Abgabe zum Praktikum Mess- und Regelungstechnik Revision 1

Simon Klüpfel, Lukas Zeller
Robotik und Telematik
Universität Würzburg
Am Hubland, D-97074 Würzburg
simon.kluepfel@stud-mail.uni-wuerzburg.de, lukas.zeller@stud-mail.uni-wuerzburg.de

17. November 2022

1 Einleitung

1020 PS auf 2162kg, und von 0 auf 100 km/h in 2,1 Sekunden[17]. Dazu noch ein Level an autonomen Fahren das uns wie von Zauberhand durch den Verkehr bringt und die kognitiven Faehigkeiten des Menschen teilweise bereits uebersteigt. Die Rede ist vom Tesla Model S. Die enormen Fahrleistungen lassen sich durch modernsten Maschinenbau und Elektrotechnik sowie jahrelanger Entwicklung erreichen. Spannender ist nach Meinung der Autoren schon das (teil-)autonome Fahren und die Fortbewegung eines zwei Tonnen schweren Fahrzeugs in einer dynamischen Umgebung wie dem Strassenverkehr, ohne einen Unfall zu verursachen oder im Worst-Case Menschen zu verletzen oder gar zu toeten. Dass dies nicht passiert spricht einmal fuer das Know-How der Tesla-Ingenieure als auch der Zuverlaessigkeit der verwendeten Technik. In dieser Arbeit wird ein kleiner Einblick in die Technologie der Lokalisierung gegebenen und ein Vergleich angestellt zwischen zwei Methoden, der AMCL- und Odometrie-Lokalisierung.

2 Zur Lokalisierung

Aus [10], Folie 5-2, lassen sich drei Grundvoraussetzungen sowie das Ziel unserer Roboterlokalisierung definieren: Der Roboter bewegt sich in der Umgebung mittels bekannter Stuerbefehle, nimmt seine Umwelt mit Sensoren wahr und hat eine Karte seiner Umgebung zum Abgleich gespeichert. Das Ziel ist eine Positonsabschaetzung zum Zeitpunkt t in X- und Y-Position sowie ein Heading welches die Richtung angibt. Als Referenz ist ein in der Umgebung verankertes Koordinatensystem gegeben. Weiter unterscheidet man zum Beginn der Lokalisierung zwischen lokaler und globaler Selbstlokalisierung. Im ersten Fall ist die Initialposition des Roboters grob bekannt, das Ziel ist es bei Bewegung des Roboters eine Neuberechnung der Position durchzufuehren, dieses Verfahren nennt man *position tracking*. Bei der schwierigeren globalen Selbstlokalisierung kennen wir unsere Initialposition dagegen nicht, sie muss aus der Roboterbewegung und neugewonnenen Sensordaten errechnet werden. Hier kann das *kidnapped robot problem* eine Rolle spielen, bei der der Roboter in Zuge einer nicht abgeschlossenen Lokalisierung umgesetzt wird. Bei der Sensorik die zur Lageschaetzung verwendet wird unterscheidet man zwischen *propriozeptiven*, d.h. internen, und *exteriozeptiven*, externen, Sensoren. Ersteres misst die lokal am Roboter anfallenden *onboard* [1] Eigenschaften wie die Drehzahl der Radmotoren oder die Temperatur, die exteriozeptiven Sensoren liefern Messungen aus der

Umgebung des Roboters, als Beispiel nennt [1] Distanzmessungen wie Ultraschall, Laser, oder eine Kameraaufnahme der Umgebung.

3 Kurze Einführung in den Volksbot und ROS

In dieser Arbeit wurde das Robot Operating System, kurz *ROS*, auf einem Roboter, dem *Volksbot* verwendet. Dazu eine kurze Einfuehrung aus [6]: Die Volksbot-Plattform wurde vom Fraunhofer Institut IAIS aus St. Augustin entwickelt. Die Plattform besteht aus einer Roboterbaukasten untereinander kompatibler Bauteile, die eine langwierige Prototypenentwicklung zu vermeiden versucht indem sie eine Vielzahl von Varianten anbietet, die sich fuer die gewuenschte Anwendung jeweils passgenau zusammenstellen lassen. In Zentrum des Volksbot-Systems steht die Idee einer schnellen und kostenguenstigen Realisierung eines Roboters der sich sowohl fuer Lehr- und Entwicklungszwecke als auch fuer die Industrie eignet. Der in dieser Arbeit verwendete Roboter ist der *RT3-2* mit zwei bzw. einem (*RT-3*) passiven Rad und jeweils zwei angetriebenen Raeder die vorne rechts und links angebracht sind. Das Kuerzel RT steht dabei fuer *Rough Terrain*. Die technischen Daten sind wie folgt:

Abmessungen	580x520x315mm (L x B x H)
Gewicht	17kg
Raddurchmesser	260x85mm (aktive Räder)
	200mm (passive Räder)
Maximale Geschwindigkeit	2,2 $\frac{m}{s}$
Maximale Zuladung	25kg

aus [8]

Auf dem Roboter ist ein LIDAR-Sensor LMS100 der Firma *SICK* installiert. Er scannt seine Umgebung via eines 905nm-Infrarotlaser mit 25/50Hz und einer Reichweite von 5-20m unter einem Oeffnungswinkel von 270° [16]. Die Daten werden via Ethernet-Anschluss zu dem als ROS-Plattform verwendeten Laptop gesendet. Neben der Steuerung beziehungsweise Regelung ueber *ROS* lässt sich der Roboter via eines Joystick, ähnlich eines Gamecontrollers, manuell steuern. Das *Robot Operating System* stammt aus dem Bedarf fuer ein Open-Source-Roboter-Framework in der Robotercommunity. Es entstand Mitte der 2000er-Jahre aus Projekten an der *Stanford University* wie dem *STanford AI Robot - STAIR*. 2007 konnte mithilfe von *Willow Garage Inc*, einer amerikanischen Robotikfirma, und derer Resourcen die ROS-Konzepte in erste Implementationen umgesetzt werden. Heute hat *ROS* zehntausende Nutzer auf der ganzen Welt, die Anwendungen reichen von Amateurrobotern bis zu grossen automatisierten Industrieanlagen. *ROS* an sich besteht dabei aus vielen kleinen Programmen die ueber Nachrichten, die *Messages*, miteinander kommunizieren. Die Nachrichten werden dabei untereinander direkt verschickt ohne einen zentralen Bus oder aehnliches. Insgesamt folgt *ROS* damit einer Graphenarchitektur mit Knoten (englisch *nodes*), den Programmen, und Kanten (*edges*), sie entsprechen den Nachrichten. [13] Es werden sowohl vorgegebene ROS-Nodes, die vom Institut für Robotik und Telematik zur Verfügung gestellt wurden, als auch angepasste Nodes die die Autoren selbst erstellt beziehungsweise geändert haben, verwendet.

4 Odometrie, GMapping und AMCL im Überblick

Die Odometrie

Die Odometrie errechnet die Bewegung des Roboters ueber die Radgeschwindigkeit und -bewegung des *Volksbot*. Im Zentrum dieser Rechnung stehen die mathematischen Zusammenhaenge im Kreisbogen. Aus [2] wurde dazu folgende Grafik entnommen:

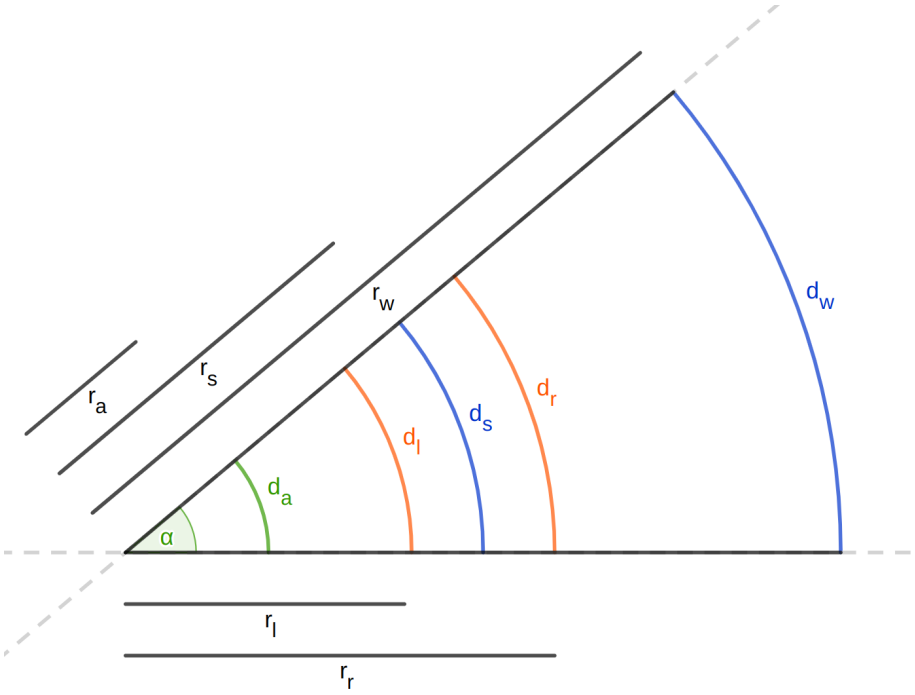

Abbildung 1: Aus [2]: Ein Kreissegment mit verschiedenen eingezeichneten Boegen bei unterschiedlichen Radien. Die Radienlaengen sind an der oberen Kreiskante aufgetragen. Der Winkel α des Kreissegments ist an der Spitze eingezeichnet

Dazu betrachten wir die mathematischen Ueberlegungen aus [2]:
Fuer zwei Kreisboegen d_l und d_r mit dem Radius r_l bzw. r_r ueber einen Winkel α gilt:
$\alpha = \frac{d_l}{r_l} = \frac{d_r}{r_r}$ oder auch $d_l = \alpha * r_l, d_r = \alpha * r_r$
Fuer einen Kreisbogen d_w mit dem Radius $r_w = r_r + r_l$ ueber α gilt:
$\alpha = \frac{d_w}{r_w} = \frac{d_w}{r_r + r_l}$ oder auch $d_w = \alpha * (r_r + r_l)$
Daraus folgt:
$d_r + d_l = \alpha * r_r + a * r_l = \alpha * (r_r + r_l) = d_w$
$\alpha = \frac{d_w}{r_w} = \frac{d_r + d_l}{r_r + r_l}$
Analog gilt fuer einen Kreisbogen d_a mit $r_a = r_r - r_l$:
$\alpha = \frac{d_a}{r_a} = \frac{d_r - d_l}{r_r - r_l}$
Sei r_s nun der Durchschnitt aus r_r und r_l. Sei d_s der Kreisbogen mit Radius r_s ueber α. Es gilt:
$r_s = \frac{(r_r + r_l)}{2}$
$d_s = \alpha * r_s = \alpha * \frac{(r_r + r_l)}{2} = \frac{\alpha * r_r + \alpha * r_l}{2} = \frac{d_r + d_l}{2}$

Die Sehne s unter einem Kreisbogen d_s mit dem Radius r_s ueber α hat die Laenge:
$s = 2 * r_s * sin(\frac{\alpha}{2})$

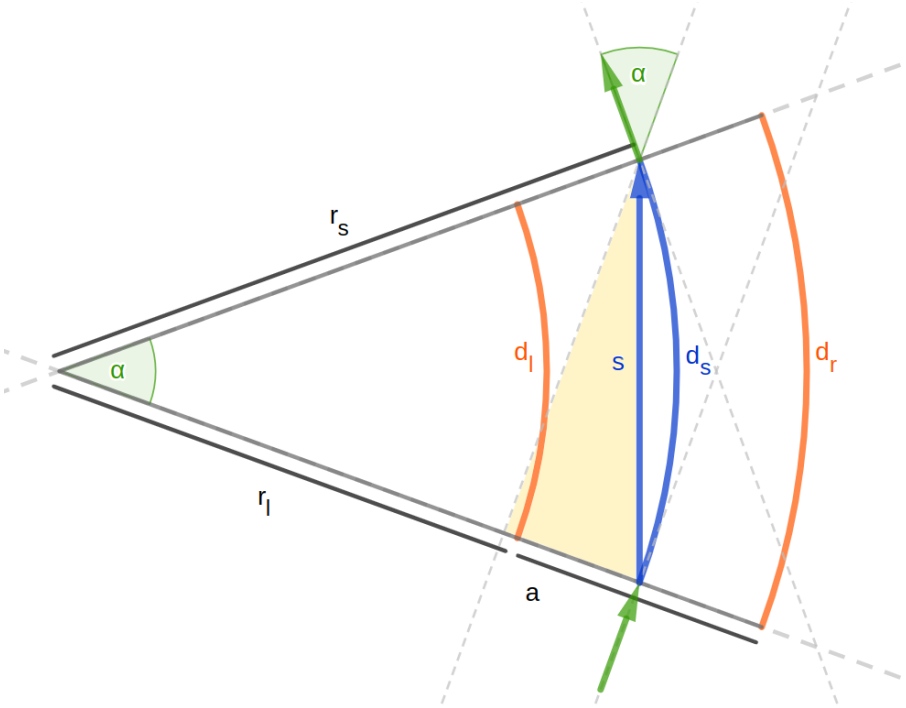

Abbildung 2: Aus [2]: Skizze des Kreissegments mit eingezeichneter Sehne im Kreisbogen der durch die Mitte des Roboters geht. Der Winkel α ist am Ende des Kreisbogens d_s eingezeichnet

Um nun die Distanz- und Richtungsaenderung ermitteln zu koennen betrachtet man die Sehne s des Kreisbogens d_s. Aus [2]: "Bekannt ist der Radabstand α, sowie die vom linken bzw. rechten Rad im betrachtetem Intervall zurueckgelegte Strecke d_l und d_r. Gesucht ist die zurueckgelegte Strecke s unter dem gefahrenen Kreisbogen d_s sowie die Aenderung des Blickwinkels $\alpha_{Richtungsaenderung}$."
Wir loesen also fuer $\alpha_{Richtungsaenderung}$, d_S, und s:
$\alpha = \frac{d_a}{r_a} = \frac{(d_r - d_l)}{r_r - r_l} = \frac{d_r - d_l}{a}$
$d_s = \frac{(d_r + d_l)}{2}$
$\alpha_{Richtungsaenderung} = \frac{d_s}{r_s}$ bzw. $r_s = \frac{d_s}{\alpha_{Richtungsaenderung}}$
Schlussendlich berechnet sich die Strecke s aus:
$s = 2 * r_s * sin(\frac{\alpha}{2}) = 2 * \frac{d_s}{\alpha} * sin(\frac{\alpha}{2}) = a * \frac{d_r + d_l}{d_r - d_l} * sin(\frac{d_r - d_l}{2a})$
Man beachte den Unterschied zwischen α und a. Ausserdem muss der Fall $d_r = d_l$ bzw. $\alpha = 0$ extra betrachtet werden. Aufgrund der Einfachheit dieser Sonderfaelle wird hier aus Platzgruenden nicht extra darauf eingegangen.

Praktische Aspekte der Odometrie

[11] fasst wie folgt zusammen: Odometrie liefert gute Ergebnisse in der Koppelnavigation ueber kurze Distanzen, jedoch summieren sich Fehler sehr schnell auf. In jeder Iteration der Odometrie enthalten unsere Ergebnisse Fehler und Rauschen in der X- und Y-Richtung als auch in der Ausrichtung θ, dem *Heading* des Roboters. Der Fehler im Heading ist der Killer der

Genauigkeit, da wir den fehlerbehafteten Wert aus der Berechnung in jeder weiteren Iteration als Grundlage nehmen.
Im Original:
Odometry can provide good dead-reckoning over short distances, but error accumulates very rapidly. At every step, we inject error (noise) into not just x and y, but also θ. The error in θ is the killer, since every error in θ will be amplified by future iterations.

Weiter nennt *OLSON* drei Probleme in der praktischen Nutzung der Odometrie:

1. Sensorenfehler (*sensor error*): Die Messwerte fuer d_l und d_r rauschen, da die Radsensoren zur Messung der Drehung und Bewegung keine idealen Ergebnisse liefern

2. Rutschen (*slippage*): Der Roboter unterliegt bei der Kurvenfahrt radseitig immer einem Rutschen. Fuer die Odometrie bedeutet dies das man selbst aus angenommmenen perfekten Odometriedaten den realen Pfad nicht ableiten kann

3. Fehler im Abschaetzen vom d_s oder des Raddurchmessers: Wir kennen die Mittellinie d_s des Roboters nie perfekt genau. Ausserdem haben die Raeder ebenfalls einen Groessenunterschied. Diese Unterschiede moegen klein anmuten, fuehren aber dazu dass die Odometrie ueber die Zeit einen Drift in eine Richtung aufweist und sich Kleinstfehler ueber einen lange genugen Intervall zu grossen Abweichungen summieren.

gMapping

gMapping ist ein auf *SLAM* basierender Algorithmus. *SLAM* steht fuer *simultaneous localisation and mapping*[9] Hierbei lokalisiert sich der Roboter in seiner Umgebung und kartiert diese gleichzeitig. Es braucht keine externe Karte um die Umgebung zu kartieren. [3] schreibt dazu:
It is considered to be a complex problem, because for localization a robot needs a consistent map and for acquiring a map a robot requires a good estimate of its location. This mutual dependency between the pose and the map estimates makes the SLAM problem hard and requires searching for a solution in a high-dimensional space
Der gMapping-Algorithmus nutzt zur Loesung des SLAM-Problems einen auf den Satz von Rao-Blackwell basierenden mathematischen Partikel-Filter und LiDAR-Daten[5], in unserem Experiment die des *LMS100*. Dei einzelnen Partikel stellen jeweils die geschaetzte Position des Roboters dar.
gMapping macht die aus diesen Berechnungen erzeugten Karten in einem *map-Topic* verfuegbar, die dann durch RViz als eine Karte dargestellt werden[15]

AMCL

AMCL, die *Adaptive Monte Carlo Localization* lokalisiert den Roboter in der Karte. Verwendet wird dafuer erneut ein Filter, der den Monte-Carlo-Algorithmus nutzt um sich in der Karte zurechtzufinden.[12]. Der Partikelfilter startet dabei als Punktwolke, und verbessert sich ueber mehrere Interationen mithilfe neuer Messungen, heisst die Anzahl der Punkte geht zurueck, bis er schliesslich auf einen Punkt konvergiert. Die Position des Roboters kann durch gleiche oder aehnliche Kartenelemente uneindeutig sein, was sich durch Bewegung des Roboters und damit neugewonnenen Daten auf einen eindeutigen Punkt zurueckfuehren laesst.[4]. In *ROS* published AMCL die daraus gewonnene Position im `amcl_pose`- Topic, und man kann den Roboter dadurch in RViz auf der Karte verfolgen [14].

5 Die Funktionsweise der Pfadverfolgung und ihre Güte

Zuerst betrachten wir das Modell aus [7].

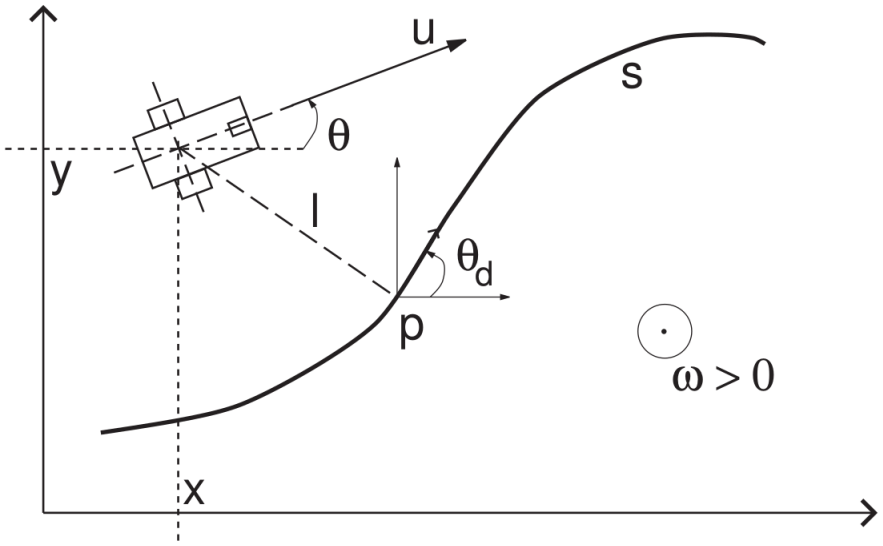

Abbildung 3: Aus [7]: Das Model zur Pfadverfolgung. Man erkennt ein stilisiertes Fahrzeug und einen Pfad. Ausserdem sind die auftretenden Groessen eingezeichnet

Gegeben ist ein nichtlinearer Pfad. Wir unterteilen diesen zuerst in lineare Teilstuecke. Unser Roboter hat dabei ein Position bestehend aus (x, y, θ). x ist die X-Koordinate, y die Y-Koordinate, und θ unser Heading. Das Heading setzt sich aus der Orientierung und Bewegungsrichtung des Roboters zur X-Achse zusammen. Der Vektor u bezeichnet den Bewegungsvektor des Fahrzeugs. l ist der Abstand der Fahrzeugmitte zum Pfad als orthogonaler Projektion. θ_d ist der Winkel des linearisierten Pfades zur X-Achse. Die Winkeldifferenz θ - θ_d bezeichnet [7] als $\tilde{\theta}$. Die Winkelgeschwindigkeit des Fahrzeugs ist ω.
ω berechnet sich dabei aus dem allgemeinen Reglergesetz als:
$\omega = \frac{u\kappa(s)cos(\tilde{\theta})}{1-l\kappa(s)} - hul\frac{sin(\tilde{\theta})}{\tilde{\theta}} - y\tilde{\theta} : h, y > 0$
Die Kruemmung ist bei einem linearen Pfad null.
$\kappa(s) = 0 \forall s$
Daher vereinfacht sich das Reglergesetz zu:
$\omega = -hul\frac{sin(\tilde{\theta})}{\tilde{\theta}} - y\tilde{\theta} : h, y > 0$

6 Experimenteller Vergleich der Qualität der Pfadverfolgung bei Verwendung von AMCL- bzw. Odometrie-Lokalisierung

Nun soll betrachtet werden wie genau die Lokalisierungsmethoden einen vorgegebenen Pfad verfolgen. Desweiteren soll ein Vergleich zwischen beiden Methoden angestellt werden. Demnach ist die Lokalisierungsmethode besser, welche in der Verwendung eine geringere Abweichung vom vorgegebenen Pfad aufweist. Wir gaben dem Controller hier einen Pfad vor, welcher mit Hilfe der 3D-Design-Applikation *Blender* so erstellt wurde, dass er sowohl gerade Linien als auch weiche und harte Kurven enthält.

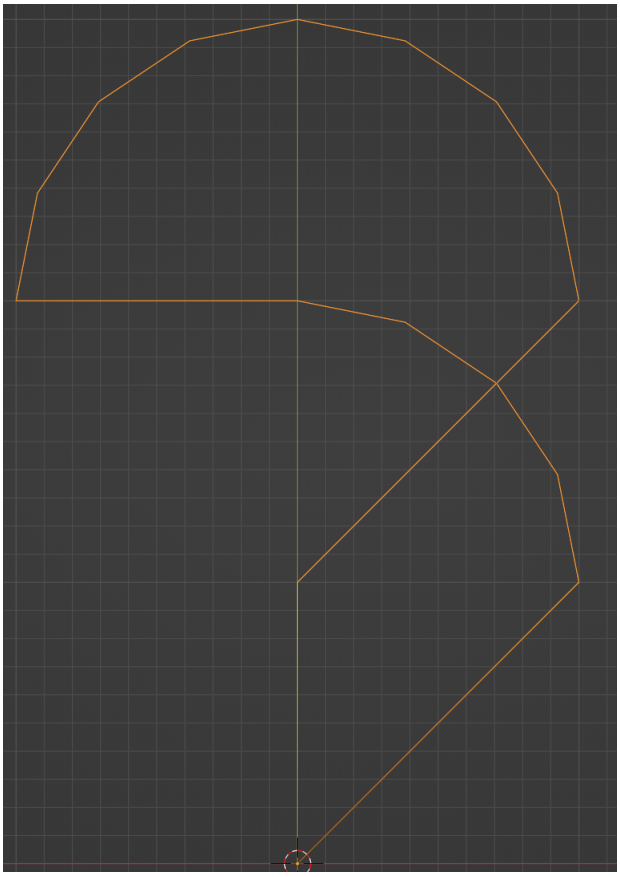

Abbildung 4: Der Pfad, wie er in Blender erstellt wurde. Man erkennt eine Linienstruktur bestehend aus gerade und gekurvten Teilstrecken, die einen geschlossenen Pfad bilden

Start und Ende sind hier der Cursor unten im Bild, Anfangsrichtung ist gerade nach oben. Die verwendete Pfaddatei ist in `catkin_ws` als `Pfad.dat` zu finden. Dieser Pfad wurde auf dem Boden des Testgeländes mit Wollfäden markiert. Der Controller hat die Koordinaten des erstellten Pfades als relativ interpretiert. Dadurch konnten wir den Roboter am Startpunkt des Pfades platzieren, und in die Geradeaus-Richtung des Pfades orientieren. Dies wurde mithilfe durch ein Maßband, das durch die gerade Mittellinie des Pfades bis zum Roboter führt, und diesen dann symmetrisch teilt, möglichst exakt getan. Dazu wurde im Ursprung des Roboters in der Mitte der Vorderradachse ein Schraubenzieher befestigt, der gerade nach unten zeigt um die Position auf dem Boden leichter einsehen zu können. Es ergibt sich dennoch ein systematischer Fehler, da die Startausrichtung und -Platzierung des Roboters nicht perfekt mit der des Pfades am Boden übereinstimmt. Der geschätzte Maximalfehler für die Position des Roboters beträgt 2cm, der der Ausrichtung 2°.
Dadurch ergibt sich im maximalen Abstand vom Startpunkt ein maximaler Fehler von:
$0,02m + tan(2^\circ) * 3m = 0,12m$
An allen anderen Orten ist der systematische Fehler durch den geringeren Abstand vom Startpunkt kleiner. Nachdem der Roboter platziert wurde, wurde ein Node gestartet das den benannten Pfad abfährt, und dabei alle 2 Sekunden anhält, sodass seine Position auf dem Boden markiert werden kann. Dies wird für beide Lokalisierungsmethoden durchgeführt. Danach wird der senkrechte Abstand jedes Punktes zum markierten Pfad gemessen und in

einer Tabelle eingetragen. Die Messergebnisse sind in Anhang[1] einzusehen.

7 Beschreibung der Ergebnisse

Der auf dem Boden markierte Pfad und die markierten Punkte, an denen der Roboter anhielt. Diese wurden jeweils auf einem Bild in Blau bei der Pfadverfolgung unter Verwendung von AMCL, und in Rot bei der Verwendung von Odometrie, zusätzlich zur besseren Sicht nachbearbeitet, eingekreist und mit der jeweiligen Nummer des Anhaltens gekennzeichnet, startend bei 0 für die Initialposition. Man beobachtet das der Pfad anfangs, bis etwa Punkt 7, mit beiden Lokalisierungsmethoden etwa gleich gut verfolgt wird. Der Durchschnitt der Abweichungen vom Pfad von Haltepunkt 1 bis zu Haltepunkt 7 liegt bei Odometrie bei 8cm, bei AMCL 4,2cm. Die folgenden Punkte zeigen bei beiden Lokalisierungsmethoden eine deutlich größere Abweichung auf: Punkte 8-13 bei der Verwendung von Odometrie haben eine durchschnittliche Abweichung von 51,2cm, bei der Verwendung von AMCL ist bei den Punkten 8-14 eine durchschnittliche Abweichung von 22,9cm aufgetreten. Hierbei ist zu bemerken dass durch das beiden Ansätze in der Realität stark verschiedene Pfade verfolgt wurden (siehe Abbildung) und es dazu kam dass AMCL einen längeren Weg zurückgelegt hat, und somit einen Haltepunkt mehr in der Messtabelle aufweist als Odometrie. Anhand der Markierungen auf dem Boden erkennt man auch dass der Odometrie-Ansatz sich nach Haltepunkt 7 durchgehend auf einer Seite des zu verfolgenden Pfades befindet, und immer weiter in diese Richtung von diesem abdriftet, während der AMCL Ansatz die Seite wechselt, in welche der Positionsfehler vorliegt.

8 Diskussion und Auswertung der Fahrtwege unter AMCL und Odometrie

Die durchschnittliche Abweichung aller Fehlerpunkte von AMCL liegt bei 12,6cm, bei der Odometrie sind es 25,9cm. Zu Beginn der Verfolgung des Pfades lagen beide Methoden etwa gleich nahe am Zielpfad, jedoch fing die Odometrie-Lokalisation ab der ca. Hälfte des Pfades an eine immer größer werdende Abweichung vom Zielpfad aufzubauen bis zu einem maximal gemessenem Fehler von 1,13m, während bei AMCL keine größere Pfadabweichung als 36cm gemessen wurde. Das bereits beschriebene Abdriften der Haltepunkte des Odometrie Ansatz lässt darauf schließen das die sich akkumulierenden Fehler in den Radumdrehungen, welche das alleinige Lokalisierungsmerkmal der Odometrie ist, einen Drift in der Positionierung verursachen, was bewirkt das Odometrie für eine Lokalisierung auf Pfaden, die nicht kurz sind, keine der Realität akkuraten Ergebnisse liefern kann. Da AMCL nicht allein auf die Radumdrehungen angewiesen ist, kann es die Fehler dieser Informationsquelle ausgleichen und so auch auf längeren Pfaden bis zu einem gewissen Grad die Position genau bestimmen.

9 Zusammenfassung und Ausblick

Literatur

[1] Dr. Mohamed Oubbati. Einführung in die Robotik - Vorlesung 3 Universität Ulm. `https://www.uni-ulm.de/fileadmin/website_uni_ulm/iui.inst.130/Arbeitsgruppen/Robotics/Robotik/Vorlesung_03.pdf`. abgerufen: 27.11.2022.

[1] Siehe Tabelle 1

[2] TU Dresden. ODOMETRIE. https://robolab.inf.tu-dresden.de/spring/task/odometry/. abgerufen: 26.11.2022.

[3] Giorgio Grisetti et al. Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters. *IEEE Transactions on Robotics*, 23, 2007.

[4] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte carlo localization: Efficient position estimation for mobile robots. *AAAI/IAAI*, 1999(343-349):2–2, 1999.

[5] Giorgio Grisetti et al. openSLAM-GMapping. https://openslam-org.github.io/gmapping.html. abgerufen: 28.11.2022.

[6] Hartmut Surmann et al. The Volksbot. https://www.volksbot.de/surmann/papers/Simpar2008-volksbot.pdf, 2008. abgerufen: 28.11.2022.

[7] Giovanni Indiveri and Maria Letizia Corradini. Switching linear path following for bounded curvature car-like vehicles. *IFAC Proceedings Volumes*, 37(8):185–190, 2004.

[8] Fraunhofer Institut IAIS. Volksbot. https://www.volksbot.de/rt3-de.php. abgerufen: 24.11.2022.

[9] MathWorks Inc. SLAM (Simultaneous Localization and Mapping). https://de.mathworks.com/discovery/slam.html. abgerufen: 28.11.2022.

[10] O. Bittel. Lokalisierung HTWG Konstanz. http://www-home.htwg-konstanz.de/~bittel/ain_robo/Vorlesung/05_Lokalisierung.pdf. abgerufen: 27.11.2022.

[11] Edwin Olson. A primer on odometry and motor control. *Electronic Group Discuss*, 12, 2004.

[12] O'Reilly. Understanding AMCL. https://learning.oreilly.com/library/view/ros-programming-building/9781788627436/78d422a7-dcbd-4cf9-a0c4-eec4d636d335.xhtml. abgerufen: 30.11.2022.

[13] Morgan Quigley, Brian Gerkey, and William D Smart. *Programming Robots with ROS: a practical introduction to the Robot Operating System.* Ö'Reilly Media, Inc.", 2015.

[14] Ros.org. AMCL. http://wiki.ros.org/amcl. abgerufen: 30.11.2022.

[15] Ros.org. gmapping. http://wiki.ros.org/gmapping. abgerufen: 30.11.2022.

[16] SICK Sensor Intelligence. 2D-LiDAR-Sensoren LMS1xx / Indoor. https://www.sick.com/de/de/mess-und-detektionsloesungen/2d-lidar-sensoren/lms1xx/lms100-10000/p/p109841. abgerufen: 28.11.2022.

[17] Tesla Deutschland. Model S Plaid. http://www.tesla.com/de_de/models. abgerufen: 26.11.2022.