Karlsruher Institut für Technologie

Graphbasiertes SLAM mit integrierter Kalibrierung für mobile Roboter

Diplomarbeit
von

Leonie Sautter

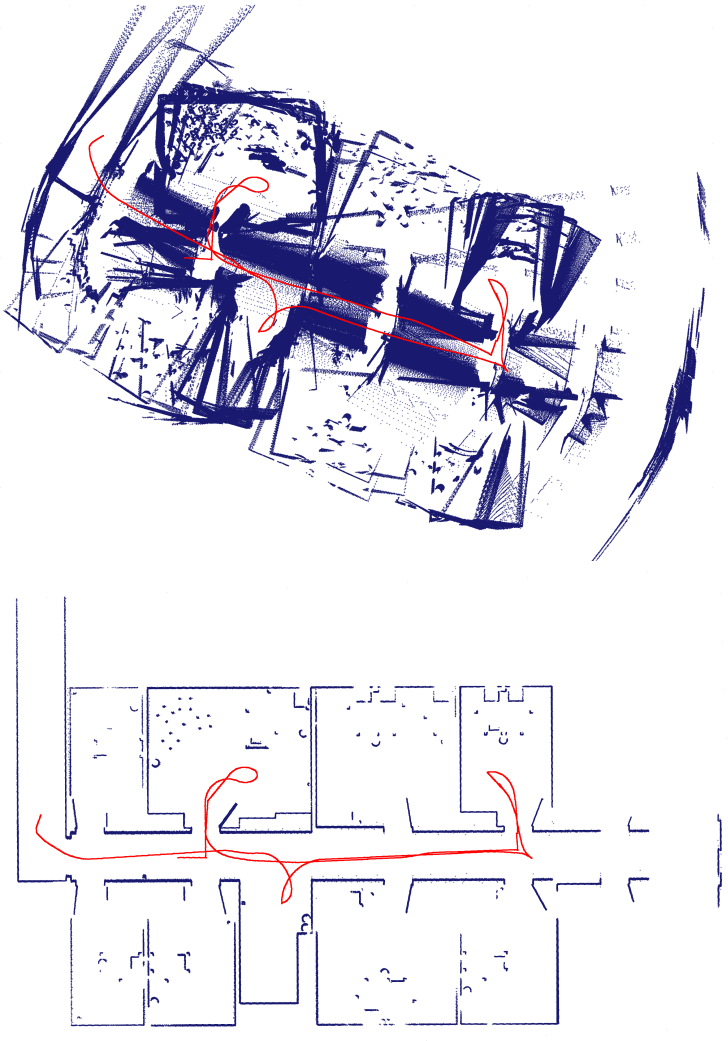
Fakultät für Informatik
Institut für Anthropomatik
und
FZI Forschungszentrum Informatik

Erstgutachter:	Prof. Dr.–Ing. Rüdiger Dillmann
Zweitgutachter:	Prof. Dr.–Ing. Uwe D. Hanebeck
Betreuender Mitarbeiter:	Dipl.–Inform. Jan Oberländer

Bearbeitungszeit: 11. Dezember 2014 – 10. Juni 2015

Graphbasiertes SLAM mit integrierter Kalibrierung für mobile Roboter

von
Leonie Sautter



Diplomarbeit
im Juni 2015

FZI

Diplomarbeit, FZI
Fakultät für Informatik, 2015
Gutachter: Prof. Dr.–Ing. Rüdiger Dillmann, Prof. Dr.–Ing. Uwe D. Hanebeck

Abteilung Interaktive Diagnose- und Servicesysteme
FZI Forschungszentrum Informatik

Erklärung

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

Karlsruhe, *Leonie Sautter*
im Juni 2015

Inhaltsverzeichnis

1 Einleitung

Die vorliegende Arbeit befasst sich mit dem Problem, Sensoren und Kinematik eines mobilen Roboters automatisch zu kalibrieren. Eine fehlende oder schlechte Kalibrierung von Robotern führt zu vergleichsweise leicht zu behebenden Unsicherheiten und Ungenauigkeiten bei der Berechnung der genauen Fahrtroute (auch Trajektorie genannt). Trotzdem werden in der Praxis Roboter oft nur unzureichend oder zu selten kalibriert, da die Kalibrierung als zu aufwändig angesehen wird. Eine automatische Kalibrierung während einer Fahrt des Roboters, für die keine externen Messungen oder Karten der Umgebung benötigt werden, könnte Abhilfe schaffen. Bei der Programmierung eines mobilen Roboters stellt sich sehr schnell die Frage, wie die genaue Position des Roboter zu einem spezifischen Zeitpunkt bestimmt werden kann. Ein erster Ansatz bestünde darin, mit sogenannter Koppelnavigation die Bewegungen des Roboters nachzuvollziehen. Dabei wird zum Beispiel mit Hilfe eines Sensors, der die Anzahl der Radumdrehungen misst, berechnet, wie sich der Roboter bewegt haben müsste. Dieses Vorgehen wird auch Odometrie genannt. Dabei fällt direkt auf, dass Effekte wie Rutschen oder Durchdrehen der Räder nicht berücksichtigt werden können und dadurch Fehler entstehen, die sich mit der Zeit immer weiter verstärken.

Eine Methode zur Verringerung von Odometriefehlern besteht darin, Sensoren auf dem Roboter zu installieren, die Auskunft über die Umgebung oder die Position des Roboters geben und diese Informationen mit einer Karte der Umgebung abzugleichen. Beispiele für solche Sensoren sind Abstandssensoren oder GPS-Sensoren.

Hierbei ergibt sich das offensichtliche Problem, dass eine möglichst korrekte Karte der Umgebung benötigt wird. Die Idee, die Karte der Umgebung vom Roboter selbst mit Hilfe der Sensordaten erstellen zu lassen, führt zur Definition des „Simultaneous Localization and Mapping“ -Problems, kurz SLAM. Dieser Ansatz hat den großen Vorteil, dass keine initialen Informationen über die Umgebung benötigt werden. Des Weiteren können durch die Positionierung des Roboters in der Karte die Odometriefehler teilweise korrigiert werden.

Verschiedene Fehlerquellen können allerdings auch bei SLAM dazu führen, dass der Fehler in der Positionierung des Roboters über die Zeit hinweg stark ansteigt. Die Karte der Umgebung, die der Roboter während der Fahrt erstellt, wird aus den einzelnen Sensor-Scans zusammengesetzt. Der Vorgang des Ausrichtens eines neuen Scans an der bereits erstellten Umgebungskarte wird mit Scan Matching bezeichnet. Als Anhaltspunkt für die Stelle, an der der Scan in die Karte eingefügt werden muss, kann dabei die, mit Hilfe der Odometrie berechnete, Schätzung der Position des Roboters verwendet werden.

Wenn es nun zum Beispiel durch eine Schiefstellung des Sensors oder eines Fehlers in der Odometrie dazu kommt, dass Scans immer wieder leicht verschoben oder verdreht in die Karte eingefügt werden, addieren sich diese Fehler über die Zeit hinweg auf. Dadurch kann auch der Fehler bei der Positionsbestimmung des Roboters mit der Zeit beliebig groß werden.

Eine Möglichkeit der Fehlerkorrektur besteht in einer absoluten Positionsbestimmung, zum Beispiel mithilfe eines GPS-Sensors. Wenn dies allerdings nicht möglich ist, bleibt nur der Versuch, die Fehler soweit möglich von vornherein zu vermeiden. Für einen großen Teil der möglichen Fehlerquellen können die daraus resultierenden Fehler durch Kalibrierung von Odometrie und Umgebungssensoren minimiert werden. Ein gut kalibrierter Roboter macht im Vergleich deutlich kleinere Fehler bei längeren Fahrten und erleichtert damit auch das Scan Matching, da nur in einem kleineren Umfeld der vermuteten Position nach einem passenden Kartenausschnitt gesucht werden muss. Außerdem müssen bei einer genau kalibrierten Odometrie die Sensormessungen nicht so oft durchgeführt werden, da davon ausgegangen werden kann, dass die Schätzung der Trajektorie des Roboters nur mit Hilfe der Odometrie bereits relativ genau ist.

Methoden zur Kalibrierung von mobilen Robotern gibt es schon lange und auch der große Nutzen einer genauen Kalibrierung ist schon länger bekannt. Da sich einige für die Kalibrierung wichtige Einflussfaktoren, wie zum Beispiel die Beladung des Roboters oder der Luftdruck in den Reifen, häufig ändern, kann Kalibrierung nicht als einmalige Prozedur angesehen werden, sondern sollte regelmäßig wiederholt werden. Daher ist es sinnvoll, die Prozedur so einfach wie möglich zu gestalten. Trotzdem sind viele häufig genutzte Methoden zur Kalibrierung relativ aufwändig und beinhalten beispielsweise die genaue Abmessung der Umgebung oder der Position des Roboters im Verhältnis zu festen Objekten der Umgebung und das Abfahren einer vorgegebenen Route. In dieser Arbeit wird nun ein Verfahren entwickelt, das keine Kenntnis der Umgebung oder der Position des Roboters erfordert. Des Weiteren wird es ermöglicht, die Kalibrierung auf Basis der gesammelten Sensormesswerte einer beliebigen Fahrt des Roboters unabhängig von einer vorgegebenen Route durchzuführen. Für bestimmte Typen von Robotern gibt es bereits solche Verfahren, die sich auf eine bestimmte Kombination von Sensoren und Roboterkinematiken beschränken. In dieser Arbeit wird hingegen ein allgemeines Framework zur Kalibrierung möglichst vieler verschiedener Robotertypen entwickelt.

Dafür wird im theoretischen Teil der Arbeit in Kapitel 3 das zu lösende Optimierungsproblem modelliert. Dabei wird eine bestehende Modellierung des SLAM-Problems genutzt und um Faktoren für die Kalibrierung erweitert. Im praktischen Teil in Kapitel 4 und 5 wird das resultierende Verfahren implementiert und mit verschiedenen Robotern sowie in Simulationen erfolgreich getestet. Die Grundlagen dieser Arbeit werden in Kapitel 2 beschrieben. Dort wird das SLAM-Problem mathematisch definiert sowie verschiedene Verfahren zu seiner Lösung vorgestellt. Anschließend werden Methoden zur Kalibrierung mobiler Roboter beschrieben und der aktuelle Stand der Forschung zu diesem Thema vorgestellt. Außerdem wird noch eine kurze Einführung in Lie-Gruppen gegeben, da diese im theoretischen Teil der Arbeit relevant werden.

2 Grundlagen

Da in dieser Arbeit die automatische Kalibrierung mobiler Roboter während des SLAM-Prozesses betrachtet wird, beschreibt dieses Kapitel zuerst das SLAM-Problem allgemein und die verschiedenen Lösungsansätze. Die in der Literatur zu findenden Verfahren lassen sich im Wesentlichen in drei Bereiche unterteilen: Methoden, die auf dem Kalman-Filter, Partikelfiltern oder auf graphbasierter Optimierung beruhen. In dieser Arbeit wird ein graphbasiertes Verfahren verwendet. In Kapitel 2.3 wird auf die verschiedenen Ansätze zur manuellen und automatischen Kalibrierung eingegangen.

In dem graphbasierten Verfahren werden die Bewegungen von Objekten in der Ebene als Elemente einer speziellen Lie-Gruppe betrachtet. In Kapitel 2.4 findet sich eine kurze Einführung in die Grundlagen von Lie-Gruppen. Des Weiteren wird die in Kapitel 3 verwendete Lie-Gruppe definiert und motiviert.

2.1 Das SLAM-Problem

Das SLAM-Problem (Simultaneous Localization and Mapping) wirft die Frage auf, ob es für einen Roboter möglich ist, sich zurecht zu finden, wenn er in eine unbekannte Umgebung gesetzt wird. Dafür muss der Roboter inkrementell eine Karte seiner Umgebung erstellen und gleichzeitig seine Position in dieser Karte feststellen [TBF05]. Die Lösung dieses Problems ist damit ein wichtiger Schritt auf dem Weg zu autonomen Robotern. Auch für die automatische Kalibrierung von mobilen Robotern ist ein funktionierendes SLAM-Verfahren Voraussetzung, falls kein Vorwissen über die Umgebung vorhanden ist. Vorausgesetzt wird in dieser Arbeit nur, dass die Umgebung statisch ist, das heißt sie verändert sich nicht während der Erstellung der Karte.

2.1.1 Mathematische Beschreibung

Mathematisch wird das SLAM-Problem hier analog zu [DWB06] probabilistisch beschrieben. Das Ziel eines SLAM-Verfahrens besteht darin, aus Sensor- und Odometriedaten die tatsächliche Robotertrajektorie und ein Modell der Umgebung, Karte genannt, zu schätzen. Die Robotertrajektorie besteht dabei aus einzelnen Roboterposen für jeden Zeitpunkt, während die Karte auf verschiedene Arten repräsentiert werden kann, beispielsweise durch Landmarken, das heißt auffällige Punkte in der Umgebung. Alle diese Roboterposen und Landmarkenpositionen sowie die Messwerte stehen

nun als Zufallsvariablen in Abhängigkeit zueinander. Die hier vorgestellten Verfahren zielen darauf ab, die gemeinsame Wahrscheinlichkeitsverteilung aller Zufallsvariablen aus Karte und Roboter-Trajektorie abhängig von den Messwerten zu schätzen. Anhand dieser Schätzung kann dann für gegebene Messwerte der Erwartungswert der Verteilung und damit die wahrscheinlichste Robotertrajektorie und Umgebungskarte bestimmt werden. In den folgenden Abschnitten werden diese Zusammenhänge genauer beschrieben und formalisiert.

Die Trajektorie des Roboters $X_{1:T}$ wird mit einer Folge von Roboterposen x_t beschrieben, wobei t die Zeit beschreibt. Die Notation $X_{1:T}$, gibt dabei an, dass es sich um die Folge der Posen von Zeitpunkt 1 bis T handelt. Eine Roboterpose besteht meistens aus drei Koordinaten. Zwei beschreiben die Position des Roboters in der Ebene und die dritte die Orientierung in Form eines Winkels.

$$X_{1:T} = (x_0, x_1, \ldots, x_T), \qquad x_t = (a_t, b_t, \theta_t) \tag{2.1}$$

Die Folge $U_{1:T}$ der Odometriemessungen u_t und die Folge $Z_{1:T}$ der Messungen der Umgebungssensoren z_t ermöglichen es $X_{1:T}$ und ein Modell der Umgebung m zu schätzen. Dabei kann m zum Beispiel die Position von Landmarken enthalten oder aber eine Belegtheitskarte der Umgebung. Eine Belegtheitskarte besteht aus einem Gitter, in dem einzelne Zellen als belegt markiert werden, wenn sich an dieser Stelle in der Umgebung laut einer Messung ein Hindernis beispielsweise eine Wand befindet.

Die oben genannte A-posteriori-Verteilung kann nun wie folgt beschrieben werden:

$$P(X_{1:T}, m | Z_{1:T}, U_{1:T}, x_0) \tag{2.2}$$

Der Startpunkt x_0 wird dabei meistens auf den Ursprung gesetzt.

Um diese Verteilung schätzen zu können, werden noch ein Bewegungsmodell und ein Messmodell benötigt. Das Bewegungsmodell besteht aus Wahrscheinlichkeitsverteilungen, die beschreiben, wie aus einer Odometriemessung und der letzten Roboterpose die nächste Roboterpose geschätzt werden kann:

$$P(x_t | x_{t-1}, u_t) \tag{2.3}$$

Dabei wird angenommen, dass die Übergänge vom einen Zustand zum Nächsten in der Folge $X_{1:T}$ der Markovbedingung genügen. Das heißt, der neue Zustand hängt zu jedem Zeitpunkt nur von dem direkt vorhergehenden Zustand und der aktuellen Odometriemessung ab.

Das Messmodell beschreibt, was die Sensoren ausgeben, wenn eine Roboterposition und eine Karte gegeben sind. Auch dies ist eine Wahrscheinlichkeitsverteilung, da die Sensormessungen nicht immer exakt sind.

$$P(z_t | x_t, m) \tag{2.4}$$

In dieser Arbeit, sowie in den meisten Anwendungen, werden Bewegungs- und Messmodell als

multivariate Gauß-Verteilungen $\mathcal{N}(\mu, \sigma)$ beschrieben:

$$P(x_t|x_{t-1}, u_t) \sim \mathcal{N}(g(x_{t-1}, u_t), Q_t), \qquad P(z_t|x_t, m) \sim \mathcal{N}(h(x_t, m), R_t) \tag{2.5}$$

Dabei modelliert g die Kinematik des Roboters und h gibt an, wie die Beobachtungen der Sensoren die Umgebung abbilden. Die Matrizen Q_t und R_t sind die Kovarianzen der zugehörigen Verteilungen.

2.1.2 Varianten und Teilprobleme

Allgemein werden zwei Varianten des SLAM-Problems unterschieden: Offline oder full SLAM und online SLAM [TBF05]. Das Offline-SLAM-Problem wurde bereits im letzten Abschnitt beschrieben. Dabei wird meist nach der Fahrt des Roboters die wahrscheinlichste Karte und Trajektorie mit Hilfe der gesamten geloggten Daten berechnet.

Bei online SLAM geht es dagegen darum, die Berechnung in Echtzeit während der Fahrt durchführen zu können. Da das Offline-SLAM-Problem einen hohen Rechenaufwand beinhaltet wird ein etwas einfacheres Problem betrachtet. Statt der gesamten Trajektorie des Roboters wird jetzt inkrementell immer nur die wahrscheinlichste aktuelle Roboterpose gesucht. Die A-posteriori-Verteilung des Online-SLAM-Problems sieht dann wie folgt aus:

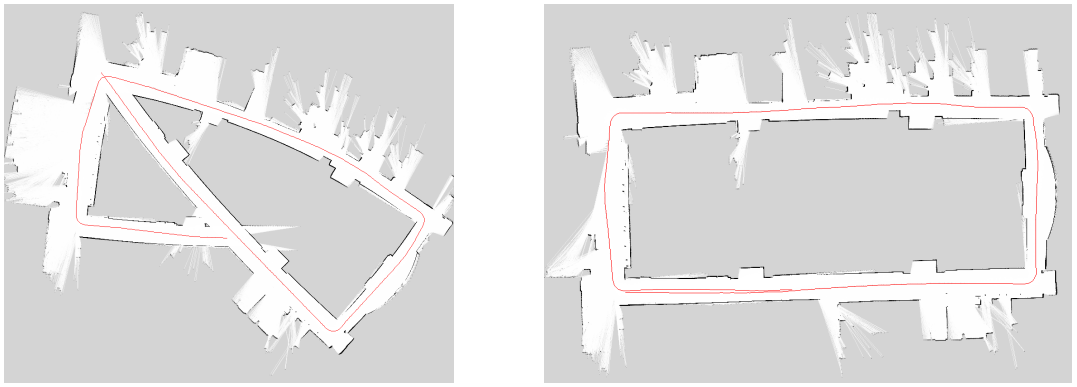
$$P(x_t, m|Z_{1:t}, U_{1:t}, x_0) \tag{2.6}$$

Wird SLAM offline gelöst, berechnet ein Computer nach der Fahrt des Roboters die wahrscheinlichste Karte und Trajektorie auf Basis der gesamten geloggten Daten. Wird SLAM hingegen online gelöst, berechnet der Roboter inkrementell für jeden Zeitpunkt die A-posteriori-Verteilung, allerdings nicht für die gesamte Trajektorie sondern nur für die aktuelle Roboterposition. Aufgrund der inkrementellen Berechnung muss hierbei zu jedem Zeitpunkt nur der Übergang vom letzten zum aktuellen Zustand betrachtet werden, was den Rechenaufwand stark verringert. Offline-SLAM-Algorithmen können wegen des hohen Rechenaufwands in den meisten Fällen nur in sehr kleinen Umgebungen online ausgeführt werden. Eine Ausnahme bilden auf dem „Smoothing-and-Mapping-Ansatz“ beruhende Verfahren, die mit kleineren Einschränkungen auch für den online-Einsatz geeignet sind [KRD08].

Ein weiteres Problem, das innerhalb eines SLAM-Verfahrens gelöst werden muss, besteht darin, neue Messdaten in das Umgebungsmodell zu integrieren. Wird zum Beispiel die Umgebung mit Hilfe einer festen Anzahl an Landmarken, die als Punkte in der Ebene repräsentiert werden, modelliert, stellt sich die Frage, wie bei einer neuen Sensormessung die gemessenen Landmarken mit denen in der Karte identifizieren werden kann. Häufiger geben Sensoren allerdings Abstände mit zugehörigen Messwinkeln zurück. Solche Messungen können durch eine Punktwolke rund um den Roboter dargestellt werden. In diesem Fall kann die Punktwolke als lokaler Kartenaus-

schnitt verstanden werden, der nun an der richtigen Stelle in die Gesamtkarte eingepasst werden muss. Das Problem, die richtige Stelle für einen neuen Kartenausschnitt zu finden, entspricht dann dem Problem, gemessene Landmarken zu identifizieren. Die Schätzung der aktuellen Position des Roboters mit Hilfe der Odometriedaten gibt natürlich einen ersten Ansatz für die Position des Kartenausschnitts in der Gesamtkarte. Da die Daten fehlerbehaftet sind, ist es mit dieser Schätzung allein allerdings nicht möglich das Kartenupdate korrekt durchzuführen. Das beschriebene Problem wird in der Literatur auch Korrespondenzproblem genannt. Im Falle der Kartenmodellierung mit Punktwolken werden die Algorithmen für das Problem Scan Matcher genannt.

Da sich in der Praxis durch schlechte Kalibrierung oder zufällige Fehler wie Sensorrauschen immer kleine Fehler beim Scan Matching ergeben, die sich über die Zeit hinweg aufsummieren, kann es bei SLAM trotzdem passieren, dass die berechnete Position des Roboters nach einer Weile stark von der wirklichen Position abweicht. Durch eine gute Kalibrierung kann dieser Effekt zwar nicht komplett abgewendet aber verlangsamt werden. Vollumfänglich korrigiert werden können solche Fehler nur durch eine absolute Positionsbestimmung, zum Beispiel durch einen GPS-Sensor, oder durch das Wiedererkennen vor längerer Zeit gescannter Orte. Im zweiten Fall erkennt der Roboter beim Scan Matching, dass er sich am gleichen Ort schon einmal befunden hat und kann dann die aktuelle Position gleich der früheren Position setzen. Dieses Verfahren wird Loop Closing genannt. Da der Fehler, der dadurch korrigiert wird, über die gesamte Fahrt seit dem ersten Verlassen der aktuellen Position akkumuliert wurde, muss dieser auch über die gesamte Trajektorie des Roboters verteilt korrigiert werden. Die Abbildung 2.1 zeigt in einem Beispiel die vom Roboter produzierte Karte mit und ohne Loop Closing.



(a) Karte ohne Loop Closing berechnet

(b) Gleiche Karte mit Loop Closing

Abb. 2.1: Diese beiden Abbildungen aus [KD05] veranschaulichen den großen Einfluss, den Loop Closing auf die Exaktheit der berechneten Karte haben kann. Abbildung (a) zeigt den Zustand der Karte ohne Loop Closing, Abbildung (b) die Karte nachdem Loop Closing durchgeführt wurde. Die Trajektorie wurde dabei gegen den Uhrzeigersinn abgefahren. Erst am Ende der Trajektorie erkennt der Roboter, dass er die gleiche Umgebung schon einmal gescannt hat, da die Messungen vorher nicht charakeristisch genug sind. Die Winkelabweichung wird über die ganze Strecke korrigiert.

2.2 Lösungsansätze für das SLAM-Problem

In diesem Kapitel werden die drei grundlegenden Verfahren für SLAM kurz vorgestellt und ihre Vor- und Nachteile aufgezeigt.

2.2.1 Erweiterter Kalman-Filter

Der von Rudolf E. Kalman 1960 eingeführte Kalman-Filter ist eine Sammlung mathematischer Sätze, die eingesetzt werden können, falls aus fehlerbehafteten Messungen auf den unbekannten Zustand eines Systems geschlossen werden soll [Kal60]. In Verbindung mit dem SLAM-Problem wurde der Kalman-Filter erstmals 1990 von Smith et al. gebracht [SSC90]. Er kann verwendet werden um das online-SLAM-Problem zu lösen. Dafür werden die gesuchten Variablen, nämlich die aktuelle Roboterposition und Karte der Umgebung, zu einem einzigen Zustandsvektor zusammengefasst. Weiter wird angenommen, dass die A-posteriori-Verteilung normalverteilt ist mit Parametern μ_t und Σ_t, wobei der Mittelwert μ_t der besten Schätzung für den Zustand zum Zeitpunkt t entspricht.

$$P(x_t, m | Z_{1:t}, U_{1:t}) \sim \mathcal{N}(\mu_t, \Sigma_t) \qquad [2.7]$$

Der erweiterte Kalman-Filter (EKF) ist im Gegensatz zu dem konventionellen Kalman-Filter auch geeignet, wenn die Bewegungsfunktion g und die Messfunktion h nicht linear sind. Dies ist bei mobilen Robotern meistens der Fall, da zum Beispiel durch Drehungen des Roboters trigonometrische Terme in die Bewegungsfunktion eingehen.

Mit dem erweiterten Kalman-Filter kann nun inkrementell die jeweils beste Schätzung des Roboters zu jedem Zeitpunkt berechnet werden. Dabei wird jeweils der Zustandsvektor μ_t und die Kovarianzmatrix Σ_t, die die Unsicherheit der Schätzung beschreibt, aktualisiert. Dies geschieht in zwei Schritten, dem Vorhersage- und dem Beobachtungsschritt.

Im Vorhersageschritt wird mit Hilfe der Odometrie eine Vorhersage dafür abgegeben, wo der Roboter sich einen Zeitschritt später wahrscheinlich befinden wird. Genauer gesagt wird mit Hilfe des Bewegungsmodells $P(x_t | x_{t-1}, u_t)$ und der geschätzten Verteilung über dem letzten Zustand $\mathcal{N}(\mu_t, \Sigma_t)$ eine Schätzung der Wahrscheinlichkeitsverteilung über dem aktuellen Zustand berechnet. Diese Schätzung basiert dabei nur auf der Odometrie, die Messdaten werden erst einmal außer acht gelassen.

Im Beobachtungsschritt werden nun die Messdaten verwendet um die getroffene Vorhersage zu korrigieren. Dabei wird ein Parameter genannt Kalman Gain berechnet, der anzeigt, ob der Messung oder der Odometrie in der Schätzung mehr vertraut wird.

Das Hauptproblem des Kalman-Filter-Ansatzes ist, dass in jedem Schritt die gemeinsame Kovarianzmatrix Σ aktualisiert werden muss. In dem Fall, dass die Umgebung durch N Landmarken

beschrieben wird ist die Anzahl der Zeilen und Spalten der Matrix jeweils $3+2N$. Des Weiteren ist die Matrix nicht dünn besetzt und gerade die Elemente, die nicht auf der Diagonalen liegen, sind wichtig für die Konvergenz des Verfahrens [DWB06]. Damit wächst der Rechenaufwand quadratisch mit der Zahl der Landmarken.

2.2.2 Partikelfilter

Partikelfilter nutzen die Idee, mit rekursivem Monte-Carlo-Sampling [MU49] die A-posteriori-Verteilung approximieren zu können. Geht die Anzahl der zufällig gezogenen Datenpunkte, hier Partikel genannt, gegen unendlich, nähert sich ihre Verteilung der tatsächlichen A-posteriori-Verteilung an. Der große Vorteil der Methode ist, dass sie nicht parametrisch ist, das heißt es werden keine Annahmen über die Form der A-posteriori-Verteilung getroffen (im Fall des EKF wurde eine Gauß-Verteilung vorausgesetzt).

Ein Partikel steht dabei für eine Schätzung des Zustands des Systems. Da in diesem Datenpunkt einerseits die Trajektorie $X_{1:T}$ und andererseits eine Beschreibung der Umgebung enthalten sein muss, hat der Zustandsraum eine sehr hohe Dimension. Ein großes Problem dieses Ansatzes ergibt sich daraus, dass die Anzahl der benötigten Partikel, um die A-posteriori-Verteilung gut genug approximieren zu können, exponentiell mit der Dimension wächst. Erstmals gelöst wurde dieses Problem 2002 mit dem FastSLAM-Algorithmus [MTKW02]. Montemerlo et al. nutzen dafür den Satz von Rao-Blackwell, der es erlaubt, den Zustandsraum wie folgt zu faktorisieren [Bla47], [Rao45]:

$$P(x_t, m|Z_{1:t}, U_{1:t}, x_0) = P(x_t|Z_{1:t}, U_{1:t}, x_0) \prod_i P(l_i|X_{1:t}, Z_{1:t}, U_{1:t}, x_0) \quad [2.8]$$

In diesem Fall wird die Umgebung m durch Landmarken $(l_1, \ldots, l_M)$ beschrieben. Mit Hilfe der Faktorisierung können nun die Partikel aus einem Zustandsraum kleinerer Dimension, in diesem Fall aus der A-posteriori-Verteilung für den Pfad $P(x_t|Z_{1:t}, U_{1:t}, x_0)$, gesamplet werden. Der Algorithmus liest nun inkrementell die Odometrie- und Sensordaten ein. Zu jedem Zeitpunkt gibt es dabei K Partikel, die in zwei Schritten, analog zum EKF, angepasst werden. Im Vorhersageschritt werden basierend auf dem Bewegungsmodell und den neuen Odometriedaten neue Roboterpositionen gesamplet. Für jeden Partikel kann nun die Verteilung $P(l_i|X_{1:t}, Z_{1:t}, U_{1:t}, x_0)$ für jede Landmarke l_i durch eine Gauß-Verteilung mit Dimension zwei dargestellt werden. Im Beobachtungsschritt berechnet FastSLAM zuerst für jeden Partikel die Wahrscheinlichkeit, dass die beobachtete Messung tatsächlich gemessen wird. Nun werden aus allen Partikeln mit Zurücklegen neue Partikel gezogen, wobei die Wahrscheinlichkeit, dass ein Partikel gezogen wird, der eben berechneten entspricht. Anschließend werden die Gauß-Verteilungen der Landmarken angepasst.

Damit löst FastSLAM sowohl das offline-SLAM-Problem, da jeder Partikel die gesamte Robotertrajektorie enthält, als auch das online-SLAM-Problem, da beim Update in jedem Zeitschritt nur auf die jeweils letzte Roboterposition zugegriffen wird.

2.2.3 Graphbasierte Optimierung

Bei der graphbasierten Optimierung wird die zu optimierende Wahrscheinlichkeitsverteilung als Graph dargestellt. Knoten des Graphen stellen dabei Zufallsvariablen dar, Kanten repräsentieren stochastische Abhängigkeiten zwischen den verbundenen Knoten. Das heißt jede Kante steht für eine Wahrscheinlichkeitsverteilung über den inzidenten Knoten. Damit lässt sich aus dem Graph direkt eine Faktorisierung der gemeinsamen Wahrscheinlichkeitsverteilung aller Zufallsvariablen ablesen. Die gemeinsame Wahrscheinlichkeitsverteilung ist genau das Produkt der Wahrscheinlichkeitsverteilungen aller Kanten [Del12]. Diese Faktorisierung kann nun genutzt werden, um die gemeinsame Verteilung effizienter zu berechnen. Implizit wird dabei ausgenutzt, dass viele der Variablen oft nur von wenigen anderen Variablen direkt abhängig sind [KGS+11].

Graphbasierte Optimierungsverfahren werden hauptsächlich in dem Bereich Computer Vision und der Robotik verwendet. Beispiele für Probleme, bei denen graphbasierte Verfahren Anwendung finden, sind zum Beispiel Bündelblockausgleichung oder Structure from Motion [BSSC13], [GKN12]. Bei letzterem Problem soll aus einer Folge von 2D-Bildern von einer bewegten Kamera die 3D-Umgebung rekonstruiert werden. Bei der Bündelblockausgleichung wird ein 3D-Raum mit mehreren Kameras beobachtet. Nun sollen die geschätzten Positionen von Kameras und Punkten im Raum so angepasst werden, dass Fehler wie Bildverzerrungen und Messfehler minimiert werden und dabei auch noch möglichst gleichmäßig auf die verschiedenen Beobachtungen aufgeteilt werden.

Bei der Anwendung auf das SLAM-Problem wird die A-posteriori-Verteilung als Graph dargestellt. Dabei repräsentieren die Knoten Positionen des Roboters oder der Landmarken oder Sensormessungen zu bestimmten Zeitpunkten. Die Kanten dazwischen bilden stochastische Abhängigkeiten zwischen den Variablen ab. So kann mit Hilfe eines Graphen eine Faktorisierung der gemeinsamen Verteilung dargestellt werden [TL08].

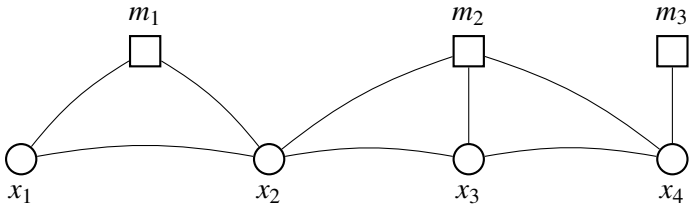


Abb. 2.2: Beispiel einer graphischen Darstellung mit Landmarken. Die runden Knoten stehen dabei für die Position des Roboters zu verschiedenen Zeitpunkten, die eckigen, für die Position der beobachteten Landmarken.

Abbildung 2.2 zeigt ein Beispiel eines solchen Graphen für landmarkenbasiertes SLAM. Die Knoten x_1 bis x_4 repräsentieren Zufallsvariablen für die Roboterposition zu den Zeitpunkten $t = 1$ bis 4. Die aufeinander folgenden Knoten sind jeweils durch eine Kante verbunden, da jede Position von der vorhergehenden Position abhängig ist. Die Knoten m_1 bis m_3 repräsentieren Zufallsvariablen

für die Position der beobachteten Landmarken. Wenn der Roboter eine Landmarke beobachtet hat, führt das zu einer Abhängigkeit der geschätzten Position der Landmarke von der aktuellen Roboterposition, die auch durch eine Kante dargestellt wird.

Um einen Graphen wie in Abbildung 2.2 erstellen zu können, muss offensichtlich auch das Korrespondenzproblem gelöst werden. Nur damit kann für zwei Beobachtungen einer Landmarke zu unterschiedlichen Zeitpunkten festgestellt werden, ob dieselbe oder zwei unterschiedliche Landmarken beobachtet wurden. In dem in dieser Arbeit vorgestellten Algorithmus werden statt einzelnen Landmarken ganze Scans der Umgebung verwendet. Diese werden mit Hilfe eines Scan Matchers relativ zueinander ausgerichtet, was der Lösung des Korrespondenzproblems entspricht.

Die erste Veröffentlichung, in der ein Graph mit Sensorpositionen als Knoten und ihren Relationen als Kanten verwendet wird, stammt aus 1997 von Feng Lu und Evangelos Milios [LM97]. Diese Arbeit hat das Ziel, mit Hilfe der Sensoren mobiler Roboter eine möglichst genaue Umgebungskarte zu erstellen. Die Autoren stellen fest, dass inkrementelle Methoden, bei denen immer jeweils ein Scan zu einer Gesamtkarte hinzugefügt wird, zu Inkonsistenzen neigen. Deshalb führen sie eine Graphstruktur ein, um Informationen über die relativen Positionen von Scans zueinander zu speichern. Dabei wird zwar in jedem Schritt berechnet, wie der aktuelle Scan relativ zum letzten eingefügt werden müsste, diese Information wird aber zuerst nur im Graph gespeichert. Erst nach der Erstellung des kompletten Graphen wird optimiert und damit eine global konsistente Umgebungskarte erzeugt.

Weiterentwickelt wurde diese Methode von Sebastian Thrun und Michael Montemerlo in ihrem Algorithmus GraphSLAM [TM06]. Der Algorithmus löst das SLAM-Problem offline und erstellt dafür einen Graphen wie in Abbildung 2.2. Die Umgebungskarte wird hierbei durch Positionen von Landmarken (hier Features genannt) repräsentiert. Das resultierende Korrespondenzproblem löst ein Greedy-Algorithmus. Die Funktionen der Kanten des Graphen werden dann linearisiert und in einer Informationsmatrix repräsentiert. Diese Matrix und damit auch der zugehörige Graph wird nun vereinfacht und zwar werden Kanten zwischen Roboterposen und Features durch Kanten nur zwischen Roboterposen ersetzt. Insgesamt ist es möglich alle Knoten die Features repräsentieren so aus dem Graphen und der Matrix zu eliminieren, dass die A-Posteriori-Wahrscheinlichkeit sich nicht ändert. Damit kann die Dimensionalität des Problems stark verringert und das Problem nun mit einem Standard-Optimierungsverfahren gelöst werden. Anschließend können die Positionen der Features aus den optimierten Roboterposen zurückgewonnen werden. Für die nötige Linearisierung wird allerdings eine initiale Schätzung der Robotertrajektorie benötigt. Diese wird durch iteratives Anwenden des Bewegungsmodells mit den Odometriemessungen berechnet. Für längere Trajektorien wird die initiale Schätzung immer schlechter, was letztendlich dazu führt, dass der Algorithmus divergiert. Trotzdem konnten mit dem Algorithmus erstmals korrekte Karten mit bis zu 10^8 Features erstellt werden.

In dieser Arbeit werden Faktorgraphen zur Darstellung verwendet [KF09]. Diese unterscheiden sich von dem oben gezeigten Beispiel dadurch, dass die Wahrscheinlichkeitsverteilungen durch

eigene Knoten, genannt Faktoren, repräsentiert werden. Das Resultat ist ein bipartiter Graph, wie er in Abbildung 2.3 zu sehen ist. Der Faktor f_1, der durch eine neu hinzugekommene Kante mit x_1 verbunden ist, stellt dabei die nur von x_1 abhängende A-priori-Verteilung dar. Die Wirkung des Faktors f_1 ist es, die erste Roboterpose x_1 auf $(0,0,0)^T$ festzulegen. Dies ist nötig, falls alle weiteren Faktoren nur Wahrscheinlichkeitsverteilungen über die Posen relativ zueinander enthalten.

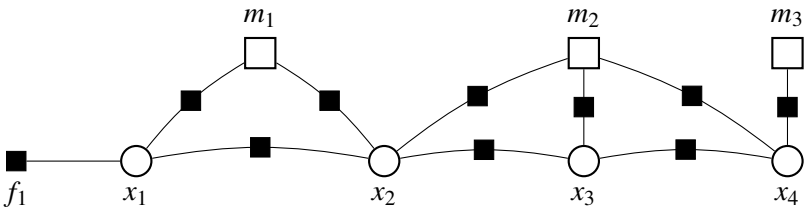


Abb. 2.3: Beispiel eines Faktorgraphen. Die kleinen eckigen Knoten stehen dabei für die Faktoren.

Der Graph für eine Roboterfahrt wird während oder nach der Fahrt inkrementell aus den Sensor- und Odometriewerten erstellt. Der Algorithmus löst das SLAM-Problem nun offline, indem er die wahrscheinlichste Trajektorie und Umgebungskarte berechnet.

$$X^*_{1:T}, m^* = \underset{X_{1:T},m}{\operatorname{argmax}} P(X_{1:T}, m | Z_{1:T}, U_{1:T}) \qquad [2.9]$$

Dies wird durch die Faktorisierung erst effizient machbar. Durch Logarithmierung der Funktion wird aus dem Produkt eine Summe:

$$\log P(X_{1:T}, m | Z_{1:T}, U_{1:T}) = \text{const} + \sum_t \log P(x_t | x_{t-1}, u_t) + \sum_t \log P(z_t | x_t, m) \qquad [2.10]$$

Der konstante Term steht für die A-priori-Verteilung über der ersten Roboterposition. Für jeden weiteren Summand in der Summe gibt es eine korrespondierende Kante im Graphen. Die einzelnen Summanden sind allerdings in der Regel nicht linear. Zum Beispiel gibt es durch die Verwendung der dritten Dimension von Roboterposen, die für die Orientierung des Roboters steht, trigonometrische Anteile in der Funktion des Bewegungsmodells. Um die Summe mit einem Standard-Optimierungsverfahren minimieren zu können, muss deshalb eine lokale Linearisierung der Optimierungsfunktion stattfinden. Dies implementiert beispielsweise der auch in dieser Arbeit verwendete Levenberg-Marquardt-Algorithmus, der auf der Methode der kleinsten Quadrate beruht [Lev44], [Mar63].

Das graphbasierte Verfahren skaliert deutlich besser als die Kalman-Filter-Methode. Speicher- und Zeitkomplexität für die Erstellung des Graphen sind linear. Die Optimierung kann bei großen Karten trotzdem sehr aufwändig werden, so dass das Verfahren so kaum online, das heißt in Echtzeit während der Fahrt, verwendet werden kann. Dafür wird anders als beim Kalman-Filter über die gesamte Trajektorie optimiert und nicht nur die jeweils aktuelle Roboterposition geschätzt.

Aktuelle Erweiterungen des graphbasierten Verfahrens, die auf einem 2008 von Michael Kaess et al. vorgestellten Ansatz beruhen, können durch die Verwendung der QR-Zerlegung [GVL96] einer dünn besetzten Matrix, die Rechenzeit deutlich reduzieren und sind damit auch für den online-Einsatz geeignet [KRD08]. Dabei wird in jedem Zeitschritt über die gesamte Trajektorie optimiert. Trotzdem kann die Aktualisierung der dünn besetzten Matrix bei dem Eintreffen neuer Messwerte mithilfe der QR-Zerlegung in konstanter Zeit durchgeführt werden, da nur wenige Einträge der Matrix neu berechnet werden müssen. Allerdings nimmt die Rechenzeit, die für jeden neuen Messwert benötigt wird, mit der Länge und der Anzahl der Schleifen in der Trajektorie stark zu. Dies führt dazu, dass ab einer bestimmten Komplexität der Trajektorie die Aktualisierungsschritte am Anfang der Fahrt zwar sehr schnell gehen, sie gegen Ende der Trajektorie jedoch für den Echtzeiteinsatz zu lange brauchen.

2.3 Kalibrierung mobiler Roboter

Die Vorteile einer genauen Kalibrierung mobiler Roboter liegen auf der Hand. Geringere Fehler in der Positionsbestimmung und gegebenenfalls auch der Kartierung der Umgebung lassen sich in manchen Fällen nur durch Kalibrierung erreichen. Dies ist zum Beispiel der Fall, falls die Sensoren des Roboters temporär nicht funktionsfähig sind (z.B. GPS-Sensor bei der Fahrt in einem Gebäude) oder falls er nur Zugriff auf eine sehr ungenaue oder nicht aktuelle Karte der Umgebung hat. Der Roboter braucht nun eine sehr genau kalibrierte Odometrie, damit der Fehler in der Positionsbestimmung nicht schon nach kurzer Zeit enorm wächst. Dieses Beispiel zeigt, dass es sich auch bei sehr genauen externen Sensoren lohnt, die Odometrie zu kalibrieren, falls die Möglichkeit eines temporären Sensorausfalls besteht.

Selbst wenn der Roboter die Möglichkeit hat, seine Umgebung kontinuierlich mit Hilfe von Sensoren zu erfassen, hilft eine Kalibrierung der Odometrie sowie der Sensoren, Fehler zu verringern und den Rechenaufwand klein zu halten. So kann die Frequenz der Sensormessungen bei einem Roboter mit genau kalibrierter Odometrie verringert werden, ohne die Genauigkeit der Positionsbestimmung im gleichen Maße zu beeinträchtigen.

Die Fehler in der Positionsbestimmung, die bei einem mobilen Roboter entstehen, lassen sich nach ihren Ursachen in systematische und nicht-systematische Fehler aufteilen. Systematische Fehler entstehen dadurch, dass bestimmte Eigenschaften des Roboters oder der Umgebung entweder gar nicht berücksichtigt, oder fehlerhaft quantifiziert werden. Sie treten in einem beschränkten Zeitrahmen entweder mit konstanter oder sich auf regelmäßige Art und Weise ändernder Stärke auf. Zum Beispiel kann der Durchmesser der Räder des Roboters von dem in der Berechnung der Trajektorie angenommen Durchmesser abweichen. Der Roboter fährt dann durchgehend um einen bestimmten Faktor weiter oder kürzer als berechnet. Weitere Fehlerquellen, die in diese Kategorie gehören, sind verschieden große Räder oder ein ungenau auf dem Roboter platzierter Sensor. Im Gegensatz dazu sind nicht-systematische Fehler solche, die nicht vorhersagbar, zufällig und

mit unterschiedlicher Stärke auftreten [Top72]. Sie können, wie beispielsweise das Rutschen eines Rades auf glattem Untergrund, durch die Beschaffung der Umgebung entstehen oder rein zufällig sein, wie das Rauschen von Sensoren. Nicht-systematische Fehler lassen sich ihrer Natur nach kaum vorhersehen und daher nur sehr schwer vermeiden. Systematische Fehler können hingegen durch eine genaue Kalibrierung des Roboters stark eingeschränkt werden.

Eine Auflistung möglicher Fehlerquellen findet sich in [BF96]. Dort werden bezogen auf einen Roboter mit Differentialantrieb folgende systematische Fehler genannt:

- die Durchmesser der Räder sind nicht gleich
- das Mittel der Durchmesser der Räder weicht von dem für Odometriemessungen verwendeten Wert ab
- die Räder sind nicht parallel zueinander
- der Abstand der Räder ist nicht genau bekannt (Durch die Breite der Räder rutschen diese in Kurven an fast allen Stellen der Auflagefläche leicht. Für die Odometrie wird aber der Abstand zwischen den nicht rutschenden Punkten auf dem Rad benötigt.)
- der Radumdrehungsmesser des Roboters (Encoder) hat eine ungenügende Auflösung
- der Encoder hat eine zu kleine Messrate
- die allgemeine Untergrundbeschaffenheit ist nicht bekannt

Des Weiteren werden folgende nicht-systematische Fehler aufgelistet:

- Fahrt über unebenen Boden
- Fahrt über unerwartete Objekte auf dem Boden
- Wegrutschen der Räder durch:
  - glatten Untergrund
  - zu schnelle Beschleunigung
  - zu schnelle Drehungen
  - externe Kräfte (z.B: Kollision mit Objekten in der Umgebung)
  - interne Kräfte
  - nicht punktförmige Kontaktfläche zwischen Rad und Untergrund

Der letzte systematische Fehler wurde der Liste aus [BF96] hinzugefügt, da eine einheitliche Un-

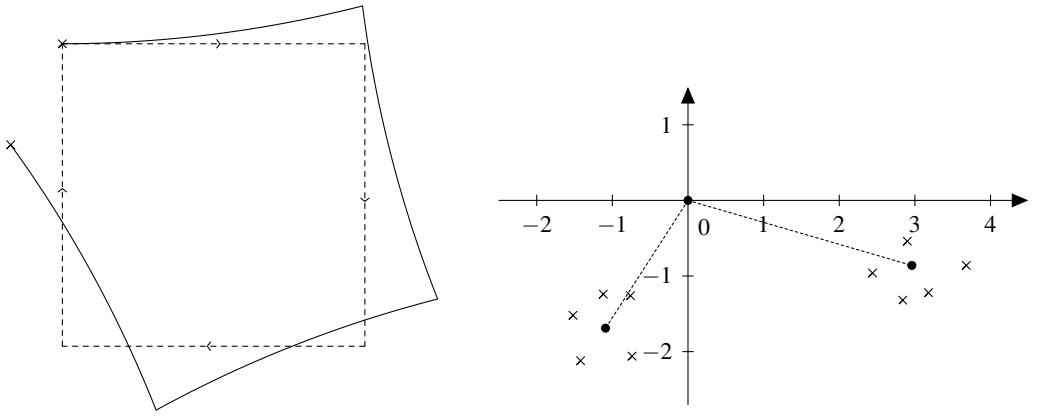
tergrundbeschaffenheit zur Fehlerminimierung in das Bewegungsmodell des Roboters mit aufgenommen werden kann [KGB12].

In dieser Arbeit wird sich auf die Minimierung der systematischen Fehler beschränkt. Diese wirken sich einerseits besonders stark auf den Fehler aus, da sie sich über die Zeit hinweg aufsummieren. Andererseits sind sie auch deutlich leichter zu beheben, da sie in unserem Szenario meist darauf beruhen, dass ein Parameter nicht genau bekannt ist oder im Modell nicht berücksichtigt wird. Nicht-systematische Fehler hängen oft vollständig von der jeweiligen Umgebung des Roboters ab. Sollte ein Roboter auch im Hinblick auf nicht-systematische Fehler kalibriert werden, müsste die Kalibrierung für jede neue Umgebung aktualisiert werden. Des Weiteren kann ein Roboter während einer einzigen Fahrt auf sehr unterschiedliche Umgebungen treffen (z.B. geteerte Wege und Kieswege). Trotz all dieser Schwierigkeiten gibt es Ansätze auch diese Fehler zu minimieren. So stellte Johann Borenstein 1994 einen Roboteraufbau vor, in dem zwei einzelne Fahrzeuge mit jeweils eigener Odometriemessung durch eine Art flexible Koppelung verbunden sind [Bor94]. Über die Koppelung kann ein Fahrzeug die relative Position des Anderen berechnen. Wenn nun ein Fahrzeug durch nicht-systematische Fehler beeinflusst wird, können diese durch den Vergleich mit der Position des anderen Fahrzeugs korrigiert werden.

Für die Kalibrierung mobiler Roboter gibt es eine Vielzahl Methoden. Oft wird nur die Odometrie kalibriert und die genaue Kalibrierung der Sensoren, falls vorhanden, wird vorausgesetzt. Ältere Methoden zur Kalibrierung der Odometrie gehen oft so vor, dass eine vorgegebene Route nur mit Hilfe der Odometrie abgefahren und die Roboterposition vor und nach der Fahrt manuell genau bestimmt wird. Dann werden aus der Differenz zwischen erwartetem und tatsächlichem Endpunkt die Kalibrierungsparameter berechnet.

Ein Beispiel für ein solches Verfahren ist UMBmark, das 1996 von Borenstein und Feng vorgestellt wurde [BF96]. Dabei wird zuerst der Fehler, der durch ein zu großes oder zu kleines Mittel der Raddurchmesser entsteht, korrigiert. Dies wird erreicht, indem der Roboter eine vorgegebene Anzahl Radumdrehungen geradeaus fährt und dann die abgefahrene Strecke gemessen wird. Danach fährt der Roboter ein Quadrat mit vorgegebener Seitenlänge ab, einmal im Uhrzeigersinn, einmal gegen den Uhrzeigersinn. Das heißt er fährt viermal jeweils eine bestimmte Strecke geradeaus und dreht dann um 90°. Durch die fehlende Kalibrierung wird der Roboter dabei einen Fehler machen und nicht wieder genau beim Startpunkt ankommen. Abbildung 2.4a zeigt eine mögliche Trajektorie für die Fahrt des Roboters im Uhrzeigersinn. Die tatsächlichen Endpunkte des Roboters werden für beide Fahrtrichtungen relativ zum Startpunkt gemessen. Aus diesen Informationen lassen sich nun das tatsächliche Verhältnis der Raddurchmesser zueinander und der Abstand der Räder bestimmen. Die Genauigkeit der Kalibrierung wird erhöht, wenn das Quadrat in jeder Richtung mehrfach abgefahren wird. Kleine Unterschiede in den Endpositionen ergeben sich aus nicht-systematischen Fehlern. Für die Berechnung der Kalibrierungsparameter wird nun der Schwerpunkt der gemessenen Endpunkte (pro Fahrtrichtung) genutzt (siehe Abbildung 2.4b).

Ein weiteres Verfahren für die Korrektur von Odometriefehlern nutzt ein auf den Boden gemaltes



(a) Beispiel für eine Trajektorie

(b) Endpunkte der Trajektorien mit ihren Schwerpunkten

Abb. 2.4: Die linke Abbildung zeigt eine Trajektorie, wie sie ein Roboter mit einem Linksdrall im UMBmark-Test gefahren sein könnte. Gestrichelt sieht man die optimale Trajektorie. In der rechten Abbildung sieht man die Endpunkte der Robotertrajektorien für Fahrten mit und gegen den Uhrzeigersinn und die jeweiligen Schwerpunkte mit ihrem Abstand vom Nullpunkt.

Gitter [Mä97]. Der Roboter braucht nun einen nach unten zeigenden Sensor, der die Gitterlinien erkennen kann. In dem vorgestellten Verfahren werden mit den Informationen über die überfahrenen Gitterlinien sowohl systematische als auch nicht-systematische Fehler ausgeglichen.

Gemeinsam ist diesen Methoden, dass entweder eine Modifikation der Umgebung vorgenommen werden muss, oder die Position des Roboters vor und nach der Fahrt genau bestimmt werden muss. Neuere Methoden reduzieren den Aufwand, der für die Kalibrierung der Odometrie nötig ist, indem sie die Positionsbestimmung vom Roboter selbst mit Hilfe von Sensoren erledigen lassen. Dabei wird angenommen, dass die Sensoren bereits kalibriert sind. Des Weiteren wird oft eine Karte der Umgebung benötigt [MTTS03, LBAR98]. Ein erstes Verfahren zur Kalibrierung der Odometrie, bei dem keine zusätzlichen Informationen benötigt werden, wurde 1999 von Nicholas Roy und Sebastian Thrun vorgestellt [RT99]. Dabei werden jeweils zwei aufeinander folgende Scans der Sensoren so zueinander ausgerichtet, dass die Umgebung möglichst genau aufeinander passt (Scan Matching). Die Verschiebung und Rotation der zwei Scans kann nun genutzt werden um die Odometrie automatisch zu kalibrieren. Dabei muss keine bestimmte Route abgefahren werden sondern die Kalibrierung kann kontinuierlich während der Fahrt geschehen. Allerdings erstellt der Roboter nicht wie bei SLAM eine komplette Umgebungskarte und es wird auch nicht versucht in den aktuellen Scans bereits früher gescannte Umgebungen wieder zu erkennen (Loop Closing).

Die bis jetzt beschriebenen Methoden kalibrieren alle nur die Odometrie eines Roboters. Viele der Methoden setzen dabei bereits kalibrierte Sensoren voraus. Auch für das Problem der Sensorkalibrierung gibt es einige Methoden. In dieser Arbeit beschränkt sich die Kalibrierung darauf, die Position und Orientierung der Sensoren auf dem Roboter möglichst genau festzustellen. Dabei wird vorausgesetzt, dass die sensorspezifischen Parameter bereits vom Hersteller kalibriert

wurden. Eine recht naheliegende Vorgehensweise bei der Kalibrierung der Sensoren gleicht die Sensordaten während der Fahrt mit einer vorhandenen Karte der Umgebung ab und bestimmt so die Sensorposition relativ zum Roboter [UHS07].

Gianluca Antonelli et al. beschreiben eine Methode, gleichzeitig die Odometrie und die Kamera eines Roboters zu kalibrieren [ACGM10]. Allerdings werden für das Verfahren ein Satz bekannter Landmarken, sowie Fotoaufnahmen dieser Landmarken benötigt. In dieser Arbeit soll ein Verfahren entwickelt werden, dass gleichzeitig Sensoren sowie die Odometrie kalibriert und dabei keinerlei Vorwissen über die Umgebung benötigt. Dafür soll SLAM verwendet werden und das Verfahren so erweitert werden, dass gleichzeitig die Trajektorie und die Kalibrierungsparameter optimiert werden. Das bedeutet, dass die in Kapitel 2.1.1 eingeführten Funktionen für das Bewegungs- und Messmodell g und h nicht fest vorgegeben sind, sondern durch Parameter wie den Abstand der Räder oder die Position des Sensors mit optimiert werden.

Im Jahr 2008 stellten Censi et al. ein Verfahren vor, das gleichzeitig die Kinematik und die Position eines Laserscanners auf einem mobilen Roboter kalibriert und kein Vorwissen über die Umgebung erfordert [CMO08]. Hierbei wird ein Roboter mit Differentialantrieb vorausgesetzt, das heißt die beiden nicht drehbaren Räder können unabhängig voneinander angetrieben werden. Kurven in der Robotertrajektorie kommen durch verschiedene Geschwindigkeiten der Räder zustande. Als Kalibrierungsparameter werden für die Odometrie die Radien der Räder r_L und r_R und der Abstand der Räder von einander b gewählt. Für die Kalibrierung des Sensors werden die horizontale Position des Sensors im Koordinatensystems des Roboters (l_x, l_y) und die horizontale Ausrichtung des Sensors als Winkel zur Längsachse des Roboters l_θ als Parameter gewählt. Dabei wird angenommen, dass der Sensor horizontal auf dem Roboter angebracht ist.

In dem Verfahren wird nun zuerst über Scan Matching eine Schätzung für die Rotation des Roboters berechnet. Diese ist, anders als bei der Translation des Roboters, immer gleich der Rotation des Sensors und damit gleich der Rotation, die aus einem Scan Matching ablesbar ist. Die Rotation des Roboters wiederum hängt von den Parametern der Kinematik r_L, r_R und b ab. Mittels Least-Squares-Optimierung über alle Scan Matches kann damit eine Schätzung für das Verhältnis von r_L und r_R zu b gefunden werden. Nun kann für die Parameter b, l_x, l_y und l_θ ein Optimierungsproblem aufgestellt werden, das mittels der Methode der Lagrange-Multiplikatoren gelöst werden kann. Die Parameter r_L und r_R können dann über die vorher berechneten Verhältnisse zu b berechnet werden. Dass das Verfahrens nur mit konstanten Kalibrierungsparametern umgehen kann, ist ein Schwachpunkt. Zusätzlich kann aufgrund der Nutzung des Scan Matching in der Feststellung der Roboterrotation dazu kommen, dass in großen, verschlungenen Umgebungen keine akkurate Karte gefunden wird [KGB12].

Ein weiteres Verfahren für die simultane Kalibrierung von Odometrie und Sensoren wurde 2012 von Kümmerle et al. vorgestellt [KGB12]. Dabei wird ein graphbasiertes Verfahren für SLAM verwendet, welches gleichzeitig die Kalibrierungsparameter optimiert. Die einzigen benötigten Informationen sind ungefähre initiale Schätzungen für die Parameter. Als Robotermodell wird

hierbei wieder ein mobiler Roboter mit Differentialantrieb und einem fest montierten Laserscanner gewählt. Auch die gewählten Parameter für die Kalibrierung sind die gleichen wie in [CMO08].

In dem Verfahren wird dann ein Hypergraph aufgebaut, in dem jeder Knoten einer Roboterposition, einer Sensorposition oder Odometrieparametern entspricht. Dieser Graph, der allerdings kein Faktorgraph ist, wie er in Kapitel 2.2.3 beschrieben wurde, kann dann für die gemeinsame Optimierung von Karte, Trajektorie und Kalibrierung verwendet werden. Dafür wird das g^{2}o-Toolkit verwendet [KGS$^+$11].

Eine Übersicht über die verschiedenen in diesem Kapitel vorgestellten Methoden mit ihren Eigenschaften findet sich in Tabelle 2.1.

In der vorliegenden Arbeit soll ähnlich zu [KGB12] ein graphisches Optimierungsverfahren für SLAM um die Optimierung der Kalibrierungsparameter erweitert werden. Dabei werden anders als bei Kümmerle et al. Faktorgraphen zur Modellierung und die GTSAM-Toolbox zur Optimierung verwendet. Dies liegt vor allem an der besseren Verwendbarkeit der Toolbox, die sich wiederum aus einer besseren Dokumentation ableitet. Des Weiteren kann das in der vorliegenden Arbeit entwickelte Verfahren und die zugehörige Implementierung für die Kalibrierung von Robotern mit unterschiedlichen Kinematiken und unterschiedlich vielen Laserscannern verwendet werden. Außerdem ist die Implementierung leicht für die Verwendung weiterer Sensortypen anpassbar.

2.4 Einführung in Lie-Gruppen und die spezielle euklidische Gruppe $\mathbf{SE}(2)$

Um die Trajektorie eines Roboters optimieren zu können, wird eine mathematische Beschreibung von Bewegungen und Positionen, inklusive Orientierung, benötigt. Weiter soll es leicht möglich sein, das Ausführen zweier Bewegungen nacheinander zu berechnen. Schließlich müssen für die Optimierung, über die Methode der kleinsten Quadrate, Ableitungen von Bewegungsfunktionen gebildet werden können.

Alle diese Eigenschaften finden sich in einer mathematischen Struktur, den Lie-Gruppen. Eine Lie-Gruppe ist einerseits eine mathematische Gruppe und andererseits eine differenzierbare Mannigfaltigkeit. Im folgenden werden die Definitionen von Gruppen und Mannigfaltigkeiten kurz wiederholt und Beispiele zur Anschauung gegeben.

2.4.1 Lie-Gruppe

Eine Gruppe $G = (M, *)$ bezeichnet in der Mathematik eine Menge von Objekten M mit einer Verknüpfung $*$, die je zwei Elementen der Gruppe ein drittes zuweist: $* : G \times G \mapsto G$. Außerdem

Autoren	Borenstein, Feng	Mächler	Larsen et al.	Martinelli et al.
Quelle	[BF96]	[Mä97]	[LBAR98]	[MTTS03]
Spezielle Umgebung	wenn möglich Zimmerecke	Gitter auf dem Boden	Markierung an der Wand	nein
Benötigtes Vorwissen	nein	Größe der Gitterzellen	Existenz der Markierung	Karte von Landmarken
mauelle Messungen	ja	nein	nein	nein
feste Trajektorie	ja	nein	ja	nein
Kalibrierung der Odometrie	ja	Nur Korrektur der Fehler	ja	ja
Kalibrierung der externen Sensoren	nein	nein	nein	nein
Minimierung nicht systematischer Fehler	nein	ja	nein	ja
kinematische Modelle	Differentialantrieb	Differentialantrieb	Differentialantrieb	Diff. + Synch. Drive
externe Sensoren	keine	zur Erkennung des Gitters	ja	ja

Autoren	Roy, Thrun	Underwood et al.	Antonelli et al.	Censi et al.	Kümmerle et al.
Quelle	[RT99]	[UHS07]	[ACGM10]	[CMO08]	[KGB12]
Spezielle Umgebung	nein	einfach, konstruiert	nein	nein	nein
Benötigtes Vorwissen	nein	Umgebungskarte	Fotos von Landmarken	nein	nein
mauelle Messungen	nein	nein	nein	nein	nein
feste Trajektorie	nein	nein	nein	nein	nein
Kalibrierung der Odometrie	ja	nein	ja	ja	ja
Kalibrierung der externen Sensoren	nein	ja	ja	ja	ja
Minimierung nicht systematischer Fehler	ja	nein	nein	nein	nein
kinematische Modelle	verschiedene	verschiedene	Differentialantrieb	Differentialantrieb	Differentialantrieb
externe Sensoren	ja	Abstandssensor	Kamera	Abstandssensor	Abstandssensor

Tab. 2.1: Die Tabelle zeigt die betrachteten Methoden zur Kalibrierung mobiler Roboter mit ihren Eigenschaften in der Reihenfolge, in der sie im Text besprochen werden.

müssen folgende drei Eigenschaften erfüllt sein:

- Assoziativität: Für $a,b,c \in G$ gilt $(a*b)*c = a*(b*c)$.
- Neutrales Element: Es existiert ein Element $e \in G$, so dass für alle $a \in G$ gilt: $a*e = e*a = a$.
- Inverses Element: Zu jedem Element $a \in G$ existiert ein Element $a^{-1} \in G$, so dass $a*a^{-1} = a^{-1}*a = e$.

Ein einfaches Beispiel für eine Gruppe ist die Menge der ganzen Zahlen mit der Addition als Verknüpfung $(\mathbb{Z},+)$. Die einzelnen Eigenschaften lassen sich leicht überprüfen und es stellt sich heraus, dass 0 das neutrale Element und $-a$ das inverse Element zu $a \in \mathbb{Z}$ ist.

Eine n-dimensionale Mannigfaltigkeit M ist ein topologischer Raum, so dass für jeden Punkt $p \in M$ eine Umgebung von p existiert, die homöomorph zu einer offenen Teilmenge des $\mathbb{R}^n$ ist. Homöomorph sind zwei Räume M und N dabei, wenn eine bijektive Funktion $\phi : M \mapsto N$ existiert für die sowohl ϕ als auch ϕ^{-1} stetig sind.

Anschaulich gesprochen heißt das, dass eine Mannigfaltigkeit lokal dem euklidischen Raum $\mathbb{R}^n$ ähnelt. Als Beispiel dient die Oberfläche einer Kugel, die in einer kleinen Umgebung eines Punktes der Ebene $\mathbb{R}^2$ ähnelt, allerdings global nicht durch eine Karte in der Ebene darstellbar ist (dafür muss die Kugeloberfläche an einer Stelle aufgeschnitten werden).

Den Homöomorphismus ϕ, der eine offene Teilmenge der Mannigfaltigkeit M auf den $\mathbb{R}^n$ abbildet, wird auch Karte genannt. Ein Atlas ist nun eine Menge von Karten, so dass für jeden Punkt $p \in M$ mindestens eine Karte existiert, die p auf den $\mathbb{R}^n$ abbildet. Über zwei Karten ϕ und *psi* kann nun mit $\phi \circ \psi^{-1}$ eine Funktion definiert werden, die einen Kartenwechsel beschreibt. Sind alle diese Kartenwechselabbildungen eines Atlas beliebig oft differenzierbar, heißt die zugehörige Mannigfaltigkeit ebenfalls differenzierbare Mannigfaltigkeit.

Mit Hilfe dieser Eigenschaft können nun auch Konzepte wie Ableitungen auf der Mannigfaltigkeit definiert werden.

Eine Lie-Gruppe G ist eine Teilmenge des $\mathbb{R}^n$ mit folgenden Eigenschaften:

- $(G,\cdot)$ ist eine Gruppe.
- G ist eine differenzierbare Mannigfaltigkeit.
- Die Gruppenverknüpfung $\cdot : G \times G \mapsto G$ und die Funktion zur Inversenbildung $^{-1} : G \mapsto G$ sind beliebig oft differenzierbar.

Eine mathematisch tiefergehende Einführung in Lie-Gruppen findet sich in [Gal11]. In dieser Arbeit wird sich darauf beschränkt, die für die Berechnungen in Kapitel 3 nötigen Begriffe kurz einzuführen.

Ein Beispiel für eine Lie-Gruppe sind die invertierbaren $n \times n$-Matrizen [Sel04]. Mit der Matrixmultiplikation bilden diese offensichtlich eine Gruppe. Weiter sind die Matrixmultiplikation sowie Invertierung differenzierbar. Die Mannigfaltigkeit ist einfach eine offene Teilmenge in $\mathbb{R}^{n^2}$. Diese Lie-Gruppe wird auch allgemeine lineare Gruppe genannt und mit $\mathrm{GL}(n)$ bezeichnet.

2.4.2 Lie-Algebra

Eine Lie-Algebra $\mathfrak{g}$ ist ein Vektorraum über einem Körper K zusammen mit einer Verknüpfung $[\cdot,\cdot] : \mathfrak{g} \times \mathfrak{g} \mapsto \mathfrak{g}$, Lie-Klammer genannt, sodass für Elemente $a, b \in K$ und $x, y, z \in \mathfrak{g}$ gilt:

- Jacobi-Identität: $[z,[x,y]] = [[z,x]y] + [x,[z,y]]$
- Bilinearität: $[ax+by,z] = a[x,z] + b[y,z]$ und $[z,ax+by] = a[z,x] + b[z,y]$
- $[x,x] = 0$

Ein Tangentialraum T_pM einer differenzierbaren Mannigfaltigkeit M an einem Punkt $p \in M$ ist ein Vektorraum, der M an dem Punkt p linear approximiert. Der Tangentialraum wird von den Vektoren $v = \frac{\partial \gamma}{\partial x}(0) \in T_pM$ aufgespannt, wobei $\gamma : (-\varepsilon, \varepsilon) \mapsto M$ eine differenzierbare Kurve mit $\gamma(0) = p$ ist.

Die Lie-Algebra $\mathfrak{g}$ zu einer Lie-Gruppe G ist immer gleich dem Tangentialraum an dem neutralen Element e von G [Bla10]:

$$\mathfrak{g} = T_eG. \tag{2.11}$$

Die Exponentialabbildung $\exp : \mathfrak{g} \mapsto G$ ist eine Abbildung, die einem Element der Lie-Algebra $\mathfrak{g}$ ein Element der Lie-Gruppe G zuordnet. Ihre Inverse wird, falls sie existiert, mit $\log : G \mapsto \mathfrak{g}$ bezeichnet. Für Matrixgruppen wie $\mathrm{GL}(n)$ ist die Exponentialabbildung gerade das Matrixexponential definiert als

$$e^A = \sum_{k=0}^{\infty} \frac{A^k}{k!} \quad \text{für } A \in \mathbb{R}^{n \times n}. \tag{2.12}$$

Eine Lie-Gruppe G kann auch dargestellt werden als Gruppe von linearen Transformationen über der Lie-Algebra $\mathfrak{g}$. Dafür wird durch eine Abbildung jedem Element $g \in G$ eine Funktion $\mathrm{Adj}_g : \mathfrak{g} \mapsto \mathfrak{g}$ zugewiesen. Für Matrixgruppen ist diese adjungierte Darstellung für $g \in G$ und $a \in \mathfrak{g}$ wie folgt definiert:

$$\mathrm{Adj}_g(a) = gag^{-1} \tag{2.13}$$

2.4.3 Die spezielle euklidische Gruppe $\mathbf{SE}(2)$

Bis jetzt wurde die Pose eines Roboters in dieser Arbeit durch seine Koordinaten im $\mathbb{R}^2$ und seine Orientierung in Form eines Winkels angegeben. Nun wird die Pose des Roboters einfach als der Endzustand des Roboters nach einer Bewegung vom Ursprung des Koordinatensystems aus betrachtet. Diese Bewegung kann zwar auch einfach durch die Endpose kodiert werden, es ist jedoch vorteilhaft, sie statt dessen als Abbildung $A : \mathbb{R}^2 \mapsto \mathbb{R}^2$ zu verstehen, die alle Punkte des Roboters auf ihre neue Position abbildet.

2.4.3.1 Bewegungen eines Roboters mathematisch beschrieben

Es fällt auf, dass sich die Bewegungen zu allen möglichen Endposen des Roboters immer durch eine Translation und eine Rotation beschreiben lassen. Die neue Orientierung des Roboters ist dabei implizit durch die Rotation in der Abbildung gegeben. Mathematisch lässt sich so eine Abbildung am besten über homogenen Koordinaten beschreiben. Jeder Punkt $(a_x, a_y) \in \mathbb{R}^2$ wird dabei als Vektor $\begin{pmatrix} a_x \\ a_y \\ 1 \end{pmatrix}$ beschrieben. Dann lässt sich eine Verschiebung um den Vektor $\begin{pmatrix} t_x \\ t_y \end{pmatrix}$ durch folgende Abbildung beschreiben:

$$\begin{pmatrix} a_x \\ a_y \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_x \\ a_y \\ 1 \end{pmatrix} = \begin{pmatrix} a_x + t_x \\ a_y + t_y \\ 1 \end{pmatrix} \quad [2.14]$$

Eine Rotation um den Ursprung mit dem Winkel θ kann wie folgt dargestellt werden:

$$\begin{pmatrix} a_x \\ a_y \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_x \\ a_y \\ 1 \end{pmatrix} = \begin{pmatrix} a_x\cos\theta - a_y\sin\theta \\ a_x\sin\theta + a_y\cos\theta \\ 1 \end{pmatrix} \quad [2.15]$$

Wobei $R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$ die Rotationsmatrix für den Winkel θ genannt wird. Alle Rotationsmatrizen im $\mathbb{R}^2$ bilden zusammen die so genannte spezielle orthogonale Gruppe $\mathrm{SO}(2)$, auch Kreisgruppe genannt.

Zusammengenommen lässt sich eine Bewegung des Roboters dann durch eine 3×3-Abbildungsmatrix beschreiben, die der Pose $\begin{pmatrix} t_x & t_y & \theta \end{pmatrix}^T$ entspricht:

$$\begin{pmatrix} a_x \\ a_y \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_x \\ a_y \\ 1 \end{pmatrix} = \begin{pmatrix} R_\theta & t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} a_x \\ a_y \\ 1 \end{pmatrix} = \begin{pmatrix} a_x\cos\theta - a_y\sin\theta + t_x \\ a_x\sin\theta + a_y\cos\theta + t_y \\ 1 \end{pmatrix} \quad [2.16]$$

Ein Vorteil dieser mathematischen Beschreibung ist nun, dass das Ausführen mehrerer Bewegungen nacheinander einfach durch Matrixmultiplikation der jeweiligen Abbildungsmatrizen berechnet werden kann. Allgemeiner gesagt bilden die Matrizen dieser Form zusammen mit der Matrixmultiplikation eine Gruppe und wie im nächsten Abschnitt gezeigt wird, sogar eine Lie-Gruppe genannt $\mathrm{SE}(2)$.

2.4.3.2 Die Lie-Gruppe $\mathbf{SE}(2)$

In diesem Kapitel wird gezeigt, dass die Menge

$$\mathrm{SE}(2) = \left\{ \begin{pmatrix} R_\theta & t \\ 0 & 1 \end{pmatrix} \in \mathbb{R}^{3\times 3} \mid t \in \mathbb{R}^2, R_\theta \in \mathrm{SO}(2) \right\} \tag{2.17}$$

eine Lie-Gruppe ist.

Satz 1. *Die Menge $SE(2)$ bildet zusammen mit der Matrixmultiplikation eine Gruppe.*

Beweis. Um dies zu beweisen wird zuerst gezeigt, dass die Menge unter der Multiplikation abgeschlossen ist, das heißt, dass das Produkt wiederum ein Element der Menge ist. Seien dafür

$$A = \begin{pmatrix} R_\alpha & a_t \\ 0 & 1 \end{pmatrix}, B = \begin{pmatrix} R_\beta & b_t \\ 0 & 1 \end{pmatrix} \in \mathrm{SE}(2) \quad \text{mit } a_t = \begin{pmatrix} a_x \\ a_y \end{pmatrix} \text{ und } b_t = \begin{pmatrix} b_x \\ b_y \end{pmatrix}.$$

Es gilt

$$\begin{pmatrix} R_\alpha & a_t \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} R_\beta & b_t \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} R_\alpha \cdot R_\beta & R_\alpha b_t + a_t \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} R_{\alpha+\beta} & R_\alpha b_t + a_t \\ 0 & 1 \end{pmatrix}. \tag{2.18}$$

Offensichtlich liegt damit auch das Produkt zweier Elemente aus $\mathrm{SE}(2)$ wieder in $\mathrm{SE}(2)$. Dass die Matrixmultiplikation außerdem assoziativ ist, ist allgemein bekannt. Das neutrale Element der Gruppe ist die Einheitsmatrix I_3:

$$\begin{pmatrix} R_\alpha & a_t \\ 0 & 1 \end{pmatrix} \cdot I_3 = I_3 \cdot \begin{pmatrix} R_\alpha & a_t \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} R_\alpha & a_t \\ 0 & 1 \end{pmatrix} \tag{2.19}$$

Auch das inverse Element lässt sich einfach berechnen:

$$\begin{pmatrix} R_\alpha & a_t \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} R_{-\alpha} & -R_{-\alpha} a_t \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} R_{-\alpha} & -R_{-\alpha} a_t \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} R_\alpha & a_t \\ 0 & 1 \end{pmatrix} = I_3 \tag{2.20}$$

Damit sind alle Bedingungen einer Gruppe erfüllt. □

Dass $\mathrm{SE}(2)$ auch eine differenzierbare Mannigfaltigkeit ist, wird hier nicht explizit beweisen, allerdings kann dieses Ergebnis auch direkt aus einem Theorem ableitet werden, welches besagt, dass eine abgeschlossene Untergruppe von $\mathrm{GL}(n)$ eine Untermannigfaltigkeit ist [War71, Theorem 3.42]. Da, wie schon bei $\mathrm{GL}(n)$ bemerkt, die Matrixmultiplikation so wie die Matrixinvertierung beliebig oft differenzierbar sind, ist damit $\mathrm{SE}(2)$ eine Lie-Gruppe.

2.4.3.3 Die Lie-Algebra $\mathfrak{se}(2)$

Nach der Definition der Lie-Algebra ist diese isomorph zu dem Tangentialraum von $\mathrm{SE}(2)$ an dem neutralen Element I_3. Der Tangentialraum wird wiederum von folgenden Vektoren aufgespannt

$$v = \frac{\partial \gamma}{\partial x}(I_3) \in T_{I_3}\mathrm{SE}(2), \tag{2.21}$$

wobei $\gamma : (-\varepsilon, \varepsilon) \mapsto \mathrm{SE}(2)$ eine differenzierbare Kurve mit $\gamma(0) = I_3$ ist. Diese Vektoren können nun für drei Kurven bestimmt werden und es zeigt sich, dass sie den Tangentialraum $T_{I_3}\mathrm{SE}(2)$ und damit die Lie-Algebra $\mathfrak{se}(2)$ aufspannen.

$$\gamma_1 : a_x \mapsto \begin{pmatrix} 1 & 0 & a_x \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \frac{\partial \gamma_1}{\partial a_x}(0) = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{2.22}$$

$$\gamma_2 : a_y \mapsto \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & a_y \\ 0 & 0 & 1 \end{pmatrix}, \quad \frac{\partial \gamma_2}{\partial a_y}(0) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \tag{2.23}$$

$$\gamma_3 : \alpha \mapsto \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} R_\alpha & 0 \\ 0 & 1 \end{pmatrix}, \quad \frac{\partial \gamma_2}{\partial \alpha}(0) = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{2.24}$$

Beachte dabei, dass I_3 das Element ist, das der Pose mit $a_x = a_y = a_\theta = 0$ entspricht, weshalb die Ableitung jeweils an der Stelle 0 ausgewertet wird. Die berechneten Matrizen werden auch Erzeuger der Lie-Algebra genannt und mit G_1 bis G_3 bezeichnet.

Elemente aus $\mathfrak{se}(2)$ haben nun die Form

$$s_x G_1 + s_y G_2 + \sigma G_3 = \begin{pmatrix} 0 & -\sigma & s_x \\ \sigma & 0 & s_y \\ 0 & 0 & 0 \end{pmatrix} \quad \text{mit} \quad \begin{pmatrix} s_x \\ s_y \\ \sigma \end{pmatrix} \in \mathbb{R}^3. \tag{2.25}$$

Da ein Element aus $\mathfrak{se}(2)$ vollständig durch den Vektor $\begin{pmatrix} s_x & s_2 & \sigma \end{pmatrix}^T$ bestimmt ist, werden an Stelle des Elements oft auch einfach die Faktoren der Erzeuger in Form eines Vektor angegeben. Dann wird kurz $\begin{pmatrix} s_x & s_2 & \sigma \end{pmatrix}^T \in \mathfrak{se}(2)$ geschrieben.

Für $A, B \in \mathfrak{se}(2)$ wird die Lie-Klammer als $[A,B] = AB - BA$ definiert. Es lässt sich leicht nachprüfen, dass diese Definition alle Bedingungen der Lie-Klammer erfüllt.

Im vorigen Kapitel wurde für eine Lie-Gruppe die adjungierte Darstellung eingeführt. Diese weist einem Gruppenelement A die Funktion

$$\mathrm{Adj}_A : \mathfrak{se}(2) \mapsto \mathfrak{se}(2) \quad \text{mit } \mathrm{Adj}_A(S) = ASA^{-1} \tag{2.26}$$

zu.

Sei nun

$$A = \begin{pmatrix} R_\alpha & a_t \\ 0 & 1 \end{pmatrix} \in \mathrm{SE}(2) \quad \text{und} \quad s = \begin{pmatrix} s_x \\ s_y \\ \sigma \end{pmatrix}, \quad S = \begin{pmatrix} 0 & -\sigma & s_x \\ \sigma & 0 & s_y \\ 0 & 0 & 0 \end{pmatrix} \in \mathfrak{se}(2). \tag{2.27}$$

Dann gilt

$$\begin{aligned} \mathrm{Adj}_A(S) = A \cdot S \cdot A^{-1} &= \begin{pmatrix} R_\alpha & a_t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & -\sigma & s_x \\ \sigma & 0 & s_y \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} R_\alpha^T & -R_\alpha^T \cdot a_t \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} R_\alpha s_t + \sigma \begin{pmatrix} a_y \\ -a_x \end{pmatrix} \\ \sigma \end{pmatrix} = \begin{pmatrix} R_\alpha & \begin{matrix} a_y \\ -a_x \end{matrix} \\ 0 \quad 0 & 1 \end{pmatrix} \cdot s \end{aligned} \tag{2.28}$$

Die Funktion $\mathrm{Adj}_A(S)$ kann also auch als Matrix dargestellt werden, die hier mit Ad_A bezeichnet wird:

$$\mathrm{Adj}_A(S) = \mathrm{Ad}_A \cdot s \quad \text{mit} \quad \mathrm{Ad}_A = \begin{pmatrix} R_\alpha & \begin{matrix} a_y \\ -a_x \end{matrix} \\ 0 \quad 0 & 1 \end{pmatrix}. \tag{2.29}$$

Ein Element aus der Lie-Algebra kann nun mit der Exponentialabbildung auf ein Element der Lie-Gruppe abgebildet werden. Da $\mathrm{SE}(2)$ eine Matrixgruppe ist, entspricht die Exponentialabbildung gerade dem Matrixexponential, das in geschlossener Form berechnet werden kann.

Sei $S = \begin{pmatrix} 0 & -\sigma & s_x \\ \sigma & 0 & s_y \\ 0 & 0 & 0 \end{pmatrix}$ ein Element der Lie-Algebra $\mathfrak{se}(2)$. Dann gilt

$$\begin{aligned} e^S &= \exp \begin{pmatrix} 0 & -\sigma & s_x \\ \sigma & 0 & s_y \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} R_\sigma & V \cdot s \\ 0 & 1 \end{pmatrix} \quad \text{mit} \quad V = \frac{1}{\sigma} \begin{pmatrix} \sin\sigma & -(1-\cos\sigma) \\ 1-\cos\sigma & \sin\sigma \end{pmatrix} \\ &= \begin{pmatrix} \cos\sigma & -\sin\sigma & \frac{1}{\sigma}(s_x \sin\sigma - s_y(1-\cos\sigma)) \\ \sin\sigma & \cos\sigma & \frac{1}{\sigma}(s_x(1-\cos\sigma) + s_y \sin\sigma) \\ 0 & 0 & 1 \end{pmatrix} \in \mathrm{SE}(2) \end{aligned} \tag{2.30}$$

Die Berechnung der geschlossenen Form des Matrixexponentials kann mit Hilfe der Gleichung $S^3 = -\sigma^2 \cdot S$ und der Reihenentwicklung von Sinus und Kosinus nachgerechnet werden.

2.5 Zusammenfassung

In diesem Kapitel wurden viele der für diese Arbeit relevanten Begriffe eingeführt und die weitere Vorgehensweise motiviert. Da die Kalibrierung in den SLAM-Prozess integriert werden soll, wurden das SLAM-Problem und verschiedene Ansätze zur seiner Lösung vorgestellt. Dabei wurde zwischen offline und online SLAM unterschieden, wobei Online-SLAM-Verfahren meist ein leichteres Problem lösen, aber dafür auch in Echtzeit berechnet werden können. Der historisch gesehen erste Lösungsansatz für SLAM funktioniert mit Kalman-Filtern, hat heute aber fast nur noch theoretische Bedeutung, da der Rechenaufwand im Vergleich zu anderen Verfahren viel zu hoch ist. FastSLAM, das einen partikelbasierten Ansatz verfolgt, hat den Vorteil, dass es gleichzeitig das Offline- und das Online-SLAM-Problem löst. Der graphbasierte Ansatz wiederum skaliert sehr gut mit großen Umgebungen, war allerdings lange nur offline lösbar. Mittlerweile gibt es allerdings auch Verfahren, die sich mit kleineren Einschränkungen auch für den Online-Einsatz eignen [KRD08].

Des Weiteren wurden die Vorteile einer möglichst exakten Kalibrierung mobiler Roboter erläutert, wobei festgestellt wurde, dass die Kalibrierung aufgrund sich mit der Zeit ändernder Einflussfaktoren möglichst häufig durchgeführt werden sollte. Dies spricht dafür den Kalibrierungsaufwand möglichst weit zu minimieren und befürwortet damit automatische Kalibrierungsverfahren, wie sie in dieser Arbeit verwendet werden. Es wurde aber auch festgestellt, dass sich nicht alle Fehler durch Kalibrierung beheben lassen, sondern dass nur die systematischen Fehler minimiert werden können.

Bei der Beschreibung verschiedener bereits bestehender automatischer Kalibrierungsverfahren fiel auf, dass diese jeweils für spezielle Roboterkonfigurationen aus Kinematik und Sensoren entwi-

ckelt wurden und dass ein allgemeines Kalibrierungs-Framework bis jetzt fehlt.

In der mathematischen Modellierung der automatischen Kalibrierung in Kapitel 3 müssen zusammengesetzte Roboterbewegungen berechnet und Funktionen von Roboterbewegungen abgeleitet werden. Eine mathematische Struktur, die das ermöglicht, ist die Lie-Gruppe. In diesem Kapitel wurden deshalb Lie-Gruppen formal eingeführt und für die Menge der möglichen Roboterbewegungen gezeigt, das sie eine Lie-Gruppe bilden. Des Weiteren wurde die zugehörige Lie-Algebra bestimmt, die für die Bestimmung von Ableitungen gebraucht wird.

3 Konzept

In diesem Kapitel wird ein graphbasiertes Lösungsverfahren für das SLAM-Problem so erweitert, dass gleichzeitig die Sensor- und Odometrieparameter kalibriert werden. Im ersten Abschnitt werden die verschiedenen in Kapitel 2 eingeführten Ansätze zur Lösung des SLAM-Problems verglichen und die Entscheidung für das graphbasierte Verfahren begründet. Für die Modellierung des Problems werden in dieser Arbeit Faktorgraphen verwendet [KF09]. Dies sind bipartite Graphen, in denen jeder Knoten entweder eine Zufallsvariable oder stochastische Abhängigkeiten zwischen Zufallsvariablen repräsentiert. Dieses Kapitel beschäftigt sich mit der Frage, wie die Kalibrierung von verschiedenen Sensoren und Odometrien in einem Faktorgraph abgebildet werden kann. Dafür werden die nötigen stochastischen Informationen berechnet. In Kapitel 4 werden diese dann in der Implementierung verwendet.

3.1 Vergleich und Bewertung der verschiedenen SLAM-Verfahren

In Kapitel 2.2 wurden im Wesentlichen drei verschiedene Ansätze für die Lösung des SLAM-Problems vorgestellt: Erweiterte Kalman-Filter, Partikelfilter und graphbasierte Algorithmen. Für jeden der verschiedenen Ansätze gibt es wiederum unterschiedliche Implementierungen und unterschiedliche Erweiterungen.

In dieser Arbeit fällt die Wahl auf einen graphbasierten Ansatz, und eine Implementierung mit dem GTSAM-Framework. Die Entscheidung für den graphbasierten Ansatz wird in diesem Kapitel kurz begründet. Zur Implementierung standen zwei Frameworks zur Auswahl, g2o und GTSAM. Da beide Frameworks einen ähnlichen Funktionsumfang bieten, gab die umfänglichere Dokumentation den Ausschlag für das GTSAM-Framework entschieden. Eine kurze Beschreibung des Frameworks befindet sich in Kapitel 4.4.

Der Ansatz, das SLAM-Problem mit einem erweiterten Kalman-Filter zu lösen, ist heutzutage wegen seines schlechten Laufzeitverhaltens veraltet. Er wurde vor allem in den ersten Implementierungen für das SLAM-Problem ab 1990 verwendet. Dabei wurde das Online-SLAM-Problem gelöst, wofür zu jedem Zeitschritt eine gemeinsame Kovarianzmatrix berechnet werden muss, deren Größe quadratisch mit der Anzahl der verwendeten Landmarken wächst. Für große Umgebungen ist der Algorithmus daher praktisch nicht verwendbar. Zwar gibt es erfolgreiche Reduktionen der Laufzeit dadurch, dass größere Umgebungen in kleinere Abschnitte aufgeteilt und diese erst am Ende des Algorithmus verschmolzen werden [Bai02, Kapitel 6], der erweiterte Kalman-Filter

wird aber heutzutage trotzdem kaum mehr verwendet.

Bei einer potentiellen Erweiterung für die Kalibrierung von Sensoren wäre ein weiteres Problem, dass der erweiterte Kalman-Filter ein reiner Filter ist und nicht über die gesamte Trajektorie glättet. In jedem Schritt wird immer nur die aktuell wahrscheinlichste Roboterpose berechnet, ohne dass dieses Ergebnis die vorher berechneten Schätzungen beeinflusst. Im Fall der Kalibrierung ist letzteres aber unbedingt nötig, da am Ende die wahrscheinlichste Trajektorie unter für alle Zeitpunkte gleich optimierten Kalibrierungsparametern ausgegeben werden soll. Die Kalibrierungsparameter müssen dabei im Fall einer online Berechnung fortlaufend aktualisiert und auch auf die gesamte bereits berechnete Trajektorie angewendet werden.

Nach Ausscheiden des erweiterten Kalman-Filters aus den oben beschriebenen Gründen, bleiben noch der graphbasierte Ansatz und der Ansatz mit Partikelfiltern übrig. Beide Methoden sind aktuell Stand der Technik und können Probleme in großen Umgebungen effizient lösen. Es gibt aber weitere Unterscheidungsmerkmale, die in dieser Arbeit letztlich zur Entscheidung für den graphbasierten Ansatz führen.

Ein großer Unterschied besteht darin, dass graphbasierte Ansätze das Offline-SLAM-Problem lösen und in der Regel auch offline verwendet werden. Es gibt allerdings schon erste Verfahren, die so effizient sind, dass auch die Verwendung online möglich ist [KJR+12]. Partikelfilter lösen hingegen gleichzeitig das offline sowie das online-SLAM-Problem, da einerseits zu jedem Zeitpunkt die nächste Roboterpose nur von der aktuellen Pose abhängt aber andererseits jeder Partikel die gesamte Robotertrajektorie enthält. Da in dieser Arbeit, wie in Kapitel 4.1 zu sehen ist, ein offline ablaufender Algorithmus gewählt wird, ist diese Restriktion des graphbasierten Ansatzes hier nicht wesentlich. Der Fokus dieser Arbeit liegt ohnehin primär darauf, einen vielseitig einsetzbaren Algorithmus zur automatischen Kalibrierung zu entwickeln. Laufzeiten spielen dabei erst einmal eine sekundäre Rolle, da die Kalibrierung normalerweise nicht andauernd, sondern nur ab und zu und bei geänderten Umgebungsbedingungen durchgeführt werden muss.

Für den Partikelfilter spricht die Möglichkeit, verschiedene Lösungen für das Korrespondenzproblem zuzulassen. So kann für einen Partikel der Scan Matcher ein anderes Ergebnis liefern als für einen anderen Partikel. Beide Hypothesen können so weiter verfolgt werden, während sich im graphbasierten Fall entschieden werden muss, welches Ergebnis in der Wahrscheinlichkeitsverteilung einer Kante des Graphen repräsentiert wird. Ein Nachteil der Partikelfilter-Implementierung FastSLAM ist hingegen, dass besonders bei langen Trajektorien mit mehreren Schleifen die Anzahl der benötigten Partikel sehr groß werden kann [TL08].

Ein Hauptkriterium bei der Wahl für einen der Ansätze war natürlich, wie leicht sich der Ansatz für die automatische Kalibrierung erweitern lässt. Im graphbasierten Fall werden Wahrscheinlichkeitsverteilungen über Zufallsvariablen durch einen Graph dargestellt. Dabei stellt beispielsweise eine Kante zwischen zwei aufeinander folgenden Roboterposen die Abhängigkeit der zweiten Pose von der ersten Pose und der Odometriemessung dar. Soll der Ansatz nun um die Kalibrierung

der Odometrie erweitert werden, muss der Kante nur eine weitere Abhängigkeit von den Kalibrierungsparametern, ebenfalls durch eine Zufallsvariable modelliert, hinzugefügt werden. Die Kante repräsentiert dann die neue Wahrscheinlichkeitsverteilung, die die Kalibrierungsparameter berücksichtigt. Der Rest des Graphen bleibt davon unberührt. In diesem Fall beschränkt sich die Erweiterung um die Kalibrierung also darauf, die neuen Wahrscheinlichkeitsverteilungen zu modellieren.

Die Schnelligkeit des auf einem Partikelfilter beruhenden Ansatzes liegt darin begründet, dass zu jedem Zeitschritt bei einer Odometriemessung für jeden Partikel die neue Roboterpose nur auf Basis der letzten Pose und des Bewegungsmodells bestimmt zu werden braucht. Bei einer Umgebungsmessung wird die Wahrscheinlichkeit der neuen Messung für jeden Partikel auf Grund der Umgebung und der aktuellen Roboterpose bestimmt. Werden dem Partikel Kalibrierungsparameter für Odometrie und Umgebungssensoren hinzugefügt, so müssen auch diese Parameter in jedem Zeitschritt neu gesamplet werden. Für die Odometriekalibrierung hängen die Parameter dabei von allen bisher erfolgten Odometriemessungen und der gesamten in einem Partikel enthaltenen Robotertrajektorie ab. Würde bei einem Update nur die letzte und die aktuelle Roboterpose in die Berechnung mit einbezogen werden, so müsste danach die gesamte Trajektorie, unter Berücksichtigung der neuen Kalibrierungsparameter, neu gesamplet werden. In dem FastSLAM-Algorithmus, einer populären Implementierung basierend auf Partikelfiltern, kann argumentiert werden, dass obwohl in jedem Schritt nur abhängig von der letzten Roboterpose gesamplet wird, dies trotzdem äquivalent zum Samplen direkt aus der Posteriori-Verteilung für full SLAM ist [MTKW02]. Ob dies auch bei der Erweiterung für Kalibrierung möglich ist, scheint zumindest fragwürdig zu sein.

Auf der Grundlage der beschriebenen Vor- und Nachteile der beiden Ansätze fiel in dieser Arbeit die Entscheidung für einen graphbasierten Ansatz. Da bezüglich der meisten Kriterien beide Methoden ähnlich gut geeignet sind, war die Komplexität der Anpassung für die Kalibrierung ausschlaggebend.

3.2 Graphische Modellierung des SLAM-Prozesses und der Kalibrierungsparameter

Faktorgraphen, die in dieser Arbeit zur graphischen Modellierung des SLAM-Problems verwendet werden, sind bipartite Graphen. Eine Art von Knoten (in den Abbildungen als Kreis dargestellt) repräsentieren Zufallsvariablen, zum Beispiel eine Roboterpose zu einem bestimmten Zeitpunkt. Die andere Art von Knoten (in den Abbildungen als Viereck dargestellt) werden Faktoren genannt und repräsentieren aus Messungen gewonnene oder a-priori bekannte stochastische Informationen über diese Zufallsvariablen. Der Knoten einer Zufallsvariable ist nun genau dann durch eine Kante mit einem Faktor verbunden, wenn dieser eine Information über diese Zufallsvariable enthält. Eine tiefergehende Einführung in Faktorgraphen und die Implementierung einiger Beispiele mit dem

GTSAM-Framework finden sich in [Del12].

Werden Sensoren für Umgebungsmessungen erst einmal außen vor gelassen, zeigt Abbildung 3.1 eine mögliche Modellierung der Roboterbewegungen in einem Faktorgraph. Die Knoten x_1 bis x_4 repräsentieren dabei Zufallsvariablen für die Roboterposen zu den Zeitpunkten $t = 1$ bis 4. Der Faktor f_1 modelliert die A-Priori-Wahrscheinlichkeit für die erste Roboterpose. Die restlichen Faktoren f_2 bis f_4 beschreiben die, sich aus den gemessenen Odometriewerten o_1 bis o_3 ergebende, Abhängigkeit von der jeweils vorhergehenden Pose. Genauer gesagt stehen die Faktoren für eine Measurement Likelihoodfunktion:

$$f(x_i, x_{i+1}) = L(x_i, x_{i+1}; o_i). \tag{3.1}$$

Der gesamte Graph beschreibt nun in faktorisierter Form eine Funktion $f(X)$, die zur gemeinsamen Wahrscheinlichkeitsverteilung aller Zufallsvariablen proportional ist:

$$P(X|O) = P(x_1, x_2, x_3, x_4|o_1, o_2, o_3) \propto f(X) = f(x_1) f(x_1, x_2; o_1) f(x_2, x_3; o_2) f(x_3, x_4; o_3). \tag{3.2}$$

Es soll nun die wahrscheinlichste Menge von Roboterposen $X = \{x_1, x_2, x_3, x_4\}$ und damit die wahrscheinlichste Robotertrajektorie $X = \text{argmax}_X P(X|O)$ gefunden werden. Dies entspricht einer Glättung im Gegensatz zu Filtern, die bei vielen inkrementellen Verfahren verwendet werden. Mit dieser Optimierung wird nicht nur inkrementell immer die wahrscheinlichste letzte Roboterposition betrachtet sondern die wahrscheinlichste Gesamt-Trajektorie. Dadurch dass die Roboterposen, die in die Faktoren eingehen, einen Parameter für die Orientierung enthalten, gehen nichtlineare Terme in die Verteilungen der Faktoren ein. Zur Optimierung müssen daher nichtlineare Optimierungsverfahren wie der Levenberg-Marquardt-Algorithmus verwendet werden.

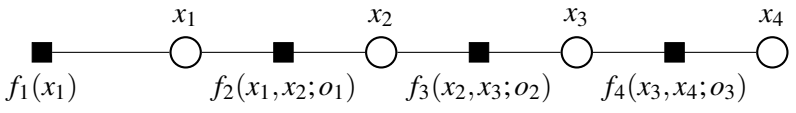


Abb. 3.1: Beispiel eines Faktorgraphen für einen Roboter, der seine Position nur mit Hilfe von Odometrie bestimmen kann [Del12].

Werden nun auch externe Sensoren betrachtet, kommen weitere Faktoren hinzu. In Abbildung 3.2 ist ein Faktorgraph für einen mobilen Roboter mit zwei Laserscannern dargestellt. In diesem Beispiel wird zu jedem Zeitpunkt t nur ein Scan durchgeführt. Dabei werden abwechselnd die beiden Laserscanner verwendet.

Um die Scans zu einer Karte der Umgebung zusammenzusetzen wird für jeden Scan ein Match zu dem vorigen Scan des selben Sensors gesucht. Wird ein Match gefunden, kann aus der Verschiebung und Verdrehung der Scans zueinander die wahrscheinliche Bewegung des Roboters zwischen den beiden Zeitpunkten berechnet werden. Diese Informationen werden nun als zusätzliche Faktoren f_7 bis f_{10} in den Faktorgraphen aufgenommen.

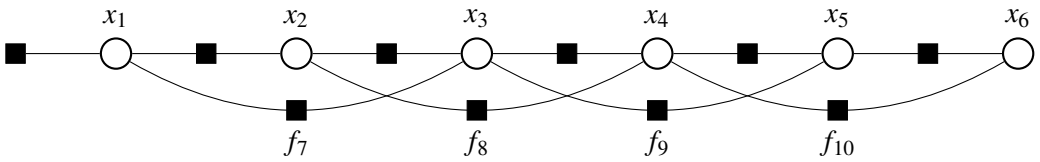


Abb. 3.2: Faktorgraph für einen Roboter, mit zwei Laserscannern, die abwechselnd messen.

Dabei muss allerdings beachtet werden, dass die aus den Scans berechnete Bewegung die Bewegung des Laserscanners wiedergibt. Diese muss nicht unbedingt gleich der Bewegung des Roboters sein, wie ein Beispiel in Abbildung 3.3 zeigt. Als Bewegung eines Roboters wird immer die Bewegung eines festen Punkts auf dem Roboter angegeben. Die Sensoren haben relativ zu diesem Punkt eine feste Pose (Koordinaten und Orientierung). Diese Rototranslation muss noch aus der Bewegung herausgerechnet werden, bevor sie in die Faktoren eingehen kann.

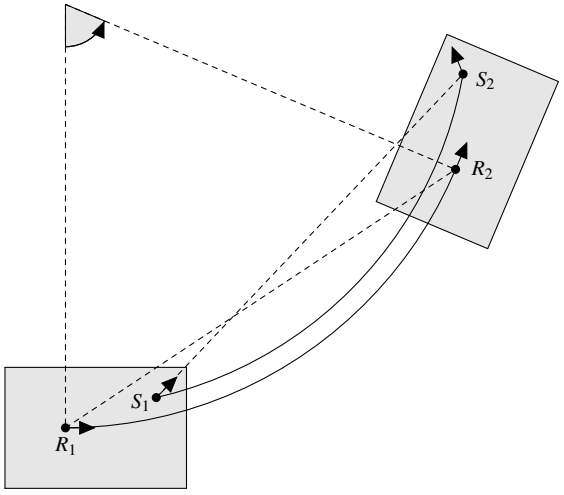


Abb. 3.3: Diese Abbildung zeigt einen Roboter, der sich auf einem Kreisbogen bewegt, mit den dazugehörigen Posen R_1 und R_2 zu den Zeitpunkten 1 und 2. Die Posen des Sensors sind S_1 und S_2. Offensichtlich ist die Bewegung des Sensors nicht gleich der Bewegung des Roboters.

Eine andere Möglichkeit dieses Problem zu lösen wäre, die Sensorposen explizit als weitere Zufallsvariablen zu modellieren. Der resultierende Faktorgraph ist in Abbildung 3.4 zu sehen.

In diesem Fall können die Informationen aus dem Scan-Matching als Faktoren direkt zwischen den Sensorposen modelliert werden. Die Faktoren zwischen Sensor- und Roboterposen repräsentieren dann, unabhängig von den Messungen, den Zusammenhang zwischen den beiden Zufallsvariablen für Sensor- und Roboterpose. Dieser Zusammenhang ist fest durch die Pose des Sensors relativ zum Roboter gegeben. In der Praxis ist es sinnvoll, auch diesen Zusammenhang als Wahrscheinlichkeitsverteilung mit einem kleinen Gaußschem Rauschen zu modellieren, da Fehler, die durch leichte Ungenauigkeiten in den Zeitpunkten der Messungen entstehen, ausgeglichen werden können.

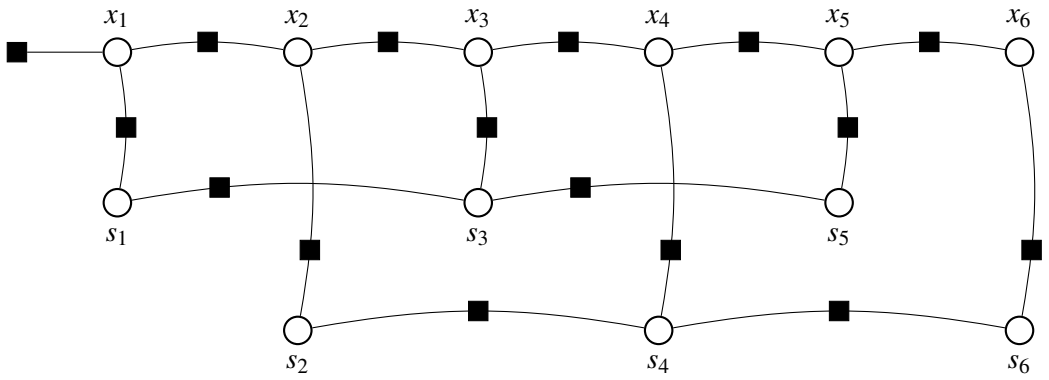


Abb. 3.4: In diesem Faktorgraph sind die Sensorposen explizit als Knoten s_1 bis s_6 modelliert.

Soll auch die Pose eines Sensors relativ zum Roboter variabel modelliert werden, so muss dies explizit als Zufallsvariable und damit als Knoten im Faktorgraph geschehen. Die Werte dieser beiden neuen Knoten gehen dabei jeweils in die Faktoren zwischen Roboter- und Sensorpose ein. Damit sind automatisch die Kalibrierungsparameter für die Sensoren in dem Faktorgraphen integriert worden. Abbildung 3.5 zeigt den Faktorgraphen mit den neuen Kalibrierungsknoten für die Sensoren. Die neu entstandenen Faktoren, die Roboterpose mit Sensorpose und Kalibrierungsparameter in Bezug setzen, beschreiben jetzt die Wahrscheinlichkeiten, für alle Kombination aus Roboterpose, Sensorpose und Kalibrierungsparameter zu einem Zeitpunkt t. Die mathematische Modellierung dieser Verteilung wird genauer in Kapitel 3.3 besprochen.

In dieser Arbeit werden die Sensorparameter wie gerade beschrieben modelliert, da dies übersichtlicher und leichter in der Implementierung ist. Der Vollständigkeit halber sei jedoch erwähnt, dass die Parameter zur Sensorkalibrierung auch modelliert werden können, wenn die Sensorposen nicht explizit als Knoten dargestellt werden (siehe Abbildung 3.6).

Analog zu den Parametern für die Sensorkalibrierung wird auch die Odometriekalibrierung modelliert. Die Zufallsvariable ist hierbei allerdings keine Rototranslation sondern ein Vektor, der aus einzelnen Parametern besteht, mit deren Hilfe die Odometrie berechnet wird. Diese Parameter können zum Beispiel im Fall eines Roboters mit Differentialantrieb die Radien der beiden Räder und ihr Abstand voneinander sein.

Jeder Faktor im Faktorgraphen, dessen Likelihoodfunktion von einer Odometriemessung abhängt, wird nun durch eine zusätzliche Kante mit dem Kalibrierungs-Knoten verbunden. In den Abbildungen 3.5 und 3.6 sind die hinzugekommenen Knoten und Kanten, für die zwei Möglichkeiten der Modellierung des Faktorgraphen, in grauer Farbe dargestellt.

Bei einer längeren Fahrt eines Roboters kommen jetzt noch einzelne Kanten für gefundene Loop-Closings hinzu. Wenn für zwei zeitlich weiter auseinander liegende Scans ein Loop Closing Match gefunden wurde, wird dafür genau wie beim Scan-Matching konsekutiver Scans eine Kante zwischen den zwei Knoten für die zugehörigen Sensorposen beziehungsweise falls die Sensorposen

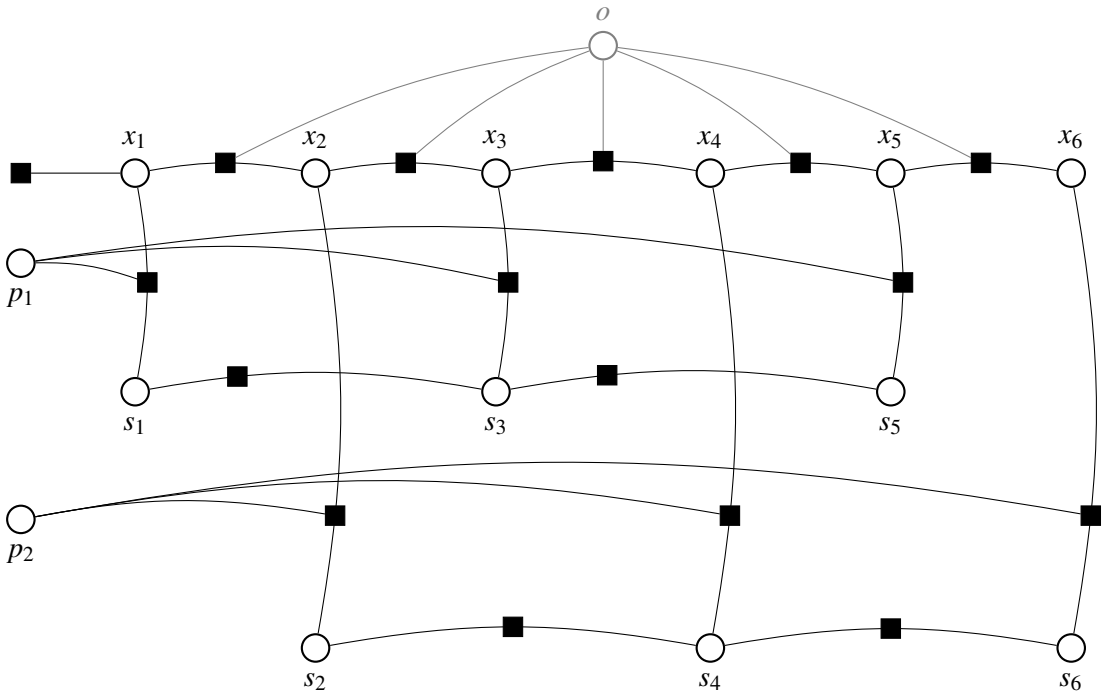


Abb. 3.5: Beispiel für einen Faktorgraphen, der die Kalibrierungsparameter der verwendeten Sensoren als Knoten p_1 und p_2 modelliert. Die Knoten s_1 bis s_3 stehen für die Posen des ersten, s_4 bis s_6 für die Posen des zweiten Sensors. In grauer Farbe sind der Knoten für die Odometriekalibrierung o und die zugehörigen Kanten zu sehen.

nicht explizit modelliert wurden, direkt zwischen den Roboterposen zu dem Zeitpunkt der Scans, hinzugefügt.

Nun ist klar, wie die Struktur des Faktorgraphen für die Fahrt eines Roboters aussehen wird, allerdings ist noch nicht klar, wie die Likelihoodfunktionen der einzelnen Faktoren modelliert werden und wie diese aus den aufgezeichneten Odometrie- und Sensormessungen der Fahrt bestimmt werden. Die mathematische Modellierung der Faktoren ist der theoretische Kern dieser Arbeit und wird im folgenden Abschnitt behandelt.

3.3 Mathematische Modellierung der Faktoren

Faktoren im Faktorgraph repräsentieren, wie bereits weiter oben beschrieben, Wahrscheinlichkeitsverteilungen über den mit ihnen durch Kanten verbundenen Zufallsvariablen. Um die gemeinsame Wahrscheinlichkeitsverteilung des Faktorgraphen optimieren zu können und so die wahrscheinlichste Robotertrajektorie zu finden, müssen zuerst die einzelnen Wahrscheinlichkeitsverteilungen der Faktoren mathematisch beschrieben werden.

Wie im vorigen Kapitel bereits erwähnt, wird für die Kalibrierung in dieser Arbeit ein Faktorgraph erstellt, in dem die Posen der Sensoren explizit als Zufallsvariablen modelliert werden. Ein Bei-

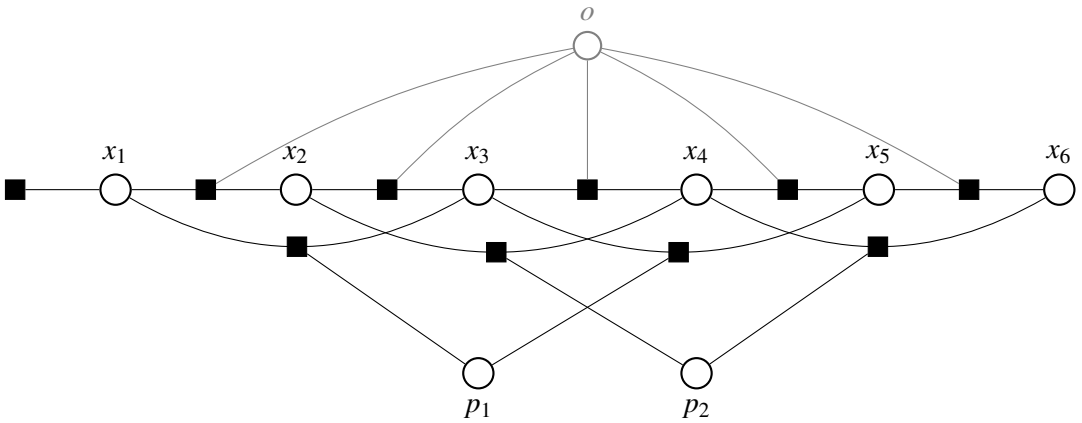


Abb. 3.6: Faktorgraph für SLAM und Kalibrierung ohne explizite Modellierung der Sensorposen.

spiel für einen solchen Faktorgraphen ist in Abbildung 3.5 zu sehen. Dort wird der Faktorgraph für einen Roboter mit zwei Sensoren die abwechselnd messen dargestellt.

Bei Faktoren, deren stochastische Information aus einer Messung gewonnen wurde, wird vereinfachend angenommen, dass diese Messungen grundsätzlich mit einem Gaußschen Rauschen belegt sind. Dies ist auch in der Literatur eine häufig gemachte Annahme [TM06].

3.3.1 Unäre und binäre Faktoren

Ein Faktor dessen zugehöriger Knoten im Faktorgraph Grad 1 hat, wird unärer Faktor genannt. Analog werden binäre und ternäre Faktoren benannt. Der einzige unäre Faktor in Abbildung 3.5 beschreibt die A-Priori-Wahrscheinlichkeit der Anfangspose der Roboters. Im Normalfall startet der Roboter im Ursprung des Koordinatensystems mit einem Orientierungswinkel von 0°, was der Pose $p = (x, y, \theta)^T = (0, 0, 0)^T$ entspricht. Um dies zu erzwingen wird die Diracverteilung zur Pose p verwendet. In dieser Verteilung ist die Wahrscheinlichkeit der Pose p 1 während alle anderen Posen die Wahrscheinlichkeit 0 haben.

Die binären Faktoren in dem Faktorgraphen aus Abbildung 3.5 modellieren die Abhängigkeiten von Sensorposen zueinander, die mittels Scan-Matching oder Loop Closing gefunden werden. Beim Scan-Matching werden hier immer nur verschiedene Scans desselben Sensors verglichen, beim Loop Closing hingegen können die beiden involvierten Scans auch von unterschiedlichen Sensoren aufgenommen worden sein. Es wird hier vorausgesetzt, dass beim Vorgang des Scan-Matchings oder Loop Closings eine Pose p zurückgegeben wird, die die Pose des Sensors des zweiten Scans im Koordinatensystem des Sensors des ersten Scans darstellt. Anders gesagt gibt p an, wie sich der Sensor des ersten Scans von seiner Pose zu diesem Zeitpunkt aus bewegen müsste, um die Pose des Sensors des zweiten Scans zum Zeitpunkt der Messung einzunehmen, oder wie der erste Scan verschoben und gedreht werden muss damit er passend über dem zweiten Scan liegt.

Der Faktor selbst ist definiert durch eine Funktion f, die nur von den Zufallsvariablen für die beiden Sensorpositionen abhängt, da zum Zeitpunkt der Optimierung die Messwerte fest und bekannt sind. Sie gibt für zwei Werte der Zufallsvariablen an, wie plausibel diese Werte bei den gemachten Messungen sind. Dafür wird f als die Measurement Likelihoodfunktion für die Zufallsvariablen der ersten und zweiten Sensorposition, $\mathcal{A}$ und $\mathcal{B}$ und die Messung p definiert [Del12]. Diese hat die Eigenschaft, dass sie zur Wahrscheinlichkeitsdichtefunktion der Verteilung $P(\mathcal{A},\mathcal{B}|p)$ proportional ist. In Abbildung 3.5 stehen die Knoten s_1 und s_3 zum Beispiel für solche Zufallsvariablen $\mathcal{A}$ und $\mathcal{B}$. Der Faktor zwischen den beiden Knoten repräsentiert dann die Funktion $f(\mathcal{A},\mathcal{B})$. Dadurch, dass die Messung mit Gaußschem Rauschen belegt ist, enthält auch die Measurement Likelihoodfunktion dieses Rauschen und wird wie folgt modelliert:

$$f(\mathcal{A},\mathcal{B}) = L(\mathcal{A},\mathcal{B};p) = \exp\left\{-\frac{1}{2}\|h(\mathcal{A},\mathcal{B}) \ominus p\|^2_\Sigma\right\} \tag{3.3}$$

Die Funktion $h(\mathcal{A},\mathcal{B})$ ist dabei die Messfunktion, die angibt, welche Messung bei den gegebenen Posen erwartet wird, während Σ die Kovarianzmatrix des Gaußschen Rauschens ist. In diesem Fall gilt

$$h(a,b) = b \ominus a = \begin{pmatrix} R_{-\alpha} & 0 \\ 0 & 1 \end{pmatrix}(b-a) = \begin{pmatrix} (b_x-a_x)\cos\alpha + (b_y-a_y)\sin\alpha \\ -(b_x-a_x)\sin\alpha + (b_y-a_y)\cos\alpha \\ \beta-\alpha \end{pmatrix} \tag{3.4}$$

wobei $a = (a_x, a_y, \alpha)^T$ und $b = (b_x, b_y, \beta)^T$ Sensorposen sind und

$$R_\alpha = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \tag{3.5}$$

die Rotationsmatrix für den Winkel α ist. Es gilt $R_{-\alpha} = R_\alpha^{-1} = R_\alpha^T$. Die Funktion $b \ominus a$ berechnet die Pose von b relativ zum Koordinatensystem von a. In diesem Kapitel werden für Posen Kleinbuchstaben verwendet, während Zufallsvariablen mit Großbuchstaben in kalligraphischer Schrift und Elemente einer Lie-Gruppe mit normalen Großbuchstaben bezeichnet werden.

Die gleiche Berechnung kann auch mit Hilfe der, in Kapitel 2.4 vorgestellten, Lie-Gruppe $\mathrm{SE}(2)$ ausgeführt werden. Im Folgenden wird gezeigt, dass auch diese Rechnung zum selben Ergebnis führt. Ein Element der Lie-Gruppe setzt sich aus einer Rotation und einer Translation zusammen und hat die Form

$$A' = \begin{pmatrix} R_\alpha & a_t \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & a_x \\ 0 & 1 & a_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} R_\alpha & 0 \\ 0 & 1 \end{pmatrix} \quad \text{mit } a_t = \begin{pmatrix} a_x \\ a_y \end{pmatrix} \in \mathbb{R}^2. \tag{3.6}$$

Mit $A' \cdot x$ mit $x \in \mathbb{R}^2$ wird im globalen Koordinatensystem beschrieben, wie sich ein Punkt x auf einem Roboter bewegen würde, wenn sich dieser zuerst um den Winkel α dreht und dann um den Vektor a_t verschoben wird. Jetzt soll aber die Bewegung jeweils aus dem Koordinatensystem des

Roboters gesehen beschrieben werden. Dies entspricht einer Rotation um $-\alpha$ und einer Verschiebung um $-a_t$. Außerdem entspricht die Bewegung der Pose $a = (a_x, a_y, \alpha)^T$ nur dann, wenn zuerst die Verschiebung und erst danach die Drehung um α vorgenommen wird. Zusammengenommen wird die Pose a nun durch folgendes Element A der Lie-Gruppe repräsentiert:

$$A = \begin{pmatrix} R_{-\alpha} & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} I_2 & -a_t \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} R_\alpha^T & -R_\alpha^T a_t \\ 0 & 1 \end{pmatrix} \in \mathrm{SE}(2) \tag{3.7}$$

Die Funktion $b \ominus a$ kann nun einfach durch $B \cdot A^{-1}$ beschrieben werden. Es folgt:

$$\begin{aligned} h(A,B) = BA^{-1} &= \begin{pmatrix} R_\beta^T & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} I_2 & -b_t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} I_2 & a_t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R_\alpha & 0 \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} R_\beta^T & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} I_2 & -(b_t - a_t) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R_\alpha & 0 \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} R_\beta^T & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R_\alpha & -(b_t - a_t) \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} R_{\beta-\alpha}^T & -R_\beta^T (b_t - a_t) \\ 0 & 1 \end{pmatrix} \end{aligned} \tag{3.8}$$

Das Lie-Gruppen-Element

$$C = h(A,B) = \begin{pmatrix} R_{\beta-\alpha}^T & -R_\beta^T (b_t - a_t) \\ 0 & 1 \end{pmatrix}$$

entspricht wiederum einer Pose $c = \begin{pmatrix} c_x & c_y & \gamma \end{pmatrix}^T$ mit $c_t = \begin{pmatrix} c_x & c_y \end{pmatrix}^T$. Es gilt

$$R_\gamma^T = R_{\beta-\alpha}^T \quad \Rightarrow \gamma = \beta - \alpha \tag{3.9}$$

und

$$\begin{aligned} & -R_\gamma^T \cdot c_t = -R_\beta^T (b_t - a_t) \\ \Rightarrow \quad & R_{\beta-\alpha}^T \cdot c_t = R_\beta^T (b_t - a_t) \\ \Rightarrow \quad & c_t = R_{\beta-\alpha} \cdot R_\beta^T (b_t - a_t) \\ \Rightarrow \quad & c_t = R_\alpha^T (b_t - a_t). \end{aligned} \tag{3.10}$$

Damit ist

$$c = \begin{pmatrix} c_x \\ c_y \\ \gamma \end{pmatrix} = \begin{pmatrix} (b_x - a_x)\cos\alpha + (b_y - a_y)\sin\alpha \\ -(b_x - a_x)\sin\alpha + (b_y - a_y)\cos\alpha \\ \beta - \alpha \end{pmatrix}, \tag{3.11}$$

was dem oben ausgerechneten Wert für $h(a,b)$ entspricht.

In der Optimierung des Faktorgraphen wird die Summe der quadratischen Fehler minimiert. Die Fehlerfunktion für diesen Faktor ist die Differenz zwischen der erwarteten Messung bei den gegebenen Posen und der tatsächlichen Messung $p = (p_x, p_y, p_\theta)^T$ im Messkoordinatensystem, oder anders gesagt, die Pose der erwarteten Messung relativ zum Koordinatensystem der tatsächlichen Messung:

$$\begin{aligned} E(a,b) = h(a,b) \ominus p &= \begin{pmatrix} R_{-p_\theta} & 0 \\ 0 & 1 \end{pmatrix} \left(\begin{pmatrix} R_{-\alpha} & 0 \\ 0 & 1 \end{pmatrix} (b-a) - p \right) \\ &= \begin{pmatrix} R_{-\alpha-p_\theta} & 0 \\ 0 & 1 \end{pmatrix} (b-a) - \begin{pmatrix} R_{-p_\theta} & 0 \\ 0 & 1 \end{pmatrix} p. \end{aligned} \tag{3.12}$$

Ausgerechnet ergibt dies

$$E(a,b) = \begin{pmatrix} (b_x - a_x)\cos(\alpha + p_\theta) + (b_y - a_y)\sin(\alpha + p_\theta) - p_x \cos p_\theta - p_y \sin p_\theta \\ -(b_x - a_x)\sin(\alpha + p_\theta) + (b_y - a_y)\cos(\alpha + p_\theta) + p_x \sin p_\theta - p_y \cos p_\theta \\ \beta - \alpha - p_\theta \end{pmatrix}. \tag{3.13}$$

Für den Gradientenabstieg in der Optimierung wird außerdem die Jacobimatrix der Messfunktion $h(a,b)$ benötigt. Die Jacobimatrix einer Funktion $f : \mathbb{R}^n \to \mathbb{R}^m$ ist definiert als

$$J(x) := \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(x) & \frac{\partial f_1}{\partial x_2}(x) & \dots & \frac{\partial f_1}{\partial x_n}(x) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(x) & \frac{\partial f_m}{\partial x_2}(x) & \dots & \frac{\partial f_m}{\partial x_n}(x) \end{pmatrix}. \tag{3.14}$$

In diesem Fall wird die gesamte Jacobimatrix der Übersicht halber in zwei Untermatrizen aufgeteilt, die unabhängig von einander berechnet werden können. Die erste Matrix J_a enthält die Ableitungen nach a, die zweite Matrix J_b die Ableitungen nach b. Um die gesamte Jacobimatrix J zu erhalten, müssen nur die beiden Untermatrizen nebeneinander geschrieben werden $J = \begin{pmatrix} J_a & J_b \end{pmatrix}$.

Wie bereits in Kapitel 2.4 angedeutet, wird sich dem Konzept der Lie-Gruppe bedient, um die Jacobimatrizen zu bestimmen. Für die genaue Herleitung der Ableitungsregeln müsste eine deutlich tiefergehende Einführung in Mannigfaltigkeiten und Vektorfelder gegeben werden. Dies würde den Rahmen dieser Arbeit sprengen, die genaue mathematische Theorie kann aber in [War71, Kapitel 3] nachgelesen werden. Eine Beschreibung einer analogen Herleitung der Jacobimatrizen für die Rotationsgruppe $\mathrm{SO}(3)$ findet sich in [Ead13]. Für die hier berechneten Ableitungen genügt es zu wissen, dass ein Produkt von zwei Elementen einer Lie-Gruppe nach einem der Elemente abgeleitet werden kann, indem dieses von links mit der Exponentialabbildung eines Tangentialvektors multipliziert wird. Das Resultat kann dann nach dem Tangentialvektor um das neutrale Element

abgeleitet werden. Es gilt also für $X, Y \in \mathrm{SE}(2)$ und $\delta \in T_{I_3}\mathrm{SE}(2) \cong \mathfrak{se}(2)$

$$\frac{\partial XY}{\partial Y} = \frac{\partial}{\partial \delta}\left(X \cdot \left(e^{\delta} \cdot Y\right)\right)\Big|_{\delta=0} \quad \text{und} \quad \frac{\partial XY}{\partial X} = \frac{\partial}{\partial \delta}\left(\left(e^{\delta} \cdot X\right) \cdot Y\right)\Big|_{\delta=0}. \tag{3.15}$$

Weiter gilt

$$\frac{\partial}{\partial \delta}\left(\exp\left(X \cdot \delta\right) \cdot Y\right)\big|_{\delta=0} = X. \tag{3.16}$$

In Kapitel 2.4.3.3 wurde außerdem die adjungierte Matrix Ad_A berechnet, für die gilt

$$\mathrm{Ad}_A \cdot s = \begin{pmatrix} R_\alpha & a_y \\ & -a_x \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} s_x \\ s_y \\ \sigma \end{pmatrix} = ASA^{-1} \tag{3.17}$$

$$\text{mit} \quad A = \begin{pmatrix} R_\alpha & a_x \\ & a_y \\ 0 & 1 \end{pmatrix} \in \mathrm{SE}(2) \quad \text{und} \quad S = \begin{pmatrix} 0 & -\sigma & s_x \\ \sigma & 0 & s_y \\ 0 & 0 & 0 \end{pmatrix} \in \mathfrak{se}(2).$$

Damit kann eine weitere Rechenregel gezeigt werden, die bei der Berechnung der Jacobimatrizen angewendet wird.

$$\exp(\mathrm{Ad}_X \cdot \delta) \cdot X = \exp(X\delta X^{-1}) \cdot X = X \cdot \exp(\delta) \cdot X^{-1} \cdot X = X \cdot \exp(\delta) \tag{3.18}$$

Mit Hilfe der adjungierten Matrix kann also $\exp(\delta)$ in der Gleichung nach links verschoben werden:

$$X \cdot \exp(\delta) = \exp(\mathrm{Ad}_X \cdot \delta) \cdot X \tag{3.19}$$

Mit den vorgestellten Rechenregeln können die Jacobimatrizen der Messfunktion $h(A,B) = B \cdot A^{-1}$ wie folgt berechnet werden:

$$J_b = \frac{\partial BA^{-1}}{\partial B} = \frac{\partial}{\partial \delta}\left(\left(\exp(\delta) \cdot B\right) \cdot A^{-1}\right)\big|_{\delta=0} = I_3 \tag{3.20}$$

$$\begin{aligned} J_a = \frac{\partial BA^{-1}}{\partial A} &= \frac{\partial}{\partial \delta}\left(B \cdot \left(\exp(\delta) \cdot A\right)^{-1}\right)\big|_{\delta=0} \\ &= \frac{\partial}{\partial \delta}\left(BA^{-1} \cdot \exp(-\delta)\right)\big|_{\delta=0} \\ &= \frac{\partial}{\partial \delta}\left(\exp(-\mathrm{Ad}_{BA^{-1}}\delta) \cdot BA^{-1}\right)\big|_{\delta=0} \\ &= -\mathrm{Ad}_{BA^{-1}} \end{aligned} \tag{3.21}$$

Mit $B \cdot A^{-1} = \begin{pmatrix} R_{\beta-\alpha}^T & -R_\beta^T(b_t - a_t) \\ 0 & 1 \end{pmatrix}$ und $\mathrm{Ad}_C = \begin{pmatrix} R_\gamma & c_y \\ & -c_x \\ 0 & 1 \end{pmatrix}$ für $C = \begin{pmatrix} R_\gamma & c_x \\ & c_y \\ 0 & 1 \end{pmatrix}$ ergibt sich:

$$-\mathrm{Ad}_{BA^{-1}} = \begin{pmatrix} -R_{\beta-\alpha}^T & v_2 \\ & -v_1 \\ 0 & -1 \end{pmatrix} \quad \text{mit} \quad v = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = -R_\beta^T(b_t - a_t) \tag{3.22}$$

Eingesetzt ergibt sich die Jacobimatrix für die partielle Ableitung nach a folgendermaßen

$$J_a = -\mathrm{Adj}_{BA^{-1}} = \begin{pmatrix} -\cos(\beta-\alpha) & -\sin(\beta-\alpha) & -(b_x-a_x)\sin\beta + (b_y-a_y)\cos\beta \\ \sin(\beta-\alpha) & -\cos(\beta-\alpha) & -(b_x-a_x)\cos\beta - (b_y-a_y)\sin\beta \\ 0 & 0 & -1 \end{pmatrix}. \tag{3.23}$$

Das Konzept zur Erstellung binärer Faktoren lässt sich für n–äre Faktoren erweitern. Der Faktor beschreibt dann den stochastischen Zusammenhang zwischen n Zufallsvariablen $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$ abhängig von einer Messung p. Die Messfunktion $h(a_1, \ldots, a_n)$ gibt nun den wahrscheinlichsten Messwert bei gegebenen Werten $a_1, \ldots, a_n$ der Zufallsvariablen zurück. Die Funktion f des Faktors ist dann

$$f(\mathcal{A}_1, \ldots, \mathcal{A}_n) = L(\mathcal{A}_1, \ldots, \mathcal{A}_n; p) = \exp\left\{-\frac{1}{2}\|h(\mathcal{A}_1, \ldots, \mathcal{A}_n) \ominus p\|_\Sigma^2\right\}. \tag{3.24}$$

3.3.2 Faktoren für die Kalibrierung der Sensoren

Die Faktoren für die Kalibrierung der Sensoren sind ternäre Faktoren, die von den Zufallsvariablen für die aktuelle Roboter- und Sensorpose und für die Kalibrierungsparameter abhängen. Anders als bei den binären Faktoren gibt es hierbei keine Messungen als zusätzliche Eingabe. Der Kalibrierungsparameter für einen Sensor ist dreidimensional und gibt die Sensorpose relativ zum Koordinatensystem des Roboters an.

Es wird hierbei davon ausgegangen, dass die Pose des Sensors auf dem Roboter während einer Fahrt konstant ist. Deshalb sollte es nicht möglich sein, dass die Sensorpose sich relativ zum Roboterkoordinatensystem von einem Zeitpunkt zum anderen ändert. Wie im unären Fall dürften diese Faktoren daher im Optimierungsergebnis keinen Fehler haben, das heißt, die Rototranslation von Roboter- zu Sensorpose sollte genau dem Kalibrierungsparameter entsprechen. Die Likelihoodfunktion, die proportional zu der gemeinsame Wahrscheinlichkeitsverteilung der drei Zufallsvariablen ist, wäre dann folgende:

$$f(\mathcal{A}, \mathcal{B}, \mathcal{C}) = L(\mathcal{A}, \mathcal{B}, \mathcal{C}) = \begin{cases} 1 & \text{wenn } \mathcal{C} = \mathcal{B} \ominus \mathcal{A} \\ 0 & \text{sonst} \end{cases} \tag{3.25}$$

Dabei sind $\mathcal{A}$ und $\mathcal{B}$ die Zufallsvariablen, die die Roboter- bzw. Sensorpose modellieren und $\mathcal{C}$ die

Variable für die Kalibrierungsparameter.

In der Realität kann diese Verteilung allerdings nicht vorausgesetzt werden. Sind beispielsweise die Zeitstempel der Odometrie- und Sensormessungen nicht genau synchron, könnte es passieren, dass eine Sensormessung einer, nur vermeintlich gleichzeitig erfolgten, Odometriemessung zugeordnet wird. Tatsächlich wäre die Sensormessung allerdings erst erfolgt, nachdem sich der Roboter bereits ein Stück weiterbewegt hätte. In diesem Fall wäre es nicht sinnvoll, eine feste Pose des Sensors auf dem Roboter zu erzwingen. Die Funktion des Faktors wird deshalb wie bei den binären Faktoren mit einem Gaußschen Rauschen belegt:

$$f(\mathcal{A},\mathcal{B},\mathcal{C}) = L(\mathcal{A},\mathcal{B},\mathcal{C}) = \exp\left\{-\frac{1}{2}\|\mathcal{B}\ominus\mathcal{A}\ominus\mathcal{C}\|^2_\Sigma\right\} \tag{3.26}$$

Die Fehlerfunktion $E(a,b,c)$ für drei Posen

$$a = \begin{pmatrix} a_x & a_y & \alpha \end{pmatrix}^T,\ b = \begin{pmatrix} b_x & b_y & \beta \end{pmatrix}^T \text{ und } c = \begin{pmatrix} c_x & c_y & \gamma \end{pmatrix}^T$$

gibt die Differenz zwischen tatsächlicher Sensorpose relativ zum Roboterkoordinatensystem und Kalibrierungsparameter in dessen Koordinatensystem an. Sie ist äquivalent zu der Fehlerfunktion der binären Faktoren, wenn der Kalibrierungsparameter gleich dem Messwert gesetzt wird:

$$E(a,b,c) = (b\ominus a)\ominus c = \begin{pmatrix} R_{-\alpha-\gamma} & 0 \\ 0 & 1 \end{pmatrix}(b-a) - \begin{pmatrix} R_{-\gamma} & 0 \\ 0 & 1 \end{pmatrix} c. \tag{3.27}$$

In der Lie-Gruppe $\mathrm{SE}(2)$ entspricht dies der Funktion

$$\begin{aligned} E(A,B,C) = BA^{-1}C^{-1} &= \begin{pmatrix} R_\beta^T & -R_\beta^T b_t \\ 0 & 1 \end{pmatrix}\begin{pmatrix} R_\alpha & a_t \\ 0 & 1 \end{pmatrix}\begin{pmatrix} R_\gamma & c_t \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} R^T_{\beta-\alpha-\gamma} & R^T_{\beta-\alpha}\cdot c_t - R^T_\beta(b_t-a_t) \\ 0 & 1 \end{pmatrix}. \end{aligned} \tag{3.28}$$

Für die Pose $d = \begin{pmatrix} d_x & d_y & \delta \end{pmatrix}^T$, $d_t = \begin{pmatrix} d_x & d_y \end{pmatrix}^T$, die dem Element $E(A,B,C)$ der Lie-Gruppe entspricht, gilt

$$R^T_\delta = R^T_{\beta-\alpha-\gamma} \quad \Rightarrow \delta = \beta-\alpha-\gamma \tag{3.29}$$

und

$$\begin{aligned} -R_\delta^T \cdot d_t &= R_{\beta-\alpha}^T \cdot c_t - R_\beta^T (b_t - a_t) \\ \Rightarrow \quad R_{\beta-\alpha-\gamma}^T \cdot d_t &= R_\beta^T (b_t - a_t) - R_{\beta-\alpha}^T \cdot c_t \\ \Rightarrow \quad d_t &= R_{\beta-\alpha-\gamma}(R_\beta^T (b_t - a_t) - R_{\beta-\alpha}^T \cdot c_t) \\ \Rightarrow \quad d_t &= R_{\beta-\alpha-\gamma} R_\beta^T (b_t - a_t) - R_{\beta-\alpha-\gamma} R_{\beta-\alpha}^T \cdot c_t \\ \Rightarrow \quad d_t &= R_{\alpha+\gamma}^T (b_t - a_t) - R_\gamma^T \cdot c_t \end{aligned} \tag{3.30}$$

Damit gilt:

$$\begin{aligned} d = \begin{pmatrix} d_t \\ \delta \end{pmatrix} &= \begin{pmatrix} R_{\alpha+\gamma}^T (b_t - a_t) - R_\gamma^T \cdot c_t \\ \beta - \alpha - \gamma \end{pmatrix} \\ &= \begin{pmatrix} (b_x - a_x)\cos(\alpha+\gamma) + (b_y - a_y)\sin(\alpha+\gamma) - c_x \cos\gamma - c_y \sin\gamma \\ -(b_x - a_x)\sin(\alpha+\gamma) + (b_y - a_y)\cos(\alpha+\gamma) + c_x \sin\gamma - c_y \cos\gamma \\ \beta - \alpha - \gamma \end{pmatrix} \\ &= E(a,b,c) \end{aligned} \tag{3.31}$$

Die Jacobimatrizen J_a, J_b und J_c werden wieder einzeln mit Hilfe der in den Gleichungen 3.15, 3.16 und 3.19 vorgestellten Regeln berechnet.

$$J_b = \frac{\partial BA^{-1}C^{-1}}{\partial B} = \frac{\partial}{\partial \delta} \left((\exp(\delta) \cdot B) \cdot A^{-1}C^{-1} \right)\big|_{\delta=0} = I_3 \tag{3.32}$$

$$\begin{aligned} J_a = \frac{\partial BA^{-1}C^{-1}}{\partial A} &= \frac{\partial}{\partial \delta} \left(B \cdot (\exp(\delta) \cdot A)^{-1} \cdot C^{-1} \right)\big|_{\delta=0} \\ &= \frac{\partial}{\partial \delta} \left(BA^{-1} \cdot \exp(-\delta) \cdot C^{-1} \right)\big|_{\delta=0} \\ &= \frac{\partial}{\partial \delta} \left(\exp(-\mathrm{Adj}_{BA^{-1}}\delta) \cdot BA^{-1}C^{-1} \right)\big|_{\delta=0} \\ &= -\mathrm{Adj}_{BA^{-1}} \end{aligned} \tag{3.33}$$

$$\begin{aligned} J_c = \frac{\partial BA^{-1}C^{-1}}{\partial C} &= \frac{\partial}{\partial \delta} \left(BA^{-1} \cdot (\exp(\delta) \cdot C)^{-1} \right)\big|_{\delta=0} \\ &= \frac{\partial}{\partial \delta} \left(BA^{-1}C^{-1} \cdot \exp(-\delta) \right)\big|_{\delta=0} \\ &= \frac{\partial}{\partial \delta} \left(\exp(-\mathrm{Adj}_{BA^{-1}C^{-1}}\delta) \cdot BA^{-1}C^{-1} \right)\big|_{\delta=0} \\ &= -\mathrm{Adj}_{BA^{-1}C^{-1}} \end{aligned} \tag{3.34}$$

Die Matrix $-\mathrm{Adj}_{BA^{-1}}$ wurde bereits in Gleichung 3.23 für die binären Faktoren berechnet. Die Jacobimatrix $J_c = -\mathrm{Adj}_{BA^{-1}C^{-1}}$ lässt sich wie folgt berechnen:

$$J_c = -\mathrm{Adj}_{BA^{-1}C^{-1}} = \begin{pmatrix} -R^T_{\beta-\alpha-\gamma} & -v_2 \\ 0 & v_1 \\ & -1 \end{pmatrix} \quad \text{mit} \quad v = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = R^T_{\beta-\alpha} \cdot c_t - R^T_{\beta}(b_t - a_t) \quad [3.35]$$

Ausgerechnet ergibt sich:

$$\begin{aligned} -v_2 &= c_x \sin(\beta-\alpha) - c_y \cos(\beta-\alpha) - (b_x - a_x)\sin\beta + (b_y - a_y)\cos\beta \\ v_1 &= c_x \cos(\beta-\alpha) + c_y \sin(\beta-\alpha) - (b_x - a_x)\cos\beta - (b_y - a_y)\sin\beta \end{aligned} \quad [3.36]$$

3.3.3 Faktoren für die Kalibrierung der Odometrie

Für die Modellierung der Odometriekalibrierung können verschiedene Ansätze verwendet werden. Entweder die Modellierung der Faktoren wird auf einem spezifischen Bewegungsmodell für eine bestimmte Kinematik aufgebaut oder es wird stattdessen ein allgemeines stochastisches Bewegungsmodell verwendet, das für verschiedene Roboterkinematiken verwendet werden kann.

Beide Möglichkeiten haben ihre eigenen Vorteile. Wird ein speziell auf die Kinematik des Roboters ausgerichtetes Bewegungsmodell verwendet, entsprechen die Kalibrierungsparameter physikalischen Größen, die am Roboter messbar sind. So wird zum Beispiel bei der Kalibrierung von Robotern mit Differentialantrieb oft der Abstand der Räder als ein Kalibrierungsparameter gewählt. Fällt die Entscheidung zugunsten eines allgemeinen stochastischen Bewegungsmodells, ist die Zuordnung von Kalibrierungsparametern zu physikalischen Eigenschaften des Roboters nicht von vornherein klar. Außerdem kann ein allgemeines Modell sich nur stochastisch an die Gegebenheiten der Roboterkinematik annähern und es ist nicht offensichtlich, ob eine Kinematik mit dem Modell überhaupt genau darstellbar ist. Bei einem speziell für eine Kinematik entwickelten Modell ist im Gegensatz dazu evident, welche vereinfachten Annahmen getroffen wurden, und das Modell ist unter der Voraussetzung exakt, dass diese Annahmen gültig sind. Andererseits ermöglicht ein stochastischen Bewegungsmodell die Einbeziehung nicht-systematische Fehler in die Modellbildung. Außerdem kann für eine Positionsbestimmung mittels Odometrie eine Varianz angeben werden.

In diesem Kapitel wird zuerst die Modellierung der Faktoren für einen Roboter mit Differentialantrieb und ein entsprechendes Bewegungsmodell angegeben. Danach wird eine Modellierung eines Kalibrierungsfaktors betrachtet, die wegen eines allgemeinen stochastischen Bewegungsmodells für verschiedene Kinematiken verwendbar ist.

3.3.3.1 Kalibrierung der Odometrie eines Roboters mit Differentialantrieb

Ein Roboter mit Differentialantrieb kann seine beiden angetriebenen Räder einzeln mit jeweils einem Motor antreiben. Drehungen des Roboters resultieren daraus, dass seine Räder mit unterschiedlicher Geschwindigkeit oder sogar in entgegengesetzte Richtungen drehen.

Die Drehgeschwindigkeit der Räder wird dabei über jeweils einen Drehgeber (Rotary Encoder) an jedem Rad gemessen. Dies ist ein Sensor der die Drehwinkel des Rades misst und nach einer Drehung um einen bestimmten Winkel einen Impuls zurück gibt. Mit der Anzahl der Impulse pro Radumdrehung und dem Radius des Rades kann aus der Anzahl gemessener Impulse die vom Rad zurückgelegte Strecke berechnet werden. Als Kalibrierungsparameter werden in diesem Abschnitt – wie auch häufig in der Literatur – die Radien und der Abstand der Räder r_R, r_L und b verwendet [KGB12].

Ein Faktor zur Kalibrierung der Odometrie hängt nun jeweils von den Zufallsvariablen für zwei aufeinander folgende Roboterposen $\mathcal{A}$ und $\mathcal{B}$ und der dreidimensionalen Zufallsvariable für den Kalibrierungsparameter $\mathcal{P}$ ab. Des Weiteren wird die Anzahl der Impulse der Drehgeber für beide Räder während der Bewegung von der ersten zur zweiten Pose gemessen. Auch diese Messung m geht in die Funktion des Faktors mit ein. Das Bewegungsmodell des Roboters gibt die Wahrscheinlichkeit $P(b|a,m,p)$ einer zweiten Pose b unter der Bedingung einer ersten Pose a und der Odometriemessung m an. In unserem Fall ist das Bewegungsmodell zusätzlich auch noch von dem Kalibrierungsparameter p abhängig. Da die Wahrscheinlichkeit der ersten Pose a stochastisch unabhängig von der danach gemachten Unter der Annahme, dass die Wahrscheinlichkeit der ersten Pose a unabhängig von m und p gleichverteilt ist gilt:

$$P(b|a,m,p) = P(a,b|m,p) \propto L(a,b,p;m) \qquad [3.37]$$

Die Measurement Likelihoodfunktion $L(a,b,p;m)$ wird im folgenden Abschnitt berechnet.

3.3.3.1.1 Berechnung des Bewegungsmodells

Aus der Odometriemessung und den Kalibrierungsparametern lässt sich eindeutig eine Schätzung für die Roboterbewegung bestimmen. Vereinfachend wird dabei angenommen, dass die Geschwindigkeit der Räder zwischen den beiden Roboterposen konstant ist. Diese Annahme ist theoretisch keine große Einschränkung, da sich jede Bewegung als Abfolge von Teilbewegungen mit konstanter Geschwindigkeit modellieren lässt. Eine exakte Modellierung von Beschleunigungen ist ohnehin nicht möglich, da die Messungen der Radbewegungen diskret erfolgen.

Bei konstanter Geschwindigkeit bewegen sich beide Räder des Roboters auf einem Kreisbogen um einen gemeinsamen Mittelpunkt (siehe Abbildung 3.7). Weiter wird angenommen, dass der Ursprung des Roboterkoordinatensystems in der Mitte zwischen den Mittelpunkten der beiden Rädern liegt. Im Folgenden wird nun aus den Messwerten und Kalibrierungsparametern eine Pose

o berechnet, indem aus dem messbaren Kreisbogen auf die nicht direkt zugänglichen Werte von Winkel und Radius zurückgeschlossen wird. Diese Pose ist eine Schätzung für die zweite Roboterpose relativ zum Koordinatensystem der ersten Pose.

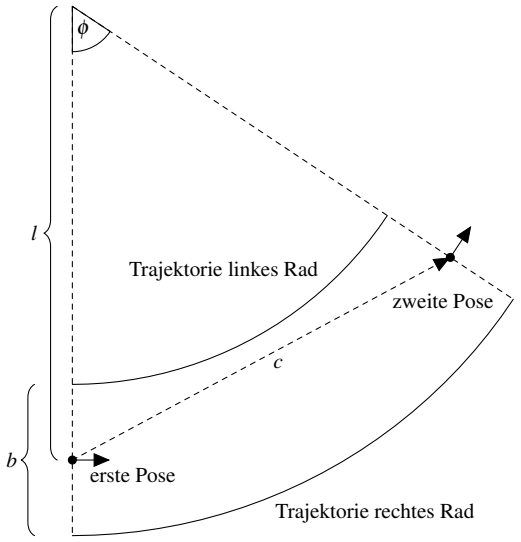


Abb. 3.7: Die Abbildung zeigt die Trajektorien der beiden Räder des Roboters auf dem Weg von der ersten zur zweiten Pose.

Sei l der Radius des Kreisbogens auf dem der Roboter sich in dem betrachteten Zeitraum bewegt und ϕ der im Bogenmaß gemessene Winkel, um den sich der Roboter in dem betrachteten Zeitraum dreht. Dann ergibt sich die Strecke, die das rechte bzw. linke Rad (s_R bzw. s_L) zurücklegt, wie folgt:

$$s_R = \left(l + \frac{b}{2}\right)\phi \qquad [3.38]$$

$$s_L = \left(l - \frac{b}{2}\right)\phi. \qquad [3.39]$$

Außerdem kann die von den Rädern zurückgelegte Strecke auch aus der Anzahl gemessener Impulse für das rechte bzw. linke Rad $N_{R/L}$, der Anzahl der Impulse pro Radumdrehung C_e und dem Radius des jeweiligen Rades $r_{R/L}$ berechnet werden:

$$s_{R/L} = \frac{2\pi}{C_e} \cdot r_{R/L} \cdot N_{R/L} = \text{konst} \cdot r_{R/L} \cdot N_{R/L} \qquad [3.40]$$

Aus diesen vier Gleichungen können s_R und s_L eliminiert werden und damit l und ϕ abhängig von den bekannten Parametern berechnet werden:

$$\phi = \frac{s_R - s_L}{b} \qquad = \frac{2\pi}{C_e b}(r_R N_R - r_L N_L) \qquad [3.41]$$

$$l = \frac{b}{2} \cdot \frac{s_R + s_L}{s_R - s_L} = \frac{b}{2} \cdot \frac{r_R N_R + r_L N_L}{r_R N_R - r_L N_L} \qquad [3.42]$$

Die dritte Dimension der Pose o, ihre Orientierung, entspricht dem Winkel der Rotation, die der Roboter von der ersten zur zweiten Pose durchläuft. Dies ist nach Definition einfach ϕ. Die ersten beiden Dimensionen entsprechen einer an der ersten Pose beginnenden Translation c (siehe Abbildung 3.7). Diese Translation kann wie folgt berechnet werden:

$$c = \begin{pmatrix} 0 \\ l \end{pmatrix} - R_\phi \begin{pmatrix} 0 \\ l \end{pmatrix} \qquad [3.43]$$

Dies entspricht der Kombination zweier Translationen, von der ersten Pose zum Drehmittelpunkt und dann von dort mit der negierten und um ϕ gedrehten ersten Translation zur zweiten Pose.

Damit ergibt sich die Pose o als

$$o = \begin{pmatrix} R_\phi & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ -l \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ l \\ \phi \end{pmatrix} = \begin{pmatrix} l \sin\phi \\ l(1-\cos\phi) \\ \phi \end{pmatrix} \qquad [3.44]$$

Die Pose o beschreibt nun also eine aus den Messwerten berechnete Roboterbewegung. Da sie aus Messwerten berechnet wurde, ist sie als Schätzung der korrekten Pose mit einem Gaußschen Rauschen behaftet. Die Measurement Likelihoodfunktion kann damit wie folgt aufgestellt werden:

$$L(a,b,p;m) = L(a,b;o) = \exp\left\{-\frac{1}{2}\|h(a,b) \ominus o\|^2_\Sigma\right\} = \exp\left\{-\frac{1}{2}\|(b \ominus a) \ominus o\|^2_\Sigma\right\} \qquad [3.45]$$

Dabei ist $h(a,b) = b \ominus a$ die Messfunktion, die angibt, welcher Wert für o bei den gegebenen Werten von a und b erwartet wird und Σ die Kovarianz des Gaußschen Rauschens.

3.3.3.1.2 Modell des Faktors Die von dem Faktor repräsentierte Funktion $f(a,b,p)$ wird wie schon zuvor gleich der Measurement Likelihoodfunktion $L(a,b,p;m)$ gesetzt. Aus den Berechnungen des letzten Abschnitts ergibt sich

$$f(a,b,p) = L(a,b,p;m) = L(a,b;o) = \exp\left\{-\frac{1}{2}\|(b \ominus a) \ominus o\|^2_\Sigma\right\} \qquad [3.46]$$

wobei

$$o = \begin{pmatrix} l \sin\phi \\ l(1-\cos\phi) \\ \phi \end{pmatrix}, \qquad \phi = \frac{2\pi}{C_e b}(r_R N_R - r_L N_L), \qquad l = \frac{b}{2} \cdot \frac{r_R N_R + r_L N_L}{r_R N_R - r_L N_L},$$

$$p = (r_R, r_L, b)^T \qquad \text{und} \qquad m = (N_R, N_L)^T.$$

Die Fehlerfunktion im Koordinatensystem von o berechnet, ergibt sich wie folgt:

$$\begin{aligned} E(a,b,p) &= h(a,b) \ominus o = (b \ominus a) \ominus o \\ &= \begin{pmatrix} R_{-o_\theta} & 0 \\ 0 & 1 \end{pmatrix} \left(\begin{pmatrix} R_{-\alpha} & 0 \\ 0 & 1 \end{pmatrix} (b-a) - o \right) \\ &= \begin{pmatrix} R_{-o_\theta-\alpha} & 0 \\ 0 & 1 \end{pmatrix} (b-a) - \begin{pmatrix} R_{-o_\theta} & 0 \\ 0 & 1 \end{pmatrix} o \end{aligned} \quad [3.47]$$

Werden für o die berechneten Werte eingesetzt, lässt sich dies noch weiter vereinfachen:

$$\begin{aligned} E(a,b,p) &= \begin{pmatrix} R_{-\phi-\alpha} & 0 \\ 0 & 1 \end{pmatrix} (b-a) - \begin{pmatrix} R_{-\phi} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} l\sin\phi \\ l(1-\cos\phi) \\ \phi \end{pmatrix} \\ &= \begin{pmatrix} R_{-\phi-\alpha} & 0 \\ 0 & 1 \end{pmatrix} (b-a) - \begin{pmatrix} l \\ 0 \\ \phi \end{pmatrix} \\ &= \begin{pmatrix} (b_x-a_x)\cos(\alpha+\phi) + (b_y-a_y)\sin(\alpha+\phi) - l \\ -(b_x-a_x)\sin(\alpha+\phi) + (b_y-a_y)\cos(\alpha+\phi) \\ \beta-\alpha-\phi \end{pmatrix} \end{aligned} \quad [3.48]$$

3.3.3.2 Allgemeine Kalibrierung der Odometrie über ein stochastisches Modell

Ein Bewegungsmodell eines Roboters ist eine Wahrscheinlichkeitsverteilung $P(b'|a,m)$, die die Wahrscheinlichkeit einer neuen Roboterpose b' auf Basis der letzten Pose a und der Odometriemessung m angibt. Das Bewegungsmodell als Verteilung anstatt als feste Funktion anzugeben ist insofern sinnvoll, als bei jeder Roboterbewegung aufgrund nicht-systematischer Fehlerquellen probabilistische Anteile enthalten sind. Die gleiche Folge von Bewegungsbefehlen wird praktisch nie dazu führen, dass auch der Roboter an der exakt gleichen Endposition ankommt. Des Weiteren hat ein probabilistisches Bewegungsmodell nicht so große Anforderungen an die Korrektheit des Bewegungsmodells [Thr00]. So können allgemeine Bewegungsmodelle, die für verschiedene Arten von Roboterkinematiken verwendet werden können, aufgestellt werden. In Kapitel 3.3.3.1 wurde auch ein probabilistisches Modell verwendet. Der Unterschied zu dem Verfahren hier besteht darin, dass dort zuerst die Roboterpose b möglichst exakt aus den gegebenen Werten berechnet wird. Erst danach wird das Bewegungsmodell als multivariate Gaußverteilung mit dem berechneten Wert als Mittel aufgestellt. Hier wird hingegen versucht, die Bewegung eines Roboters anhand verschiedener Parameter allgemein zu beschreiben. Die Mittelwerte und Kovarianzen der Verteilung werden dann für einen speziellen Roboter gelernt.

3.3.3.2.1 Stochastische Bewegungsmodelle Im letzten Kapitel wurden als Odometriemessungen die Anzahl der Impulse zweier Drehgeber verwendet. Diese Parameter können natürlich nur verwendet werden, wenn die Odometrie genauso gemessen wird. Da dies nicht immer der Fall ist, wird statt dessen von Nicholas Roy und Sebastian Thrun 1999 in [RT99] vorgeschlagen, als Parameter die Rotation des Roboters t von $a = \begin{pmatrix} a_x & a_y & \alpha \end{pmatrix}^T$ zu $b' = \begin{pmatrix} b'_x & b'_y & \beta' \end{pmatrix}^T$ und den Abstand der beiden Posen d zu wählen. Unter der Annahme, dass der Roboter von einer Pose zur nächsten jeweils zuerst um den Winkel t dreht und dann die Strecke d geradeaus fährt (siehe Abbildung 3.8), lässt sich die neue Pose b' wie folgt berechnen [RT99]:

$$\begin{aligned} b'_x &= a_x + D\cos(\alpha + T) \\ b'_y &= a_y + D\sin(\alpha + T) \\ \beta' &= \alpha + T \mod 2\pi \end{aligned} \qquad [3.49]$$

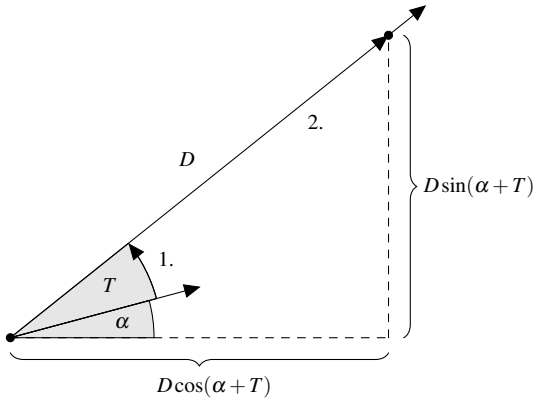


Abb. 3.8: Die Abbildung zeigt die Schritte der Bewegung des Roboter nach dem Modell. Zuerst dreht der Roboter um den Winkel T dann fährt er die Strecke D vorwärts.

Dabei sind D und T die Erwartungswerte der Zufallsvariablen $\mathcal{D}$ und $\mathcal{T}$, die von d und t abhängen und den tatsächlichen Abstand der Posen und die tatsächliche Rotation des Roboters modellieren. Ist die Odometrie eines Roboters perfekt kalibriert, treten keine systematischen Fehler sondern nur noch nicht-systematische Fehler auf. Die nicht-systematischen Fehler (ε_d und ε_t) haben dann einen Erwartungswert von 0 und wirken sich additiv auf die Odometrieparameter aus:

$$\mathcal{D} = d + \varepsilon_d, \qquad \mathcal{T} = t + \varepsilon_t, \qquad D = \mathbb{E}(\mathcal{D}) = d, \qquad T = \mathbb{E}(\mathcal{T}) = t \qquad [3.50]$$

Normalerweise erfolgen Drehung und Vorwärtsbewegung des Roboters allerdings nicht getrennt voneinander. Im Gegenteil wurde im Fall des Roboters mit Differentialantrieb davon ausgegangen, dass der Roboter gleichzeitig zu einer Vorwärtsbewegung mit konstanter Geschwindigkeit ebenfalls mit konstanter Geschwindigkeit dreht und dadurch auf einem Kreisbogen fährt. Unter dieser

Annahme wird mit den gleichen Parametern ein verbessertes Modell aufgestellt [EP04]:

$$\begin{aligned} b'_x &= a_x + D\cos\left(\alpha + \frac{T}{2}\right) \\ b'_y &= a_y + D\sin\left(\alpha + \frac{T}{2}\right) \\ \beta' &= \alpha + T \mod 2\pi \end{aligned} \qquad [3.51]$$

Die Bewegung des Roboters auf einem Kreisbogen wird hierbei dadurch approximiert, dass der Roboter zuerst um den Winkel $T/2$ dreht, dann eine Strecke der Länge D geradeaus fährt und am Ende wiederum um den Winkel $T/2$ dreht (siehe Abbildung 3.9). Dieses Modell setzt wie-

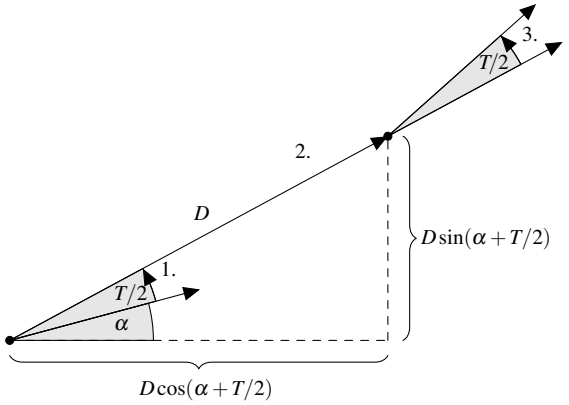


Abb. 3.9: Die Drehung des Roboters mit konstanter Geschwindigkeit während der Fahrt wird durch eine Drehung um $T/2$, Geradeausfahren der Strecke D und eine weitere Drehung um $T/2$ modelliert.

derum voraus, dass der Roboter sich relativ zu seiner eigenen Orientierung immer nur geradeaus vorwärts bewegen kann. Des Weiteren wird die Annahme einer konstanten Geschwindigkeit des Roboters gemacht. Beide Annahmen gelten selbst für einen Roboter mit Differentialantrieb nicht immer. Selbst auf sehr gut geeigneten Oberflächen kann es passieren, dass ein Roboter seitwärts rutscht, oder die Geschwindigkeit während einer Drehung nicht konstant ist [EP04]. Um auch diese Bewegungen modellieren zu können, wird ein weiterer Parameter C für den Bewegungsanteil orthogonal zur primären Bewegungsrichtung eingeführt. Als Bewegungsrichtung von C wird dabei die Orthogonale gewählt, die aus der primären Bewegungsrichtung durch eine Drehung um 90° im mathematisch positiven Drehsinn entsteht. Trotzdem ist bei einer idealen, das heißt nicht fehlerbehafteten, Bewegung eines Roboters mit Differentialantrieb der Parameter C gleich 0. Deshalb ist es auch nicht nötig, die Odometriemessung um einen Parameter für die Seitwärtsbewegung zu erweitern. Die primäre Bewegungsrichtung wird wie schon im vorigen Modell als um den Winkel $T/2$ gedreht zur Anfangsrichtung angenommen. Das heißt die Bewegung des Roboters wird dadurch modelliert, dass der Roboter sich erst um den Winkel $T/2$ dreht, dann eine Strecke der Länge D vorwärts und eine Strecke der Länge C seitwärts fährt und schließlich wieder um den Winkel $T/2$ dreht (siehe Abbildung 3.10). Die Gleichungen für das Bewegungsmodell stellen Austin I. Eliazar

und Ronald Parr wie folgt auf [EP04]:

$$\begin{aligned} b'_x &= a_x + D\cos\left(\alpha + \frac{T}{2}\right) + C\cos\left(\alpha + \frac{T+\pi}{2}\right) \\ b'_y &= a_y + D\sin\left(\alpha + \frac{T}{2}\right) + C\sin\left(\alpha + \frac{T+\pi}{2}\right) \\ \beta' &= \alpha + T \mod 2\pi \end{aligned} \tag{3.52}$$

Dabei sind D, C und T die Erwartungswerte der normalverteilten Zufallsvariablen $\mathcal{D}$, $\mathcal{C}$ und $\mathcal{T}$,

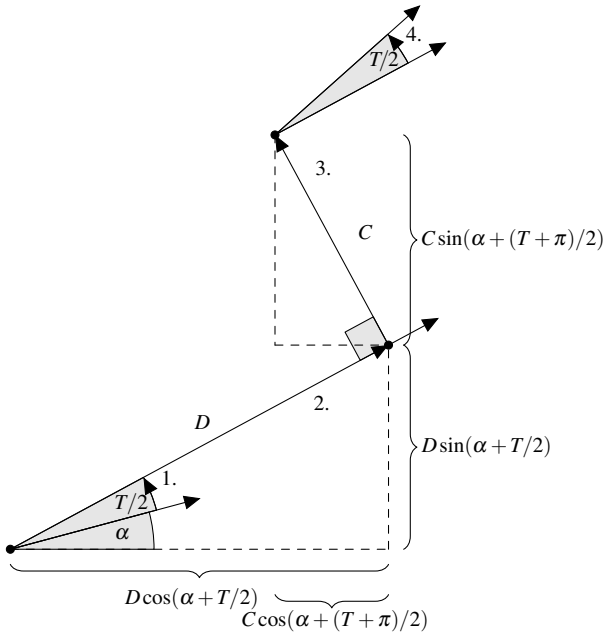


Abb. 3.10: In diesem Bewegungsmodell dreht der Roboter zuerst um $T/2$, fährt dann die Strecke D geradeaus, anschließend wird er um die Strecke C seitwärts verschoben und schließlich dreht er noch einmal um den Winkel $T/2$.

die wiederum von d und t abhängen. Eliazar und Parr gehen davon aus, dass die Mittelwerte der Verteilungen linear und die Varianzen quadratisch mit d und t skalieren. Es ergeben sich folgende Verteilungen:

$$\begin{aligned} \mathcal{D} &\sim \mathcal{N}(d\mu_{D_d} + t\mu_{D_t},\, d^2\sigma^2_{D_d} + t^2\sigma^2_{D_t}) \\ \mathcal{C} &\sim \mathcal{N}(d\mu_{C_d} + t\mu_{C_t},\; d^2\sigma^2_{C_d} + t^2\sigma^2_{C_t}) \\ \mathcal{T} &\sim \mathcal{N}(d\mu_{T_d} + t\mu_{T_t},\; d^2\sigma^2_{T_d} + t^2\sigma^2_{T_t}) \end{aligned} \tag{3.53}$$

Die Parameter μ_{I_j} geben dabei den Einfluss des Odometriemesswerts j auf den Mittelwert der Verteilung von I an. Ebenso gibt $\sigma^2_{I_j}$ den Einfluss des Odometriemesswerts j auf die Varianz der Verteilung von I an.

Eine weitere, am Forschungszentrum Informatik am KIT entwickelte, Erweiterung wird nötig, wenn Roboter betrachtet werden sollen, die sich omnidirektional zu ihrer eigenen Orientierung bewegen können. In diesem Fall reichen die zwei Odometrieparameter d und t nicht mehr aus, um

alle möglichen idealen Roboterbewegungen zu beschreiben. Es wird zusätzlich ein dritter Parameter c eingeführt, so dass die Odometrie nun drei Werte d, c und t zurückliefert. Der Parameter d gibt die in der primären Bewegungsrichtung, c die in der zur primären Richtung orthogonalen Richtung zurückgelegte Strecke an. Die Gleichungen für das Bewegungsmodell sind identisch zu denen des letzten Modells (siehe Gleichungen 3.52), nur die Wahrscheinlichkeitsverteilungen von $\mathcal{D}$, $\mathcal{C}$ und $\mathcal{T}$ müssen an den neuen Odometrieparameter angepasst werden:

$$\begin{aligned}
\mathcal{D} &\sim \mathcal{N}(d\mu_{D_d} + c\mu_{D_c} + t\mu_{D_t},\, d^2\sigma^2_{D_d} + c^2\sigma^2_{D_c} + t^2\sigma^2_{D_t}) \\
\mathcal{C} &\sim \mathcal{N}(d\mu_{C_d} + c\mu_{C_c} + t\mu_{C_t},\, d^2\sigma^2_{C_d} + c^2\sigma^2_{C_c} + t^2\sigma^2_{C_t}) \\
\mathcal{T} &\sim \mathcal{N}(d\mu_{T_d} + c\mu_{T_c} + t\mu_{T_t},\, d^2\sigma^2_{T_d} + c^2\sigma^2_{T_c} + t^2\sigma^2_{T_t})
\end{aligned} \qquad [3.54]$$

Dieses allgemeine und damit auch flexible Bewegungsmodell wird im nächsten Abschnitt für die Modellierung der Faktoren für die Kalibrierung der Odometrie verwendet.

3.3.3.2.2 Modellierung der Odometriefaktoren im Faktorgraphen Die Odometriefaktoren im Faktorgraphen hängen von drei Zufallsvariablen ($\mathcal{A}$ für die erste Pose, $\mathcal{B}$ für die zweite Pose $\mathcal{P}$ für die Kalibrierungsparameter) und der Odometriemessung m ab. In dem Bewegungsmodell wird angenommen, dass die Messwerte in Form der Parameter d, c und t vorliegen. In der Praxis wird statt dessen oft die geschätzte zweite Roboterpose in absoluten Koordinaten angeben. In dieser Arbeit wird hingegen immer angenommen, dass die Messung die zweite Roboterpose relativ zum Koordinatensystem der ersten Pose a angibt. Dies hat den Vorteil, dass die Anfangspose des Roboters nicht bekannt sein muss. Außerdem lässt sich die relative Pose wie schon gesehen leicht aus der absoluten Pose und der letzten Roboterpose berechnen. Tatsächlich ist es im Normalfall nicht möglich, die tatsächlich gemessenen Impulse der Drehgeber auszulesen, da von den Sensoren stattdessen bereits fertige Schätzungen für die Roboterbewegung zurückgeben werden. Deshalb ist die praktische Nutzung des Faktoren aus Kapitel 3.3.3.1 oft nicht möglich.

Werden Posen als Bewegungen und damit als Elemente der Lie-Gruppe $\mathrm{SE}(2)$ betrachtet, so sind sowohl die normale Schreibweise einer Pose $a = \begin{pmatrix} a_x & a_y & \alpha \end{pmatrix}^T$ als auch die neue Form $\begin{pmatrix} d & c & t \end{pmatrix}^T$ Parametrisierungen des Elements $A = \begin{pmatrix} R_\alpha^T & -R_\alpha^T a_t \\ 0 & 1 \end{pmatrix}$ der Lie-Gruppe im $\mathbb{R}^3$. Die neue Parametrisierung wird in dieser Arbeit d-c-t-Parametrisierung genannt. Sie kann aus der Pose a einfach durch eine Rotation um $-\alpha/2$ berechnet werden.

$$\begin{pmatrix} d \\ c \\ t \end{pmatrix} = \begin{pmatrix} R_{\alpha/2}^T & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} a_t \\ \alpha \end{pmatrix} \qquad [3.55]$$

Das mit der d-c-t-Parametrisierung $\begin{pmatrix} d & c & t \end{pmatrix}^T$ beschriebene Element A der Lie-Gruppe ist dann

$$A = \begin{pmatrix} R_\alpha^T & -R_\alpha^T a_t \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} R_t^T & -R_{t/2}^T \begin{pmatrix} d \\ c \end{pmatrix} \\ 0 \qquad 1 \end{pmatrix}. \tag{3.56}$$

Mit Hilfe des Bewegungsmodells und der Kalibrierungsparameter wird aus einer Pose in d-c-t-Form eine neue gewichtete Pose $\begin{pmatrix} D & C & T \end{pmatrix}^T$ ebenfalls in d-c-t-Form berechnet. Die einzelnen Werte der neuen Pose sind die Erwartungswerte der Normalverteilungen in Gleichung 3.54. Der Erwartungswert P der Zufallsvariablen $\mathcal{P}$ für die Kalibrierungsparameter enthält dabei die Gewichtungsparameter μ_{I_j}, die die Mittelwerte der Normalverteilungen für $\mathcal{D}$, $\mathcal{C}$ und $\mathcal{T}$ bestimmen. Dies wird am Ende dieses Kapitels gezeigt.

$$P = (\mu_{D_d}, \mu_{D_c}, \mu_{D_t}, \mu_{C_d}, \mu_{C_c}, \mu_{C_t}, \mu_{T_d}, \mu_{T_c}, \mu_{T_t})^T. \tag{3.57}$$

Um die Funktion des Odometriefaktoren zu bestimmen, wird die wahrscheinlichste Pose, die mit dem Bewegungsmodell aus einer Messung m berechnet wird, bestimmt. Diese besteht gerade aus den Erwartungswerten von $\mathcal{D}$, $\mathcal{C}$ und $\mathcal{T}$:

$$\begin{aligned} D &= d\mu_{D_d} + c\mu_{D_c} + t\mu_{D_t} \\ C &= d\mu_{C_d} + c\mu_{C_c} + t\mu_{C_t} \\ T &= d\mu_{T_d} + c\mu_{T_c} + t\mu_{T_t} \end{aligned} \tag{3.58}$$

bzw.

$$\begin{pmatrix} D \\ C \\ T \end{pmatrix} = P' \cdot \begin{pmatrix} d \\ c \\ t \end{pmatrix} \quad \text{mit} \quad P' = \begin{pmatrix} \mu_{D_d} & \mu_{D_c} & \mu_{D_t} \\ \mu_{C_d} & \mu_{C_c} & \mu_{C_t} \\ \mu_{T_d} & \mu_{T_c} & \mu_{T_t} \end{pmatrix} \tag{3.59}$$

Anstatt nun mit Gleichung 3.52 eine neue Pose b' aus D, C, T und der letzten Pose a zu berechnen, wird aus der d-c-t-Form einfach wieder ein Element der Lie-Gruppe berechnet. Es repräsentiert die Bewegung des Roboters von der Pose a zu der geschätzten zweiten Pose b' und könnte, um die Pose b' zu berechnen, einfach von links mit dem Element A der Lie-Gruppe multipliziert werden.

Sei also $E \in \mathrm{SE}(2)$ das Element der Lie-Gruppe, das durch $\begin{pmatrix} D & C & T \end{pmatrix}^T$ repräsentiert wird. Dann gilt mit Gleichung 3.56:

$$E = \begin{pmatrix} R_T^T & -R_{T/2}^T \begin{pmatrix} D \\ C \end{pmatrix} \\ 0 \qquad 1 \end{pmatrix} = \begin{pmatrix} R^T_{\mu_{T_d} d + \mu_{T_c} c + \mu_{T_t} t} & -R^T_{(\mu_{T_d} d + \mu_{T_c} c + \mu_{T_t} t)/2} \begin{pmatrix} \mu_{D_d} d + \mu_{D_c} c + \mu_{D_t} t \\ \mu_{C_d} d + \mu_{C_c} c + \mu_{C_t} t \end{pmatrix} \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad 1 \end{pmatrix}. \tag{3.60}$$

Die Parametrisierung von E in Form einer Pose $e = \begin{pmatrix} e_x & e_y & \varepsilon \end{pmatrix}^T$ ergibt sich wie folgt:

$$e = \begin{pmatrix} e_x \\ e_y \\ \varepsilon \end{pmatrix} = \begin{pmatrix} R_{T/2} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} D \\ C \\ T \end{pmatrix} = \begin{pmatrix} D\cos(T/2) - C\sin(T/2) \\ D\sin(T/2) + C\cos(T/2) \\ T \end{pmatrix}. \tag{3.61}$$

Die Fehlerfunktion des Faktors, berechnet aus den Elementen der Lie-Gruppe, ist dann mit $A = \begin{pmatrix} R_\alpha^T & -R_\alpha^T a_t \\ 0 & 1 \end{pmatrix}$ und $B = \begin{pmatrix} R_\beta^T & -R_\beta^T b_t \\ 0 & 1 \end{pmatrix}$ aus SE(2):

$$\begin{aligned} E(A,B,P) &= BA^{-1}E^{-1} \\ &= \begin{pmatrix} R_\beta^T & -R_\beta^T b_t \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} R_\alpha & a_t \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} R_T & R_{T/2} \begin{pmatrix} D \\ C \end{pmatrix} \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} R_{\beta-\alpha-T}^T & R_{\beta-\alpha-T/2}^T \begin{pmatrix} D \\ C \end{pmatrix} - R_\beta^T (b_t - a_t) \\ 0 & 1 \end{pmatrix}. \end{aligned} \tag{3.62}$$

Die zu $E(A,B,P)$ gehörende Pose $t = \begin{pmatrix} t_t \\ \theta \end{pmatrix}$ kann folgendermaßen berechnet werden:

$$R_\theta^T = R_{\beta-\alpha-T}^T \quad \Rightarrow \theta = \beta - \alpha - T \tag{3.63}$$

und

$$\begin{aligned} & -R_\theta^T \cdot t_t = R_{\beta-\alpha-T/2}^T \begin{pmatrix} D \\ C \end{pmatrix} - R_\beta^T (b_t - a_t) \\ \Rightarrow \quad & -R_{\beta-\alpha-T}^T \cdot t_t = R_{\beta-\alpha-T/2}^T \begin{pmatrix} D \\ C \end{pmatrix} - R_\beta^T (b_t - a_t) \\ \Rightarrow \quad & t_t = R_{\beta-\alpha-T} \cdot R_\beta^T (b_t - a_t) - R_{\beta-\alpha-T} \cdot R_{\beta-\alpha-T/2}^T \begin{pmatrix} D \\ C \end{pmatrix} \\ \Rightarrow \quad & t_t = R_{\alpha+T}^T (b_t - a_t) - R_{T/2}^T \begin{pmatrix} D \\ C \end{pmatrix} \end{aligned} \tag{3.64}$$

Es folgt

$$t = \begin{pmatrix} t_t \\ \theta \end{pmatrix} = \begin{pmatrix} R_{\alpha+T}^T (b_t - a_t) - R_{T/2}^T \begin{pmatrix} D \\ C \end{pmatrix} \\ \beta - \alpha - T \end{pmatrix}. \tag{3.65}$$

Zur Überprüfung wird die Fehlerfunktion auch noch in Form der Posen a, b und e berechnet.

Vergleichen mit Gleichung 3.65 ergibt, dass die berechneten Posen übereinstimmen.

$$\begin{aligned} E(a,b,p) &= h(a,b) \ominus e = (b \ominus a) \ominus e \\ &= \begin{pmatrix} R_{\varepsilon+\alpha}^T & 0 \\ 0 & 1 \end{pmatrix} (b-a) - \begin{pmatrix} R_{\varepsilon}^T & 0 \\ 0 & 1 \end{pmatrix} e \\ &= \begin{pmatrix} R_{T+\alpha}^T & 0 \\ 0 & 1 \end{pmatrix} (b-a) - \begin{pmatrix} R_{T/2}^T & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} D \\ C \\ T \end{pmatrix}. \end{aligned} \tag{3.66}$$

Die Funktion des Faktors ergibt sich nun wieder aus der Likelihoodfunktion:

$$f(a,b,p) = L(a,b,p;m) = L(a,b;e) = \exp\left\{ -\frac{1}{2} \| (b \ominus a) \ominus e \|_{\Sigma}^2 \right\} \tag{3.67}$$

Die Jacobimatrizen werden wie schon bei dem Faktor für die Sensorkalibrierung getrennt für A, B und P berechnet. Die Jacobimatrizen für J_a für A und J_b für B lassen sich mit dem gleichen Verfahren wie vorher folgendermaßen berechnen.

$$J_b = \frac{\partial BA^{-1}E^{-1}}{\partial B} = \frac{\partial}{\partial \delta} \left((\exp(\delta) \cdot B) \cdot A^{-1}E^{-1} \right) \big|_{\delta=0} = I_3 \tag{3.68}$$

$$\begin{aligned} J_a = \frac{\partial BA^{-1}E^{-1}}{\partial A} &= \frac{\partial}{\partial \delta} \left(B \cdot (\exp(\delta) \cdot A)^{-1} \cdot E^{-1} \right) \big|_{\delta=0} \\ &= \frac{\partial}{\partial \delta} \left(BA^{-1} \cdot \exp(-\delta) \cdot E^{-1} \right) \big|_{\delta=0} \\ &= \frac{\partial}{\partial \delta} \left(\exp(-\mathrm{Adj}_{BA^{-1}} \delta) \cdot BA^{-1}E^{-1} \right) \big|_{\delta=0} \\ &= -\mathrm{Adj}_{BA^{-1}} \end{aligned} \tag{3.69}$$

Für die Jacobimatrix J_p für den Kalibrierungsparameter P wird zuerst die Kettenregel angewendet, da E von P abhängt.

$$\frac{\partial BA^{-1}E^{-1}}{\partial P} = \frac{\partial BA^{-1}X^{-1}}{\partial X}(E) \cdot \frac{E}{\partial P} \tag{3.70}$$

Der erste Faktor kann wie folgt berechnet werden:

$$\begin{aligned} \frac{\partial BA^{-1}X^{-1}}{\partial X}(E) &= \frac{\partial}{\partial \delta} \left(BA^{-1} \cdot (\exp(\delta) \cdot E)^{-1} \right) \big|_{\delta=0} \\ &= \frac{\partial}{\partial \delta} \left(BA^{-1}E^{-1} \cdot \exp(-\delta) \right) \big|_{\delta=0} \\ &= \frac{\partial}{\partial \delta} \left(\exp(-\mathrm{Adj}_{BA^{-1}E^{-1}} \delta) \cdot BA^{-1}E^{-1} \right) \big|_{\delta=0} \end{aligned} \tag{3.71}$$

$$= -\mathrm{Adj}_{BA^{-1}E^{-1}} \in \mathbb{R}^{3\times 3}$$

Da P kein Element der Lie-Gruppe $\mathrm{SE}(2)$ ist, kann der zweite Faktor $\frac{E}{\partial P}$ in Gleichung 3.70 nicht auf die selbe Art und Weise berechnet werden. Die Funktion

$$f : \mathbb{R}^9 \mapsto \mathrm{SE}(2) \quad \text{mit} \quad f(P) = E \tag{3.72}$$

wendet das Bewegungsmodell auf die gegebene Messung an. Da die Funktion so nicht differenziert werden kann, wird für den Wertebereich $\mathrm{SE}(2)$ eine Parametrisierung gewählt, in diesem Fall die d-c-t-Parametrisierung. Dann sieht die gleiche Funktion folgendermaßen aus:

$$f : \mathbb{R}^9 \mapsto \mathbb{R}^3 \quad \text{mit} \quad f(P) = \begin{pmatrix} D \\ C \\ T \end{pmatrix} = \begin{pmatrix} d\mu_{D_d} + c\mu_{D_c} + t\mu_{D_t} \\ d\mu_{C_d} + c\mu_{C_c} + t\mu_{C_t} \\ d\mu_{T_d} + c\mu_{T_c} + t\mu_{T_t} \end{pmatrix} \tag{3.73}$$

Diese Funktion kann nun ganz normal partiell abgeleitet werden. Es ergibt sich

$$\frac{E}{\partial P} = \frac{f(P)}{\partial P} = \begin{pmatrix} d & c & t & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & d & c & t & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & d & c & t \end{pmatrix}. \tag{3.74}$$

Die Jacobimatrix der gesamten Funktion ergibt sich dann nach der Kettenregel aus dem Produkt der einzelnen Ableitungen.

$$J_p = \frac{\partial BA^{-1}E^{-1}}{\partial P} = -\mathrm{Adj}_{BA^{-1}E^{-1}} \cdot \begin{pmatrix} d & c & t & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & d & c & t & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & d & c & t \end{pmatrix} \tag{3.75}$$

Soll die Jacobimatrix für bestimmte Werte für P und die Messung $m = \begin{pmatrix} m_t \\ \nu \end{pmatrix}$ ausgerechnet werden, so müssen nur die Werte nacheinander in die folgenden Gleichungen eingesetzt werden.

$$\begin{pmatrix} d \\ c \\ t \end{pmatrix} = \begin{pmatrix} R^T_{\nu/2} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} m_t \\ \nu \end{pmatrix} \tag{3.76}$$

$$\begin{pmatrix} D \\ C \\ T \end{pmatrix} = P' \cdot \begin{pmatrix} d \\ c \\ t \end{pmatrix} \quad \text{mit} \quad P' = \begin{pmatrix} \mu_{D_d} & \mu_{D_c} & \mu_{D_t} \\ \mu_{C_d} & \mu_{C_c} & \mu_{C_t} \\ \mu_{T_d} & \mu_{T_c} & \mu_{T_t} \end{pmatrix} \tag{3.77}$$

$$-\operatorname{Adj}_{BA^{-1}E^{-1}} = \begin{pmatrix} -R^T_{\beta-\alpha-T} & \begin{matrix} -v_2 \\ v_1 \end{matrix} \\ \begin{matrix} 0 & -1 \end{matrix} \end{pmatrix} \quad \text{mit} \quad v = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = R^T_{\beta-\alpha-T/2} \begin{pmatrix} D \\ C \end{pmatrix} - R^T_\beta (b_t - a_t) \tag{3.78}$$

Als Ergebnis der Optimierung des Faktorgraphen wird für jede Zufallsvariable der wahrscheinlichste Wert sowie eine Kovarianz für diesen Wert zurück gegeben. Im Falle des Kalibrierungsparameters P heißt dies, dass nicht nur der Wert des Parameters geschätzt werden kann, sondern auch seine Varianz. Diese kann wiederum direkt im Bewegungsmodell verwendet werden (siehe Gleichung 3.54). Es bleibt noch zu zeigen, wie die Kovarianzmatrix, die nach der Optimierung für P zurückgegeben wird, mit den σ_{I_j} aus dem Bewegungsmodell zusammenhängt. Laut den Gleichungen des Bewegungsmodells sind die Zufallsvariablen $\mathcal{D}$, $\mathcal{C}$ und $\mathcal{T}$ gleichverteilt. Die Werte d, c und t sind konstante Messungen. Die Zufallsvariablen $\mathcal{P}_i$ für die einzelnen Kalibrierungsparameter beschreiben jeweils den Einfluss eines einzelnen Odometriemesswerts auf die Verteilung von $\mathcal{D}, \mathcal{C}$ und $\mathcal{T}$ ($\mathcal{P} = (\mathcal{P}_1 \dots \mathcal{P}_9)^T$, $i \in \{1, \dots 9\}$). Es gilt:

$$\begin{aligned} \mathcal{D} &\sim d\mathcal{P}_1 + c\mathcal{P}_2 + t\mathcal{P}_3 \\ \mathcal{C} &\sim d\mathcal{P}_4 + c\mathcal{P}_5 + t\mathcal{P}_6 \\ \mathcal{T} &\sim d\mathcal{P}_7 + c\mathcal{P}_8 + t\mathcal{P}_9 \end{aligned} \tag{3.79}$$

Der Satz von Cramér sagt aus, dass zwei unabhängige Zufallsvariablen, deren Summe normalverteilt ist, auch normalverteilt sind. Da die Messwerte konstant und unabhängig voneinander sind, sind auch die Einflüsse verschiedener Messwerte auf eine Verteilung unabhängig voneinander. Das heißt die $\mathcal{P}_i$, die eine Verteilung beeinflussen, sind untereinander stochastisch unabhängig. Damit kann der Satz von Cramér angewendet werden und die Zufallsvariablen $\mathcal{P}_i$ sind ebenfalls normalverteilt. Es gelten die folgenden Regeln für zwei unabhängige Zufallsvariablen $\mathcal{X}$, $\mathcal{Y}$ und Konstanten α, β:

$$\mathcal{X} \sim \mathcal{N}(\mu_x, \sigma_x^2), \quad \mathcal{Y} \sim \mathcal{N}(\mu_y, \sigma_y^2) \quad \Rightarrow \quad \mathcal{X} + \mathcal{Y} \sim \mathcal{N}(\mu_x + \mu_y, \sigma_x^2 + \sigma_y^2) \tag{3.80}$$

$$\mathcal{X} \sim \mathcal{N}(\mu, \sigma^2), \quad \mathcal{Y} = \alpha\mathcal{X} + \beta \quad \Rightarrow \quad \mathcal{Y} \sim \mathcal{N}(\alpha\mu + \beta, \alpha^2\sigma^2) \tag{3.81}$$

Damit und aus der Semantik des Bewegungsmodells kann nun auch die genaue Verteilung der $\mathcal{P}_i$ angegeben werden:

$$\begin{aligned} \mathcal{P}_1 &\sim \mathcal{N}(\mu_{D_d}, \sigma^2_{D_d}) \\ \mathcal{P}_2 &\sim \mathcal{N}(\mu_{D_c}, \sigma^2_{D_c}) \\ &\vdots \\ \mathcal{P}_9 &\sim \mathcal{N}(\mu_{T_t}, \sigma^2_{T_t}) \end{aligned} \tag{3.82}$$

In der Kovarianzmatrix, die für $\mathcal{P}$ nach der Optimierung ausgegeben wird, entspricht die Varianz eines Parameters $\mathcal{P}_i$ dem i-ten Eintrag auf der Diagonalen. Die Diagonalelemente der Kovarianzmatrix sind also gerade die im Bewegungsmodell verwendeten Parameter σ_{I_j}.

3.4 Zusammenfassung

In diesem Kapitel wurden zuerst verschiedene Ansätze zur Lösung des SLAM-Problems verglichen. Daraufhin wurde sich in dieser Arbeit dafür entschieden, einen graphbasierten Ansatz zu verwenden und diesen um die Möglichkeit zur automatischen Kalibrierung zu erweitern. Die Entscheidung gegen einen Ansatz mit Partikelfiltern beruhte dabei hauptsächlich auf der leichten Erweiterbarkeit des graphbasierten Ansatzes.

Im Anschluss an den Vergleich der Methoden wurde das verwendete graphische Modell, der Faktorgraph, eingeführt und die Struktur des Graphen für das SLAM-Problem erläutert. Für die Kalibrierung mobiler Roboter wurden neue Faktoren eingeführt und diese in die bereits vorhandene Struktur eingebettet.

Die mathematische Modellierung der Faktoren wurde ausführlich beschrieben und es wurden zwei verschiedene Möglichkeiten für die Modellierung der Odometriekalibrierung vorgestellt. Bei der ersten Möglichkeit wird für einen Roboter mit Differentialantrieb die Bewegung direkt aus den gemessenen Encoderticks hergeleitet, bei der zweiten Möglichkeit wird ein allgemeines Bewegungsmodell als Grundlage angenommen.

Die erste Variante hat den Vorteil, dass die Berechnung auf den physikalischen Eigenschaften des Roboters beruht und die Kalibrierungsparameter leicht an dem Roboter nachgemessen werden können. Jedoch kann der Faktor nur für Roboter mit Differentialantrieb verwendet werden und es muss möglich sein, die Encoderticks auszulesen. Dies ist jedoch bei den verwendeten Sensoren oft nicht der Fall. Die zweite Variante hat den Vorteil, dass sie für verschiedene Odometrien angewendet werden kann. Dafür wird ein stochastisches Modell verwendet, das nicht direkt aus der Roboterodometrie abgeleitet werden kann. Die Güte des Modells beeinflusst dabei auch die Güte der berechneten Parameter.

Da ein Ziel dieser Arbeit die Wiederverwendbarkeit der Implementierung für verschiedene Roboterplattformen mit unterschiedlichen Kinematiken und Sensoren ist, wird in der Umsetzung, die im nächsten Kapitel beschrieben wird, die zweite Variante für die Kalibrierung der Odometrie verwendet.

4 Umsetzung

Für die automatische Kalibrierung mobiler Roboter wurde ein Algorithmus entwickelt, der gleichzeitig das SLAM-Problem löst und Odometrie und Sensoren kalibriert. Dafür erstellt der Algorithmus aus den von dem Roboter gelieferten Messwerten von Odometrie- und Umfeldsensoren einen Faktorgraphen wie er in Kapitel 3 beschrieben wurde. Dieser Faktorgraph beinhaltet dabei die neu implementierten Faktoren für die Kalibrierung. Ist der Faktorgraph für die Fahrt eines Roboters erstellt, können die wahrscheinlichste Trajektorie, Umgebungskarte und die wahrscheinlichsten Kalibrierungsparameter mittels eines gegebenen Optimierungsverfahrens bestimmt werden.

Dieses Kapitel beschreibt den Aufbau des Algorithmus und die Vorgehensweise bei der Erstellung genauer. Zuerst wird das Szenario, in dem der Algorithmus verwendet werden kann, beschrieben. Dabei werden auch die zugehörigen Entscheidungen begründet. Danach wird in Kapitel 4.2 die genaue Funktionsweise erläutert und auf Konfigurationsmöglichkeiten eingegangen. Außerdem werden für die Funktion des SLAM-Verfahrens und die Performanz des Algorithmus wichtige Punkte aufgezeigt. Da ein Fokus bei der Implementierung auf der Wiederverwendbarkeit des Algorithmus für verschiedene Roboterplattformen liegt, beschäftigt sich Kapitel 4.3 explizit mit diesem Thema. Die Ergebnisse der Verwendung des Algorithmus mit verschiedenen Robotern findet sich in Kapitel 5. Zum Abschluss werden die in der Implementierung verwendeten Bibliotheken und Frameworks kurz vorgestellt und die Entscheidung für diese begründet.

4.1 Einsatzszenario

Bereits in der Einleitung dieser Arbeit wurde die Wichtigkeit der Kalibrierung mobiler Roboter hervorgehoben und begründet. Weiter wurde festgestellt, dass Abnutzungserscheinungen am Roboter sowie der Einsatz in Umgebungen mit unterschiedlichen Untergrundeigenschaften eine häufige Rekalibrierung des Roboters nötig machen können. Deshalb ist es vor allem wichtig, dass der Kalibrierungsalgorithmus ohne großen Aufwand eingesetzt werden kann. Faktoren, die diesen Aufwand beeinflussen, sind vor allem nötige manuell auszuführende Schritte, wie zum Beispiel das Vorbereiten einer bestimmten Umgebung, das Abfahren einer bestimmten Trajektorie mit dem Roboter oder das Anpassen des Algorithmus für die eigene Roboterplattform. Bei dem entwickelten Algorithmus wurde daher darauf geachtet, dass fast kein Vorwissen über den Roboter oder die Umgebung nötig ist. Die einzige Ausnahme besteht darin, dass initiale Schätzungen für die Kalibrierungsparameter benötigt werden, die allerdings nicht sehr genau sein müssen. So sollte die Position und Ausrichtung der Umgebungssensoren auf ca. 15 cm beziehungsweise 15° genau

gemessen werden. Ein erstes Bewegungsmodell, dessen Güte in allen in Kapitel 5.3 beschriebenen Versuchen ausreichend war, liefern gängige Odometriesensoren automatisch mit, indem sie bereits fertige Schätzungen für die Bewegung des Roboters zurück geben.

Des Weiteren kann der Algorithmus auf eine beliebige Robotertrajektorie angewendet werden, es wird nur eine Aufzeichnung aller Sensormessungen benötigt. Trotzdem werden die Ergebnisse der Kalibrierung bei einer längeren und verschlungeneren Trajektorie tendenziell besser sein, da in dem Fall die bei der Optimierung verwendeten Daten umfangreicher und variabler sind. Trotzdem konnten in Simulationen, die in Kapitel 5.2 beschrieben sind, die Parameter bereits bei einer sehr kurzen Trajektorie korrekt kalibriert werden. Auch ist der Algorithmus vielseitig und ohne große Konfiguration einsetzbar. Genauer wird darauf in Kapitel 4.3 eingegangen.

Bei dem Entwurf eines Kalibrierungsalgorithmus ist grundsätzlich zu entscheiden, ob dieser online eingesetzt werden soll oder ob ein Offline-Algorithmus vorzuziehen ist. In dieser Arbeit wurde sich dafür entschieden, einen Offline-Algorithmus zu verwenden. Bei der Verwendung eines Online-Algorithmus würden die Kalibrierungsparameter während der Fahrt zu jedem Zeitpunkt neu optimiert. Dies stellt einerseits einen großen Rechenaufwand dar, andererseits ist es aber nicht sinnvoll, wenn zu jedem Punkt auf einer Trajektorie unterschiedliche Kalibrierungsparameter gelten. Falls sich zu einem beliebigen Zeitpunkt während der Ausführung des Online-Algorithmus die optimierten Kalibrierungsparameter im Vergleich zum letzten Zeitpunkt ändern, sollten diese geänderten Parameter auch auf die vorigen Berechnungen angewendet werden. Das heißt, dass ein Filter, wie er in den meisten Lösungen des Online-SLAM-Problems verwendet wird, allein nicht ausreicht, sondern der Algorithmus über die gesamte Trajektorie glätten muss. Dies ist äquivalent dazu, dass der Algorithmus zu jedem Zeitpunkt das Offline-SLAM-Problem mit der Erweiterung für die Kalibrierung lösen muss. Erste Verfahren, die Offline-SLAM online lösen können, gibt es bereits, allerdings stellt sich die Frage, ob Online-Kalibrierung überhaupt nötig oder wünschenswert ist.

Ein großer Vorteil der Online-Kalibrierung ist dabei natürlich, dass sie ohne weiteren Aufwand im Hintergrund abläuft. Außerdem ist, wie bereits argumentiert, eine häufige Kalibrierung sinnvoll, allerdings ist es offensichtlich nicht sinnvoll, für jeden Punkt auf der Robotertrajektorie unterschiedliche Kalibrierungsparameter zuzulassen. Statt dessen sollten die Parameter in einem Zeitraum, in dem sich die Eigenschaften des Roboters selbst und der Umgebung nicht wesentlich verändern, konstant bleiben. Ein Online-Algorithmus müsste also in die Berechnung der aktuellen Kalibrierungsparameter jeweils eine bestimmte Historie einbeziehen. Dabei stellt sich nun die Frage, wie umfassend die gewählte Historie sein soll. Ein Beispiel wäre, dass die Kalibrierung immer nur mit den Daten der aktuellen Fahrt des Roboters bestimmt wird. Dies ist einerseits sinnvoll, da es vorstellbar ist, dass der Roboter mit jeder neuen Fahrt eine neue Umgebung antrifft. Die Daten, die in der alten Umgebung aufgenommen wurden, in die aktuelle Berechnung einzubeziehen, kann zu schlechteren Ergebnissen führen. Auf der anderen Seite kann sich das Beschränken der Historie auf die aktuelle Fahrt auch nachteilig auswirken, so zum Beispiel bei mehreren kurzen Fahrten in der gleichen Umgebung mit kurzem zeitlichen Abstand. Bei einem Offline-Algorithmus hat der

Nutzer des Roboters hingegen die volle Kontrolle darüber, mit welchen Daten kalibriert wird. Ein weiterer Vorteil eines nur bei Bedarf ausgeführten Offline-Algorithmus besteht darin, dass die berechneten Kalibrierungsparameter durchaus auch in anderen SLAM-Verfahren verwendet werden können. So kann beispielsweise die Kalibrierung mit graphbasiertem SLAM wie hier vorgestellt vorgenommen werden, dann aber für die weiteren Fahrten ein entsprechend eingestellter Partikelfilter verwendet werden, falls dieser für die aktuelle Situation besser geeignet ist.

Die Kalibrierung ausschließlich online vorzunehmen, kann also auch Nachteile haben, allerdings wäre es natürlich wünschenswert, einen Algorithmus mit der Möglichkeit zum Online-Einsatz zur Verfügung zu haben. Aufgrund des erwarteten hohen Rechenaufwands und der Einschränkung auf Offline-SLAM-Verfahren mit der Möglichkeit zum Online-Einsatz, wurde sich in dieser Arbeit ausschließlich für einen Offline-Kalibrierungsalgorithmus entschieden.

Als Eingabe benötigt der Algorithmus ausschließlich eine Aufzeichnung der Messwerte eines Roboters in einem gängigen Format sowie die initialen Schätzungen der Kalibrierungsparameter. Als Ausgabe werden die optimalen Kalibrierungsparameter sowie die mit diesen Parametern berechnete optimale Robotertrajektorie und Umgebungskarte zurück gegeben. Außerdem gibt der Algorithmus die Kovarianzen für alle Werte mit aus, sodass auch eine Aussage über die Qualität der gefundenen Parameter getroffen werden kann.

Voraussetzungen für den Einsatz des Algorithmus sind nur, dass der Roboter zumindest mit einem Odometriesensor und einem Sensor zur Erkennung der Umgebung ausgestattet ist. Getestet wird die Kalibrierung in Kapitel 5 mit verschiedenen Kinematiken und verschieden vielen Umgebungssensoren.

4.2 Aufbau des Algorithmus

In diesem Kapitel wird die genaue Funktionsweise des Algorithmus erläutert. Dafür wird zuerst ein Algorithmus, der das SLAM-Problem ohne Kalibrierung löst, eingeführt. Anschließend werden nötige Anpassungen und Änderungen für die Kalibrierung erläutert.

Der Ablauf des Algorithmus lässt sich grundlegend in vier einzelne Aufgaben aufteilen, die nacheinander abgearbeitet werden:

1. Initialisierung und Konfiguration
2. Inkrementelle Erstellung des Faktorgraphen
3. Optimierung des Faktorgraphen
4. Ausgabe der Ergebnisse

Die Hauptaufgabe ist dabei die Erstellung des Faktorgraphen. Abbildung 4.1 zeigt beispielhaft,

wie die Struktur eines solchen Faktorgraphen für das reine SLAM-Problem aussehen kann. Die Optimierung des Faktorgraphen kann von einem vorhandenen Optimierer mit Hilfe des Levenberg-Marquardt-Algorithmus vorgenommen werden.

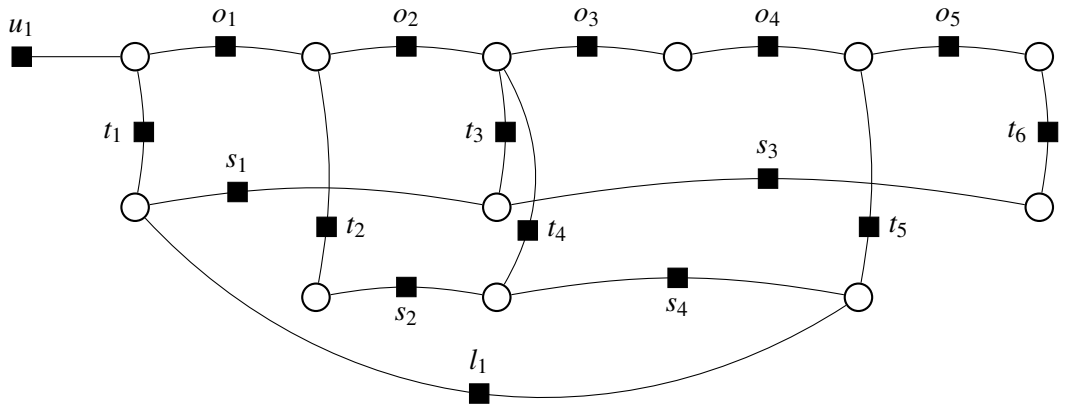


Abb. 4.1: Diese Abbildung zeigt ein Beispiel für einen Faktorgraphen, der mit dem Algorithmus für SLAM ohne Kalibrierung erstellt wurde.

Als Eingabe benötigt der Algorithmus nur eine Aufzeichnung der Sensormessungen während der Fahrt des Roboters. Optional können noch einige Konfigurationsparameter übergeben werden, deren Einfluss auf den Algorithmus später erklärt werden. Initialisiert wird der Faktorgraph mit einem ersten unären Faktor, der wie in Kapitel 3.3.1 beschrieben, die A-Priori-Verteilung für die erste Roboterpose enthält (siehe Faktor u_1 in Abbildung 4.1). Da in unseren Anwendungsszenarien alle weiteren Faktoren die Roboter- und Sensorposen nur relativ zueinander beschreiben, wird mit diesem ersten Faktor der Startpunkt des Roboters auf den Ursprung und die Orientierung auf $0°$ festgelegt.

Um die weiteren Faktoren dem Faktorgraphen hinzuzufügen, liest der Algorithmus nun jeweils eine Sensormessung ein. Ist die Sensormessung eine Odometriemessung kann, wie in Kapitel 3.3.1 beschrieben, ein binärer Faktor in den Faktorgraphen eingefügt werden, der die Abhängigkeit der aktuellen Roboterpose von der vorhergehenden beschreibt (siehe Faktoren o_1 bis o_5 in Abbildung 4.1). Für die Kalibrierung werden diese Faktoren dann durch Faktoren mit einer weiteren Abhängigkeit von den Odometrie-Kalibrierungsparametern ersetzt. Für die neue Zufallsvariable, welche die aktuelle Roboterpose im Faktorgraphen repräsentiert, wird dann mit Hilfe der initialen Schätzung der letzten Roboterpose und der aktuellen Messung eine erste Schätzung berechnet und diese als Anfangswert für den Knoten in den Faktorgraphen eingefügt.

Stammt die Messung hingegen von einem Umgebungssensor, sollen zwei neue Faktoren in den Faktorgraphen eingefügt werden. Der erste bringt die Position des Sensors in Bezug zu der aktuellen Roboterpose, der zweite beschreibt die aktuelle Position des Sensors abhängig von seiner letzten Position und der aktuellen Messung. Für den ersten Faktor muss zuerst bestimmt werden, welcher Knoten im Faktorgraph die Roboterpose zum Zeitpunkt der Sensormessung beschreibt. Der hier vorgestellte Algorithmus verwendet dafür einfach den zuletzt eingefügten Knoten, der

eine Roboterpose beschreibt. Diesem Vorgehen liegt die Annahme zu Grunde, dass sehr häufig Messungen gemacht werden und dass die Messungen in chronologischer Reihenfolge aufgezeichnet werden. Da alle Messungen üblicherweise mit einem Zeitstempel gespeichert werden, könnte auch dieser Zeitstempel verwendet werden, um den passenden Knoten zu finden. Dies kann allerdings zu weiteren Problemen führen, falls die Uhren der einzelnen Sensoren nicht synchronisiert sind. Zwischen den Knoten für Roboter- und Sensorpose wird dann ein binärer Faktor eingefügt, der unabhängig von der aktuellen Messung ist und die in diesem Fall a priori bekannte Rototranslation zwischen den Posen beschreibt (siehe Faktoren t_1 bis t_6 in Abbildung 4.1).

Danach wird versucht, mittels Scan Matching eine Beziehung zu der vorigen Messung des gleichen Sensors herzustellen. Ist das Scan Matching erfolgreich, kann ein binärer Faktor zwischen den Knoten für die beiden Messungen im Faktorgraphen eingefügt werden (siehe Faktoren s_1 bis s_4 in Abbildung 4.1). Die initiale Schätzung für den neuen Knoten, der die aktuelle Sensorpose beschreibt, kann, falls ein Match gefunden wurde, aus diesem Match und dem Schätzwert für die letzte Sensorpose berechnet werden. Ist dies nicht der Fall, wird der Wert aus der Schätzung für die zugehörige Roboterpose und der gegebenen Rototranslation zwischen Roboter- und Sensorpose berechnet.

Schließlich wird noch für alle weiter zurückliegenden Messungen der Umgebungssensoren versucht ein Match zu finden (Loop Closing). Ist dies erfolgreich wird für den jeweiligen zu der Messung gehörenden Knoten ein binärer Faktor zwischen diesem und dem aktuellen Knoten eingefügt (siehe Faktor l_1 in Abbildung 4.1). Die Abbildung Algorithmus 1 zeigt den beschriebenen Algorithmus in Pseudocode.

4.2.1 Einfügen der Kalibrierung und weitere Anpassungen des grundlegenden Algorithmus

Damit der im vorigen Kapitel vorgestellte Algorithmus für das SLAM-Problem ohne Kalibrierung funktioniert und um seine Effizienz zu steigern, können noch einige kleinere Anpassungen vorgenommen werden. In Abbildung 4.1 ist an den Faktoren o_3 und o_4 ersichtlich, dass es passieren kann, dass mehrere Odometriefaktoren aufeinander folgen, ohne dass die dazwischenliegenden Knoten mit weiteren Faktoren verbunden sind. Diese Ketten von Odometriefaktoren können in der Praxis sehr lang werden und sind unnötig, da sie sich zu einem einzigen Faktor zusammenfassen lassen. Um dies im Algorithmus umzusetzen, werden jeweils erst beim Lesen des Messung eines Umgebungssensors der Knoten für die zugehörige Roboterpose und der Odometriefaktor eingefügt. Dafür werden die zwischen zwei Umgebungsmessungen gemessenen Odometriewerte aufsummiert (siehe Zeile 6 in Algorithmus 2).

Eine weitere Anpassung wird nötig, wenn der Roboter während seiner Fahrt eine Pause einlegt. Die Sensoren messen in diesem Zeitraum weiter und es werden auch weiter Knoten in den Faktorgraphen eingefügt. Dies stellt nicht nur eine Verschwendung von Speicher- und Rechenkapazitä-

Algorithmus 1 : Erster Algorithmus für SLAM ohne Kalibrierung

Data : Logfile l

```
1  read configuration;
2  initialize Factorgraph f;
3  foreach mesurement p in l do
4      if p is odometry measurement then
5          set r := last node for a robot pose in f;
6          insert odometry factor f(r, r'; m) into f;                    /* r' is a new node */
7          compute the initial estimate for r' from r and p;
8      else if p is range sensor measurement then
9          set r := last node for a robot pose in f;
10         set s := last node in f that corresponds to a measurement of the current sensor;
11         insert translation factor f(r', s') into f;                   /* s' is a new node */
12         try scan matching p with the measurement corresponding to s;
13         if match m is successful then
14             compute the initial estimate for s' from s and m;
15             insert scan matching factor f(s, s'; m) into f;
16         else
17             compute the initial estimate for s' from translation from r';
18         end
19         foreach previous range sensor node s'' do
20             try scan matching p with the measurement corresponding to s'';
21             if match m' is successful then
22                 insert loop closing factor f(s'', s'; m')into f;
23             end
24         end
25     end
26 end
27 optimize f;
28 print results;
```

ten dar, sondern kann auch zu Fehlern führen, da während der Optimierung jeder dieser Faktoren mit einer bestimmten Wahrscheinlichkeit von dem geschätzten Wert abweichen darf. Diese Wahrscheinlichkeiten summieren sich über alle Faktoren, die während dieser Pause erstellt wurden, auf. Daher ist es sinnvoll, einen Schwellwert für die minimale Bewegung des Roboters einzuführen. Bleibt die aus den Odometriemessungen berechnete Bewegung des Roboters unter dem Schwellwert, werden keine Faktoren in den Faktorgraphen eingefügt (siehe Zeile 7 in Algorithmus 2).

Um den Algorithmus für die Kalibrierung anzupassen, müssen statt den binären Faktoren zwischen den Knoten zweier Roboterposen die Faktoren zur Kalibrierung der Odometrie, die in Kapitel 3.3.3.2.2 eingeführt wurden, verwendet werden. Außerdem müssen die Faktoren, die die Sensorpose relativ zur Roboterpose darstellen, durch die Faktoren zur Kalibrierung der Sensoren (siehe Kapitel 3.3.2) ersetzt werden. Die Abbildung Algorithmus 2 zeigt den Algorithmus in der aktuellen, angepassten Form. Mit den initialen Schätzungen für die Kalibrierungsknoten sind neue Eingabewerte des Algorithmus hinzugekommen. Ein Schritt des Algorithmus wird zur Ver-

Algorithmus 2 : Angepasster und optimierter Algorithmus für SLAM mit Kalibrierung

Data : Logfile l, threshold for robot motion o_{min}, initial estimate for the odometry calibration node c_o and the sensor calibration nodes

```
1  read configuration;
2  initialize Factorgraph f with unary factor to first node r;
3  set relative odometry motion variable o = (0,0,0);
4  foreach mesurement p in l do
5      if p is odometry measurement then
6          add relative motion of p to o;
7      else if p is range sensor measurement and o > o_min then
8          insert odometry calibration factor f(r, r', c_o; o) into f;        /* r' is a new node */
9          compute the initial estimate for r' from r and o;
10         set c_s := sensor calibration node for the current sensor;
11         insert sensor calibration factor f(r', s', c_s) into f;            /* s' is a new node */
12         set s := last node in f that corresponds to a measurement of the current sensor;
13         try scan matching p with the measurement corresponding to s;
14         if match m is successful then
15             compute the initial estimate for s' from s and m;
16             insert scan matching factor f(s, s'; m) into f;
17         else
18             compute the initial estimate for s' from c_s and r';
19         end
20         foreach previous range sensor node s'' do
21             try scan matching p with the measurement corresponding to s'';
22             if match m' is successful then
23                 insert loop closing factor f(s'', s'; m')into f;
24             end
25         end
26         set o := (0,0,0), r := r';
27     end
28 end
29 optimize f;
30 print results;
```

anschaulichung in Abbildung 4.2 dargestellt.

4.3 Wiederverwendbarkeit des Algorithmus

Besondere Aufmerksamkeit galt bei dem Entwurf des Algorithmus der Wiederverwendbarkeit und Vielseitigkeit. So soll durch diese Arbeit nicht nur eine Kalibrierungsmethode für eine spezielle Roboterplattform gefunden werden, sondern statt dessen ein Algorithmus implementiert werden, der die Kalibrierung während dem SLAM-Verfahren für möglichst viele unterschiedliche Roboter möglich macht.

Ein erstes Problem bei der Verwendung des Algorithmus mit verschiedenen Robotern ist dabei das

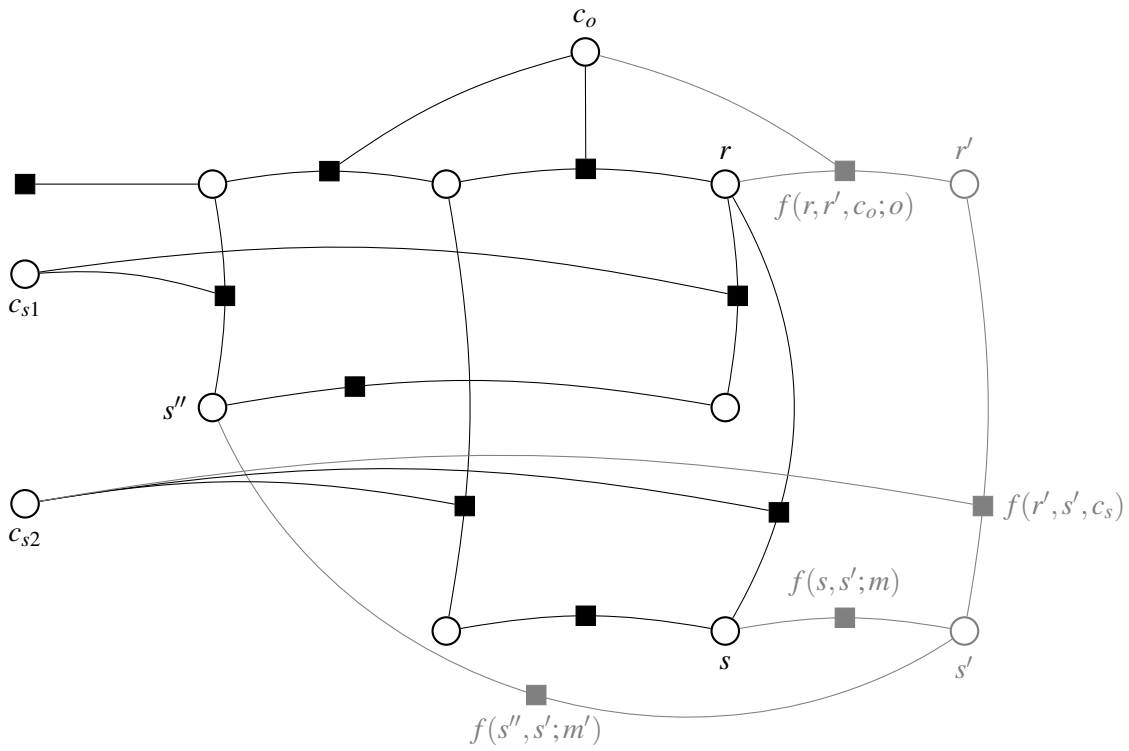


Abb. 4.2: Diese Abbildung zeigt in grau die in einem Schritt des Algorithmus 2 hinzugefügten Knoten und Faktoren.

Einlesen der Messungen des Roboters. Es wird hierbei davon ausgegangen, dass die Messungen in einer Logdatei zur Verfügung stehen. Für diese Logdateien gibt es allerdings kein allgemein anerkanntes Format. Der Algorithmus kann allerdings mit Hilfe eines am Forschungszentrum für Informatik am Karlsruher Institut für Technologie entwickelten Tools Logdateien verschiedener Formate öffnen. Darunter befinden sich unter anderem das von der Carnegie Mellon University (CMU) verwendete Format CARMEN und das an der University of Southern California (USC) verwendete Format. Des Weiteren kann die Logdatei des bekannten und auch in Kapitel 5 verwendeten Datensatzes aus einem Forschungslabor von Intel eingelesen werden. Funktionalität, die das Lesen weiterer Formate möglich macht, kann leicht hinzugefügt werden.

Ein weiterer Punkt bei den Einsatzmöglichkeiten des Algorithmus sind Anzahl und Art der von dem Roboter verwendeten Sensoren. Grundvoraussetzung ist hierbei, dass der Roboter über die Möglichkeit verfügt, seine Bewegung mittels Odometrie zu messen. Damit die Kalibrierung der Odometrie funktionieren kann, wird zusätzlich noch mindestens ein externer Sensor für die Wahrnehmung der Umgebung benötigt.

In allen betrachteten Fällen, wird die Odometriemessung für einen bestimmten Zeitpunkt in einem meist mehrdimensionalen Messvektor zurückgegeben. Damit aus einem Odometriemesswert die Bewegung des Roboters abgelesen werden kann, muss aus diesem die relative Bewegung direkt

berechnet werden können. Diese relative Bewegung in Form einer Translation und einer Rotation wird von dem Algorithmus als Eingabe erwartet. Liefert der Odometriesensor nicht direkt diesen Wert, kann wie oben erwähnt einfach eigene Funktionalität zum Einlesen und Konvertieren der Messwerte implementiert werden.

Verschiedenste Roboterkinematiken, das heißt Fortbewegungsmöglichkeiten, bestimmt durch den mechanischen Aufbau des Roboters, können dabei verwendet werden. Das in Kapitel 3.3.3.2.1 eingeführte Bewegungsmodell abstrahiert die mechanischen Einzelheiten und ist dadurch universell einsetzbar. Lediglich bei den initialen Schätzwerten für die Kalibrierung der Odometrie wird das Wissen über die verwendete Kinematik eingebracht. So werden bei einem Roboter mit Differentialantrieb laterale Bewegungen initial eher niedrig gewichtet, während dies bei einem sich omnidirektional bewegenden Roboter nicht der Fall sein muss.

Im Fall der Umgebungssensoren wurde in dieser Arbeit ausschließlich mit 2D-Abstandssensoren gearbeitet. Einer Erweiterung für andere Umgebungssensoren ist allerdings leicht möglich, so lange eine Möglichkeit zum Scan Matching von zwei Sensormessungen besteht. Der Algorithmus kann mit einer beliebigen Anzahl Umgebungssensoren umgehen. Die verwendete Zahl wird dabei direkt aus der Logdatei gelesen und muss deshalb nicht extra angegeben werden.

Des Weiteren ist für das korrekte Erstellen des Faktorgraphen unerheblich, in welcher Reihenfolge oder wie regelmäßig die verschiedenen Sensoren ihr Messungen machen, so lange die Odometriemessungen häufig genug erfolgen. Diese Einschränkung liegt daran, dass in dem Algorithmus die Sensorposen jeweils mit der Roboterpose zur Zeit der letzten Odometriemessung assoziiert werden. Im Normalfall stellt dies zwar kein Problem dar, falls es aber doch zu schlechten Ergebnissen führen sollte, könnten die Zeitstempel der Messungen verwendet und damit einer Sensorpose das gewichtete Mittel zweier zeitlich benachbarter Roboterposen zugewiesen werden. Dies wurde bis jetzt nicht gemacht, da dies wiederum zu Problemen führen kann, falls die internen Uhren der Sensoren nicht synchronisiert sind.

4.4 Beschreibung der verwendeten Bibliotheken und Frameworks

Der in den vorigen Abschnitten beschriebene Algorithmus für SLAM mit integrierter Kalibrierung, wurde in der Programmiersprache C++ implementiert. Dabei wurden einige externe Bibliotheken verwendet. Außerdem ist der Algorithmus in ein Softwarepaket für SLAM, das am Forschungszentrum Informatik am Karlsruher Institut für Technologie entwickelt wurde, eingebunden. Dieses Softwarepaket mit dem Namen icl_slam stellt unter anderem Funktionalität zum Einlesen der Logdateien, Schnittstellen für externe Bibliotheken, Klassen für Bewegungsmodelle und Sensormessungen und vieles mehr zur Verfügung. Viele der Funktionen und Klassen aus icl_slam wurden bei der Erstellung des Algorithmus verwendet.

Für die Erstellung und Optimierung der Faktorgraphen wurde die von Frank Dellaert et al. am

Georgia Institute of Technology entwickelte Bibliothek GTSAM verwendet [Del12]. Diese bietet sowohl fertige Faktoren für bestimmte Anwendungsszenarien als auch ein Interface mit dessen Hilfe eigene Faktoren implementiert werden können. Des Weiteren stehen verschiedene Methoden zur Optimierung eines fertig aufgebauten Faktorgraphen zur Verfügung. Der in diesem Algorithmus verwendete Optimierer implementiert den Levenberg-Marquardt-Algorithmus, der seinerseits auf dem Gauß-Newton-Verfahren und damit auf der Methode der kleinsten Quadrate aufbaut. Für Matrizenrechnungen wird in dem Algorithmus sowie in GTSAM die Eigen-Bibliothek verwendet.

Das Scan Matching wird in dem implementierten Algorithmus von dem Canonical Scan Matcher, kurz CSM, von Andrea Censi übernommen. Der verwendete Algorithmus ist sowohl in der Lage Scan Matching zwischen verschiedenen 2D-Umgebungsscans durchzuführen, als auch eine Konfidenz zu einem gefundenen Match anzugeben [Cen08]. Unter anderem wird CSM auch in dem bekannten Framework Robot Operation System (ROS) verwendet.

So wie der Algorithmus für SLAM und Kalibrierung bis jetzt beschrieben wurde, muss für das Loop Closing für jeden Scan Scan Matching mit jedem vorhergehenden Scan durchgeführt werden (siehe Zeile 21 in Algorithmus 2). Dies ist extrem rechenaufwändig und wird deshalb in der Praxis so nicht gemacht. Statt dessen werden mit Hilfe der FLIRTLib- und GFLIP-Bibliotheken von Gian Diego Tipaldi und Kai O. Arras für jeden Scan Features berechnet, mit denen ein schnellerer Vergleich der Scans möglich ist [TA10]. Anhand der Features werden Kandidaten für Loop Closing aus allen vorhergehenden Scans heraus gefiltert und das tatsächliche Scan Matching mit CSM dann nur noch für diese Kandidaten durchgeführt.

4.5 Zusammenfassung

In diesem Kapitel wurde der entwickelte Algorithmus für SLAM mit integrierter Kalibrierung ausführlich beschrieben. Die Entscheidung für die Entwicklung eines offline ablaufenden Algorithmus im Gegensatz zu einer kontinuierlichen Kalibrierung mittels eines Online-Algorithmus wurde begründet und es wurden allgemeine Voraussetzungen für den Einsatz des Algorithmus aufgezeigt.

Mit Hilfe eines ersten naiven Algorithmus für SLAM mit Hilfe von Faktorgraphen wurde gezeigt, wo Anpassungen, die für die Performanz aber auch die Funktionalität des Algorithmus wichtig sind, gemacht wurden. Zum Beispiel werden Ketten von Odometriefaktoren zu einem einzelnen Faktor zusammengefasst oder Messungen ignoriert, wenn sich der Roboter seit der letzten Messung fast gar nicht bewegt hat. Außerdem wurde beschrieben, welche Anpassungen an dem allgemeinen SLAM-Algorithmus für die Integration der Kalibrierungsparameter nötig sind. Dabei fällt auf, dass diese Anpassung eigentlich nur die Verwendung der neuen Kalibrierungsfaktoren und das Einführen einiger Konfigurationsparameter beinhaltet.

Genauer eingegangen wurde auf den Schwerpunkt Wiederverwendbarkeit. Dabei wurde gezeigt, dass der Algorithmus mit einer Vielzahl von Roboterplattformen einsetzbar ist und beschrieben, welche Anpassungen zur Verwendung mit weiteren Sensortypen oder Logdateiformaten vorgenommen werden können.

Auf technische Details der Implementierung wurde nicht eingegangen, allerdings wurden die verwendeten Bibliotheken und Frameworks kurz aufgezählt und vorgestellt.

5 Experimente und Ergebnisse

Im vorangegangenen Kapitel wurde die Implementierung des entwickelten Algorithmus vorgestellt, in diesem Abschnitt der Arbeit wird diese Implementierung getestet und die Ergebnisse werden evaluiert. Dafür werden einerseits Datensätze, die mit verschiedenen Robotern in unterschiedlichen Umgebungen aufgenommen wurden, verwendet und andererseits Datensätze, die aus einer Simulation entstanden sind. Mit Hilfe der Simulation können Datensätze aufgenommen werden, die nur noch minimale Fehler enthalten. Werden in diesem Szenario falsche Kalibrierungswerte eingestellt, enthält der resultierende Datensatz nur die Fehler, die durch die fehlerhafte Kalibrierung entstehen. So kann der Algorithmus unabhängig von nicht-systematischen Fehlern getestet werden. Gleichzeitig können die korrekten Daten aus der Simulation zum Vergleich betrachtet werden.

Nachdem auf diese Weise die Funktionalität des Algorithmus unter idealen Bedingungen nachgewiesen wird, kann das Verhalten mit verrauschten Daten aus realen Anwendungsszenarien getestet werden. Dabei werden in diesem Kapitel Datensätze von Robotern mit unterschiedlichen Odometrien verwendet, um die Vielseitigkeit des Algorithmus unter Beweis zu stellen.

5.1 Methodik

Bevor die durchgeführten Experimente und ihre Ergebnisse in den Abschnitten 5.2 und 5.3 beschrieben werden, wird in diesem Abschnitt die verwendete Methodik und die Faktoren für die Auswertung der Ergebnisse erläutert.

Für die Experimente wurden zwei verschiedene Datenquellen verwendet. Einerseits aus Simulationen gewonnene Datensätze, die wenig nicht-systematische Fehler enthalten und andererseits Logdateien, die während real durchgeführten Fahrten mit verschiedenen Robotern gewonnen wurden. Mit Hilfe, der aus Simulationen gewonnenen Daten kann nachgewiesen werden, dass der Algorithmus unter idealen Bedingungen die gewünschten Ergebnisse liefert und geeignet ist, die Roboterodometrie sowie die Laserscanner zu kalibrieren. Im Gegensatz dazu dienen die Datensätze von tatsächlich durchgeführten Fahrten mit verschiedenen Robotern dazu, einerseits die Robustheit des Algorithmus bei der Verwendung verrauschter Daten sowie andererseits die Vielseitigkeit des Algorithmus im Hinblick auf unterschiedliche Roboter und Odometrien zu untersuchen. So wurde einerseits ein Roboter mit zwei 2D-Laserscannern verwendet, der sich omnidirektional fortbewegen kann, sowie andererseits ein Auto mit drei 2D-Laserscannern, dessen Odometrie komplett anders

aussieht. Außerdem wurde zur allgemeinen Vergleichbarkeit noch ein populärer Datensatz, der von Dieter Fox im Jahr 2003 in einem Forschungslabor der Intel Corporation in Seattle aufgenommen wurde, verwendet. Dort fährt ein Roboter mit Differentialantrieb und nur einem 2D-Laserscanner durch das Forschungslabor und vollendet dabei mehrere Schleifen. Dieser Datensatz ist für die vorliegende Arbeit von besonderem Interesse, da aufgrund der initial sehr schlecht kalibrierten Odometrie der Effekt der vorgenommen Kalibirerung besonders deutlich wird.

Der entwickelte Algorithmus beinhaltet die Möglichkeit, die Kalibrierung komplett abzuschalten oder Odometrie und beziehungsweise oder Umgebungssensoren zu kalibrieren. In den Experimenten wird diese Funktion verwendet und jeweils die einzelne Kalibrierung von Odometrie und Sensoren sowie die gemeinsame Kalibrierung getestet. Für einen Vergleich der ermittelten Trajektorien und Punktwolken kann das Ergebnis des SLAM-Algorithmus ohne Kalibrierung herangezogen werden. Da aber dieses Ergebnis bei den verwendeten Datensätzen überwiegend bereits sehr gut ist, wird in dieser Arbeit für die Evaluation der Kalibrierungsparameter ein anderer Vergleich gezogen. Dafür wird aus dem jeweiligen Datensatz, nur auf Basis der Odometriedaten, eine Trajektorie und eine Punktwolke berechnet. Da dabei weder ein Scan Matcher noch eine Optimierung der Trajektorie zum Einsatz kommt, ist die Qualität des Ergebnisses ausschließlich von der Kalibrierung und den nicht-systematischen Fehlern des Datensatzes abhängig. Nachdem mit Hilfe des Algorithmus eine neue Kalibrierung berechnet wurde, wird diese verwendet, um wiederum die Logdatei, wie oben beschrieben, unter Verwendung der neuen Kalibrierungsparameter auszuwerten.

Eine weitere Frage, die für die Auswertung der Experimente dieser Arbeit beantwortet werden muss, ist die, wie die Ergebnisse zweier Optimierungen verglichen werden können. Im Fall der Simulationen ist dies sehr einfach. Mit den gegebenen optimalen Kalibrierungsparametern, kann die Abweichung von diesen optimalen Werten als gutes Maß für die Güte der Kalibrierung herangezogen werden. Jedoch beeinflussen nicht alle Fehler in der Kalibrierung die Güte der resultierenden Trajektorie und Punktwolke gleich stark. Beispielsweise kann schon ein leichter Rechtsdrall eines Roboters deutlich größere Auswirkungen auf die Trajektorie haben, als wenn der Roboter auf geraden Strecken immer etwas weiter fährt als gemessen wird. Der Absolute Trajectory Error (ATE) ist ein häufig genutztes Fehlermaß, das direkt die optimierte Trajektorie mit der korrekten Trajektorie vergleicht und einen Fehlerwert ausgibt. Dieser Fehler entspricht dem arithmetischen Mittel der Abweichungen von der korrekten Trajektorie.

$$e_t = \frac{1}{n} \sum_{t=1}^{n} \|x_t - y_t\|_2 \qquad [5.1]$$

Dabei bezeichnet e_t den ATE, n die Anzahl der Punkte der Trajektorie, $x_t \in \mathbb{R}^2$ den Punkt der korrekten Trajektorie zum Zeitpunkt t und $y_t \in \mathbb{R}^2$ den Punkt in der optimierten Trajektorie zum gleichen Zeitpunkt. Wenn nur die Umgebungssensoren optimiert werden, ändert sich die Trajektorie jedoch nicht. Für diesen Fall wird in dieser Arbeit der Absolute Point Cloud Error (APCE) eingeführt. Bei diesem Fehlermaß werden die Distanzen zwischen den Punkten der optimierten

Punktwolke und den Punkten der korrekten Punktwolke gemittelt.

$$e_m = \frac{1}{n \cdot m} \sum_{t=1}^{n} \sum_{i=1}^{m} \|x_{t_i} - y_{t_i}\|_2 \quad [5.2]$$

Dabei ist n die Anzahl der Sensormessungen, m die Anzahl der Messpunkte einer Sensormessung und x_{t_i} und y_{t_i} sind Punkte der korrekten beziehungsweise der optimierten Punktwolke. Zusätzlich können für die optimierte Trajektorie sowie die optimierte Punktwolke die Standardabweichung, sowie die Quartile der Abweichung von den korrekten Daten angegeben werden.

Bei den realen Datensätzen ist der Vergleich zweier Optimierungen nicht so einfach, da die optimale Trajektorie und Umgebungskarte nicht bekannt sind. Wird ein Bild der Umgebungskarte mit eingezeichneter Trajektorie betrachtet, fällt es jedoch häufig leicht, intuitiv zu entscheiden welches Ergebnis besser ist. Für diese intuitive Entscheidung werden automatisch verschiedene Kriterien verwendet, die hier formalisiert werden sollen. Die folgende Liste enthält die, in der vorliegenden Arbeit verwendeten, Kriterien zum Vergleich zweier Ergebnisse bestehend jeweils aus Trajektorie und Umgebungskarte.

- Kanten in der Karte sind klar anstatt verschwommen und erscheinen nicht mehrfach in kleinem Abstand
- Gerade Kanten in der Umgebung werden durch gerade Strecken in der Karte wiedergegeben
- Bei dem Schluss einer Schleife in der Trajektorie passen auch die zugehörigen lokalen Kartenausschnitte zusammen
- Die Trajektorie ist fortlaufend und vollzieht keine größeren Sprünge von einem Zeitpunkt zum nächsten
- Die Trajektorie verläuft nicht in einem Zick-Zack-Kurs

Des Weiteren kann die Güte der Kalibrierungsparameter auch daran gemessen werden, wie schnell das Optimierungsverfahren im einfachen SLAM-Algorithmus konvergiert, wenn einmal die optimierten und einmal die vorher bekannten Kalibrierungsparameter verwendet werden. Auch der finale Fehler gegen den das Optimierungsverfahren konvergiert, kann ein Anhaltspunkt für die Güte der Kalibrierung sein.

Schließlich wird zur Vermeidung von Overfitting noch ein Experiment durchgeführt bei dem die Kalibrierung mit Hilfe der ersten Hälfte der Logdatei durchgeführt wird und die zweite Hälfte zur Evaluation verwendet wird (siehe Kapitel 5.3.4.1). So soll gezeigt werden, dass die Kalibrierungsparameter nicht nur für genau die Trajektorie, für die sie optimiert wurden, gelten, sondern auch bei weiteren Fahrten in ähnlichen Umgebungen verwendet werden können.

5.2 Simulation

Die Datensätze, die aus der Simulation gewonnen wurden, werden verwendet, um nachzuweisen, dass der Algorithmus tatsächlich die Laserscanner sowie die Odometrie korrekt kalibrieren kann. Dafür wird versucht, möglichst alle Fehler auch einfaches Messrauschen zu eliminieren und so eine möglichst perfekte Logdatei für die Bewegung eines Roboters zu erzeugen.

Mit dem, am Forschungszentrum Informatik des Karlsruher Instituts für Technologie (FZI) entwickelten, Tool MCA2 (Modular Controller Architecture) wird dafür ein Roboter manuell durch eine als Karte vorgegebene Umgebung gesteuert. Dabei werden die Werte, die vom Roboter gemessen werden sollten, aufgezeichnet. Wird aus dieser Logdatei nur auf Basis der Odometriemessungen eine Trajektorie und eine Punktwolke zur Beschreibung der Umgebung berechnet, so entspricht diese sehr genau der tatsächlich abgefahren Trajektorie und der verwendeten Umgebungskarte. Jedoch finden sich auch bei den, von MCA2 als korrekte Werte ausgegebenen Daten, noch kleinere Fehler und ein leichtes Rauschen in den Abstandsmessungen. Diese Restfehler konnten bei den verwendeten Datensätzen leider nicht behoben werden.

Um nun die Kalibrierung zu testen, können die korrekten Einträge der Logdatei so umgerechnet werden, dass sie die zu erwartenden Messungen bei einer fehlerhaften Kalibrierung des Roboters wiedergeben. Wird beispielsweise die Odometrie fehlerhaft kalibriert, so bleiben die Abstandsmessungen zwar gleich, allerdings verändern sich die, durch die Odometrie geschätzten, Roboterposen entsprechend der neuen Kalibrierung. Im Fall inkorrekt kalibrierter Umgebungssensoren bleiben Odometriewerte und Abstandsmessungen gleich. Nur die zusätzlich zu den Abstandsmessungen angegebenen relativen Sensorposen müssen angepasst werden. Mit einer so präparierten Logdatei kann nun die Kalibrierung von Odometrie und Umgebungssensoren einzeln sowie auch beides zusammen getestet werden.

Für die Erstellung der Logdateien wurde die Simulation einer Roboterplattform verwendet, mit der auch ein realer, im nächsten Kapitel verwendeter, Datensatz aufgezeichnet wurde. Der Roboter verwendet zwei Laserscanner als Abstandssensoren und kann sich mit Hilfe von Mecanum-Rädern omnidirektional bewegen. Da die Odometriekalibrierung insbesondere auch für omnidirektional fahrende Roboter geeignet sein soll, wurde der Roboter für die Simulation so gewählt.

Die Karte, die in Abbildung 5.1 zu sehen ist, spiegelt eine Etage eines Bürogebäudes wider und wurde so erstellt, dass sie eine große Schleife enthält. Eine Schleife sollte auch deshalb Teil der Testtrajektorie sein, da an dem Punkt, an dem die Schleife geschlossen wird, auch kleine Fehler in der Odometrie sehr gut erkannt werden können. Diese kleinen Fehler summieren sich über die Zeit hinweg auf und sorgen im Endeffekt dafür, dass der Endpunkt der Schleife nicht genau zu dem Startpunkt passt.

Für die Evaluation des entwickelten Algorithmus zur Kalibrierung mobiler Roboter werden zwei verschiedene aus Simulationen gewonnene Datensätze verwendet. Beide sind mit der gleichen

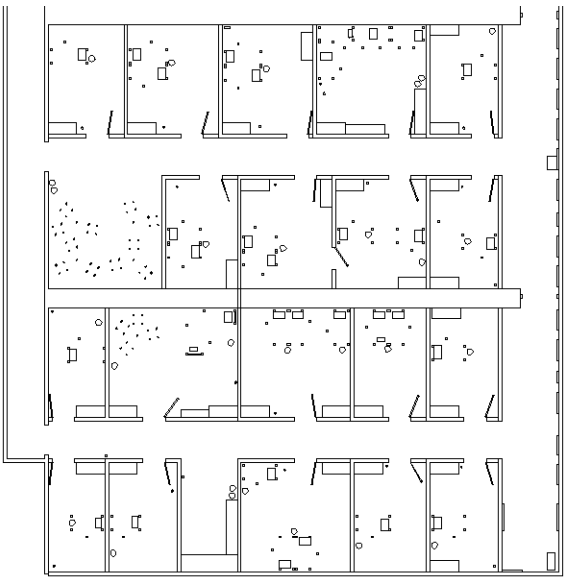


Abb. 5.1: Diese Abbildung zeigt den relevanten Ausschnitt aus der für die Simulation verwendeten Karte.

Karte und dem gleichem simulierten Roboter entstanden. Die Datensätze unterscheiden sich jedoch hinsichtlich der Länge und Form der gefahrenen Trajektorie. In dem ersten Datensatz ist die Trajektorie mit 58 Metern deutlich kürzer als die Trajektorie des zweiten Datensatzes mit 128 Metern Länge. In der Trajektorie des zweiten Datensatzes wird außerdem die Schleife in der Karte abgefahren. Weitere Eigenschaften der Datensätze sind in Tabelle 5.1 dargestellt. Der erste

	1. Datensatz	2. Datensatz
Dauer der Aufnahme	5 min 30 s	13 min 59 s
Gefahrene Strecke	58 m	128 m
Mittlere Geschwindigkeit	17,5 cm/s	15,3 cm/s
Anzahl aller Messungen	30870	91153
Anzahl der Odometriemessungen	27612	86917
Frequenz der Odometriemessungen	84 Hz	104 Hz
Anzahl der Laserscanner	2	2
Anzahl der Laserscans	3258	4236
Frequenz der Laserscans	4,9 Hz	2,5 Hz

Tab. 5.1: Diese Tabelle stellt einige Eigenschaften der in den Versuchen verwendeten Datensätze dar.

Datensatz wurde hinzugenommen, um die Kalibrierung auch bei kleinen Datenmengen testen zu können.

Abbildungen 5.2 und 5.3 zeigen die korrekten Trajektorien und Umgebungskarten, wie diese von dem Simulationswerkzeug ausgegeben wurden. Es ist gut ersichtlich, dass gerade bei längeren Trajektorien die Messwerte nicht ganz frei von Fehlern sind.

Die Funktionalität des Algorithmus für das SLAM-Problem ohne Kalibrierung wird hier nicht mehr dediziert nachgewiesen. Graphbasierte Algorithmen für SLAM wurden in der Literatur bereits vielfach getestet (vgl. Kapitel 2.2.3). Das Hauptaugenmerk dieser Arbeit liegt zudem auf der

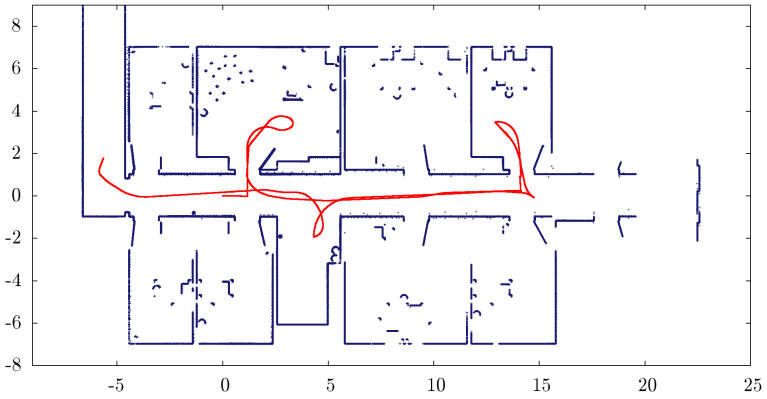


Abb. 5.2: Diese Abbildung zeigt für die erste verwendete Logdatei die nur aus den korrekten Odometriewerten berechnete Trajektorie und die zugehörige von den Abstandssensoren gemessene Umgebung in Form einer Punktwolke.

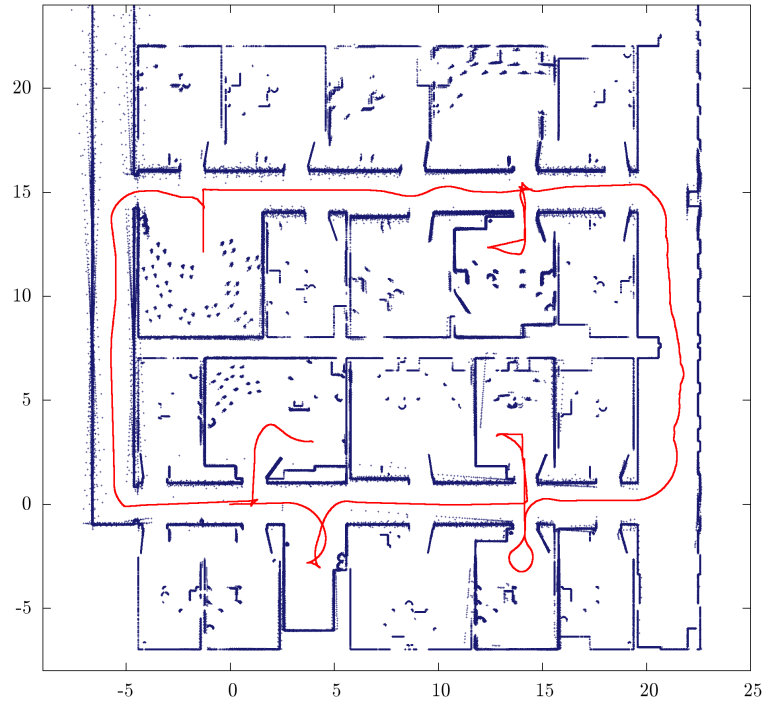


Abb. 5.3: Hier ist die korrekte Trajektorie und Punktwolke für die Umgebung wie sie aus der zweiten verwendeten Logdatei konstruiert werden kann, dargestellt. Gut erkennbar ist das leichte Rauschen in der Umgebungskarte, das existiert, obwohl die Werte von der Simulation als korrekt ausgegeben wurden.

Anpassung des Algorithmus für die Kalibrierung. Trotzdem wurde zum Vergleich der Werte auch mehrfach der reine SLAM-Algorithmus verwendet. In den beschriebenen Fällen funktionierte der Algorithmus sehr gut und konnte auch noch ziemlich schlechte initiale Kalibrierungen durch die Optimierung ausgleichen.

5.2.1 Kalibrierung der Umgebungssensoren

Für die Kalibrierung der Umgebungssensoren wurden zuerst die korrekten Werte der Logdateien so umgerechnet, dass sie der Ausgabe mit fehlerhafter Kalibrierung der Umgebungssensoren entsprechen. Für die erste Simulationsdatei war die korrekte Pose des ersten Laserscanner während der Simulation $\begin{pmatrix} 0{,}33856 & 0{,}31212 & -44{,}14056 \end{pmatrix}^T$ und die des zweiten Laserscanners $\begin{pmatrix} -0{,}33856 & -0{,}31212 & 135{,}85944 \end{pmatrix}^T$. Die ersten beiden Werte werden dabei in Metern, die Orientierung des Scanners wird als Abweichung zur Orientierung des Roboters in Grad angegeben. Als schlechte initiale Kalibrierung wurden für die Laserscanner die Posen $\begin{pmatrix} 0{,}4 & 0{,}22 & -51{,}6 \end{pmatrix}^T$ und $\begin{pmatrix} -0{,}25 & -0{,}4 & 143{,}2 \end{pmatrix}^T$ gewählt. Die Werte weichen zufällig nach oben oder unten von den optimalen Werten ab aber jeweils mindestens um 5° bei der Orientierung und um 5 cm bei den restlichen Werten. Im Normalfall sollte es kein Problem darstellen, initiale Werte, die höchstens um 5 cm beziehungsweise 5° von den korrekten Werten abweichen, manuell am Roboter zu messen.

Abbildung 5.4 zeigt die, mit den fehlerhaften Kalibrierungswerten berechnete, Logdatei. Der Absolute Point Cloud Error (APCE) im Vergleich zu den korrekten Daten, beträgt ungefähr 0,5, der ATE ist 0, da sich die Trajektorie durch die Kalibrierung nicht geändert hat. Die, von dem entwickelten Algorithmus optimierten, Kalibrierungsparameter weichen von den tatsächlichen Scannerposen um weniger als ein Zehntel Millimeter beziehungsweise weniger als ein Hundertstel Grad bei der Orientierung ab. Tabelle 5.2 zeigt die verschiedenen Werte der Kalibrierungsparameter und ihre Differenzen zu den optimalen Werten. Mit den optimierten Kalibrierungsparametern wird nun

	Sensor 1			Sensor 2		
	x_1	y_1	θ_1	x_2	y_2	θ_2
Optimal	0,33856	0,31212	-44,14056	-0,33856	-0,31212	135,85944
Initial gegeben	0,4	0,22	-51,6	-0,25	-0,4	143,2
Abweichung	0,06144	0,09212	7,45944	0,08856	0,08788	7,34056
Kalibriert	0,33854	0,31206	-44,14101	-0,33851	-0,31207	135,86167
Abweichung	0,00002	0,00006	0,00045	0,00005	0,00005	0,00223

Tab. 5.2: Die Tabelle zeigt die optimalen, die intial eingestellten und die kalibrierten Posen für die beiden Laserscanner im ersten Versuch.

eine neue Punktwolke für die Umgebung berechnet. Diese lässt sich mit dem bloßen Auge nicht von der korrekten Punktwolke, wie sie in Abbildung 5.2 zu sehen ist, unterscheiden. Der APCE hat sich von einem initialen Wert von 0,5 auf $9{,}7 \cdot 10^{-5}$ verbessert. Eine detailliertere Aufschlüsselung

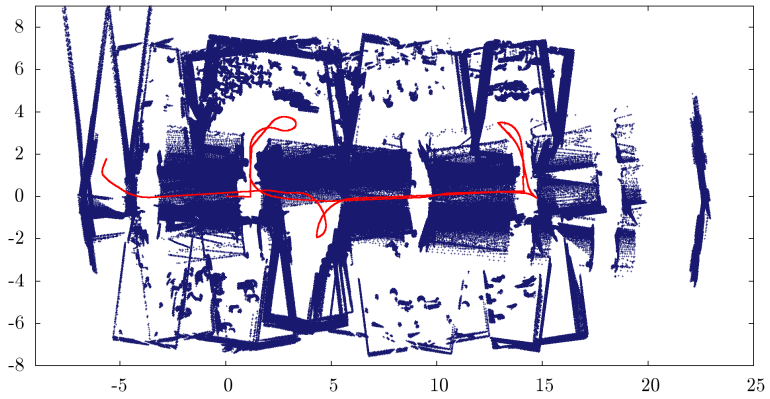


Abb. 5.4: Diese Abbildung zeigt die erste verwendete Logdatei, bei der Verwendung schlechter Kalibrierungsparameter für die Abstandssensoren. Offensichtlich hat sich an der Trajektorie nichts geändert, da nur die Kalibrierung der Laserscanner angepasst wurde.

der Fehler inklusive Standardabweichung und Quartile ist in Tabelle 5.5 dargestellt. Auch die Trajektorie und Punktwolke, die von dem SLAM-Algorithmus während der Kalibrierung ausgegeben werden, stimmen mit den korrekten Daten überein.

Da bei diesen initialen Werten die Ergebnisse so gut waren, wird die Kalibrierung noch einmal mit wesentlich schlechteren initialen Werten getestet. Diesmal wurden Werte gewählt, die mindestens 15 cm bzw. 15° von den optimalen Werten abweichen. Der APCE der resultierenden Punktwolke beträgt ca. 0,9. Die Ergebnisse des Versuchs sind in Tabelle 5.3 dargestellt. Auch mit den schlech-

	Sensor 1			Sensor 2		
	x_1	y_1	θ_1	x_2	y_2	θ_2
Optimal	0,33856	0,31212	-44,14056	-0,33856	-0,31212	135,85944
Initial gegeben	0,5	0,15	-27,0	-0,5	-0,5	120,0
Abweichung	0,16144	0,16212	17,14056	0,16144	0,18788	15,85944
Kalibriert	0,33851	0,31206	-44,14146	-0,33850	-0,31208	135,85997
Abweichung	0,00006	0,00005	0,00090	0,00007	0,00003	0,00053

Tab. 5.3: Die Tabelle zeigt das Ergebnis des zweiten Versuchs mit der ersten Logdatei. Dabei wurden initiale Werte verwendet, die um mindestens 15 cm beziehungsweise 15° von den optimalen Werten abweichen.

teren initialen Werten und einer kurzen Trajektorie kann der Algorithmus die Sensoren offensichtlich sehr gut kalibrieren. Der APCE verringert sich durch die Kalibrierung auf $8{,}4 \cdot 10^{-5}$. Weitere Werte sind in Tabelle 5.5 dargestellt.

Es stellt sich sogar heraus, dass die initialen Posen der Scanner in diesem Fall sogar beide auf $\begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T$ gesetzt werden können und die kalibrierten Parameter immer noch eine Abweichung von unter 0,1 mm beziehungsweise 0,01° haben. Der initiale APCE beträgt dabei 4,2, der der kalibrierten Daten $7{,}4 \cdot 10^{-5}$.

Auch für den zweiten Datensatz mit der längeren Trajektorie wurde die Kalibrierung getestet. Die verwendeten und die optimierten Werte finden sich in Tabelle 5.4. Abbildung 5.5 zeigt die Da-

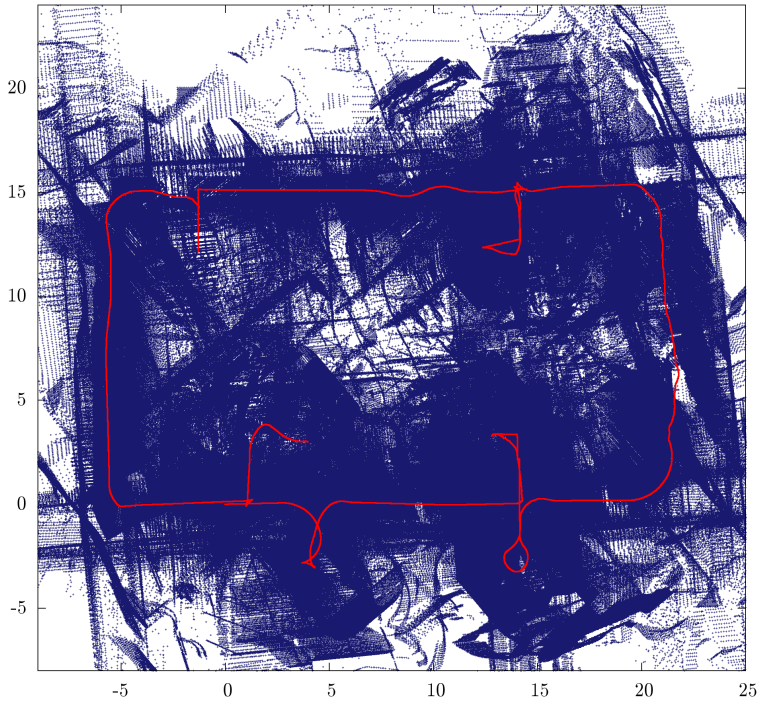


Abb. 5.5: Diese Abbildung visualisiert die zweite verwendete Logdatei mit schlechten Kalibrierungsparametern für die Abstandssensoren.

	Sensor 1			Sensor 2		
	x_1	y_1	θ_1	x_2	y_2	θ_2
Optimal	0,34	0,3	-40,0	-0,34	-0,3	160,0
Initial gegeben	0,25	0,2	-34,37747	-0,4	-0,22	154,69860
Abweichung	0,09	0,1	5,62253	0,06	0,08	5,30140
Kalibriert	0,33883	0,29762	-39,99844	-0,34074	-0,30230	159,99683
Abweichung	0,00117	0,00238	0,00156	0,00074	0,00230	0,00317

Tab. 5.4: Die Tabelle zeigt die Ergebnisse der Kalibrierung der beiden Laserscanner bei der Verwendung der zweiten aus der Simulation gewonnenen Logdatei.

ten mit einer fehlerhaften initialen Kalibrierung, die in diesem Fall wieder um mindestens 5 cm und 5° von den optimalen Werten abweicht und damit einen APCE von 4,4 aufweist. Die optimierte Kalibrierung ist ebenfalls wieder so gut, dass sich zwischen der Abbildung der korrekten Daten (siehe Abbildung 5.3) und einer Abbildung der neu kalibrierten Daten mit bloßem Auge kein Unterschied feststellen lässt. Der APCE der kalibrierten Daten beträgt 0,003. Dieser Wert ist deutlich schlechter, als bei dem Versuch mit dem ersten Datensatz. Dies könnte daran liegen, dass der zweite Datensatz ein deutlich stärkeres Rauschen enthält. Weitere Werte sind wieder in

Tabelle 5.5 aufgelistet.

Wird für diesen Datensatz jedoch als initiale Kalibrierung beider Laserscanner ebenfalls die Pose $\begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T$ gewählt, so lassen sich die Werte der Sensoren nicht mehr korrekt kalibrieren. Das Optimierungsverfahren konvergiert zwar nach wie vor, jedoch gegen ein lokales Minimum. Die zurückgegebenen Kalibrierungsparameter weichen stark von den optimalen Werten ab. Abbildung 5.6 zeigt die optimierte Trajektorie und Punktwolke des SLAM-Algorithmus mit Kalibrierung. Auch der SLAM-Algorithmus ohne integrierte Kalibrierung findet die optimale Trajektorie und Umgebungskarte bei dieser initialen Kalibrierung nicht mehr.

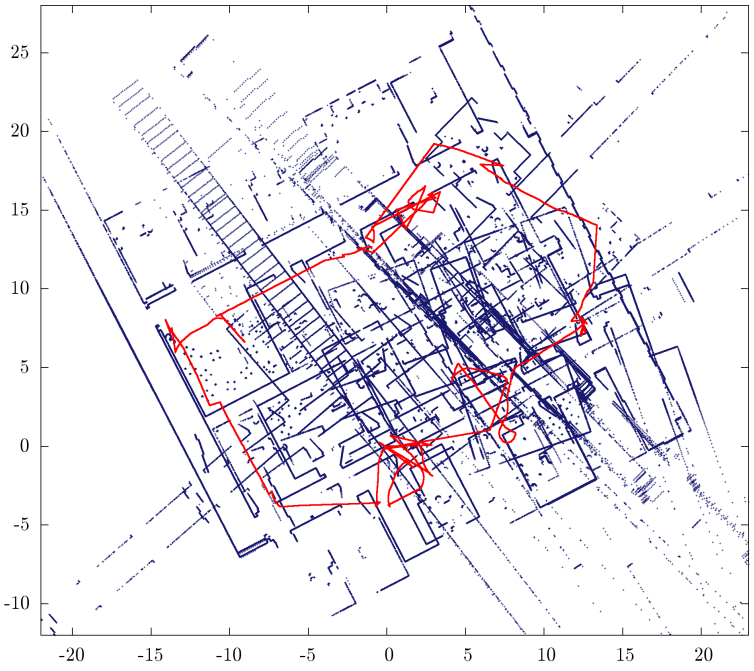


Abb. 5.6: Diese Abbildung zeigt die Ausgabe des SLAM-Algorithmus mit integrierter Kalibrierung der Umgebungssensoren, nachdem in der zweiten Logdatei beide Laserscanner mit der Pose $\begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T$ initialisiert wurden. Es ist klar erkennbar, dass der Algorithmus es nicht schafft die korrekte Lösung zu finden.

5.2.2 Kalibrierung der Odometrie

Für die Kalibrierung der Odometrie werden die gleichen Datensätze und die gleiche Methode wie schon für die Kalibrierung der Umgebungssensoren verwendet. Wie in Kapitel 3.3.3.2 beschrieben, wird die Odometrie des Roboters über ein allgemeines Bewegungsmodell mit neun Parametern beschrieben. Diese Parameter geben an, wie stark sich Vorwärts-, Seitwärts- oder Drehbewegungsanteile einer Bewegung des Roboters gegenseitig beeinflussen. Ist eine Pose, die die geschätzte Bewegung eines Roboters von einem Zeitpunkt zum nächsten darstellt, in d-c-t-Form gegeben, so kann das Bewegungsmodell, durch Multiplizieren der Pose mit der (3×3)-Matrix aus

Experiment	APCE	Stdabw.	1. Quartil	Median	3. Quartil
Kurz, Abweichung 5 cm / °:					
Punktwolke initial	0,4544	0,3706	0,2166	0,3410	0,6019
Punktwolke kalibriert	$9{,}7 \cdot 10^{-5}$	$8{,}1 \cdot 10^{-5}$	$5{,}4 \cdot 10^{-5}$	$7{,}4 \cdot 10^{-5}$	$11{,}0 \cdot 10^{-5}$
Kurz, Abweichung 15 cm / °:					
Punktwolke initial	0,8901	0,8330	0,3626	0,6608	1,2603
Punktwolke kalibriert	$8{,}4 \cdot 10^{-5}$	$3{,}7 \cdot 10^{-5}$	$6{,}5 \cdot 10^{-5}$	$7{,}7 \cdot 10^{-5}$	$9{,}4 \cdot 10^{-5}$
Kurz, initiale Posen $(0,0,0)$:					
Punktwolke initial	4,2082	4,4875	1,4742	2,8198	5,1051
Punktwolke kalibriert	$7{,}4 \cdot 10^{-5}$	$6{,}0 \cdot 10^{-5}$	$3{,}7 \cdot 10^{-5}$	$5{,}8 \cdot 10^{-5}$	$8{,}9 \cdot 10^{-5}$
Lang, Abweichung 5 cm / °:					
Punktwolke initial	4,4373	5,6609	1,3033	2,6621	5,3761
Punktwolke kalibriert	0,0025	0,0002	0,0024	0,0026	0,0027

Tab. 5.5: Diese Tabelle stellt die Fehler der Ergebnisse der verschiedenen Experimente zur Kalibrierung der Laserscanner mit den Simulations-Datensätzen dar. „Kurz“ steht dabei für die Verwendung des Datensatzes mit der kürzeren Trajektorie, „Lang“ für die Verwendung des Datensatzes mit der längeren Trajektorie.

den Parametern, auf diese Schätzung angewendet werden:

$$\begin{pmatrix} D \\ C \\ T \end{pmatrix} = \begin{pmatrix} \mu_{D_d} & \mu_{D_c} & \mu_{D_t} \\ \mu_{C_d} & \mu_{C_c} & \mu_{C_t} \\ \mu_{T_d} & \mu_{T_c} & \mu_{T_t} \end{pmatrix} \cdot \begin{pmatrix} d \\ c \\ t \end{pmatrix} \qquad [5.3]$$

Der Parameter μ_{D_t} gibt also beispielsweise den Faktor wieder, mit dem die Drehung der ursprünglichen Schätzung $\begin{pmatrix} d & c & t \end{pmatrix}^T$ in die Vorwärtsbewegung der aktualisierten Schätzung $\begin{pmatrix} D & C & T \end{pmatrix}^T$ eingeht. Bewegt sich ein Roboter beispielsweise bei jeder Drehung zusätzlich leicht vorwärts, was durch die Odometriemessung nicht widergespiegelt wird, so kann dies durch das Anpassen des Wertes μ_{D_t} korrigiert werden. Entsprechen die Parameter des Bewegungsmodells der Einheitsmatrix, so ändert das Bewegungsmodell die geschätzte Pose nicht.

Für den ersten Versuch wird die Logdatei mit den korrekten Daten so angepasst, dass sie die Ausgabe unter Verwendung eines vorgegebenen Bewegungsmodells widerspiegelt. Der Algorithmus zur Kalibrierung der Odometrie soll nun das inverse Bewegungsmodell ausgeben, welches die Anwendung des vorgegebenen Bewegungsmodells rückgängig macht. Die Matrix des inversen Modells ist dabei gerade die inverse Matrix des ursprünglichen Bewegungsmodells.

Für den ersten Datensatz mit der kürzeren Trajektorie werden die Kalibrierungsparameter wieder zufällig gewählt. Es gilt jedoch die Einschränkung, dass die resultierende Matrix in jedem Wert um mindestens 0,1 von der Einheitsmatrix abweicht. Die hier verwendeten Parameter und die des

inversen Modells sind folgende:

$$A = \begin{pmatrix} \mu_{D_d} & \mu_{D_c} & \mu_{D_t} \\ \mu_{C_d} & \mu_{C_c} & \mu_{C_t} \\ \mu_{T_d} & \mu_{T_c} & \mu_{T_t} \end{pmatrix} = \begin{pmatrix} 1{,}2 & 0{,}3 & -0{,}2 \\ 0{,}2 & 0{,}8 & -0{,}1 \\ -0{,}2 & 0{,}1 & 1{,}1 \end{pmatrix}, \quad A^{-1} = \begin{pmatrix} 0{,}9156 & -0{,}3601 & 0{,}1337 \\ -0{,}2058 & 1{,}3169 & 0{,}0823 \\ 0{,}1852 & -0{,}1852 & 0{,}9259 \end{pmatrix} \quad [5.4]$$

Abbildung 5.7 zeigt Trajektorie und Umgebungspunktwolke, wie sie bei Verwendung des fehlerhaften initialen Bewegungsmodells entstehen. Die Trajektorie ist stark verformt und deshalb ist auch in der Punktwolke keine Umgebungskarte mehr erkennbar. Als Ergebnis der Kalibrierung

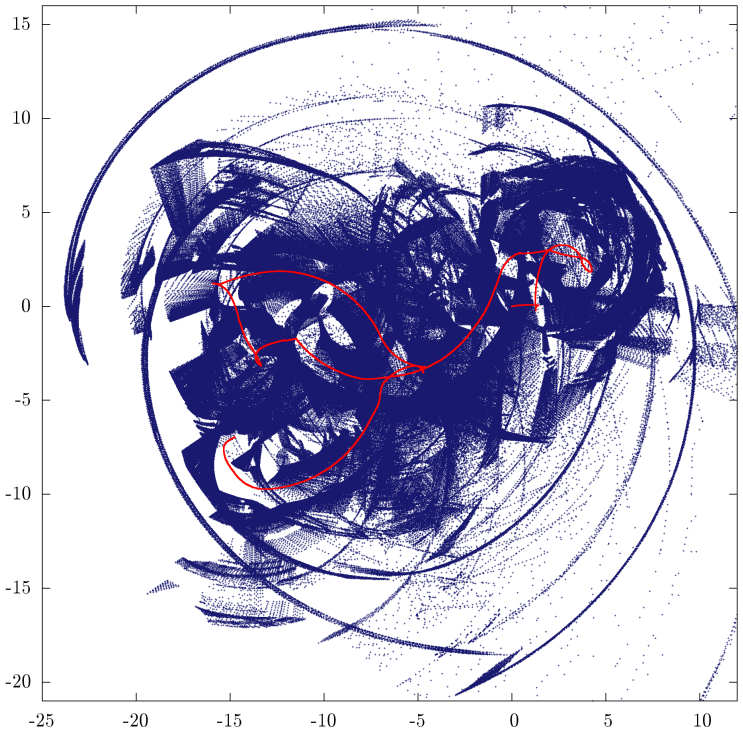


Abb. 5.7: Diese Abbildung zeigt die Trajektorie und die Punktwolke für die Umgebung, wie sie bei Verwendung des schlechten Bewegungsmodells ausgegeben wird.

gibt der Algorithmus die unten angegebene Matrix B zurück. Die Matrix $B - A^{-1}$ enthält die Differenzen zwischen den kalibrierten Werten und den optimalen Werten.

$$B = \begin{pmatrix} 0{,}9150 & -0{,}3544 & 0{,}1339 \\ -0{,}2057 & 1{,}3127 & 0{,}0808 \\ 0{,}1854 & -0{,}1842 & 0{,}9264 \end{pmatrix}, \quad B - A^{-1} = \begin{pmatrix} -0{,}0007 & 0{,}0057 & 0{,}0001 \\ 0{,}0001 & -0{,}0042 & -0{,}0015 \\ 0{,}0002 & 0{,}0010 & 0{,}0005 \end{pmatrix} \quad [5.5]$$

Der ATE beträgt vor der Kalibrierung 14,3 und wird durch die Kalibrierung auf 0,07 reduziert. Die Standardabweichung und die Quartile vor und nach Kalibrierung der Odometrie sind in Tabelle 5.6 dargestellt. Außerdem gibt der Algorithmus zusätzlich in einer zweiten Matrix Σ die Varianz der

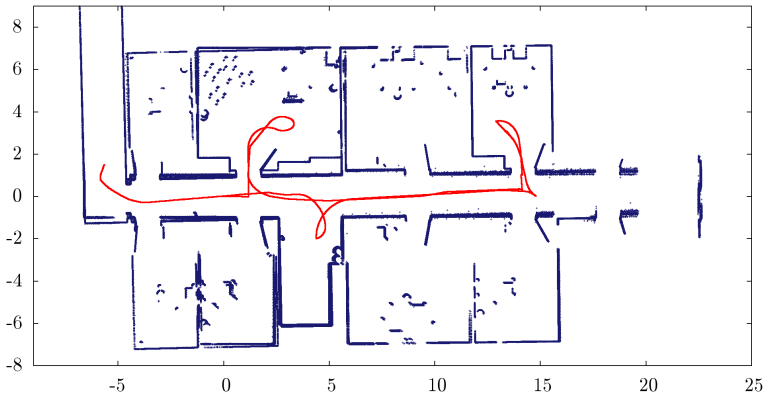


Abb. 5.8: Diese Abbildung zeigt die Visualisierung der Daten nachdem, von einer fehlerhaften Kalibrierung des Bewegungsmodells ausgehend, die vom Algorithmus ausgegebenen Kalibrierungsparameter verwendet wurden.

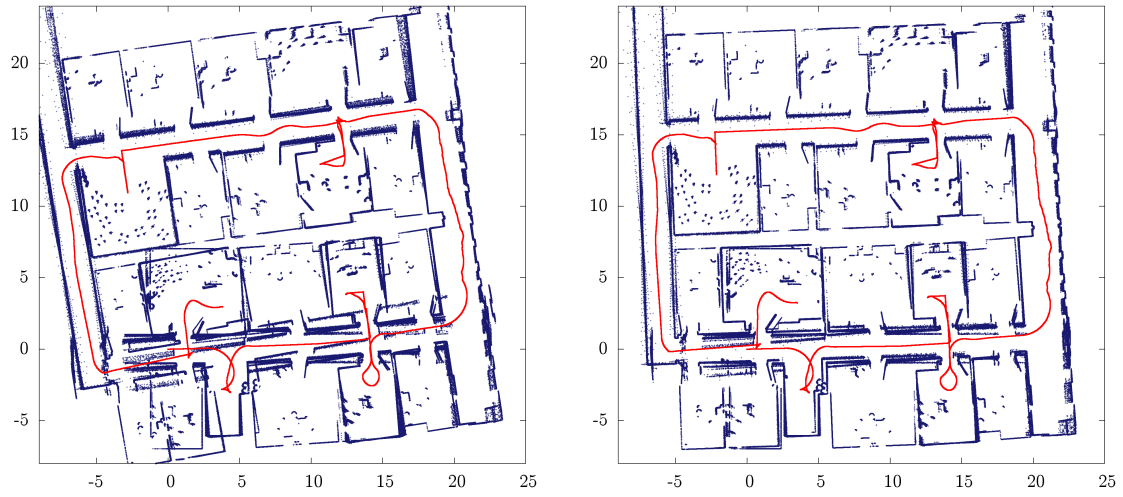
einzelnen Kalibrierungsparameter zurück.

$$\Sigma = \begin{pmatrix} 0{,}0012 & 0{,}0167 & 0{,}0015 \\ 0{,}0012 & 0{,}0167 & 0{,}0015 \\ 0{,}0012 & 0{,}0167 & 0{,}0015 \end{pmatrix} \quad [5.6]$$

Es ist gut erkennbar, dass die Werte aus Matrix B, die stärker vom Optimum abweichen, auch tendenziell mit einer höheren Varianz in Matrix Σ versehen sind.

Obwohl die durch die Kalibrierung gefundenen Werte nur minimal von den optimalen Werten abweichen, ist ein kleiner Unterschied zwischen den korrekten Daten (siehe Abbildung 5.2) und den mit dem neuen Bewegungsmodell berechneten Daten (siehe Abbildung 5.8) erkennbar. Sehr kleine Abweichungen der Parameter des Bewegungsmodells im Bereich von einigen Tausendsteln können also bereits eine sichtbare Auswirkung auf die Güte der Trajektorie haben. Trotzdem ist die Kalibrierung der Odometrieparameter immer noch sehr gut.

Das gleiche Experiment mit den initialen Parametern aus Gleichung 5.4 wird auch für die zweite Logdatei mit der längeren Trajektorie durchgeführt. Die ausgegebenen Kalibrierungsparameter in Matrix B weichen dabei etwas stärker als im vorangegangen Experiment von den optimalen Werten ab. Trotzdem liegt der absolute Fehler für jeden einzelnen Wert noch unter einem Hundertstel. Nach der Kalibrierung beträgt der ATE jedoch immer noch 1,3. Trotz des relativ großen Fehlerwerts im Vergleich zu den Experimenten mit den Umgebungssensoren, konnte der Fehlerwert von 18,5 vor der Kalibrierung deutlich reduziert werden. Abbildung 5.9a zeigt die Trajektorie und die Umgebungskarte, die aus den, mit den Werten aus Matrix B kalibrierten Daten, errechnet wurden. An der enthaltenen Schleife ist sehr gut erkennbar, dass die Trajektorie leicht von der korrekten Trajektorie abweicht. Dabei werden die kleinen Fehler in der Odometrie, wie an den verschwommenen Kanten der Punktwolke zu sehen ist, über die Zeit hinweg akkumuliert. So kommt es zu



(a) Erster Versuch mit Abweichung $> 0{,}1$ (b) Zweiter Versuch mit Abweichung $> 0{,}05$

Abb. 5.9: Diese Abbildung zeigt die Ergebnisse der beiden Versuche die Odometrie des Roboters mit Hilfe der Logdatei mit längerer Trajektorie zu kalibrieren.

der relativ großen Abweichung, wenn die Schleife geschlossen wird.

$$B = \begin{pmatrix} 0{,}9122 & -0{,}3516 & 0{,}1345 \\ -0{,}2071 & 1{,}3138 & 0{,}0784 \\ 0{,}1853 & -0{,}1817 & 0{,}9204 \end{pmatrix}, \quad B - A^{-1} = \begin{pmatrix} -0{,}0034 & 0{,}0085 & 0{,}0007 \\ -0{,}0014 & -0{,}0031 & -0{,}0039 \\ 0{,}0001 & 0{,}0035 & -0{,}0055 \end{pmatrix} \quad [5.7]$$

Wird eine bessere initiale Kalibrierung verwendet, so wird auch das Optimierungsergebnis besser. Mit einer initialen Matrix, deren Werte zwischen 0,05 und 0,10 von der Einheitsmatrix abweichen, wurden die Daten für Abbildung 5.9b erzeugt. Zwar ist das Ergebnis der Kalibrierung immer noch nicht perfekt, es kommt jedoch deutlich näher an die optimalen Werte heran. Dies spiegelt sich auch in einem geringeren ATE von 0,6 wieder.

Experiment	ATE	Stdabw.	1. Quartil	Median	3. Quartil
Kurz, Abweichung 0,1:					
Trajektorie initial	14,2966	11,3136	3,9161	12,6280	25,6117
Trajektorie kalibriert	0,0657	0,0616	0,0142	0,0597	0,0986
Lang, Abweichung 0,1:					
Trajektorie initial	18,4732	8,7156	15,8289	18,5433	22,7614
Trajektorie kalibriert	1,2728	0,8268	0,5104	1,3318	2,0636
Lang, Abweichung 0,05:					
Trajektorie initial	25,8645	17,7763	9,9963	25,2641	42,1331
Trajektorie kalibriert	0,6381	0,3903	0,3658	0,6306	0,9597

Tab. 5.6: In dieser Tabelle sind der Absolute Trajectory Error sowie Standardabweichungen und Quartile der Abweichung von den korrekten Daten dargestellt. Für jedes Experiment zur Kalibrierung der Odometrie sind jeweils die Werte vor und nach der Kalibrierung dargestellt.

5.2.3 Gleichzeitige Kalibrierung von Odometrie und Umgebungssensoren

Wie in den vorherigen Experimenten werden die beiden bereits vorgestellten Logdateien verwendet und diesmal gleichzeitig die Kalibrierung der Odometrie und der Umgebungssensoren manipuliert. Die resultierenden Logdateien werden als Eingabe für den SLAM-Algorithmus mit integrierter Kalibrierung verwendet. Tabelle 5.7 zeigt die verwendeten Werte und die Ausgabe des Algorithmus mit den Abweichungen von den optimalen Werten bei der Verwendung der ersten Logdatei mit der kürzeren Trajektorie. Die Abweichungen sind relativ hoch, obwohl der von dem

	Sensor 1			Sensor 2			Odometrie
	x_1	y_1	θ_1	x_2	y_2	θ_2	
Optimal	0,3386	0,3121	-44,1406	-0,3386	-0,3121	135,8594	$\begin{pmatrix} 0{,}9156 & -0{,}3601 & 0{,}1337 \\ -0{,}2058 & 1{,}3169 & 0{,}0823 \\ 0{,}1852 & -0{,}1852 & 0{,}9259 \end{pmatrix}$
Initial	0,4	0,22	-50,0	-0,25	-0,4	143,0	$\begin{pmatrix} 1{,}2 & 0{,}3 & -0{,}2 \\ 0{,}2 & 0{,}8 & -0{,}1 \\ -0{,}2 & 0{,}1 & 1{,}1 \end{pmatrix}$
Abweichung	0,06144	0,09212	5,85944	0,08856	0,08788	7,14056	$\begin{pmatrix} 0{,}0844 & 0{,}3601 & -0{,}1337 \\ 0{,}2058 & -0{,}3169 & -0{,}0823 \\ -0{,}1852 & 0{,}1852 & 0{,}0741 \end{pmatrix}$
Kalibriert	0,7421	-0,0437	-40,2177	0,1089	-0,7126	139,7809	$\begin{pmatrix} 0{,}8573 & -0{,}3768 & -0{,}2223 \\ -0{,}2214 & 1{,}3664 & -0{,}3019 \\ 0{,}1855 & -0{,}1853 & 0{,}9264 \end{pmatrix}$
Abweichung	0,4035	0,3558	3,9229	0,4475	0,4005	3,9215	$\begin{pmatrix} 0{,}0583 & 0{,}0167 & 0{,}3560 \\ 0{,}0156 & -0{,}0495 & 0{,}3842 \\ -0{,}0003 & 0{,}0001 & -0{,}0005 \end{pmatrix}$

Tab. 5.7: Ergebnisse der simultanen Kalibrierung von Odometrie und Umgebungssensoren unter Verwendung der ersten Logdatei. Gut erkennbar ist, dass die optimierten Werte meistens weiter von den optimalen Werten abweichen als die initial eingestellten Werte. Eine Ausnahme bilden die Parameter, die die Orientierung beziehungsweise Rotation betreffen.

Optimierungsverfahren bei der Konvergenz ausgegebene Fehler kaum höher ist, als bei funktionierenden Kalibrierungen aus den vorigen Experimenten. Auch die mit den berechneten Parametern kalibrierten Daten, zu sehen in Abbildung 5.10, sind in sich konsistent, unterscheiden sich in den Trajektorien aber trotzdem stark von den korrekten Daten. Dieser Effekt kann bei allen Experimenten mit gleichzeitiger Kalibrierung von Odometrie und Umgebungssensoren beobachtet werden, auch wenn die initialen Werte deutlich besser sind. Abbildung 5.11 zeigt den Effekt bei einem analog durchgeführten Experiment mit dem Datensatz mit der längeren Trajektorie.

Daraus lässt sich schließen, dass in diesem Fall zu viele Kalibrierungsparameter angepasst werden können. Das Bewegungsmodell ist so mächtig, dass es die Trajektorie an unterschiedliche Sensorkalibrierungen anpassen kann. Das Resultat ist ein kleiner Fehler in der Optimierung und schlechte Kalibrierungsparameter, da nur ein lokales Optimum erreicht wird. Abhilfe könnte die Verwendung eines weniger allgemeinen Bewegungsmodells mit weniger frei wählbaren Parametern liefern. Allerdings garantiert gerade dieses allgemeine Bewegungsmodell die vielseitige Einsetzbarkeit des Algorithmus. Die in dieser Arbeit verfolgte Lösungsidee besteht daher darin, zuerst nur die Umfeldsensoren zu kalibrieren und dann in einem zweiten Schritt das Bewegungsmodell, unter Verwendung der kalibrierten Umfeldsensoren, zu kalibrieren. Wie in Kapitel 5.2.2 gezeigt,

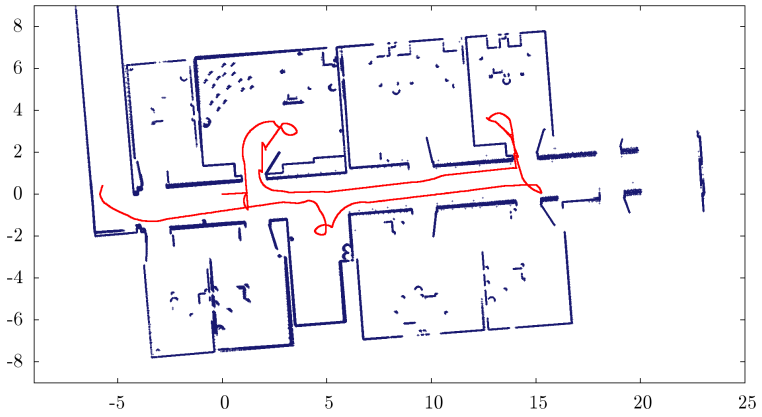


Abb. 5.10: In dieser Abbildung sind die Trajektorie und Umgebungskarte zu sehen, nachdem gleichzeitig Odometrie und Umgebungssensoren kalibriert wurden. Die Daten sind in sich konsistent, die Trajektorie weicht aber stark von der korrekten Trajektorie ab.

Abb. 5.11: Die Abbildung zeigt die Trajektorie und Umgebungskarte nach der gleichzeitigen Kalibrierung von Odometrie und Umfeldsensorik für die zweite Logdatei.

ist der Algorithmus in der Lage die Odometrie korrekt zu kalibrieren, wenn genaue Umgebungsdaten vorliegen. Dieser Ansatz wird im nächsten Abschnitt verfolgt.

5.2.4 Kalibrierung von Umgebungssensoren und Odometrie nacheinander

In diesem Kapitel soll die Idee getestet werden, zuerst die Umfeldsensoren und anschließend die Odometrie zu kalibrieren. Dies ist nötig, da sich herausgestellt hat, dass die gleichzeitige Kalibrierung von Umfeldsensoren und Odometrie keine guten Ergebnisse liefert, da der Optimierungsalgorithmus in lokalen Minima konvergiert (vgl. Abschnitt 5.2.3).

In dem in diesem Abschnitt beschriebenen Experiment kalibriert der Algorithmus in einem ersten Durchlauf nur die Umgebungssensoren. Die optimierten Werte werden dann in einem zweiten Durchlauf, bei dem die Odometrie kalibriert wird, bereits verwendet.

Als initiale Kalibrierungsparameter für die Umgebungssensoren wurden in dem Experiment Werte gewählt, die um mindestens 5 cm beziehungsweise 5° von den korrekten Werten abweichen. Für die Odometrie wurden jedoch bessere initiale Werte als in den bisherigen Experimenten verwendet. Die hier verwendeten Werte weisen eine Abweichung von 0,01 bis 0,05 zur Einheitsmatrix auf. Wie bereits in Kapitel 5.2.2 festgestellt, können schon kleine Abweichungen in der Odometriekalibrierung die Trajektorie stark beeinflussen. Wird ein sehr schlechtes initiales Bewegungsmodell gewählt, versucht das Optimierungsverfahren automatisch, bereits während der Kalibrierung der Umgebungssensoren die Fehler, die durch die inkorrekte Odometriekalibrierung entstehen, zu kompensieren. Dies führt bei den in Kapitel 5.2.4 verwendeten Werten dazu, dass nach beiden Durchläufen des Algorithmus wieder nur ein lokales Minimum gefunden wurde. Abbildung 5.12 zeigt dieses Ergebnis.

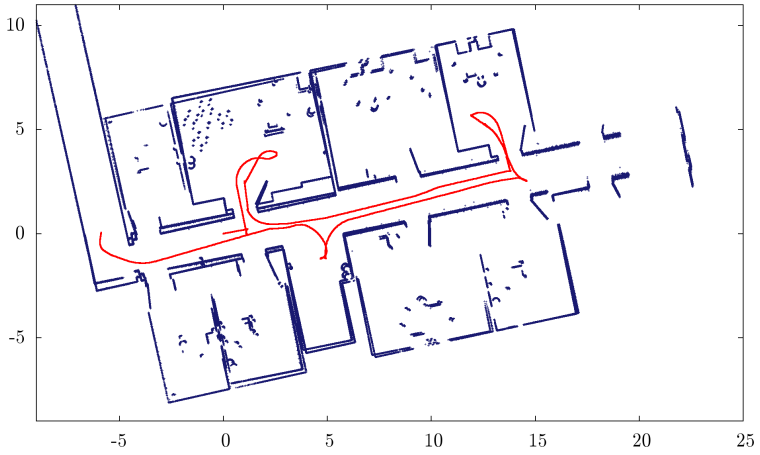


Abb. 5.12: Diese Abbildung zeigt den ersten fehlgeschlagenen Versuch Umfeldsensoren und Odometrie nacheinander zu kalibrieren. Die initiale Kalibrierung des Bewegungsmodells war in diesem Fall nicht ausreichend.

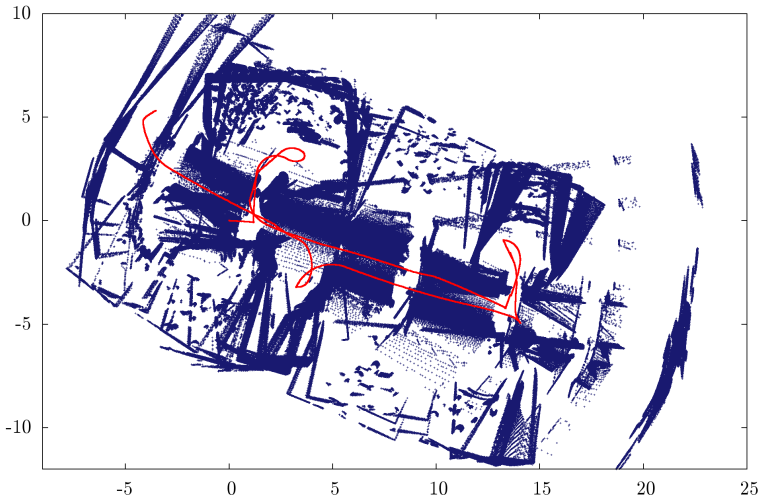
Tabelle 5.8 stellt die verwendeten Parameter und die Ergebnisse für das Experiment mit der besseren initialen Kalibrierung dar. Der ATE beträgt bei diesem Experiment vor der Kalibrierung

	Sensor 1			Sensor 2			Odometrie
	x_1	y_1	θ_1	x_2	y_2	θ_2	
Optimal	0,3386	0,3121	-44,1406	-0,3386	-0,3121	135,8594	$\begin{pmatrix} 0{,}9819 & 0{,}0296 & 0{,}0192 \\ 0{,}0403 & 1{,}0326 & -0{,}0094 \\ 0{,}0178 & -0{,}0403 & 0{,}9909 \end{pmatrix}$
Initial	0,4	0,22	-50,0	-0,25	-0,4	143,0	$\begin{pmatrix} 1{,}02 & -0{,}03 & -0{,}02 \\ -0{,}04 & 0{,}97 & 0{,}01 \\ -0{,}02 & 0{,}04 & 1{,}01 \end{pmatrix}$
Abweichung	0,06144	0,09212	5,85944	0,08856	0,08788	7,14056	$\begin{pmatrix} -0{,}0181 & 0{,}0296 & 0{,}0192 \\ 0{,}0403 & 0{,}0326 & -0{,}0094 \\ 0{,}0178 & -0{,}0403 & -0{,}0091 \end{pmatrix}$
Nacheinander kalibriert	0,3413	0,2828	46,4128	0,3601	0,3141	133,6392	$\begin{pmatrix} 0{,}9824 & 0{,}0701 & 0{,}0041 \\ 0{,}0021 & 1{,}0290 & 0{,}0000 \\ 0{,}0180 & -0{,}0415 & 0{,}9906 \end{pmatrix}$
Abweichung	0,0028	0,0293	2,2722	0,0216	0,0019	2,2202	$\begin{pmatrix} -0{,}0005 & -0{,}0406 & 0{,}0150 \\ 0{,}0382 & 0{,}0035 & -0{,}0094 \\ -0{,}0002 & 0{,}0012 & 0{,}0003 \end{pmatrix}$
Gleichzeitig kalibriert	-0,0528	0,5290	-20,7289	-0,4262	-0,3129	159,2720	$\begin{pmatrix} 0{,}8870 & -0{,}3882 & 0{,}1286 \\ 0{,}4315 & 0{,}9492 & 0{,}2360 \\ 0{,}0179 & -0{,}0398 & 0{,}9904 \end{pmatrix}$
Abweichung	-0,3914	0,2169	23,4116	-0,0876	-0,0008	23,4125	$\begin{pmatrix} 0{,}0949 & 0{,}4178 & -0{,}1094 \\ -0{,}3912 & 0{,}0834 & -0{,}2455 \\ -0{,}0000 & -0{,}0005 & 0{,}0004 \end{pmatrix}$

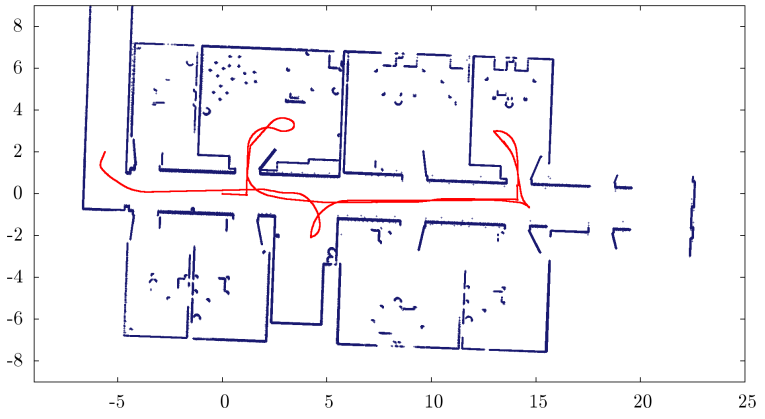
Tab. 5.8: In dieser Tabelle sind initiale Einstellungen, Ausgaben des Algorithmus und ihre Abweichungen von den optimalen Werten für die separate Kalibrierung von Odometrie und Umfeldsensorik dargestellt. Dabei wurde der erste aus der Simulation entstandene Datensatz mit der kürzeren Trajektorie verwendet. Die letzte Zeile der Tabelle zeigt zum Vergleich das Ergebnis bei simultaner Kalibrierung von Umfeldsensoren und Odometrie. Das Ergebnis der separaten Kalibrierung ist offensichtlich deutlich besser.

2,4 und nach der Kalibrierung noch 0,3. Die Werte des APCE sowie Standardabweichungen können Tabelle 5.9 entnommen werden. Die Trajektorie und Umgebungskarte, die mit diesen Kalibrierungsparametern nur aufgrund der Odometrieinformationen bestimmt wurde, ist in Abbildung 5.13b zu sehen. Im Vergleich mit den korrekten Daten, die in Abbildung 5.2 visualisiert wurden, ist erkennbar, dass das Ergebnis einerseits sehr gut ist, andererseits allerdings die gesamte Trajektorie und Karte leicht gedreht erscheinen. Deshalb ist auch der resultierende ATE mit 0,3 noch relativ hoch. Zum Vergleich sind in Abbildung 5.13a die Daten vor der Kalibrierung dargestellt.

Der Fehler lässt sich wieder darauf zurück führen, dass bei der Kalibrierung der Umfeldsensoren gleichzeitig auch Teile des Odometriefehlers ausgeglichen werden. Die Odometriekalibrierung beruht dann wiederum auf diesen neu berechneten Parametern für die Laserscanner. Obwohl diese Lösung keine perfekten Ergebnisse liefert, lässt sich doch eine deutliche Verbesserung durch die Kalibrierung feststellen. Ähnliche Ergebnisse werden bei der Kalibrierung mit der zweiten Logdatei erzielt. Die resultierenden Daten eines entsprechenden Versuchs sind in Abbildung 5.14 in Form von Trajektorie und Punktwolke dargestellt. In Tabelle 5.9 sind der ATE und weitere Werte für dieses Experiment dargestellt.



(a) Vor der Kalibrierung

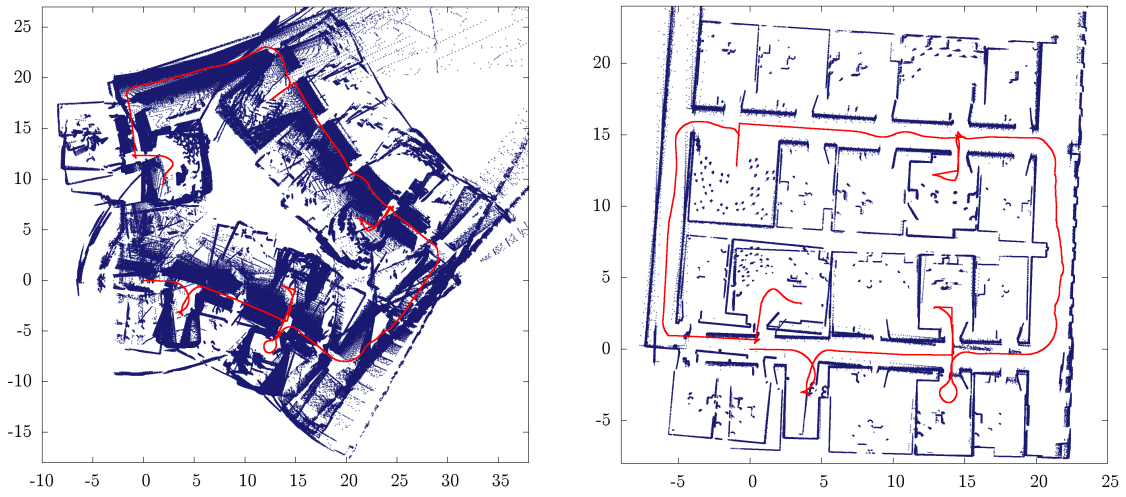


(b) Nach der Kalibrierung

Abb. 5.13: In Abbildung (a) sind die manipulierten Daten, die bei der separaten Kalibrierung von Odometrie und Umfeldsensoren verwendet wurden, zu sehen. Das Ergebnis der Kalibrierung von Odometrie und Umfeldsensoren in dieser Reihenfolge ist in Abbildung (b) dargestellt. Die Trajektorie und die Punktwolke entsprechen bis auf eine leichte Drehung sehr genau den korrekten Daten.

Experiment	ATE	Stdabw.	1. Quartil	Median	3. Quartil
Kurz:					
Trajektorie initial	2,3827	1,9331	0,6176	1,7542	4,2295
Trajektorie kalibriert	0,2895	0,1902	0,1460	0,2372	0,4871
Punktwolke initial	2,6511	2,0321	1,0542	2,8442	4,6749
Punktwolke kalibriert	0,3307	0,2003	0,1728	0,3459	0,5309
Lang:					
Trajektorie initial	8,8402	5,6353	4,5261	8,9209	13,4850
Trajektorie kalibriert	0,5602	0,2459	0,4715	0,4965	0,6947
Punktwolke initial	9,1448	6,1026	5,3748	11,2660	15,8370
Punktwolke kalibriert	0,5912	0,2957	0,4398	0,5992	0,8892

Tab. 5.9: In dieser Tabelle sind der ATE und APCE sowie Standardabweichungen und Quartile der Abweichung von den korrekten Daten dargestellt. Für beide Experimente bei denen jeweils zuerst die Laserscanner und anschließend die Odometrie kalibriert wurden, sind jeweils die Werte vor und nach der Kalibrierung dargestellt.



(a) Vor der Kalibrierung

(b) Nach der Kalibrierung

Abb. 5.14: Abbildung (a) visualisiert die zweite Logdatei, nachdem sowohl die Odometrie als auch die Abstandssensoren mit schlechten Kalibrierungsparametern initialisiert wurden. In Abbildung (b) ist das Ergebnis der Kalibrierung von Odometrie und Umfeldsensoren nacheinander zu sehen. Die korrekten Daten sind in Abbildung 5.3 dargestellt.

5.2.5 Zusammenfassung der Ergebnisse

In diesem Kapitel wurden einige Experimente zu der Kalibrierung von Odometrien und Umgebungssensoren mit Hilfe von simulierten Daten durchgeführt. Die Datensätze waren dabei unterschiedlich lang und komplex. So enthielt beispielsweise ein Datensatz eine große Schleife. Der simulierte Roboter kann sich omnidirektional bewegen, was bei der Aufnahme der Datensätze auch ausgenutzt wurde. Außerdem ist der Roboter mit zwei Laserscannern zur Abstandsmessung ausgestattet, die in entgegengesetzte Richtungen ausgerichtet sind.

Mit Hilfe dieser Daten konnte einerseits gezeigt werden, dass der Algorithmus in der Lage ist, die Odometrie oder die Umfeldsensoren einzeln korrekt zu kalibrieren. Dabei waren die jeweils anderen Sensoren bereits korrekt kalibriert. In diesem einfachsten Fall sind die Abweichungen von den optimalen Werten minimal und eventuell auf die Konvergenzkriterien des Optimierungsverfahrens zurückführbar. Dies bestätigt die Korrektheit der in Kapitel 3.3 aufgestellten mathematischen Modelle für die Faktoren.

Bei der simultanen Kalibrierung aller Sensoren stellte sich in den Experimenten allerdings heraus, dass es quasi unmöglich ist, die korrekten Kalibrierungsparameter zu finden. Das Optimierungsverfahren konvergiert bei den durchgeführten Experimenten in verschiedenen lokalen Minima, die sich einerseits dadurch auszeichnen, dass der resultierende finale Fehler des Optimierungsverfahrens klein ist, andererseits aber die Kalibrierungsparameter in keinem Zusammenhang mit den optimalen Werten stehen. Aus diesen Ergebnissen wurde geschlossen, dass das verwendete allgemeine Bewegungsmodell für die gleichzeitige Kalibrierung aller Sensoren zu mächtig ist. Das

heißt die Trajektorie kann mit Hilfe des Bewegungsmodells an verschiedene Kalibrierungen der Laserscanner angepasst werden.

Es wurden zwei Vorschläge gemacht, wie dieses Problem umgangen werden könnte. Einerseits könnte das Bewegungsmodell eingeschränkt werden, so dass es nicht mehr so viele frei wählbare Parameter enthält. Im Falle eines Roboters mit Differentialantrieb könnte beispielsweise das in Kapitel 3.3.3.2.1 vorgestellte Bewegungsmodell von Austin Eliazar und Ronald Parr verwendet werden [EP04]. Dieses Modell kommt mit drei Parametern weniger aus, indem es davon ausgeht, dass die Posen auf die das Bewegungsmodell angewendet wird keinen lateralen Anteil enthalten.

Mit der Anpassung des Bewegungsmodells für verschiedene Roboterplattformen würde natürlich ein Teil der Flexibilität des Algorithmus verloren gehen. Außerdem werden im Fall eines sich omnidirektional bewegenden Roboters alle Parameter des Bewegungsmodells benötigt. Deshalb wird in dieser Arbeit ein anderer Ansatz zur Lösung des Problems verfolgt. Dabei wird die Kalibrierung von Umgebungssensoren und Odometrie nacheinander vorgenommen. Sind die Laserscanner in einem ersten Schritt bereits gut kalibriert worden, kann in einem zweiten Schritt auch das Bewegungsmodell unter Verwendung der bereits bestimmten Parameter optimiert werden.

Die Evaluation dieses Verfahrens mit Hilfe der simulierten Daten zeigt, dass dieser Ansatz gut funktioniert, jedoch nur, wenn die initialen Schätzungen für das Bewegungsmodell genau genug sind. Im nächsten Kapitel wird dies mit Datensätzen von realen Roboterfahrten weiter evaluiert. Dabei wurde festgestellt, dass die Güte der initialen Schätzungen der Parameter in allen Fällen so gut ist, dass Kalibrierungsparameter bestimmt werden können, mit denen der SLAM-Algorithmus gegen eine sehr gute Trajektorie konvergiert.

5.3 Experimente mit realen Datensätzen

Nachdem im letzten Kapitel bereits die grundlegende Funktionalität des Algorithmus anhand von simulierten Daten getestet wurde, wird in diesem Kapitel das Verhalten bei verrauschten Daten untersucht. Dafür werden drei verschiedene Datensätze verwendet, die mit zwei unterschiedlichen Robotern und einem Auto aufgenommen wurden. Einer der Roboter kann sich dabei omnidirektional bewegen, der zweite Roboter verfügt über einen Differentialantrieb während das Autos eine auf der Ackermann-Lenkung basierende Odometrie hat. Auch die Güte der initialen Kalibrierung unterscheidet sich bei den Datensätzen stark. Einer der Roboter scheint in den Rohdaten einen starken Rechtsdrall zu haben, was auf eine ungenau kalibrierte Odometrie schließen lässt, bei dem Auto konnten die Posen der Sensoren hingegen manuell nur ungenau gemessen werden. Die Parameter des dritten Roboters sind im Vergleich relativ gut kalibriert. Außerdem haben die beiden Roboter einen beziehungsweise zwei Umgebungssensoren, während das Auto über drei Sensoren verfügt. Die drei Datensätze decken also eine große Bandbreite an unterschiedlichen Eigenschaften ab.

Nur die Kompatibilität des Algorithmus mit verschiedenen Arten von Umgebungssensoren kann in dieser Arbeit nicht getestet werden, da beide Roboter sowie das Auto mit Laserscannern bestückt sind. Um die Verwendung anderer Sensortypen im Algorithmus zu ermöglichen, müssen Methoden für das Auslesen der Messungen und das Scan Matching zwischen zwei Messungen gegeben sein. Für Laserscanner wurde diese Funktionalität nicht selbst programmiert sondern über externe Bibliotheken eingebunden. Ist dies für weitere Sensortypen ebenso möglich, so sollte der erfolgreichen Verwendung des Algorithmus nichts im Wege stehen.

Einer der verwendeten Datensätze wurde im Jahr 2003 von Dieter Fox in einem Forschungslabor der Intel Corporation in Seattle aufgenommen und ist auf der Internetseite des „Robotics Data Set Repository (Radish)“ verfügbar [HR03]. Mit diesem Datensatz wurden in der Literatur bereits häufig SLAM-Algorithmen getestet. Es existiert eine korrigierte Version von Cyrill Stachniss (siehe [Sta]), so dass die Ergebnisse der Experimente mit diesen korrigierten Daten verglichen werden können. Bei den beiden anderen Datensätzen, die von Mitarbeitern des Forschungszentrums Informatik am Karlsruher Institut für Technologie aufgenommen wurden, existieren leider keine Referenzdaten. Daher muss die Qualität der gefundenen Kalibrierungsparameter anhand der in Kapitel 5.1 vorgestellten Kriterien bestimmt werden. Zusätzlich wird das Ergebnis des SLAM-Algorithmus ohne Kalibrierung als Anhaltspunkt genommen.

Im Folgenden werden für die drei Datensätze jeweils die Kalibrierung von Odometrie und beziehungsweise oder Umgebungssensoren analog zu den im letzten Kapitel durchgeführten Versuchen getestet. Die Ergebnisse werden anschließend ausgewertet und mit den Ergebnissen aus der Simulation verglichen. Im Anschluss wird noch kurz auf das Thema Overfitting sowie die Laufzeit des Algorithmus eingegangen.

5.3.1 Experimente mit dem Intel-Datensatz

Wie bereits beschrieben, wurde für die Aufnahme der Logdatei ein Roboter mit Differentialantrieb und einem nach vorne gerichteten Laserscanner verwendet. Die Aufnahmezeit betrug dabei knapp 45 Minuten, während denen der Roboter eine Strecke von ca. 560 Metern zurücklegte. Die zurückgelegte Strecke wurde hierbei direkt aus der korrigierten Logdatei berechnet. Während der Fahrt wurde mehrfach eine große Schleife in dem Forschungslabor abgefahren. Weitere Eigenschaften der Aufnahme sind in Tabelle 5.10 zusammengefasst.

Die aus den Rohdaten nur aufgrund der Odometriemessungen berechnete Trajektorie und Punktwolke ist in Abbildung 5.15 dargestellt. Die ebenso berechnete Trajektorie und Punktwolke für die korrigierten Daten ist in Abbildung 5.16 zu sehen. Es ist gut erkennbar, dass in den Rohdaten Strecken, die eigentlich gerade sein sollten, Bögen nach rechts entsprechen. Dass dies über die ganze Trajektorie hinweg gilt, lässt auf einen großen systematischen Fehler der Odometrie schließen. Der ATE im Vergleich zu den korrigierten Daten beträgt ca. 21,3, der APCE ca. 22,2. Die genauen Werte sind im Vergleich der Experimente dieses Kapitels in Tabelle 5.11 aufgelistet.

Intel Logdatei	
Dauer der Aufnahme	44 min 51 s
Gefahrene Strecke	561,7 m
Mittlere Geschwindigkeit	20,9 cm/s
Anzahl aller Messungen	40546
Anzahl der Odometriemessungen	26915
Frequenz der Odometriemessungen	10,0 Hz
Anzahl der Laserscanner	1
Anzahl der Laserscans	13631
Frequenz der Laserscans	5,1 Hz

Tab. 5.10: Diese Tabelle stellt einige Eigenschaften der, in der Intel-Logdatei aufgenommenen, Roboterfahrt dar. Die Gesamtstrecke und die durchschnittliche Geschwindigkeit wurden dabei aus den korrigierten Daten berechnet.

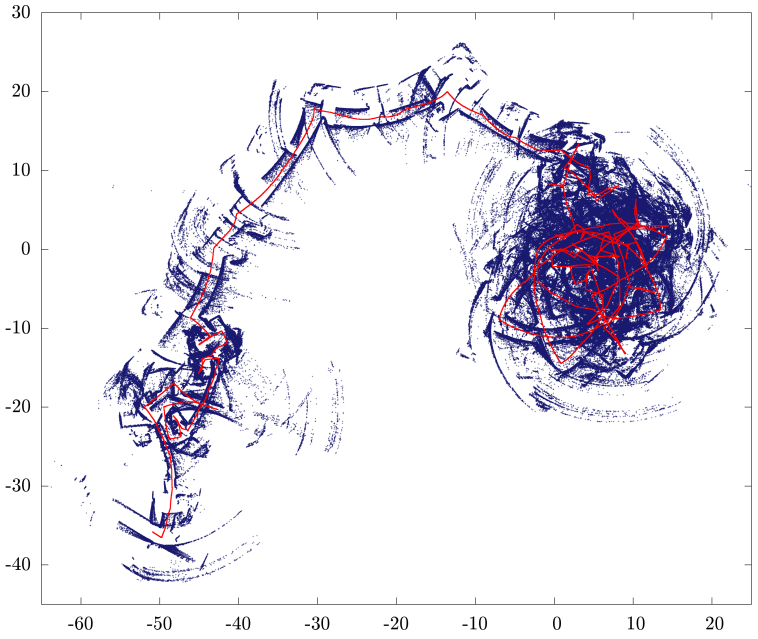


Abb. 5.15: Diese Abbildung zeigt die Trajektorie und die Punktwolke für die Intel-Logdatei, wie sie nur auf Basis der Odometriemesswerte berechnet wird.

Abb. 5.16: Hier ist die korrigierte Version der Intel-Logdatei dargestellt.

Wie schon im vorigen Kapitel werden zuerst die Odometrie und der Laserscanner jeweils einzeln kalibriert. Die optimierten Kalibrierungsparameter werden dann auf die Rohdaten angewendet. Danach werden die resultierenden Trajektorien und Punktwolken mit den korrekten Daten verglichen.

Die in der Kalibrierung berechnete Pose des Laserscanners ist $\begin{pmatrix} 0{,}0959 & 0{,}0068 & 0{,}0364 \end{pmatrix}^T$, wobei die initiale aus der Logdatei ausgelesene Schätzung für die Pose $\begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T$ ist. Der APCE sagt in diesem Fall nicht viel über die Güte der Kalibrierung des Laserscanners aus, da die, aufgrund der Odometrie bestimmte, Trajektorie sehr schlecht ist und schon dadurch der Wert des APCE sehr groß ist. Der Vollständigkeit halber sind ATE und APCE trotzdem in Tabelle 5.11 aufgelistet. Auch anhand einer Abbildung der Trajektorie mit Kalibrierung kann kaum erkannt werden, ob die Punktwolke durch die Kalibrierung besser geworden ist. Im Folgenden werden allerdings die hier berechneten Werte für die Kalibrierung des Laserscanners bei der Kalibrierung der Odometrie verwendet. Es stellt sich heraus, dass die Kalibrierung der Odometrie dadurch verbessert werden kann.

Bei der Kalibrierung der Odometrie ergibt sich die Matrix A der Parameter des Bewegungsmodells.

Verwendete Daten	ATE / APCE	Stdabw.	1. Quartil	Median	3. Quartil
Rohdaten:					
Trajektorie	21,3289	14,9467	12,2574	17,524	34,0623
Punktwolke	22,1769	15,1183	13,1613	18,5986	37,074
Kalibrierter Laserscanner:					
Trajektorie	21,3289	14,9467	12,2574	17,524	34,0623
Punktwolke	22,1911	15,1284	13,1749	18,6133	37,1056
Kalibrierte Odometrie:					
Trajektorie	7,32699	3,69542	4,8215	6,70111	10,1885
Punktwolke	7,5132	3,90517	5,06198	7,842	12,2566
Beides nacheinander kalibriert:					
Trajektorie	6,41209	3,54406	4,17884	6,85148	9,265
Punktwolke	6,74512	3,80752	5,50314	7,57375	11,0624

Tab. 5.11: Diese Tabelle stellt die Fehler für die Ergebnisse der verschiedenen Experimente mit dem Intel-Datensatz im Vergleich zu den korrigierten Daten dar.

Matrix Σ enthält die berechneten Varianzen der einzelnen Werte.

$$A = \begin{pmatrix} 0{,}9590 & -0{,}4253 & -0{,}0022 \\ 0{,}0046 & 1{,}2277 & 0{,}0922 \\ 0{,}0558 & 2{,}1987 & 0{,}9667 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} 0{,}00006 & 0{,}0746 & 0{,}0001 \\ 0{,}00006 & 0{,}0750 & 0{,}0001 \\ 0{,}00006 & 0{,}0722 & 0{,}0001 \end{pmatrix} \qquad [5.8]$$

Die zweite Spalte der Matrix A enthält die Parameter, die den Einfluss des lateralen Bewegungsanteils gewichten. Sie weichen am meisten von den Werten der Einheitsmatrix ab, was sich in den größeren Varianzen in Matrix Σ widerspiegelt. Da allerdings bei einem Roboter mit Differentialantrieb der laterale Bewegungsanteil im Idealfall gleich 0 und auch in der Praxis sehr klein ist, haben diese Werte keinen großen Einfluss auf die Trajektorie. Im Gegensatz dazu gibt der Parameter $\mu_{T_d} = A_{(3,1)}$ an, dass der Roboter sich für jeden Meter den er vorwärts zurücklegt um 0,0558 rad $\approx 3{,}2^\circ$ nach links dreht. Damit wird im Bewegungsmodell der Rechtsdrall in der Odometrie des Roboters ausgeglichen. Da der Roboter insgesamt eine Strecke von 560 Metern zurücklegt, ist der Einfluss von μ_{T_d} auf die Trajektorie insgesamt beachtlich.

Die mit den berechneten Parametern kalibrierten Rohdaten sind in Abbildung 5.17 zu sehen. Die Trajektorie ist zwar offensichtlich deutlich besser, als vor der Kalibrierung, allerdings unterscheidet sie sich durch den Einfluss nicht-systematischer Fehler immer noch sehr stark von der optimalen Trajektorie. Der ATE beträgt für die kalibrierte Trajektorie nur noch 7,3 anstatt 21,3 (siehe Tabelle 5.11).

Wie schon bei den Experimenten mit Simulationen in Kapitel 5.2.3 bemerkt, funktioniert die gleichzeitige Kalibrierung von Odometrie und Umgebungssensoren nicht, da das Bewegungsmodell zu mächtig ist und so von dem Optimierungsverfahren nur lokale Minima gefunden werden. Dies ist erwartungsgemäß auch bei den realen Datensätzen nicht anders. Zur Veranschaulichung zeigt Abbildung 5.18 das Ergebnis der Optimierung. Die Umgebungskarte ist zwar bis auf eine Drehung korrekt, die Trajektorie unterscheidet sich jedoch stark von der korrekten Trajektorie.

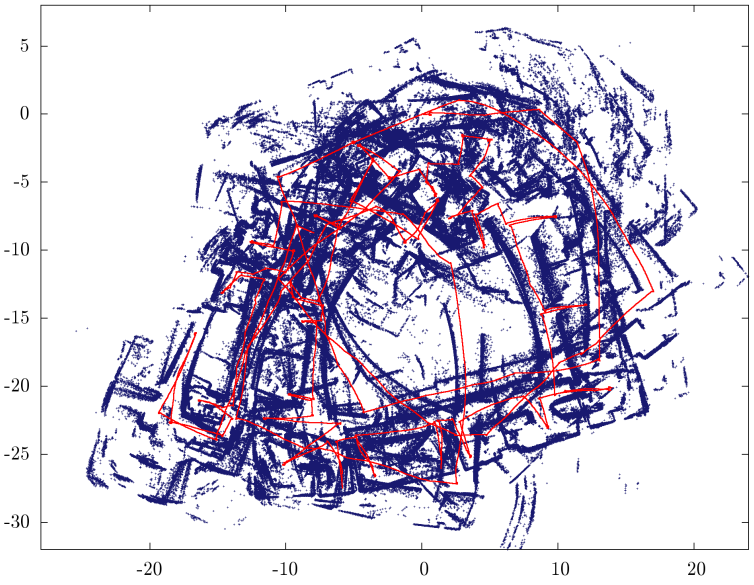


Abb. 5.17: Diese Abbildung zeigt die Trajektorie und die Punktwolke, wie sie mit den berechneten Kalibrierungsparametern des Laserscanners aus der Intel-Logdatei erzeugt werden können.

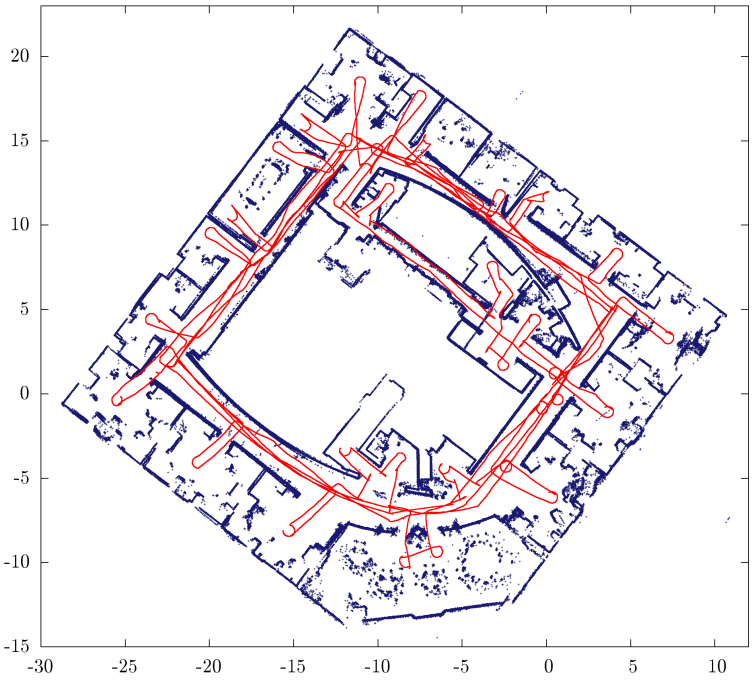


Abb. 5.18: Diese Abbildung zeigt die optimierte Punktwolke und Trajektorie, die bei gleichzeitiger Kalibrierung von Odometrie und Laserscanner ausgegeben wird.

Als Alternative wurden in Kapitel 5.2.4 Odometrie und Umgebungssensoren nacheinander kalibriert. Dies wird auch hier für den Intel-Datensatz getestet. Wie zuvor werden dafür zuerst die Umgebungssensoren, in diesem Fall der Laserscanner kalibriert. So können die Werte für die Odometrie im zweiten Schritt mit Hilfe eines bereits kalibrierten Laserscanners bestimmt werden. Das Ergebnis der Kalibrierung des Laserscanners ist wie oben bereits beschrieben die Pose $\begin{pmatrix} 0{,}0959 & 0{,}0068 & 0{,}0364 \end{pmatrix}^T$. Die anschließende Kalibrierung der Odometrie gibt die in Matrix A angegebenen Werte aus. Die Varianzen sind in Matrix Σ dargestellt.

$$A = \begin{pmatrix} 0{,}9608 & -0{,}4188 & 0{,}0020 \\ -0{,}0010 & 1{,}0049 & -0{,}0004 \\ 0{,}0543 & 2{,}1613 & 0{,}9783 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} 0{,}0008 & 1{,}4709 & 0{,}0017 \\ 0{,}0008 & 1{,}4747 & 0{,}0017 \\ 0{,}0008 & 1{,}4684 & 0{,}0017 \end{pmatrix} \qquad [5.9]$$

Mit den berechneten Kalibrierungsparametern wird aus den Rohdaten wieder eine Trajektorie und eine Punktwolke berechnet, die in Abbildung 5.19 dargestellt ist. Im Vergleich zu der reinen Odo-

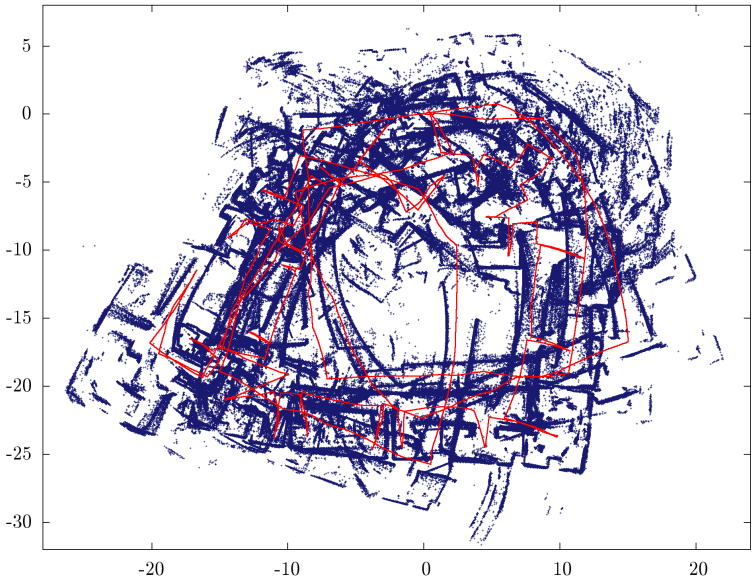


Abb. 5.19: Für die Punktwolke und Trajektorie in dieser Abbildung werden die Rohdaten mit den Kalibrierungsparametern visualisiert, die durch die Kalibrierung der Laserscanner und anschließend der Odometrie erhalten werden.

metriekalibrierung in Abbildung 5.17 ist der größte Unterschied, dass die gesamte Trajektorie und Punktwolke besser ausgerichtet sind. Dies spiegelt sich auch in dem niedrigeren ATE von 6,4 wider. Die restlichen Werte können wieder Tabelle 5.11 entnommen werden.

Abbildung 5.20 zeigt die optimierte Trajektorie und Punktwolke, die der SLAM-Algorithmus ohne Kalibrierung berechnet, wenn die hier berechneten Kalibrierungsparameter verwendet werden. Dabei konvergiert das Optimierungsverfahren in 43 Iterationen gegen einen finalen Fehler von 18,4. Zum Vergleich konvergiert das Optimierungsverfahren des SLAM-Algorithmus ohne Kalibrierung auf den Rohdaten in 50 Iterationen gegen einen Fehler von 21,7.

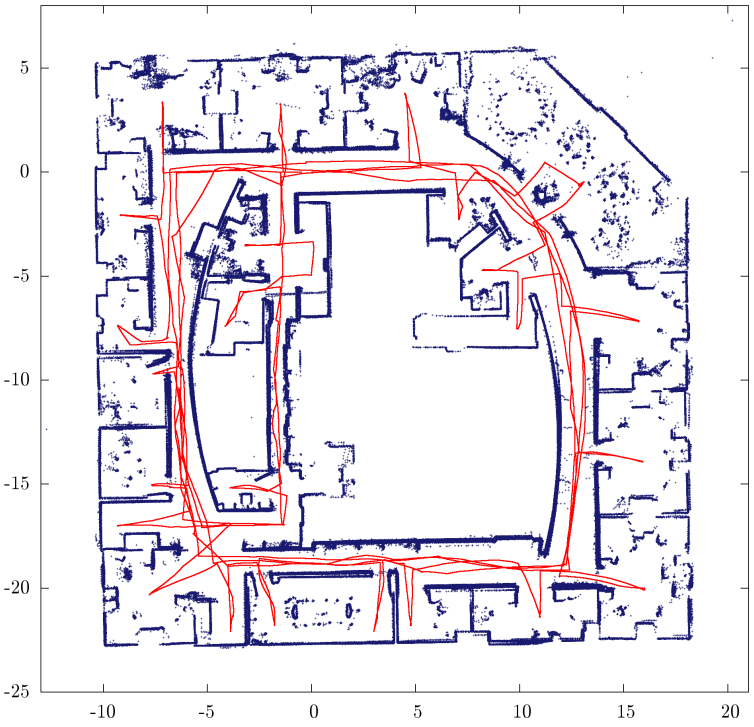


Abb. 5.20: Diese Abbildung zeigt die optimierte Trajektorie und Punktwolke, die der SLAM-Algorithmus ohne Kalibrierung ausgibt, wenn die initiale Kalibrierung an die berechneten Parameter angepasst wird.

Abschließend kann festgestellt werden, dass die Kalibrierung der Odometrie des im Intel-Datensatz verwendeten Roboters, eine deutliche Verbesserung der, nur anhand der Odometrie bestimmten, Trajektorie mit sich bringt. Außerdem wird das Ergebnis der Kalibrierung noch einmal besser, wenn vorher noch der Laserscanner kalibriert wird.

5.3.2 Experimente mit dem Roboter HoLLiE

In diesem Kapitel werden Experimente mit einem mobilen Roboter mit omnidirektionaler Kinematik durchgeführt. Die verwendete Roboterplattform HoLLiE, die am FZI entwickelt wurde, ist ein zweiarmiger Serviceroboter, der zur Fortbewegung mit Mecanum-Rädern ausgestattet ist (siehe Abbildung 5.21).

Abb. 5.21: Der Roboter HoLLiE (Quelle: www.fzi.de)

Diese Räder ermöglichen im Zusammenspiel die omnidirektionale Fortbewegung. Jedes Rad ist dabei mit kleineren tonnenförmigen Rollen besetzt, die sich frei drehen können und in einem Winkel von 45° zur Achse des Rades stehen.

Außerdem verfügt der Roboter über zwei Laserscanner, die knapp über dem Boden angebracht sind und von denen einer nach vorne und einer nach hinten ausgerichtet ist. Die Logdatei, die in diesem Kapitel für die Experimente verwendet wird, wurde innerhalb von knapp 11 Minuten aufgenommen. In dieser Zeit legte der Roboter eine Strecke von ca. 56 Metern in einem Gebäude zurück. Dabei wird zwar keine vollständige Schleife abgefahren, aber der Roboter fährt trotzdem mehrfach an der gleichen Stelle vorbei. Die Rohdaten sind in Abbildung 5.22, die mit dem SLAM-Algorithmus ohne Kalibrierung optimierten Daten in Abbildung 5.23 zu sehen. Verschiedene Eigenschaften der Fahrt des Roboters, wie sie aus der Logdatei ausgelesen wurden, sind in Tabelle 5.12 abgebildet. Dabei wird die Länge der gefahrenen Strecke aus den Odometriedaten berechnet und ist daher nur als grober Anhaltspunkt für die tatsächlich zurückgelegte Strecke zu werten.

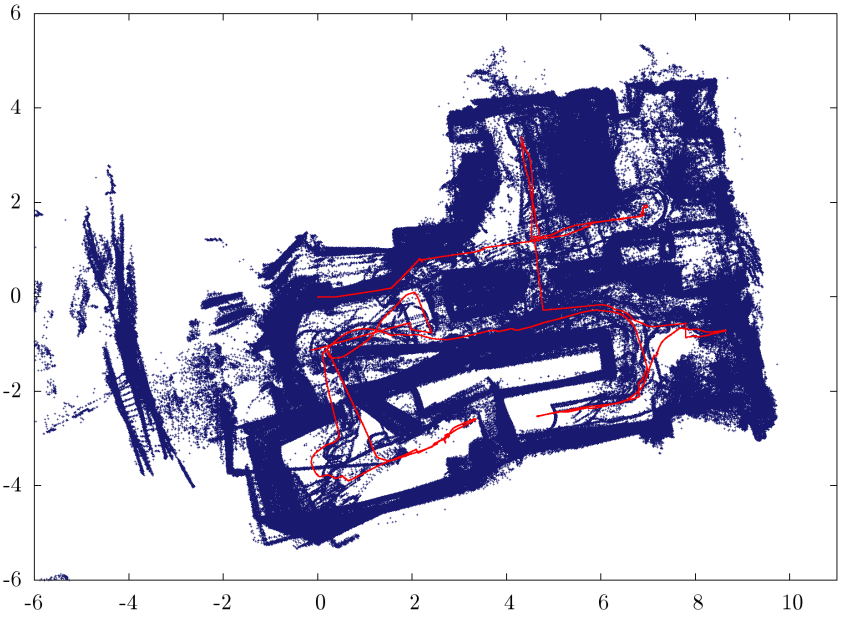


Abb. 5.22: Diese Abbildung visualisiert die Rohdaten der mit dem Roboter HoLLiE aufgenommenen Logdatei.

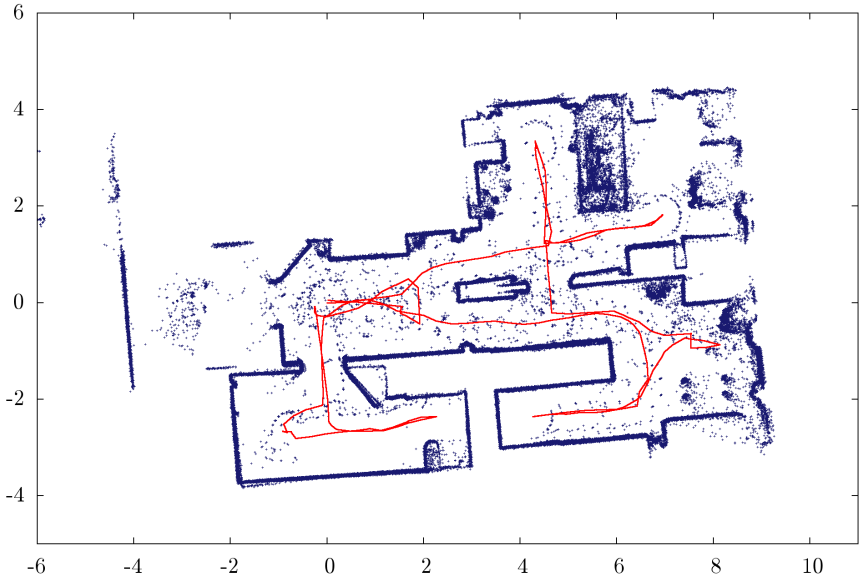


Abb. 5.23: Diese Abbildung zeigt die von dem SLAM-Algorithmus ohne Kalibrierung ausgegebene optimierte Trajektorie und Punktwolke.

HoLLiE Logdatei	
Dauer der Aufnahme	10 min 55 s
Gefahrene Strecke	56,2 m
Mittlere Geschwindigkeit	8,6 cm/s
Anzahl aller Messungen	120571
Anzahl der Odometriemessungen	93034
Frequenz der Odometriemessungen	142 Hz
Anzahl der Laserscanner	2
Anzahl der Laserscans	27537
Frequenz der Laserscans	42 Hz

Tab. 5.12: Diese Tabelle stellt einige Eigenschaften der Logdatei dar, die mit dem Roboter HoLLiE aufgenommen wurde.

An der Abbildung der Rohdaten lässt sich erkennen, dass sich bereits ohne Kalibrierung eine relativ gute Umgebungskarte ergibt. Das lässt auf eine bereits gut kalibrierte Odometrie schließen. Es sind einige klare Linien in der Punktwolke erkennbar, was darauf hindeutet, dass Kanten aus mehreren Scans genau aufeinander liegen. Dies bedeutet wiederum, dass auch die Kalibrierung der Laserscanner nicht besonders schlecht ist. Trotzdem sind einige Kanten erkennbar, die offensichtlich aufeinander liegen müssten, aber in der Abbildung weit voneinander entfernt sind. Dies liegt einerseits an kleineren Fehlern in der Kalibrierung und vor allem an nicht-systematischen Fehlern.

Wie in den anderen Experimenten werden zuerst Odometrie und Laserscanner einzeln kalibriert. Das Ergebnis der Kalibrierung der Laserscanner ist in Tabelle 5.13 dargestellt. Die berechneten

	Sensor 1			Sensor 2		
	x_1	y_1	θ_1	x_2	y_2	θ_2
Initial	0,3386	0,3121	-44,1407	-0,3386	-0,3121	135,8598
Kalibriert	0,3428	0,3092	-44,2620	-0,3413	-0,3137	136,4925

Tab. 5.13: Diese Tabelle stellt die initale Schätzung für die Posen der beiden Laserscanner des Roboters HoLLiE den kalibrierten Werten gegenüber.

Werte weichen jeweils um weniger als 5 Millimeter beziehungsweise 0,7° ab. Dies bestätigt die Vermutung, dass die Werte bereits initial gut kalibriert waren. Dementsprechend ändert sich die Abbildung der Rohdaten durch die Kalibrierung der Laserscanner kaum.

Wird nun die Odometrie einzeln kalibriert, ergeben sich die, in Matrix A dargestellten, Kalibrierungsparameter. Wieder sind in Matrix Σ die zugehörigen Varianzen dargestellt.

$$A = \begin{pmatrix} 0{,}9711 & 0{,}0139 & 0{,}0007 \\ -0{,}0031 & 0{,}9878 & 0{,}0076 \\ 0{,}0153 & -0{,}0190 & 0{,}9982 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} 0{,}0304 & 0{,}0254 & 0{,}0229 \\ 0{,}0303 & 0{,}0250 & 0{,}0229 \\ 0{,}0299 & 0{,}0240 & 0{,}0226 \end{pmatrix} \qquad [5.10]$$

Alle Werte der Matrix A weichen um weniger als 0,03 von der Einheitsmatrix ab. Trotzdem beeinflusst der Wert $A_{(3,1)} = \mu_{T_d} = 0{,}0153$ die berechnete Trajektorie des Roboters deutlich. Der Wert

sorgt dafür, dass sich in der Berechnung der Trajektorie der Roboter bei einer vorwärts zurückgelegten Strecke von einem Meter zusätzlich um ca. 1° nach links dreht. Über die gesamte Trajektorie führt dies zu einer merklichen Veränderung. Abbildung 5.24 zeigt die Rohdaten unter Verwendung der in Gleichung 5.10 dargestellten Kalibrierungsparameter. Tatsächlich ist die Trajektorie durch

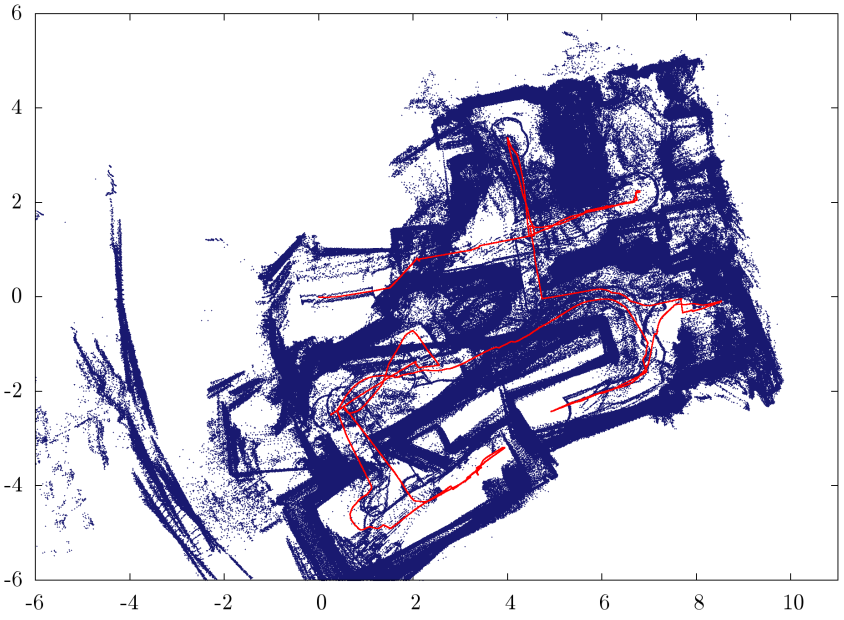


Abb. 5.24: Die Abbildung zeigt die Trajektorie und Punktwolke nach der Kalibrierung der Odometrie. Für einen besseren Vergleich zu Abbildung 5.22 wurden der gezeigte Ausschnitt gleich gewählt.

die Kalibrierung intuitiv gesehen eher schlechter geworden. Anfangs- und Endpunkt sollten nah beieinander liegen, sind aber durch die Kalibrierung weiter auseinander gerückt. Eine Erklärung für diesen Effekt könnte sein, dass Fehler in der Kalibrierung der Laserscanner durch diese Werte mit ausgeglichen werden.

Deshalb wird die Methodik der vorherigen Experimente angewendet. Die Kalibrierung aller Sensoren wird so vorgenommen, dass zuerst die Umgebungssensoren kalibriert werden und anschließend mit diesen Werten die Odometrie. Die Ergebnisse der Kalibrierung der Laserscanner wurden bereits oben beschrieben. Die neuen Werte für die Kalibrierung der Odometrie, die unter Verwendung der berechneten Posen der Laserscanner bestimmt wurden, sind in wieder in Matrix A und Σ dargestellt:

$$A = \begin{pmatrix} 0{,}9788 & 0{,}0163 & 0{,}0058 \\ -0{,}0002 & 0{,}9866 & 0{,}0083 \\ -0{,}0006 & -0{,}0105 & 0{,}9945 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} 0{,}0293 & 0{,}0222 & 0{,}0223 \\ 0{,}0293 & 0{,}0219 & 0{,}0223 \\ 0{,}0289 & 0{,}0209 & 0{,}0220 \end{pmatrix} \qquad [5.11]$$

Zwar hatte die Kalibrierung der Laserscanner alleine kaum einen sichtbaren Einfluss auf die Abbildung der Rohdaten, dennoch hat sie zu leicht unterschiedlichen Ergebnissen der Odometriekalibrierung geführt. Besonders der zuvor betrachtete einflussreiche Wert $A_{(3,1)} = \mu_{T_d}$ hat sich stark geändert und dabei sogar das Vorzeichen gewechselt. Dementsprechend sieht auch die resultieren-

de Trajektorie deutlich besser aus (siehe Abbildung 5.25).

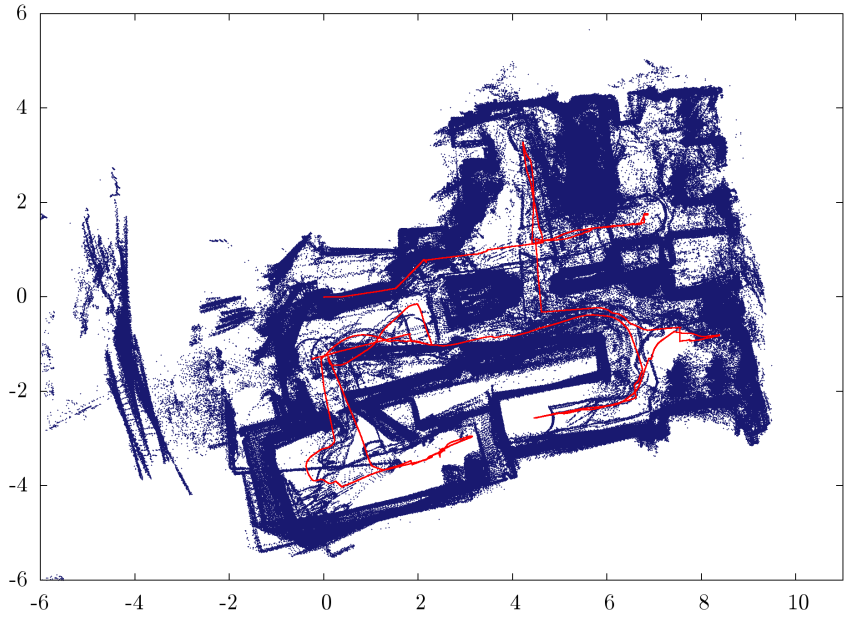


Abb. 5.25: Diese Abbildung zeigt die Trajektorie und Punktwolke nach der Kalibrierung von Laserscannern und Odometrie in dieser Reihenfolge.

Trotzdem ist aufgrund der guten initialen Kalibrierung kein großer Unterschied zu den Rohdaten ohne Kalibrierung in Abbildung 5.22 zu sehen. Wird der SLAM-Algorithmus ohne Kalibrierung einmal mit der initial gegebenen Kalibrierung und einmal mit den berechneten Kalibrierungsparametern ausgeführt, so konvergiert der Algorithmus mit den kalibrierten Parametern gegen einen Fehler von 8,9 statt 9,2 in 31 statt 38 Iterationen. Der größte Unterschied ist zu sehen, wenn der initiale Fehler, den der Algorithmus vor der Anwendung des Optimierungsverfahrens nur aufgrund der ersten Schätzungen für die verwendeten Zufallsvariablen ausgibt, betrachtet wird. Dieser liegt bei der nicht kalibrierten Version bei 360 000 und verringert sich durch die Kalibrierung auf 130 000.

Das Endergebnis der Optimierung ist dennoch kaum von dem Ergebnis in Abbildung 5.23 zu unterscheiden. Trotzdem kann aus den Experimenten mit dieser Logdatei geschlossen werden, dass sich die bereits sehr guten initialen Ergebnisse durch den Einsatz des Kalibrierungsalgorithmus nicht verschlechtert haben sondern statt dessen zu einer schnelleren Konvergenz des Optimierungsverfahrens mit leicht besserem finalem Fehler und vergleichbarem Ergebnis geführt haben. Des Weiteren wird einmal mehr die Wichtigkeit der guten Kalibrierung der Umgebungssensoren vor der Kalibrierung der Odometrie mit diesem Bewegungsmodell unterstrichen.

5.3.3 Experimente mit einem mit Sensoren ausgestatteten Auto

Die letzte für die Experimente in diesem Kapitel verwendete Logdatei wurde mit einem Auto in einer Tiefgarage aufgenommen. Bei dem Auto handelt es sich um „CoCar“ , ein instrumentiertes Testfahrzeug mit dem am FZI unter anderem an dem Thema „autonomes Fahren“ geforscht wird. Anders als bei den bisher verwendeten Datensätzen basiert die Lenkung eines Autos, genannt Ackermann- oder Achsschenkel-Lenkung, darauf, dass die Räder an einer Achse einzeln gedreht werden. Auch die auf dieser Lenkung beruhende Odometrie kann, wie in diesem Kapitel gezeigt wird, mit dem gewählten allgemeinen Bewegungsmodell kalibriert werden.

Das Auto ist mit drei Laserscannern bestückt und fuhr während der Aufnahme in einer Tiefgarage insgesamt knapp 1000 Meter weit. Weitere Eigenschaften der Aufnahme der Autofahrt finden sich in Tabelle 5.14. Die Trajektorie des Autos wurde so gewählt, dass jeder Bereich der Tiefgarage

Logdatei Tiefgarage	
Dauer der Aufnahme	9 min 43 s
Gefahrene Strecke	990,7 m
Mittlere Geschwindigkeit	1,7 m/s
Anzahl aller Messungen	102086
Anzahl der Odometriemessungen	58335
Frequenz der Odometriemessungen	100 Hz
Anzahl der Laserscanner	3
Anzahl der Laserscans	43751
Frequenz der Laserscans	25 Hz

Tab. 5.14: Diese Tabelle stellt die Eigenschaften der Logdatei dar, die mit einem Auto in einer Tiefgarage aufgenommen wurde.

mindestens einmal besucht wurde. Dabei wurden viele Punkte auch deutlich häufiger passiert. Dadurch eignet sich der Datensatz sehr gut für den SLAM-Algorithmus, da mit Hilfe von Loop Closings über die Zeit akkumulierte Fehler entdeckt und korrigiert werden können.

Da das Auto im Vergleich zu dem Roboter HoLLiE deutlich schneller fährt, ist auch der Abstand zwischen zwei Odometriemessungen oder zwei Scans der Umgebung deutlich größer. Es konnte allerdings kein negativer Einfluss dieser Eigenschaft auf die Qualität der Lösung festgestellt werden. Im Gegensatz dazu werden Odometriemessungen, die einen zu kleinen Abstand zur vorigen Messung aufweisen, bei der Erstellung des Faktorgraphen sogar ignoriert, um einerseits den Faktorgraphen möglichst klein zu halten und andererseits numerischen Problemen bei sehr kleinen Differenzen vorzubeugen. Da bei der Aufnahme mit HoLLiE auf diese Weise deutlich mehr Messungen ignoriert werden, wird der erstellte Faktorgraph für diese Logdatei im Vergleich zu der mit HoLLiE aufgenommenen deutlich größer, obwohl die Anzahl der gemachten Messungen und die Dauer der Aufnahme in beiden Fällen ähnlich sind. Damit geht auch einher, dass die Erstellung des Faktorgraphen in diesem Fall deutlich länger dauert. Bei ähnlich hochfrequenten Messungen hängt die Größe des Faktorgraphen allgemein eher von der zurückgelegten Strecke als von der Größe der Logdatei ab.

Die, nur mit Hilfe der Odometrie aus der Logdatei berechnete, Trajektorie und Punktwolke ist in Abbildung 5.26 dargestellt, die Ausgabe des SLAM-Algorithmus ohne Kalibrierung in Abbildung 5.27. Wieder ist erkennbar, dass die Odometrie bereits gut kalibriert ist. Teile der Trajektorie scheinen sehr gut zusammen zu passen, während an einzelnen Stellen Fehler in der Orientierung des Roboters entstehen.

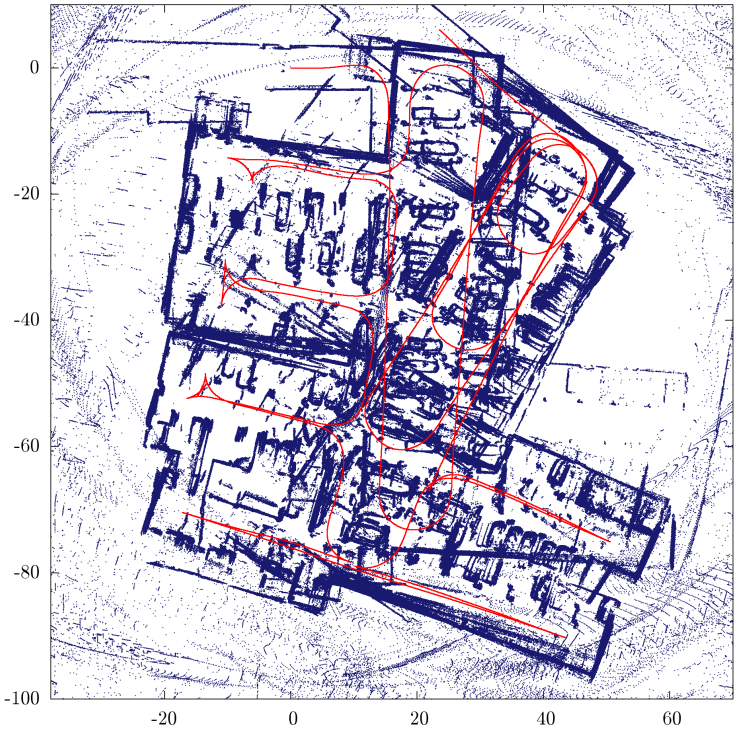


Abb. 5.26: In dieser Abbildung sind die Rohdaten der mit einem Auto in einer Tiefgarage erstellten Logdatei visualisiert.

In einem ersten Versuch werden Odometrie und Laserscanner wieder einzeln kalibriert. Das Ergebnis der Kalibrierung der Laserscanner ist in Tabelle 5.15 dargestellt. Die kalibrierten Sensorposen

	Sensor 1			Sensor 2			Sensor 3		
	x_1	y_1	θ_1	x_2	y_2	θ_2	x_3	y_3	θ_3
Initial	3,6100	0,5500	34,31	3,6100	-0,5600	-29,81	-0,9600	0,0000	180,00
Kalibriert	3,4900	0,7834	34,10	3,5070	-0,2416	-30,12	-0,9766	0,2094	179,80
Abweichung	0,1200	0,2334	0,21	0,1030	0,3184	0,31	0,0166	0,2094	0,20

Tab. 5.15: Diese Tabelle stellt die initale Schätzung für die Posen der drei Laserscanner des Testfahrzeugs CoCar den kalibrierten Werten gegenüber. Dabei sind x und y jeweils in Metern und θ in Grad angegeben.

unterscheiden sich hier deutlich von den initial angenommenen. Nur die Ausrichtung der Sensoren bleibt ungefähr gleich. Weiter ist auffällig, dass die y-Werte zwischen 20 cm und 32 cm in die gleiche Richtung abweichen. Da alle Werte um mindestens 20 cm abweichen, kann dies auch als eine Verschiebung des angenommenen Koordinatensystems des Autos um mindestens 20 cm relativ zu

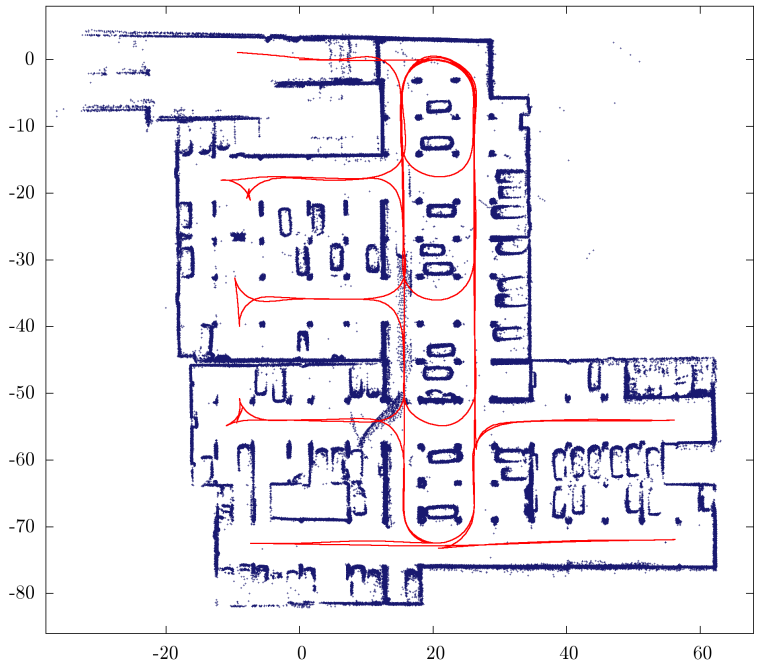


Abb. 5.27: Diese Abbildung zeigt die von dem SLAM-Algorithmus ohne Kalibrierung ausgegebene optimierte Trajektorie und Punktwolke für die mit dem Auto aufgenommene Logdatei.

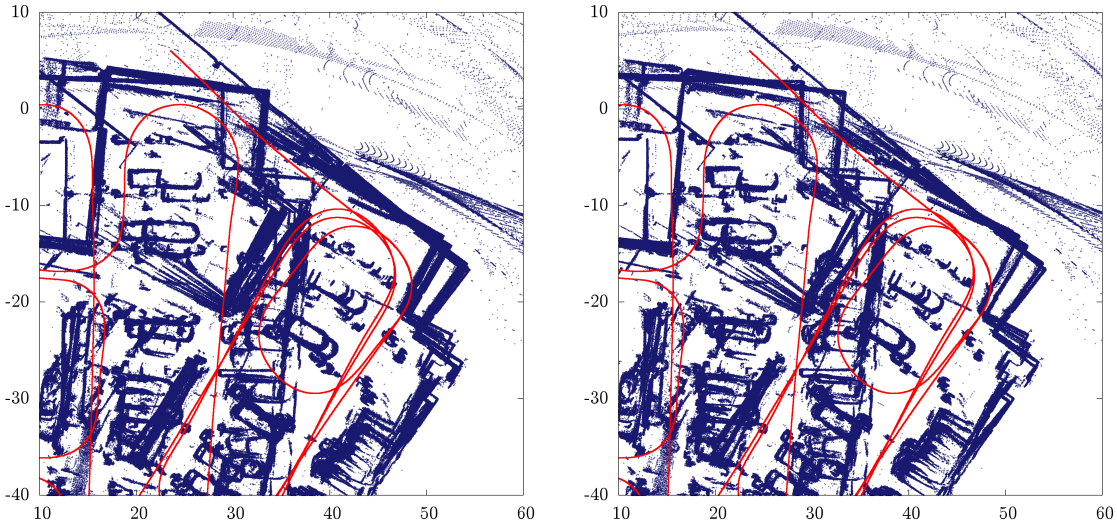
den Sensoren gewertet werden. Die Unterschiede in der Abweichung sind dann wieder durch die Kalibrierung der Sensoren begründet.

Um den Effekt der Kalibrierung der Laserscanner deutlich zu machen wird in Abbildung 5.28 ein kleiner Ausschnitt der Rohdaten vor und nach der Kalibrierung gegenübergestellt. In diesem Ausschnitt ist bei genauerem Hinsehen erkennbar, dass die Kanten in der Punktwolke durch die Kalibrierung schärfer werden. Da der Ausschnitt trotzdem einen Bereich von 50 mal 50 Metern abdeckt ist klar, dass die Auswirkung der Verschiebung eines Sensors um einige Zentimeter nicht besonders groß ist.

Die Kalibrierung der Odometrie, ohne vorherige Kalibrierung der Laserscanner, ergibt folgende in Matrix A beschriebene Kalibrierungsparameter mit den in Σ beschriebenen Varianzen:

$$A = \begin{pmatrix} 0{,}9495 & 5{,}2915 & -0{,}0711 \\ 0{,}0035 & -0{,}8103 & -0{,}0182 \\ 0{,}0025 & 0{,}5557 & 0{,}9627 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} 0{,}0010 & 170{,}8 & 0{,}1072 \\ 0{,}0012 & 292{,}8 & 0{,}1347 \\ 0{,}0006 & 157{,}9 & 0{,}0682 \end{pmatrix} \qquad [5.12]$$

Die Einträge der zweiten Spalte in Matrix A weichen am stärksten von den Werten der Einheitsmatrix ab. Diese Werte haben ebenfalls extrem hohe Varianzen und können daher nicht als verlässlich gewertet werden. Abbildung 5.29 zeigt die mit diesem Bewegungsmodell kalibrierten Daten. Als Test für die Auswirkung der Werte der zweiten Spalte wurden diese in einem zweiten Versuch gleich den Werten der Einheitsmatrix gesetzt. Die Darstellung, der so kalibrierten Daten lässt sich



(a) Vor der Kalibrierung

(b) Nach der Kalibrierung

Abb. 5.28: Abbildung (a) visualisiert die Rohdaten der Logdatei, mit der gegebenen Kalibrierung. In Abbildung (b) sind vor der Visualisierung die Posen der Laserscanner entsprechend dem Ergebnis der Kalibrierung eingestellt worden.

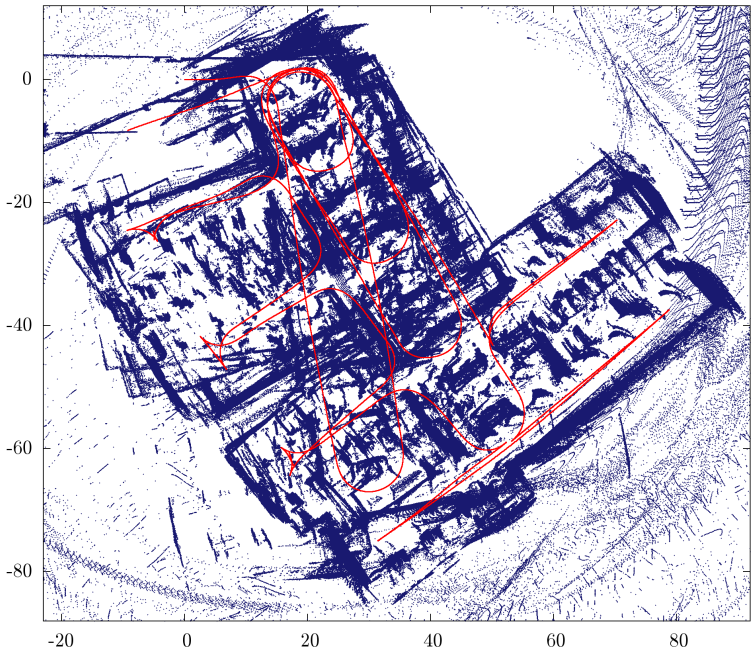


Abb. 5.29: Diese Abbildung zeigt die nur mit Hilfe der Odometrie bestimmte Trajektorie und Umgebungspunktwolke nach der Kalibrierung der Odometrie mit den Werten aus Matrix A in Gleichung 5.12.

dabei mit dem bloßen Auge nicht von der Darstellung in Abbildung 5.29 unterscheiden. Daraus lässt sich schließen, dass aufgrund der Kinematik eines Autos die Parameter, die den Einfluss der lateralen Bewegung widerspiegeln keine Rolle spielen. Dies wiederum bedeutet, dass in den rohen Odometriedaten nahezu kein lateraler Bewegungsanteil vorhanden ist.

Insgesamt unterscheidet sich die Trajektorie des Autos durch die Kalibrierung stark von der ursprünglich berechneten Trajektorie. Die neue Trajektorie scheint einerseits besser zu sein, da sie in sich bis auf die erste Runde im Parkhaus stimmig ist. Allerdings ist die Punktwolke der Umgebung durch die Kalibrierung deutlich verschwommener geworden. Dies legt den Schluss nahe, dass mit der Kalibrierung der Odometrie auch nicht-systematische Fehler ausgeglichen wurden.

Des Weiteren zeigt Abbildung 5.30 einen Ausschnitt der im Kalibrierungsalgorithmus optimierten Trajektorie. Es ist deutlich ein hin- und her-zucken der Trajektorie an einigen Stellen zu erkennen.

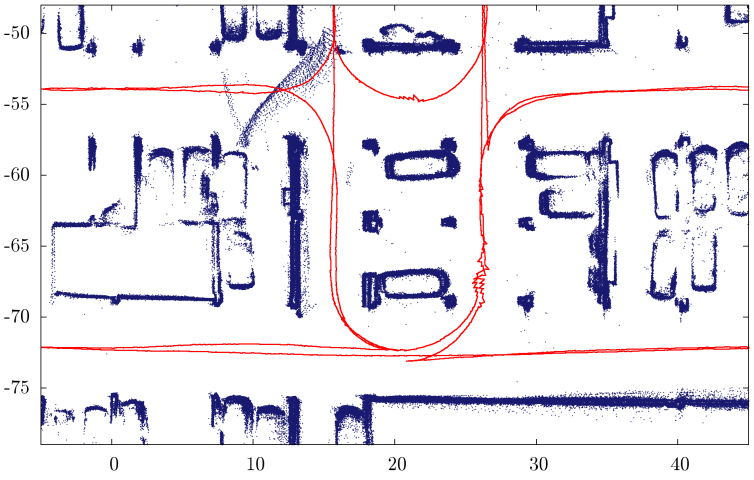


Abb. 5.30: Diese Abbildung zeigt einen Ausschnitt der, in dem Kalibrierungsverfahren für die Odometrie optimierten, Trajektorie des Autos in der Tiefgarage. Die Zick-Zack-Linien sprechen für eine schlechte initiale Kalibrierung der Laserscanner.

Dies entsteht vermutlich durch die Verwendung der fehlerhaften initialen Kalibrierungsparameter für die Laserscanner.

Wird die Odometrie unter Verwendung der oben berechneten Kalibrierungsparameter der Laserscanner kalibriert, so ergeben sich folgende Werte. Matrix A enthält dabei wieder die Kalibrierungsparameter, Matrix Σ ihre Varianzen.

$$A = \begin{pmatrix} 0{,}9495 & 5{,}2915 & -0{,}0711 \\ 0{,}0035 & -0{,}8103 & -0{,}0182 \\ 0{,}0025 & 0{,}5557 & 0{,}9627 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} 0{,}0010 & 170{,}8 & 0{,}1072 \\ 0{,}0012 & 292{,}8 & 0{,}1347 \\ 0{,}0006 & 157{,}9 & 0{,}0682 \end{pmatrix} \qquad [5.13]$$

Durch Anwendung der berechneten Kalibrierungsparameter auf die Rohdaten ergibt sich die in Abbildung 5.31 dargestellte Trajektorie und Punktwolke. Das Ergebnis sieht offensichtlich deut-

lich schlechter aus als das Ergebnis ohne Kalibrierung (siehe Abbildung 5.26). Wird die mit

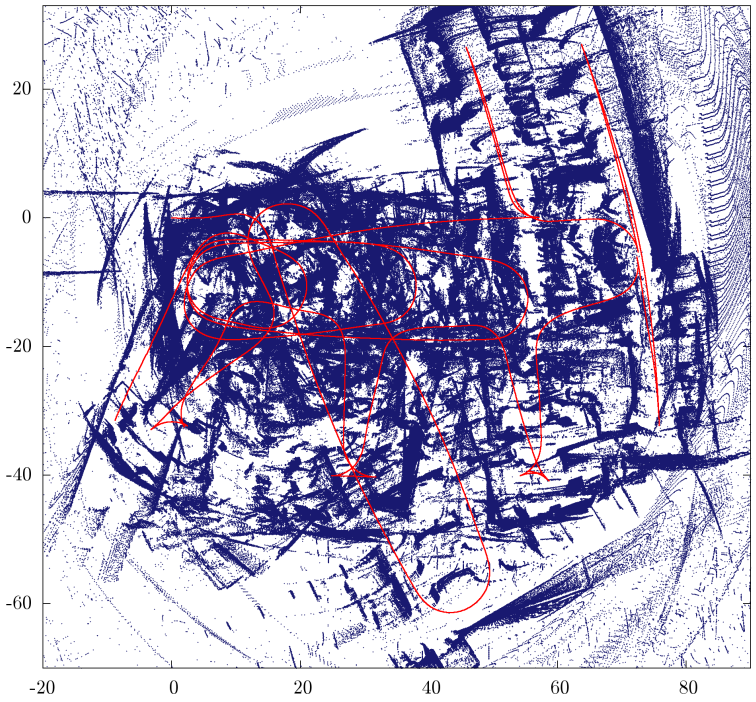


Abb. 5.31: Diese Abbildung zeigt die nur mit Hilfe der Odometrie bestimmte Trajektorie und Umgebungspunktwolke nachdem zuerst die Umgebungssensoren und anschließend die Odometrie kalibriert wurden.

dem SLAM-Algorithmus ohne Kalibrierung optimierte Trajektorie und Punktwolke aus Abbildung 5.27 als Referenz genommen, so beträgt der ATE der Rohdaten 10,7, während der ATE der kalibrierten Daten bei 36,5 liegt. Auch dies bestätigt also, dass die Rohdaten durch die Kalibrierung schlechter geworden sind.

Um die Eignung der berechneten Kalibrierungsparameter zu prüfen, wurde der SLAM-Algorithmus ohne Kalibrierung auf die mit den neuen Kalibrierungsparametern angepassten Daten angewandt. Dabei stellt sich heraus, dass die kalibrierten Rohdaten, obwohl sie so viel schlechter aussehen, im Faktorgraphen zu einem kleineren initialen Fehler von 1 073 000 statt 1 126 000 im Optimierungsverfahren und einer schnelleren Konvergenz in 54 statt 98 Iterationen führen. Dies ist trotz dem deutlich höheren ATE der kalibrierten Daten möglich, da die über den Faktorgraph aufsummierten Fehler immer nur relative statt absolute Fehler sind.

Außerdem ist die Anzahl der gefundenen Loop Closings bei Verwendung der kalibrierten Daten mit 5455 etwas größer als der Wert von 5105 ohne Kalibrierung. Um falsche Loop Closings zu vermeiden, wird im Verfahren für jeden Kandidat für ein Loop Closing ein Plausibilitätstest durchgeführt, der auf dem Abstand der beiden Sensorposen laut initialer Schätzung im Faktorgraph beruht. Offensichtlich wurde dieser Test mit den kalibrierten Daten häufiger bestanden. Trotzdem wurden keine falschen Loop Closings akzeptiert. Lediglich der finale Fehler ist mit 62,4 bei den

kalibrierten Daten geringfügig höher als der Fehler der nicht kalibrierten Daten mit 59,5.

Die optimierte Trajektorie und Punktwolke sehen ebenfalls sehr gut aus und unterscheiden sich kaum von den ohne die Kalibrierungsparameter berechneten Daten (siehe Abbildung 5.32). Nur

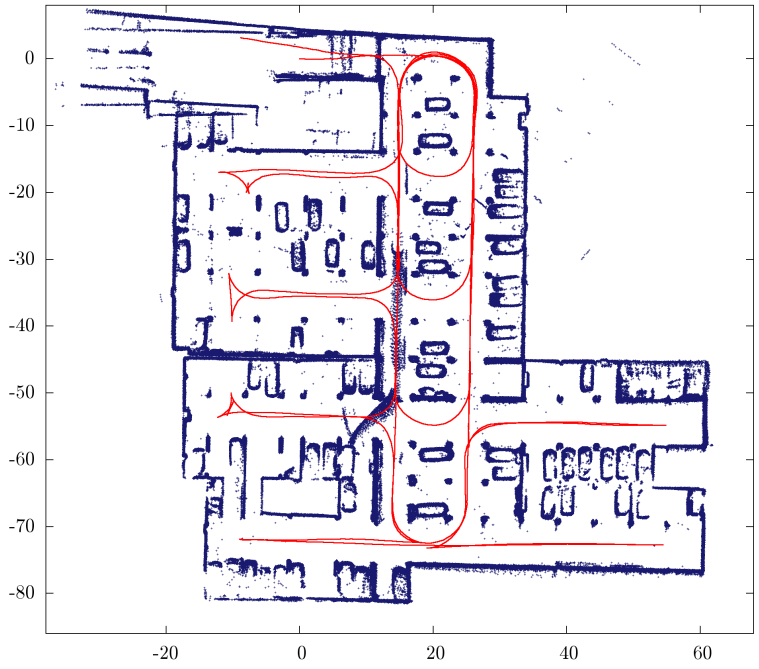


Abb. 5.32: Diese Abbildung zeigt die optimierte Trajektorie und Umgebungspunktwolke, die von dem SLAM-Algorithmus ohne Kalibrierung ausgegeben wird, wenn zuvor die berechneten Kalibrierungsparameter fest eingestellt werden.

an Start- und Endpunkt der Trajektorie laufen die Umgebungspunktwolken etwas auseinander. Dies liegt daran, dass dort keine Loop Closings zwischen dem Anfang der Trajektorie und dem Ende der Trajektorie gefunden wurden und deshalb der Einfluss des ungenau kalibrierten Bewegungsmodells, die Trajektorien verzerrt.

Insgesamt kann festgestellt werden, dass die Wahl der Kalibrierungsparameter durch das Optimierungsverfahren, zwar durch den resultierenden geringeren initialen Fehler gerechtfertigt ist, der Algorithmus also in sich konsistent arbeitet, allerdings ist das Ergebnis trotzdem nicht zufriedenstellend. Hier müsste mit weiteren Daten getestet werden, ob die Kalibrierungsparameter auch mit anderen Logdateien zu geringeren initialen Fehlern im SLAM-Algorithmus führen. Eine Vermutung wäre, dass die Kalibrierungsparameter so gewählt wurden, dass vor allem auch die nicht-systematischen Fehler in dieser Logdatei minimiert werden. Auf dieses Phänomen genannt Overfitting wird in Kapitel 5.3.4.1 noch weiter eingegangen.

5.3.4 Weitere Experimente und Ergebnisse

In diesem Kapitel wird auf ein Experiment zur Vermeidung von Overfitting eingegangen und das Laufzeitverhalten des Algorithmus diskutiert.

5.3.4.1 Overfitting der Kalibrierungsparameter

Mit dem Intel-Datensatz wird in diesem Abschnitt ein Test auf Overfitting in der Kalibrierung durchgeführt. Als Overfitting wird in der Theorie des maschinellen Lernens ein Effekt bezeichnet, der sich negativ auf die Qualität des lernenden Algorithmus auswirkt. Von Overfitting wird genau dann gesprochen, wenn sich die Ergebnisse des Algorithmus bei der Anwendung auf die Trainingsdaten zwar verbessern, aber sich bei der Anwendung auf neue Daten verschlechtern. Dies ist ein Zeichen dafür, dass der Algorithmus zufällige Eigenschaften der Trainingsdaten „auswendig“ lernt, anstatt das allgemeine Konzept zu lernen.

Übertragen auf den hier vorgestellten Kalibrierungsalgorithmus wäre von Overfitting zu sprechen, wenn mit den berechneten Kalibrierungsparametern zwar der Fehler des SLAM-Algorithmus ohne Kalibrierung angewendet auf die Logdatei sinken würde, aber gleichzeitig der Fehler bei einer weiteren Logdatei steigen würde. Die zweite verwendete Logdatei müsste dabei natürlich in ähnlicher Umgebung und mit geringem zeitlichen Abstand zur ersten aufgenommen worden sein. Da eine solche Logdatei im Fall des Intel-Datensatzes nicht zur Verfügung steht, wird der Test in diesem Fall so durchgeführt, dass die Kalibrierung mit Hilfe der ersten Hälfte der Logdatei durchgeführt wird und als Vergleichsdatensatz die zweite Hälfte verwendet wird.

Dabei werden wieder zuerst die Umgebungssensoren und danach die Odometrie kalibriert. Die Ergebnisse sind in Tabelle 5.16 im Vergleich zu den in Kapitel 5.3.1 berechneten Werten zu sehen. Die berechneten Parameter unterscheiden sich bis auf kleine Ausnahmen kaum. Für die Scanner-

	Laserscanner			Odometrie	Varianzen der Odometriewerte
	x_1	y_1	θ_1		
Alle Daten	0,0959	0,0068	0,0364	$\begin{pmatrix} 0{,}9608 & -0{,}4188 & 0{,}0020 \\ -0{,}0010 & 1{,}0049 & -0{,}0004 \\ 0{,}0543 & 2{,}1613 & 0{,}9783 \end{pmatrix}$	$\begin{pmatrix} 0{,}0008 & 1{,}4709 & 0{,}0017 \\ 0{,}0008 & 1{,}4747 & 0{,}0017 \\ 0{,}0008 & 1{,}4684 & 0{,}0017 \end{pmatrix}$
Erste Hälfte	0,0942	0,0020	-0,3992	$\begin{pmatrix} 0{,}9603 & -0{,}2738 & -0{,}0027 \\ -0{,}0046 & 0{,}9910 & 0{,}0005 \\ 0{,}0554 & 2{,}0515 & 0{,}9800 \end{pmatrix}$	$\begin{pmatrix} 0{,}0016 & 3{,}2805 & 0{,}0039 \\ 0{,}0016 & 3{,}2875 & 0{,}0039 \\ 0{,}0016 & 3{,}2775 & 0{,}0038 \end{pmatrix}$
Abweichung	0,0017	0,0048	0,4356	$\begin{pmatrix} 0{,}0005 & -0{,}145 & 0{,}0047 \\ 0{,}0036 & 0{,}0139 & -0{,}0009 \\ -0{,}0011 & 0{,}1098 & -0{,}0017 \end{pmatrix}$	$\begin{pmatrix} -0{,}0008 & -1{,}8096 & -0{,}0022 \\ -0{,}0008 & -1{,}8128 & -0{,}0022 \\ -0{,}0008 & -1{,}8091 & -0{,}0021 \end{pmatrix}$

Tab. 5.16: In dieser Tabelle sind die optimierten Kalibrierungsparameter dargestellt. Einmal wurden sie dabei mit Hilfe des kompletten Intel-Datensatzes berechnet, einmal nur mit der ersten Hälfte der Daten.

pose beträgt die Abweichung weniger als 0,5 cm beziehungsweise 0,5°. Bei den Odometrieparametern unterscheiden sich nur die Werte in der zweiten Spalte der Matrix stärker. Bei diesen

Werten ist auch die berechnete Varianz sehr groß. Dies liegt daran, dass diese Werte den Einfluss der lateralen Bewegung des Roboters auf das Bewegungsmodell ausdrücken. Da der verwendete Roboter einen Differentialantrieb hat, ist die laterale Bewegung im idealen Fall immer 0 und auch bei realen Daten sehr klein. Daher haben auch diese Kalibrierungsparameter nur einen sehr kleinen Einfluss auf die geschätzte Trajektorie des Roboters. In Abbildung 5.33 ist die ganze Trajektorie und Punktwolke, die mit den neuen Kalibrierungsparametern berechnet wurden, zu sehen.

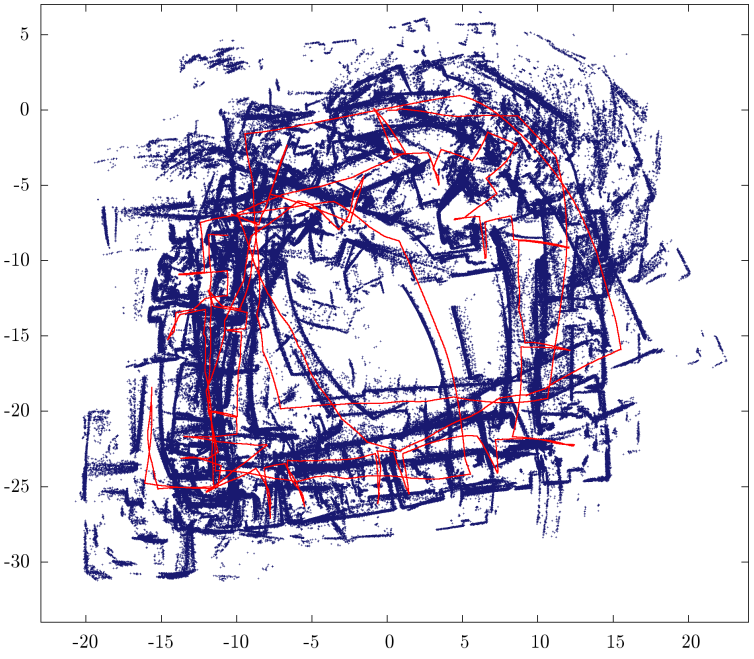


Abb. 5.33: Diese Abbildung zeigt die Trajektorie und die Punktwolke für die Intel-Logdatei, wie sie mit Hilfe der nur auf Basis der ersten Hälfte der Logdatei berechneten Kalibrierungsparameter berechnet wird.

Der ATE beträgt für die gesamte Logdatei ungefähr 6,60. Dieser Wert ist nicht wesentlich schlechter als der Wert von 6,41, der bei der Kalibrierung mit den gesamten Daten erreicht wird. Des Weiteren beträgt der Wert des ATE mit den neu berechneten Kalibrierungsparametern, wenn er nur auf der zweiten Hälfte der Daten bestimmt wird, 15,56. Bei Verwendung der alten Kalibrierungswerte, die mit der ganzen Logdatei bestimmt wurden, ist der ATE für die zweite Hälfte mit 15,31 nicht viel kleiner. Insgesamt sind die Fehler für die zweite Hälfte der Logdatei natürlich deutlich höher als für die erste Hälfte, da der erste Punkt der zweiten Hälfte schon durch die Fehler in der ersten Hälfte eine verfälschte Position hat. Für die erste Hälfte der Logdatei ergibt sich mit den neuen Kalibrierungsparametern ein ATE von 4,12 während die alten Parameter zu einem ATE von 4,35 führen.

Eine gewisse Anpassung an die Trainingsdaten ist also nachweisbar, trotzdem sind die berechneten Kalibrierungsparameter auch für die Verwendung mit den gesamten Daten geeignet. Des Weiteren muss berücksichtigt werden, dass bei der Kalibrierung auf Basis der ersten Hälfte der Logdatei

auch nur die Hälfte der Trainingsdaten zur Verfügung stehen, was sich in den höheren Varianzen der berechneten Kalibrierungsparameter widerspiegelt.

Falls die Sorge besteht, dass Overfitting bei der Kalibrierung auftreten könnte, kann auf dies auf die eben beschriebene Art getestet werden. Eine Methode, die in einem solchen Fall zu einem besseren Ergebnis führen könnte, wäre, die beiden Hälften der Logdatei einzeln für die Kalibrierung zu verwenden und anschließend die Mittelwerte der gefundenen Parameter zu bilden. Fälle in denen Overfitting ein Problem sein könnte, wären beispielsweise Logdateien mit großen nichtsystematischen Fehlern.

5.3.4.2 Laufzeiten des Algorithmus

Die Laufzeiten des Algorithmus wurden bis jetzt in der Auswertung überhaupt nicht diskutiert. Dies liegt daran, dass das Hauptziel dieser Arbeit, einen funktionierenden Algorithmus für die automatische Kalibrierung zu entwickeln, unabhängig von dessen Laufzeit ist. Da außerdem ein offline-Algorithmus verwendet wird, ist eine schnelle Laufzeit nicht entscheidend für die Funktionalität des Ansatzes, wie dies bei einem online-Einsatz der Fall wäre. Aus diesen Gründen wurde auch bei der Implementierung nicht besonders auf die Effizienz bezüglich der Rechenzeit geachtet. Trotzdem wird in diesem Abschnitt der Vollständigkeit halber grob auf die Performance des Algorithmus eingegangen.

Mit Abstand am längsten dauert innerhalb des Algorithmus die Erstellung des Faktorgraphen. Ist dieser fertig erstellt, so braucht das Optimierungsverfahren in den verwendeten Beispielen zwar auch noch zwischen 10 Sekunden bei dem mit dem Roboter HoLLiE aufgenommenen Datensatz und 5 Minuten bei dem Datensatz aus der Tiefgarage, allerdings überwiegt die zur Erstellung des Faktorgraphen verwendete Zeit in jedem Fall deutlich. Diese variiert bei den verwendeten Beispielen zwischen einigen Minuten und mehreren Stunden. Des Weiteren kann beobachtet werden, dass die benötigte Zeit zur Abarbeitung eines neuen Laserscans mit der Länge der Trajektorie stark ansteigt. Dies lässt sich damit erklären, dass für jeden neuen Scan aus allen alten Scans mit Hilfe sogenannter Features potentielle Kandidaten für ein Loop Closing gesucht werden. Sollte die Anwendung des Algorithmus für sehr lange Trajektorien nötig sein, könnte an dieser Stelle viel Zeit gespart werden, indem beispielsweise nicht mehr alle alten Scans als Kandidaten in Betracht gezogen werden. Für eine gute Kalibrierung ist dies im Normalfall nicht nötig.

Durch die Einführung der Kalibrierung steigt die für die Erstellung des Faktorgraphen benötigte Zeit nicht wesentlich an. In dem Graphen werden nur zwei bis fünf neue Knoten hinzugefügt und durch den höheren Grad der Faktoren entsteht kein messbar größerer Aufwand. Das Optimierungsverfahren braucht hingegen bei eingeschalteter Kalibrierung im Schnitt etwas länger, jedoch in keinem Fall mehr als doppelt so lang. Da die Laufzeiten sowieso nicht für den Online-Einsatz des Algorithmus geeignet sind und die für die Erstellung des Graphen benötigte Zeit überwiegt, ist dies allerdings nicht besonders relevant. So kann insgesamt festgestellt werden, dass die Laufzei-

ten des SLAM-Algorithmus durch die Verwendung der Kalibrierung nicht wesentlich steigen.

5.4 Zusammenfassung der Ergebnisse

In diesem Kapitel wurden zwei Ansätze zur Evaluation des entwickelten Verfahrens verfolgt. Zuerst wurde der Algorithmus mit Simulationsdaten, die kaum nicht-systematische Fehler enthalten, getestet. Dabei waren die optimalen Kalibrierungsparameter in den verschiedenen Versuchen jeweils bekannt, so dass die Güte der von dem Algorithmus gefundenen Parameter objektiv beurteilt werden konnte.

Dabei stellte sich heraus, dass das Verfahren in der Lage ist, sowohl die omnidirektionale Odometrie als auch die zwei Laserscanner des Roboters einzeln sehr gut zu kalibrieren. Allerdings ist es mit dem verwendeten Bewegungsmodell nicht möglich, Odometrie und Umgebungssensoren des Roboters gleichzeitig zu kalibrieren, da das resultierende Optimierungsproblem viele lokale Minima aufweist. Statt dessen wurden fortan immer zuerst die Umgebungssensoren und danach die Odometrie kalibriert. Dies führt ebenfalls zu sehr guten Kalibrierungsergebnissen, falls die initiale Schätzung des Bewegungsmodells nicht zu ungenau ist.

Anschließend wurde das Kalibrierungsverfahrens mit verschiedenen realen Datensätzen, die mit möglichst unterschiedlichen Roboterplattformen aufgenommen wurden, getestet. Hierbei besteht die zusätzliche Schwierigkeit darin, dass die Daten zum Teil erhebliche nicht-systematische Fehler enthalten. Trotzdem wurden die Parameter in den meisten Fällen zufriedenstellend kalibriert. Einzig bei einem mit einem Auto in einer Tiefgarage aufgenommenen Datensatz kam es zu dem Fall, dass die Rohdaten nach der Kalibrierung der Odometrie schlechter aussahen. Trotzdem konvergierte der SLAM-Algorithmus mit den kalibrierten Daten schneller als vorher und fand ein ähnlich gutes Ergebnis. Eine Vermutung ist, dass das berechnete Bewegungsmodell durch eine zu starke Anpassung der Kalibrierungsparameter an nicht-systematische Fehler entstanden ist. Um dies zu bestätigen oder zu widerlegen, wären allerdings noch weitere Versuche nötig. Aufgrund dieser Problematik wurden in einem Unterkapitel erste Ideen, wie Overfitting allgemein erkannt und vermieden werden könnte, vorgestellt.

6 Zusammenfassung und Ausblick

In dieser Arbeit wurde ein Verfahren für automatischen Kalibrierung der Umgebungssensoren und der Kinematik mobiler Roboter entwickelt. Dafür wurde auf einen bestehenden graphbasierten Algorithmus zur Lösung des SLAM-Problems aufgebaut und dieser um Faktoren für die Kalibrierung erweitert.

Eine verbesserte Kalibrierung kann, wie festgestellt wurde, Fehler in der Lokalisierung des Roboters und in der vom Roboter erstellten Umgebungskarte stark verringern. Dies ist gerade bei über eine längere Zeit hinweg autonom fahrenden Robotern oder Robotern, die zur Kartierung einer unbekannten Umgebung eingesetzt werden sehr wichtig. Weiter konnte gezeigt werden, dass manuelle Kalibrierungsmethoden oft aufwändig in der Durchführung oder qualitativ unzureichend sind. Zusätzlich erwies sich eine häufige Rekalibrierung bei Einsätzen des Roboters in stark unterschiedlichen Umgebungen als sinnvoll.

Ein Ziel dieser Arbeit war es deshalb den Aufwand der Kalibrierung zu minimieren. Dies ist dadurch gelungen, dass für die Ausführung des Algorithmus nur eine Aufzeichnung der Sensormessungen während einer beliebigen Fahrt des Roboters und eine initiale Schätzung der Kalibrierungsparameter benötigt werden. Bezüglich der initialen Schätzung wurde in Experimenten festgestellt, dass diese nicht besonders genau sein muss, um eine Konvergenz des Algorithmus zu gewährleisten. Bei den Umgebungssensoren sollte die Position in beiden Koordinaten auf 15 cm und die Orientierung auf 15° genau geschätzt werden. Beides stellt in der Praxis gewöhnlich kein Problem dar. Die Odometriesensoren liefern in allen betrachteten Fällen, direkt eine Schätzung der Roboterbewegung und damit bereits eine initiale Schätzung des Bewegungsmodells. Die Güte dieser Schätzung war in allen Experimenten ausreichend, um eine qualitativ hochwertige Kalibrierung zu ermöglichen.

Ein weiteres Ziel dieser Arbeit bestand darin, einen Algorithmus zu entwickeln, der verschiedene Roboter mit unterschiedlichen Kinematiken und unterschiedlicher Anzahl an Umgebungssensoren kalibrieren kann. Diese Eigenschaft grenzt den entwickelten Algorithmus gegenüber bereits existierenden Algorithmen ab, die sich jeweils auf die Kalibrierung einer speziellen Roboterplattform beschränken.

Um die Verwendung des Verfahrens mit verschiedenen Kinematiken zu ermöglichen, wurde die Bewegung des Roboters mit Hilfe eines sehr allgemeinen stochastischen Bewegungsmodells beschrieben. Dadurch konnte die Kalibrierung der Odometrie ohne Anpassung des Verfahrens mit einem Roboter mit Differentialantrieb, einer omnidirektionalen Plattform und einem Auto mit

Ackermann-Lenkung erfolgreich getestet werden.

Außerdem wurden Experimente mit ein, zwei und drei Umgebungssensoren durchgeführt, wobei die Anzahl der verwendeten Sensoren von dem Algorithmus automatisch erkannt wird. Die verwendeten Umgebungssensoren waren jedoch in allen Fällen 2D-Laserscanner. Die Verwendung des Algorithmus mit anderen Sensortypen ist dennoch möglich. Bei der Implementierung des Verfahrens wurde auf leichte Erweiterbarkeit geachtet und die nötigen Bedingungen, die für eine Verwendung anderer Sensortypen erfüllt sein müssen, beschrieben.

Der theoretische Teil der Arbeit besteht hauptsächlich daraus einen gegebenen SLAM-Algorithmus um die automatische Kalibrierung zu erweitern. Dafür wurden zuerst die Vor- und Nachteile der verschiedenen Verfahren zur Lösung des SLAM-Problems erörtert und die Möglichkeit der Erweiterung um die Kalibrierung geprüft. Mit der Entscheidung für ein graphbasiertes Verfahren, bestanden die Hauptaufgaben im Anschluss darin, die Struktur des Graphen anzupassen und neue Faktoren für die Kalibrierung zu modellieren. Letzteres konnte mit Hilfe einer mathematischen Struktur, den Lie-Gruppen, erreicht werden.

Unter Verwendung von Simulationen konnte die Korrektheit der mathematischen Modellierung und die Funktionalität des Algorithmus festgestellt werden. In der Simulation konnte der Algorithmus Odometrie und Umgebungssensoren einzeln jeweils auch bei schlechten initialen Schätzungen korrekt kalibrieren. Werden allerdings alle Sensoren gleichzeitig kalibriert, so konvergiert der Algorithmus in lokalen Minima, die stark von der optimalen Lösung abweichen können. Daraus wurde der Schluss gezogen, dass das Bewegungsmodell zu viele Freiheitsgrade hat und dadurch zu mächtig ist, um bei der gleichzeitigen Kalibrierung mit den Umgebungssensoren, ein korrektes Ergebnis zu liefern.

Eine Lösung wäre die Einschränkung des Bewegungsmodells, was allerdings auch mit einer Einschränkung der abbildbaren Odometrien einher ginge. In dieser Arbeit wurden stattdessen Umgebungssensoren und Odometrie in der Reihenfolge der Nennung nacheinander kalibriert. Dieser Ansatz führte in der Simulation zu sehr guten Ergebnissen, die im Anschluss mit realen Datensätzen verifiziert werden konnten.

Außerdem wurde festgestellt, dass Overfitting bei unvorteilhaften Trainingsdaten ein Problem darstellen könnte, und es wurden mögliche Schritte zur Vermeidung des Effekts beschrieben.

6.1 Ausblick

Im Themenbereich der automatischen Kalibrierung gibt es noch einige offene Fragestellungen und auch das hier vorgestellte Verfahren wirft einige neue Fragen auf und bietet Möglichkeiten zur Erweiterung.

Eine Möglichkeit das in dieser Arbeit vorgestellte Verfahren zu erweitern und die gleichzeitige Ka-

librierung von Umgebungssensoren und Odometrie zu ermöglichen, wäre, das Bewegungsmodell für verschiedene häufig verwendete Kinematiken anzupassen, einzuschränken und entsprechende Faktoren zu modellieren. Dann könnte der Algorithmus, unter Angabe der Kinematik des Roboters, das entsprechende Bewegungsmodell verwenden und so eventuell alle Sensoren gleichzeitig kalibrieren. Jedoch ist nicht klar, wie stark das Bewegungsmodell dafür eingeschränkt werden müsste.

Da in dieser Arbeit nur 2D-Laserscanner zur Erkennung der Umgebung verwendet wurden, wäre ein nächster Schritt, das Verfahren mit anderen Umgebungssensoren zu testen. Dafür käme beispielsweise die Verwendung von 3D-Laserscannern oder Videokameras in Frage. Auch die Kombination verschiedener Sensortypen könnte Vorteile bringen.

Das Verfahren zur automatischen Kalibrierung in einen Online-SLAM-Algorithmus zu integrieren und damit die Kalibrierung während der Fahrt möglich zu machen, ist eine weitere mögliche Erweiterung. Allerdings wäre dies voraussichtlich eine größere Herausforderung, da die Gesamtlaufzeit, des entwickelten Algorithmus dafür um eine Größenordnung zu groß ist und außerdem die benötigte Rechenzeit pro Messwert während der Ausführung stetig ansteigt. Es existieren aber zumindest bereits SLAM-Verfahren, die auf einem graphbasierten Ansatz beruhen und für den Online-Einsatz geeignet sind.

Falls dieses Ziel aufgrund von Laufzeitproblemen nicht erreicht werden kann, wäre eine vorgelagerte interessante Fragestellung, ob sich ein Online-SLAM-Algorithmus so erweitern lässt, dass er automatisch melden kann, wenn die verwendeten Kalibrierungsparameter nicht mehr korrekt sind. In diesem Fall könnte die automatische Kalibrierung mit den Daten der Fahrt im Anschluss offline durchgeführt werden.

7 Literaturverzeichnis

[ACGM10] G. Antonelli, F. Caccavale, F. Grossi, and A. Marino, "Simultaneous calibration of odometry and camera for a differential drive mobile robot," in IEEE International Conference on Robotics and Automation (ICRA), 2010, pp. 5417–5422.

[Bai02] T. Bailey, "Mobile robot localisation and mapping in extensive outdoor environments," Ph.D. dissertation, University of Sydney, 2002.

[BF96] J. Borenstein and L. Feng, "Measurement and Correction of Systematic Odometry Errors in Mobile Robots," IEEE Transactions on Robotics and Automation, vol. 12, pp. 869–880, 1996.

[Bla47] D. Blackwell, "Conditional expectation and unbiased sequential estimation." Annals of Mathematical Statistics, vol. 18, pp. 105–110, 1947.

[Bla10] J.-L. Blanco, "A tutorial on SE(3) transformation parameterizations and on-manifold optimization," University of Malaga, Technical Report, 2010.

[Bor94] J. Borenstein, "The CLAPPER: a dual-drive mobile robot with internal correction of dead-reckoning errors," in Proceedings of the IEEE International Conference on Robotics and Automation, 1994, pp. 3085–3090.

[BSSC13] D. Bendera, M. Schikoraa, J. Sturmb, and D. Cremersb, "A Graph Based Bundle Adjustment for INS-Camera Calibration," ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. 1, no. 2, pp. 39–44, 2013.

[Cen08] A. Censi, "An ICP variant using a point-to-line metric," in IEEE International Conference on Robotics and Automation (ICRA), 2008, pp. 19–25.

[CMO08] A. Censi, L. Marchionni, and G. Oriolo, "Simultaneous maximum-likelihood calibration of odometry and sensor parameters," in IEEE International Conference on Robotics and Automation (ICRA), 2008, pp. 2098–2103.

[Del12] F. Dellaert, "Factor graphs and GTSAM: A hands-on introduction," 2012. [Online]. Available: http://smartech.gatech.edu/handle/1853/45226

[DWB06] H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping (SLAM): Part I,” IEEE Robotics & Automation Magazine, vol. 13, no. 2, pp. 99–110, 2006.

[Ead13] E. Eade, “Lie groups for 2d and 3d Transformations,” 2013. [Online]. Available: http://ethaneade.org/lie.pdf

[EP04] A. I. Eliazar and R. Parr, “Learning probabilistic motion models for mobile robots,” in Proceedings of the Twenty-First International Conference on Machine Learning (ICML). ACM, 2004, p. 32.

[Gal11] J. Gallier, Geometric Methods and Applications, ser. Texts in Applied Mathematics. Springer, 2011, vol. 38.

[GKN12] G. Grisetti, R. Kummerle, and K. Ni, “Robust optimization of factor graphs by using condensed measurements,” in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012, pp. 581–588.

[GVL96] G. H. Golub and C. F. Van Loan, Matrix Computations, 3rd ed. Baltimore, Maryland: Johns Hopkins University Press, 1996.

[HR03] A. Howard and N. Roy, “The Robotics Data Set Repository (Radish),” 2003. [Online]. Available: http://radish.sourceforge.net/

[Kal60] R. E. Kalman, “A new approach to linear filtering and prediction problems,” Journal of Fluids Engineering, vol. 82, no. 1, pp. 35–45, 1960.

[KD05] M. Kaess and F. Dellaert, “A Markov chain Monte Carlo approach to closing the loop in SLAM,” in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2005, pp. 643–648.

[KF09] D. Koller and N. Friedman, Probabilistic Graphical Models: Principles and Techniques. Cambridge, Massachusetts: MIT Press, 2009.

[KGB12] R. Kümmerle, G. Grisetti, and W. Burgard, “Simultaneous Parameter Calibration, Localization, and Mapping,” Advanced Robotics, vol. 26, no. 17, pp. 2021–2041, Dec. 2012.

[KGS+11] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, “g2o: A general framework for graph optimization,” in IEEE International Conference on Robotics and Automation (ICRA), 2011, pp. 3607–3613.

[KJR+12] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, “iSAM2:

Incremental Smoothing and Mapping Using the Bayes Tree," International Journal of Robotics Research (IJRR), 2012.

[KRD08] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental Smoothing and Mapping," Transactions on Robotics (TRO), 2008.

[LBAR98] T. D. Larsen, M. Bak, N. A. Andersen, and O. Ravn, "Location Estimation for an Autonomously Guided Vehicle using an Augmented Kalman Filter to Autocalibrate the Odometry," in First International Conference on Multisource-Multisensor Information Fusion (FUSION), 1998.

[Lev44] K. Levenberg, "A Method for the Solution of Certain Non-linear Problems in Least-Squares," Quarterly of Applied Mathematics, vol. 2, no. 2, pp. 164–168, Jul. 1944.

[LM97] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," Autonomous Robots, vol. 4, no. 4, pp. 333–349, 1997.

[Mar63] D. W. Marquardt, "An Algorithm for the Least-Squares Estimation of Nonlinear Parameters," SIAM Journal of Applied Mathematics, vol. 11, no. 2, pp. 431–441, Jun. 1963.

[MTKW02] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in Proceedings of the AAAI National Conference on Artificial Intelligence (NCAI), 2002, pp. 593–598.

[MTTS03] A. Martinelli, N. Tomatis, A. Tapus, and R. Siegwart, "Simultaneous localization and odometry calibration for mobile robot," in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), vol. 2, 2003, pp. 1499–1504.

[MU49] N. Metropolis and S. Ulam, "The Monte Carlo Method," Journal of the American Statistical Association, vol. 44, no. 247, pp. 335–341, 1949.

[Mä97] P. Mächler, "Robot odometry correction using grid lines on the floor," in Proceedings of the Second International Workshop on Mechatronical Computer Systems for Perception and Action, 1997.

[Rao45] C. R. Rao, "Information and the accuracy attainable in the estimation of statistical parameters," Bulletin of the Calcutta Mathematical Society, vol. 37, pp. 81–91, 1945.

[RT99] N. Roy and S. Thrun, "Online self-calibration for mobile robots," in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), vol. 3, 1999, pp. 2292–2297.

[Sel04] J. M. Selig, “Lie groups and Lie algebras in robotics,” in Computational Noncommutative Algebra and Applications. Springer, 2004, pp. 101–125.

[SSC90] R. Smith, M. Self, and P. Cheeseman, “Estimating Uncertain Spatial Relationships in Robotics,” in Autonomous Robot Vehicles, I. Cox and G. Wilfong, Eds. Springer, 1990, pp. 167–193.

[Sta] C. Stachniss, “Corrected Logfile from the Intel Research Lab in Seattle.” [Online]. Available: http://www2.informatik.uni-freiburg.de/~stachnis/datasets/datasets/intel-lab/intel.gfs.log.gz

[TA10] G. D. Tipaldi and K. O. Arras, “FLIRT - interest regions for 2d range data,” in IEEE International Conference on Robotics and Automation (ICRA), 2010, pp. 3616–3622.

[TBF05] S. Thrun, W. Burgard, and D. Fox, Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). Cambridge, Massachusetts: The MIT Press, 2005.

[Thr00] S. Thrun, “Probabilistic Algorithms in Robotics,” AI Magazine, vol. 21, no. 4, pp. 93–109, 2000.

[TL08] S. Thrun and J. Leonard, “Simultaneous Localization and Mapping,” in Springer Handbook of Robotics, B. Siciliano and O. Khatib, Eds. Springer, 2008, pp. 871–889.

[TM06] S. Thrun and M. Montemerlo, “The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures,” International Journal of Robotics Research (IJRR), vol. 25, no. 5-6, pp. 403–429, May 2006.

[Top72] J. Topping, Errors of observation and their treatment, ser. Science paperbacks. London, United Kingdom: Chapman and Hall, 1972.

[UHS07] J. Underwood, A. Hill, and S. Scheding, “Calibration of range sensor pose on mobile platforms,” in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2007, pp. 3866–3871.

[War71] F. W. Warner, Foundations of differentiable manifolds and Lie groups. Springer, 1971, vol. 94.