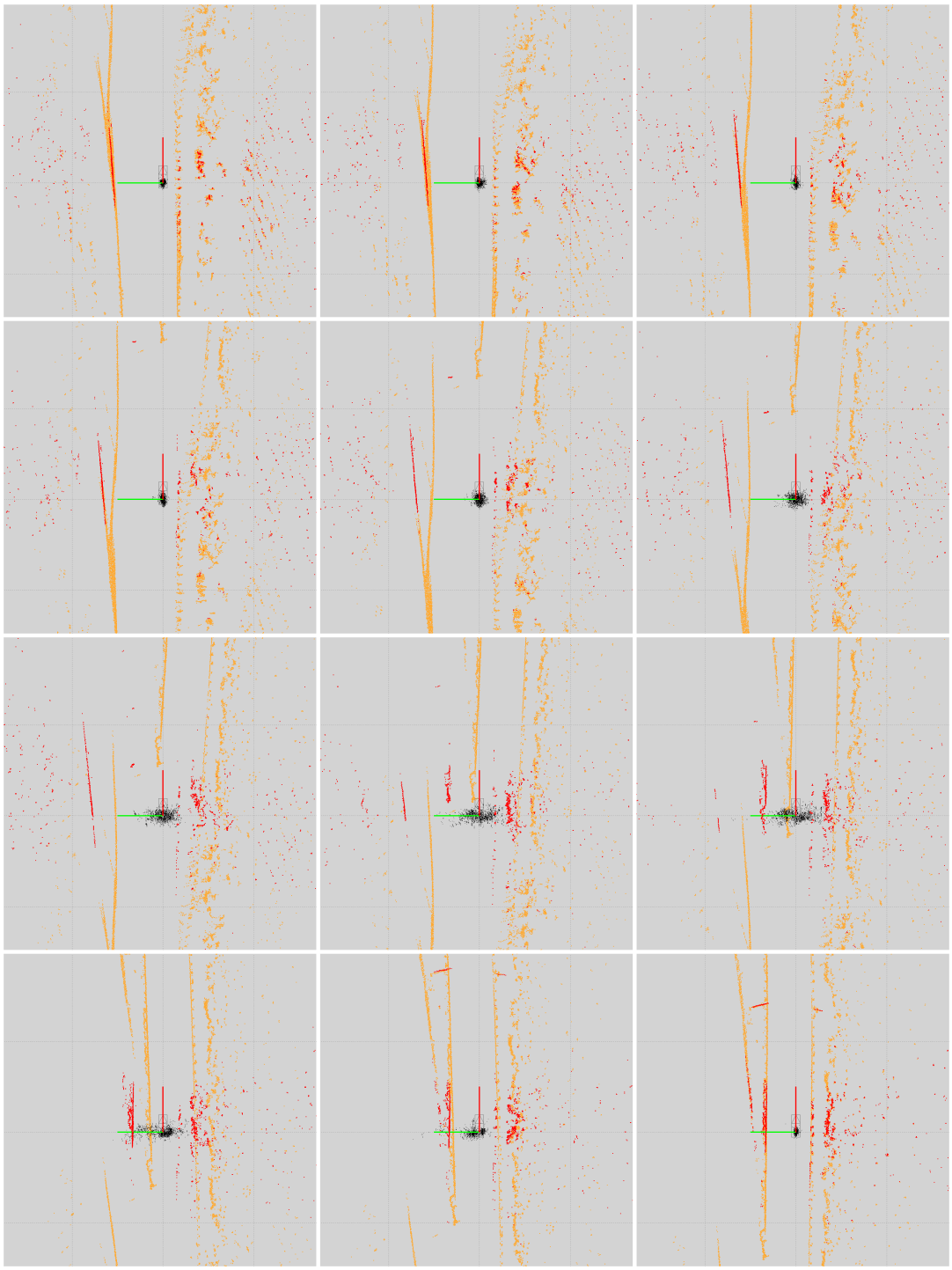


Abbildung C.9.: Von links nach rechts, von oben nach unten: Bildsequenz der Lokalisierung auf dem fehlerhaften Kartenabschnitt im Szenario „Autobahn“. Die Umgebung ist orange, die Scanpunkte rot und die Partikel des Partikelfilter schwarz dargestellt.

D. Inhalt der beigelegten CD

Die beigelegte CD enthält folgenden Inhalt:

/00_thesis Die vorliegende Arbeit im PDF Format

/01_video Beispielvideos der Lokalisierungsergebnisse

/02_eval Die Evaluationsergebnisse im CSV Format sowie als Graphen im PDF Format

- **./00_keypoints** Evaluation der Keypoint Detektion
- **./01_mapping** Evaluation der Kartierung
- **./02_localization** Evaluation der Lokalisierung

Abbildungsverzeichnis

Listings

Tabellenverzeichnis

Literaturverzeichnis

[AV98] Kai Arras and Sjur Vestli. Hybrid, High-Precision Localisation for the Mail Distributing Mobile Robot System MOPS. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA'98)*, volume 4, pages 3129–3134. IEEE, 1998. URL: http://ieeexplore.ieee.org/document/680906/, doi:10.1109/ROBOT.1998.680906.

[BDW06] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (SLAM): Part II. *IEEE Robotics and Automation Magazine*, 13(3):108–117, 2006. doi:10.1109/MRA.2006.1678144.

[BH08] Wolfram Burgard and Martial Hebert. World Modeling. In *Springer Handbook of Robotics*, pages 853–869. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. doi:10.1007/978-3-540-30301-5_37.

[Bha46] A. Bhattacharyya. On a Measure of Divergence between Two Multinomial Populations. *The Indian Journal of Statistics*, 7, 1946.

[Bla47] David Blackwell. Conditional Expectation and Unbiased Sequential Estimation. *The Annals of Mathematical Statistics*, 18(1):105–110, mar 1947. URL: http://projecteuclid.org/euclid.aoms/1177730497, doi:10.1214/aoms/1177730497.

[BM92] P.J. Besl and H.D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. URL: http://ieeexplore.ieee.org/document/121791/, doi:10.1109/34.121791.

[BTV06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3951 LNCS:404–417, 2006. doi:10.1007/11744023_32.

[CM92] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing*, 10(3):145–155, 1992. doi:10.1016/0262-8856(92)90066-C.

[Cox91] I. J. Cox. An Experiment in Guidance and Navigation of an Autonomous Mobile Robot, 1991. doi:10.1109/70.75902.

[DdFMR00] Arnaud Doucet, Nando de Freitas, Kevin Murphy, and Stuart Russell. Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks. *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI2000)*, 2000. arXiv:1301.3853.

[DK05] Frank Dellaert and Michael Kaess. Square Root SAM. *Journal of Robotics Research*, 2005.

[DW88] H.F. Durrant-Whyte. Uncertain geometry in robotics. *IEEE Journal on Robotics and Automation*, 4(1):23–31, 1988. URL: http://ieeexplore.ieee.org/document/768/, doi:10.1109/56.768.

[DWB06] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping (SLAM): part I The Essential Algorithms. *Robotics & Automation Magazine*, 2:99–110, 2006. doi:10.1109/MRA.2006.1638022.

[FH01] U Frese and G Hirzinger. Simultaneous Localization and Mapping -A Discussion. *Proceedings of the IJCAI Workshop on Reasoning with Uncertainty in Robotics*, pages 17—-26, 2001.

[Fox02] Dieter Fox. KLD-sampling: Adaptive particle filters. *Advances in Neural Information Processing Systems 14*, 14(1):713–720, 2002. URL: http://www-2.cs.cmu.edu/Groups/NIPS/NIPS2001/papers/psgz/AA62.ps.gz{%}5Cnhttps://papers.nips.cc/paper/

1998-kld-sampling-adaptive-particle-filters.pdf, doi:10.1.1.21.5786.

[GAA+12] Tom M. Gasser, Clemens Arzt, Mihiar Ayoubi, Arne Bartels, Lutz Bürkle, Jana Eier, Frank Flemisch, Dirk Häcker, Tobias Hesse, Werner Huber, Christine Lotz, Markus Maurer, Simone Ruth-Schumacher, Jürgen Schwarz, and Wolfgang Vogt. *Rechtsfolgen zunehmender Fahrzeugautomatisierung*. 2012. URL: http://bast.opus.hbz-nrw.de/volltexte/2012/587/pdf/F83.pdf.

[Goo] Google Inc. Google Maps. URL: http://maps.google.de.

[HIG15] HIGHTS. HIGHTS – High Precision Positioning for Cooperative-ITS, 2015. URL: http://www.hights.eu/.

[Ibe16] Ibeo Automotive Systems GmbH. Project HARZ, 2016. URL: https://projectharz.ibeo-as.com/.

[JKK16] Ruben Jungnickel, Michael Köhler, and Franz Korf. Efficient automotive grid maps using a sensor ray based refinement process. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 668–675. IEEE, jun 2016. URL: http://ieeexplore.ieee.org/document/7535459/, doi:10.1109/IVS.2016.7535459.

[Kal60] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35, 1960. doi:10.1115/1.3662552.

[KB88] Benjamin J. Kuipers and Yung-Tai Byun. A Robust, Qualitative Method for Robot Spatial Learning. *Proceedings of the 7th National Conference on Artificial Intelligence (AAAI)*, 1988.

[KKG+04] Gerhard K. Kraetzschmar, Gerhard K. Kraetzschmar, Guillem Pagès Gassull, Klaus Uhl, Guillem Pagès, and Gassull Klaus Uhl. Probabilistic Quadtrees for Variable-Resolution Mapping of Large Environments. *EDS.), PROCEEDINGS OF THE 5TH IFAC/EURON*, 2004. URL:

http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.59.338.

[KRC16] Fabjan Kallasi, Dario Lodi Rizzini, and Stefano Caselli. Fast Keypoint Features From Laser Scanner for Robot Localization and Mapping. *IEEE Robotics and Automation Letters*, 1(1):176–183, 2016. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7378896, doi:10.1109/LRA.2016.2517210.

[LM97] F. Lu and E. Milios. Globally Consistent Range Scan Alignment for Environment Mapping. *Autonomous Robots*, 4(4):333–349, 1997. URL: http://link.springer.com/10.1023/A:1008854305733, doi:10.1023/A:1008854305733.

[LO10] Yangming Li and Edwin B. Olson. A general purpose feature extractor for Light Detection and Ranging data. *Sensors (Switzerland)*, 10(11):10356–10375, 2010. doi:10.3390/s101110356.

[Low99] David G. Lowe. Object recognition from local scale-invariant features. *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 2:1150–1157, 1999. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=790410, arXiv:0112017, doi:10.1109/ICCV.1999.790410.

[LSRR14] Wenjie Lu, Emmanuel Seignez, F. Sergio a. Rodriguez, and Roger Reynaud. Lane marking based vehicle localization using particle filter and multi-kernel estimation. *2014 13th International Conference on Control Automation Robotics & Vision (ICARCV)*, 2014(December):601–606, 2014. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7064372, doi:10.1109/ICARCV.2014.7064372.

[MAF16] Martin Schauer, Andis Rudevics, and Fabian Kirsch. Precise Sensor-Based Vehicle Positioning. Technical report, 2016.

[ME85] H. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 116–121. Institute of Electrical and Electronics Engineers, 1985. URL: http://ieeexplore.ieee.org/document/1087316/, doi:10.1109/ROBOT.1985.1087316.

[MTKW02] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. In *AAAI-02 Proceedings*, pages 593–598. American Association for Artificial Intelligence, 2002.

[Nat00] National Imagery and Mapping Agency (NIMA). Department of Defense World Geodetic System 1984. 2000. URL: http://earth-info.nga.mil/GandG/publications/tr8350.2/wgs84fin.pdf.

[Ols09] E Olson. Real-Time Correlative Scan Matching. *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2009.

[ors] OpenRouteService. URL: http://openrouteservice.org.

[Rao45] Radhakrishna Rao. Information and the accuracy attainable in the estimation of statistical parameters. *Bull. Calcutta Math. Soc.*, 37, 1945.

[Rub88] D. B. Rubin. Using the SIR Algorithm to Simulate Posterior Distributions. In *Bayesian Statistics 3*, pages 395–402. Oxford University Press, 1988.

[SBH+15] Martin J. Schuster, Christoph Brand, Heiko Hirschmuller, Michael Suppa, and Michael Beetz. Multi-robot 6D graph SLAM connecting decoupled local reference filters. *IEEE International Conference on Intelligent Robots and Systems*, 2015-Decem(Iros):5093–5100, 2015. doi:10.1109/IROS.2015.7354094.

[SC86] R. C. Smith and P. Cheeseman. On the Representation and Estimation of Spatial Uncertainty. *The International Journal of Robotics Research*, 5(4):56–68, dec 1986. URL: http://ijr.sagepub.com/cgi/doi/10.1177/027836498600500404, doi:10.1177/027836498600500404.

[SG92] A. F. M. Smith and A. E. Gelfand. Bayesian Statistics without Tears: A Sampling-Resampling Perspective. *The American Statistician*, 46(2):84, may 1992. URL: http://www.jstor.org/stable/2684170?origin=crossref, doi:10.2307/2684170.

[Sha76] Glenn Shafer. *A mathematical theory of evidence*. Princeton University Press, 1976.

[SNS11] Roland Siegwart, Illah R. Nourbakhsh, and Davide Scaramuzza. Introduction to Autonomous Mobile Robots. mar 2011. URL: http://dl.acm.org/citation.cfm?id=1971970.

[SSC90] Randall Smith, Matthew Self, and Peter Cheeseman. Estimating Uncertain Spatial Relationships in Robotics. In *Autonomous Robot Vehicles*, pages 167–193. Springer New York, New York, NY, 1990. URL: http://link.springer.com/10.1007/978-1-4613-8997-2{_}14, doi:10.1007/978-1-4613-8997-2_14.

[TA10] Gian Diego Tipaldi and Kai O Arras. FLIRT – Interest Regions for 2D Range Data. *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3616–3622, 2010. doi:10.1109/ROBOT.2010.5509864.

[TBF05] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005. doi:10.1162/artl.2008.14.2.227.

[Thr12] S. Thrun. Artificial Intelligence for Robotics (Online Kurs), 2012. URL: https://www.udacity.com/course/artificial-intelligence-for-robotics--cs373.

[TK91] Carlo Tomasi and Takeo Kanade. Detection and Tracking of Point Features. *INTERNATIONAL JOURNAL OF COMPUTER VISION*, 9:137—-154, 1991. URL: http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.45.5770.

[TL08] Sebastian Thrun and John J Leonard. Simultaneous Localization and Mapping. *Mobile and Distributed Robotics*, pages 871–889, 2008. doi:10.1007/978-3-540-30301-5_38.

[UH08] Ranjith Unnikrishnan and Martial Hebert. Multi-scale interest regions from unorganized point clouds. *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops*, 2008. doi:10.1109/CVPRW.2008.4563030.

[vD16] Kai-Uwe von Deylen. Projekt 2 Bericht – Eigenposebestimmung mittels Matching zwischen statischen Objekten und Landmarken in einer Karte. Technical report, 2016.

[Wei11] Thorsten-Tobias Weiss. Hochgenaue Positionierung und Kartographie mit Laserscannern für Fahrerassistenzsysteme. 2011.

[WKD05] Thorsten Weiss, Nico Kaempchen, and Klaus Dietmayer. Precise ego-localization in urban areas using laserscanner and high accuracy feature maps. *IEEE Intelligent Vehicles Symposium, Proceedings*, 2005:284–289, 2005. doi:10.1109/IVS.2005.1505116.

[ZB09] Robert Zlot and Michael Bosse. Place Recognition Using Keypoint Similarities in 2D Lidar Maps. *Springer Tracts in Advanced Robotics*, 54:363–372, 2009. doi:10.1007/978-3-642-00196-3_42.

[ZGDH14] Klaus Zindler, Niklas Geiss, Konrad Doll, and Sven Heinlein. Real-time ego-motion estimation using Lidar and a vehicle model based Extended Kalman Filter. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 431–438. IEEE, oct 2014. URL: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=6957728, doi:10.1109/ITSC.2014.6957728.

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 21.02.2017 Kai-Uwe von Deylen