Implementation eines nicht linearen Reglers nach Indiveri und de Wit

Patrick Ludwig
Phillip König
Robotik und Telematik
Universität Würzburg
Am Hubland, D-97074 Würzburg
patrick.ludwig@stud-mail.uni-wuerzburg.de
philipp.koenig@stud-mail.uni-wuerzburg.de

Inhaltsverzeichnis

Diese Arbeit ist Bestandteil des Praktikums zur Mess- und Regelungstechnik

1 Einleitung

Im dieser Arbeit ist die Implementation eines nicht linearen Reglers zur Kontrolle von Unmanned Vehicle Systems(UVS) erprobt. Der Regler ist in der Lage das System entlang eines vorgegebenen Pfades zu führen. Dazu bestimmt der Regler entweder seine aktuelle Position mithilfe von Odometry, oder durch Adaptive Monte Carlo Localization (AMCL) also dem Abgleich seiner Umgebung mit einer Karte. Letzteres System verwendet einen auf einem auf dem Roboter montierten 'SICK' Laserscanner. Dieses Paper ist ergänzt durch einen im Rahmen dessen erstellten Film.

2 Verwendete Soft- und Hardware

2.1 Kenndaten des Roboters

Der verwendete Roboter ist ein Roboterbaukastensystem des Typs Volksbot RT3, entwickelt von dem Fraunhofer Institut IAIS für Rapid Prototyping speziell für Anwendungen in der Ausbildung. Den 580 x 520 x 315 mm (L x B x H) großen 17kg schweren und bis zu 2,2 m/s schnellen Roboter gibt es in zwei Ausführungen. Zwei 260 x 85 mm aktive Räder, und ein 200mm passives Rad in der ersten Variation oder zwei aktive und zwei passive Räder in der zweiten Variation. Der Roboter kann mit zusätzlichen Sensoren ausgestattet werden, und benötigt eine Recheneinheit (Laptop) als Steuerungscenter. Dabei unterstützt er eine Zuladung von maximal 25kg. Der hier verwendete Roboter ist mit einem Laser-Scanner LMS1 des Herstellers Sick ausgestattet. Der Scanner hat eine maximale Reichweite von 50m, bei 10 Prozent Remission verkürzt sich die Reichweite auf 30m. Er erreicht eine maximale Winkelauflösung von 0,25-0,5° bei einem Öffnungswinkel von 270° und einer Scanfrequenz von 25 beziehungsweise 50Hz.

Abbildung 1: Der Roboter 'IRMA'

2.2 ROS Middleware

Der Betrieb des Roboters erfolgt durch ROS, ein freies Open Source Software Framework. Diese modular aufgebaute Middleware stellt eine Verbindung zwischen einem Unix System, hier wird ein Laptop mit der Distribution Ubuntu verwendet, und dem Roboter her. Diese Strukturierung ermöglicht es, dass das Lokalisationssysteme ausgetauscht werden können, oder die Middleware um zusätzliche Funktionen und Fähigkeiten erweitert werden kann, ohne die restliche Kontrollsoftware zu verändern. Eine ausgiebige Dokumentation von ROS ist auf wiki.ros.org/ROS/Tutorials zu finden.

Die Kommunikation mit dem Roboter erfolgt über ein USB-to-Serial-Adapter, so wie einem Network-kabel für den Laserscanner über ein lokales Netzwerk zwischen kontrollierendem Computer und kontrolliertem Roboter.

2.3 Rosbag Aufzeichnungen via Joystickcontrolle

Rosbag ist eine ROS-Node die es erlaubt, sämtliche Daten des Roboters, welche über die ROS Middleware publiziert werden, inklusive des Laserscanners, aufzuzeichnen und abzuspeichern. Der Roboter besitzt die Möglichkeit einer Joystick Steuerung, die benötigt wird, um den Roboter ohne Regler zu manövrieren. Sie ermöglicht es via eines Logitech-Wireless-Gamepad die momentanen Drehgeschwindigkeiten der Vorderräder des Roboters separat einzustellen und damit den Roboter manuell zu steuern.
Das Abspielen der aufgezeichneten Daten nach Repositionierung des Roboters an der Ausgangsposition Veranschaulicht die Aufzeichnungsqualität und zeigt fundamentale Probleme, welche entstehen, wenn die vom Roboter abgefahrene Strecke nicht durch einen Regler kontrolliert ist. Zu diesen Problemen gehört das dauerhafte akkumulieren von Kleinstfehlern zwischen Input und Output, sowie der Unmöglichkeit auf Veränderungen in der Umgebung zu reagieren. Der Roboter hält den ursprünglich abgefahrenen Pfad nur ein solange es sich um einen Pfad geringer als 5m handelt welcher darüber hinaus nur minimale Drehungen im Bereich von weniger als 20 Grad aufweist.

3 Kartierung mit GMapping

Für die verbesserte Pfadfindung benötigt es eine Lokalisierung des Roboters in seiner Umgebung, wozu ein Karte (map) der Umgebung, in welcher der Roboter aktiv ist, nötig ist. Das ROS GMapping Package stellt Laser basiertes SLAM (Simultaneous Localization and Mapping) zur Verfügung. Dabei ist die erstellte Karte ein Occupancy Grid. Laserscanner Daten werden verwendet um die Wahrscheinlichkeit der Existenz eines Hindernisses wie einer Säule, Wand oder Person in der Umgebung des Roboters zu errechnen und zu speichern. Die fertige map ist keine Zeitpunktaufnahme, ein Ort kann während des Erstellens der Laserscanner Daten mehrmals aufgesucht werden und leicht unterschiedliche Hindernisse wiedergeben, alle Daten werden miteinander so verrechnet das statische Objekte in der map erhalten bleiben während temporäre oder sich bewegende herausfallen. Dies ist Problematisch in einer Umgebung mit limitierten statischen aber sehr vielen temporären oder beweglichen Objekten. Die Wahrscheinlichkeitsvoraussagen für eine solche nicht statische Umgebung im Occupancy Grid sind dann ungenau.
Auf der Karte ist ersichtlich, dass Glasscheiben für den Roboter problematisch sind, da die Laserstrahlen nicht genügend reflektiert werden, was in Bereichen mit Glasfronten zu Ungenauigkeiten in der Karte führt.
Das ROS-Package RVIZ stellt eine Visualisierungsfunktion bereit, mit der die Erstellung der Karte verfolgt werden kann
Abbildung 5 zeigt die im Rahmen dieser Arbeit erstellte Karte

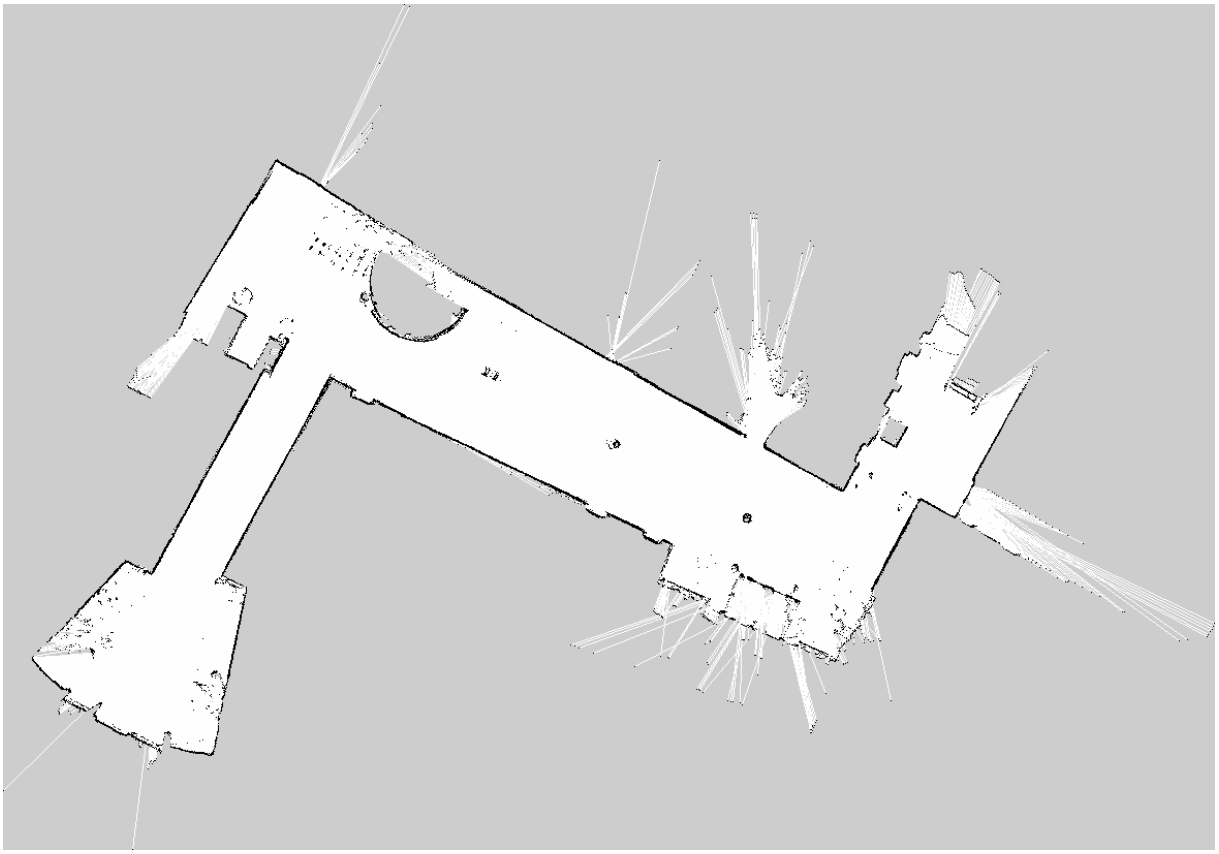

Abbildung 2: Karte des Informatikgebäude-Untergeschosses

3.1 Abspeichern der Fahrtpfade

ROS modularer Aufbau erlaubt das hinzufügen einer Node, die Positionsdaten in eine Datei extrahiert, welche als Liste von Positionspunkten gespeichert werden. Diese Positionspunkte sind als Graphen mittels Funktionsplot-Tool darstellbar. Die folgenden Abbildungen wurden durch Gnuplot erstellt und dienen der Veranschaulichung der Pfadqualität.
Die abgefahrenen Pfade sind gewählt um den Unterschied zwischen verschiedenen Fahrtmanövern darzustellen. Der erste gezeigte Pfad ist dabei in Form einer Acht, er dient zur Überprüfung von Kurvenmanövern, ein qualitativ hochwertiger Regler ist dabei sowohl fähig den erwünschten Kurvenradius einzuhalten, als auch fähig den Wechsel zwischen Kurven und Geraden zu meistern ohne dabei derartig vom Pfad abzuweichen, dass die Figur unvollständig ist.
Ein weiterer Pfad erprobt 180-Grad Kurvenmanöver und das fahren über längere Strecken der Pfad ist Form eines L. Beide Fahrten enden wieder in der Anfangsposition und Pose. Die tatsächlich resultierenden Graphen der Odometry sind in den folgenden Abbildungen 1 und 2 in lila Farbe zu sehen.
Es sei des weiteren eine grüne acht, also der erwartete Pfad für die erste Fahrt, dem ersten Graphen (Abbildung:3) hinzugefügt um die Abweichungen der Realität von der Vorgabe aufzuzeigen. Unmissverständlich ist die geringe Qualität der Positionsbestimmung der Odometry zu erkennen. Während die erwünschte Figur in Abbildung:3 noch zu erahnen ist, so ist sie doch in Abbildung:4 schon unwiedererkennbar.
Mit Hilfe von AMCL unter Verwendung eines 'SICK' Laserscanners soll nun im folgenden die Positionsbestimmung präziser gelingen.

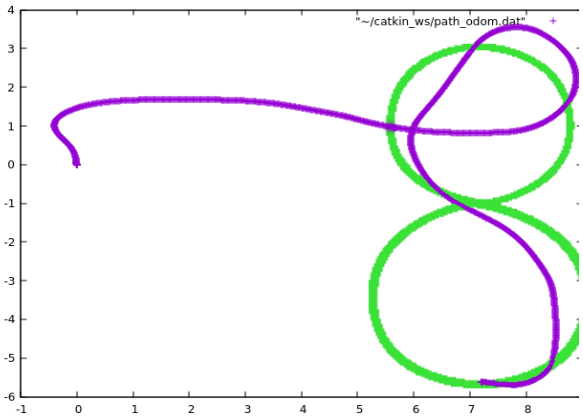

Abbildung 3: Acht-förmiger Pfad

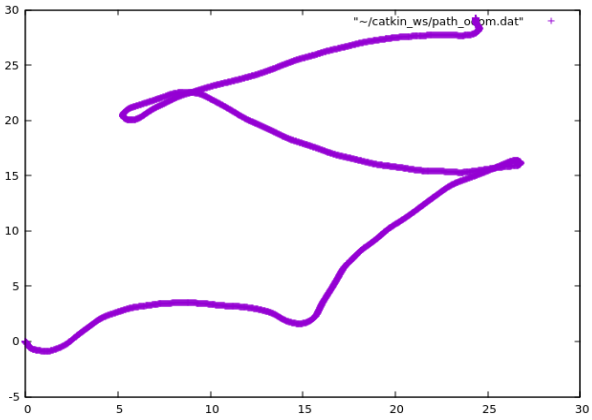

Abbildung 4: L-förmiger Pfad

4 AMCL

4.1 AMCL einrichten

AMCL Packages stehen durch ROS zur Verfügung. Ist das Roboterkontrollprogramm um die entsprechenden Nodes erweitert, steht die aus den Laserscannerdaten berechnete wahrscheinlichste Position über ein ROS-Topic zur Verfügung. Durch einen Neustart an der Startposition für die Pfadverfolgung ist sicher gestellt das alle Koordinatensysteme übereinstimmen und den selben Nullpunkt besitzen. Nur so sind aufgezeichnete Daten des Roboters verlässlich.
Damit der Regler seinen Pfad auf Grundlage der von AMCL errechneten Position verfolgen kann, wird das System während der Fahrt des Roboters getestet. Durch hinzufügen der /particlecloud lassen sich die von AMCL ermittelten Positionen als PoseArray darstellen. Dies macht eine Überprüfung sowohl während der Fahrt als auch auf Grundlage mittels rosbag aufgezeichneter Daten möglich.

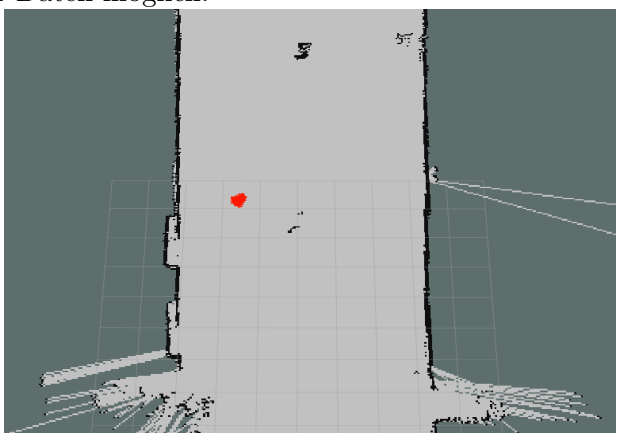

Abbildung 5: Der PoseArray in rot als Positionsangabe innerhalb der Karte

Die AMCL Daten weisen Teilweise unerwünschte Abweichungen zur Karte auf, eine experimental Versuchs-Serie ergibt die notwendige Parameter Korrektur zur Behebung des Problems. Jene Parameter sind in der AMCL config.yaml zu finden. Alle dort befindlichen alpha-Werte sind um 25% zu erhöhen, dies erbringt die gewünschte Verbesserung.
Festzustellen ist, dass AMCL auch bei hohen Geschwindigkeiten die Position zuverlässig bestimmen kann.
Um die durch AMCL berechneten Daten als Graph darstezustellen, wird das System durch

eine weitere Node erweitert, die analog zur Node, die Odometrie aufzeichnet, die von AMCL gesendeten Daten extrahiert.

4.2 AMCL Prinzip und Anwendung

Der Augmented Monte Carlo Localization Algorithmus berechnet zu jeder Zeit eine Wahrscheinliche Position des Roboters. Dazu muss ihm eine Karte der Umgebung bereits vorliegen, jene Karte ist das so genannte Occupancy Grid wobei jedweden Punkt auf der Karte eine Wahrscheinlichkeit zu geordnet ist die definiert ob dieser Punkt belegt ist oder nicht (occupied or not), ein schwarzer Punkt ist dabei Höchstwahrscheinlich ein Hindernis also belegt, hellere Farbtöne hingegen indizieren höhere Wahrscheinlichkeiten dass dort kein Objekt ist.
Nach dem erneuten abspielen eines .bag Files welches eine acht förmige Fahrt aufgezeichnet hatte und eines welches eine l-förmige Fahrt aufgezeichnet hatte, erhält man die folgenden Graphen welche hier in Grün dargestellt sind, zum Vergleich hier auch noch einmal die Odometry Graphen in Lila:

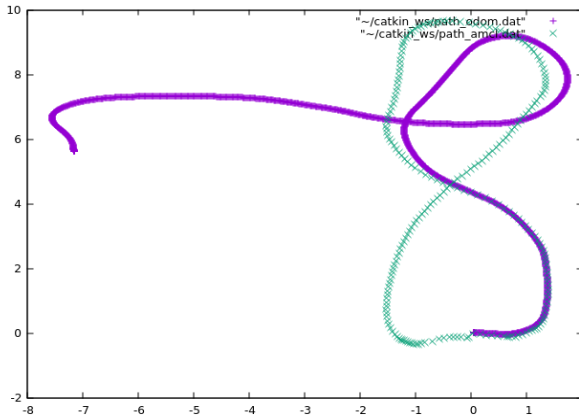

Abbildung 6: 8-förmiger Pfad

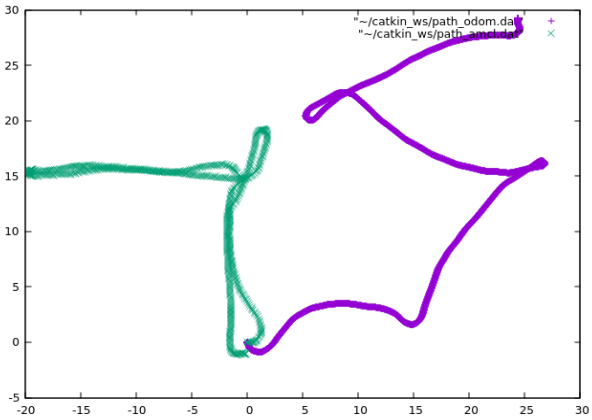

Abbildung 7: L-förmiger Pfad

Die Positionserkennung erreicht dank AMCL eine höhere Genauigkeit. Besonders wenn Wände zueinander Parallel und aus nicht Licht durch-dringlichen Materialien bestehen so ist die Positionsbestimmung zumeist akkurat bis auf ca. 20cm. Besonders fällt auf das es AMCL gelingt die Orientierung korrekt zu bestimmen die Odometry hingegen ist dazu nicht in der Lage, sie kann lediglich die gefahrene Distanz in etwa Wahrheitsgetreu wiedergeben. Das AMCL system ist dazu jedoch auch in der Lage und erweist damit klare Vorteile.

5 Implementierung der Pfadverfolgung

5.1 Grundlagen des nicht linearen Reglers

Die Erstellung des Reglers betrachtet die Prinzipien ausgelegt von Giovanni Indiveri und Maria Letizia Corradini in 'SWITCHING LINEAR PATH FOLLOWING FOR BOUNDED CURVATURE CAR-LIKE VEHICLES' [**?**]. Welche selbst der Grundlagenforschung von Canudas de Wit et al. 'A NONLINEAR PATH FOLLOWING CONTROLLER FOR THE KINEMATIC MODEL OF A CAR-LIKE ROBOT WITH BOUNDED CURVATURE IS

DESIGNED' [?] folgen. Darüber hinaus betrachtet sie Lingeman et al. 'ABOUT THE CONTROL OF HIGH SPEED INDOOR ROBOTS' [?] und 'KINEMATIC TIME-INVARIANT CONTROL OF A 2D NONHOLONOMIC VEHICLE' von Giovanni Indivieri [?]. Sei zunächst das unicycle Model gegeben durch:

$$\begin{aligned} \dot{x} &= ucos\theta \\ \dot{y} &= usin\theta \\ \dot{\theta} &= \omega \end{aligned}$$

und eine Darstellung eines vom Pfade abgekommenen Roboters Abbildung(8):

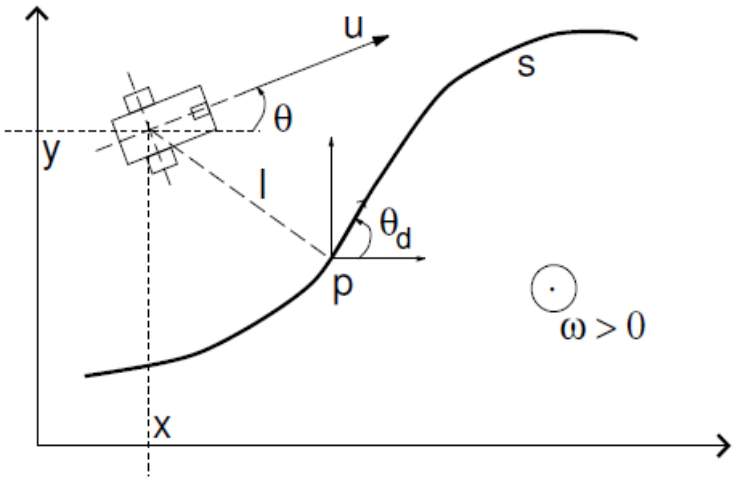

Abbildung 8

Das unicycle Modell, besteht aus den kinematischen Gleichungen welche die Änderungen der Position und der Orientierung des Roboters beschreiben. Dabei ist u die lineare Geschwindigkeit des Roboters und Theta sein momentaner Drehwinkel. Die Änderung des Drehwinkels θ selbst ist definiert als die Drehgeschwindigkeit ω. Das in Abbildung(8) gezeigte ℓ beschreibt die Distanz zwischen dem Mittelpunkt des Roboters (also seinen momentanen X und Y Koordinaten) und seiner orthogonalen Projektion p auf den zu folgenden Pfad. Der Winkel θ beschreibt die Ausrichtung des Roboters zur X-Achse. Er ist gegeben durch $\tilde{\theta} = \theta(s) - \theta_d(s)$. Der Winkel θ_d beschreibt die Ausrichtung des Pfades zur X-Achse. Um das nicht lineare Problem zu Linearisieren unterteile man den nicht linearen Pfad in lineare Teilstücke.
Berechnet man nun die gewünschte Winkelgeschwindigkeit ω nach dem steering-control-law von de Wit, so ist asymptotische Konvergenz von ℓ und θ zu 0 garantiert. Die berechnete Winkelgeschwindigkeit dreht den Roboter stets in Richtung des Pfades, präziser dreht sie ihn so dass der nächste Pfadabschnitt auf dem kürzesten Wege erreicht wird. Fortan wird der Algorithmus ad infinitum ausgeführt solange bis das Ziel erreicht ist. Der Algorithmus arbeitet also Zeit diskret.
Das angesprochene steering-control-law ist zunächst:

$$\omega = \frac{u\,\kappa(s)\,\cos\tilde{\theta}}{1 - l\,\kappa(s)} - h\,u\,l\,\frac{\sin\tilde{\theta}}{\tilde{\theta}} - \gamma\,\tilde{\theta} \;:\; h, \gamma > 0$$

Da die Krümmung κ für lineare Pfade gleich null ist, entfällt der erste Term der Gleichung jedoch, das control-law entspricht nun:

$$\omega = -h\,u\,y\,\frac{\sin\theta}{\theta} - \gamma\,\theta \;:\; h, \gamma > 0$$

Die Radgeschwindigkeiten $vLeft$ und $vRight$ berechnen sich letztendlich aus der Winkelgeschwindigkeit ω:

$$vLeft = u_0 - (\tfrac{1}{2}\omega b)$$

$$vRight = u_0 + (\tfrac{1}{2}\omega b)$$

5.2 Implementation des Reglers

Der Großteil des Reglers steht bereits zur Verfügung, er muss lediglich ordnungsgemäß eingebunden werden. Dies geschieht durch eine separate Control Node. Der Regler ist in der Lage sowohl über Odometry als auch über AMCL gesteuert zu werden, die Control Node gibt an welches Positionsbestimmungsverfahren zu verwenden ist. Es ist möglich Odometry Daten direkt eingelesen werden, AMCL Daten werden hingegen zunächst über ein tf-listener aus dem map frame genommen und in den base_link frame transformiert. Die Control Node übergibt dem Regler die aktuelle Pose als X, Y Koordinaten und dem Drehwinkel θ daraus werden die benötigten Radgeschwindigkeiten berechnet um die nächste Position zu erreichen. Die Radgeschwindigkeiten werden an die Control zurückgegeben und von dort via das Vel Topic an den Roboter weitergereicht. Daten müssen möglichst häufig kommen, AMCL kann jedoch nur limitiert oft Daten bereitstellen, über tf-transform kann bereits eine höhere Anzahl an Datenpunkten erreicht werden als durch die Standard AMCL node. Aber auch hier besteht eine Limitation, welche letztendlich die maximale Geschwindigkeit mit der der Roboter dem Pfad folgen kann einschränkt.

6 Pfadverfolgung

Der Test des Reglers erfolgt mittels eines Simulators, der aus den vom Regler zur Verfügung gestellten Radgeschwindigkeiten die neue Position berechnet und auf das odom-Topic publiziert. Die Simulation ist auf Odometrie beschränkt da eine Simulation der AMCL-Daten zu aufwändig ist. Nach erfolgreicher Simulation folgt der Test am Roboter. Der Roboter ist sowohl in der Lage dem Odometry als auch dem AMCL Pfad generell zu folgen. Jedoch bereitet bereits nach wenigen Decimetern die Datenqualität der Odometry Probleme. Der Roboter weicht zunehmend vom Pfad ab und Fehler akkumulieren sich. Der Regler ist nicht in der Lage diese Probleme festzustellen und zu korrigieren. Auch äußeren Einflüssen kann der Regler hier nicht entgegenwirken dies führt zu weiteren Diskrepanzen zwischen erwünschtem Pfad und dem tatsächlichen.
Nutzt des Reglers hingegen AMCL, kann der Roboter seinen Pfad zuverlässiger halten und Abweichungen limitieren sich auf 10 bis 30 cm. Dennoch weist auch AMCL Probleme auf, so oszilliert der Roboter in gewissen Maße um den Pfad (hauptsächlich bei Geraden zu sehen), dies liegt unter anderem an der geringen Frequenz der AMCL-Positionsbestimmung.

7 Fazit

Im tatsächlichen realen Versuch zeigt AMCL eine klare Verbesserung in Positionsbestimmung und Pfadverfolgung gegenüber Odometry, welches lediglich in einem virtuellen Raum exzellente Ergebnisse liefert. Odometry erwartet dass die Realität keine negativen Einflüsse auf das Fahrtverhalten des Roboters hat, dies ist natürlich nicht gegeben, unter anderem z.B. wegen Reibung. AMCL hingegen kann auf äußere Einflüsse reagieren und diesen Entgegenwirken, dies erlaubt den Roboter sicherer auf dem Pfad zu halten, lediglich technische Limitation bezüglich der Häufigkeit der Datenerfassung sorgen für Probleme und leichten Pfadabweichungen.

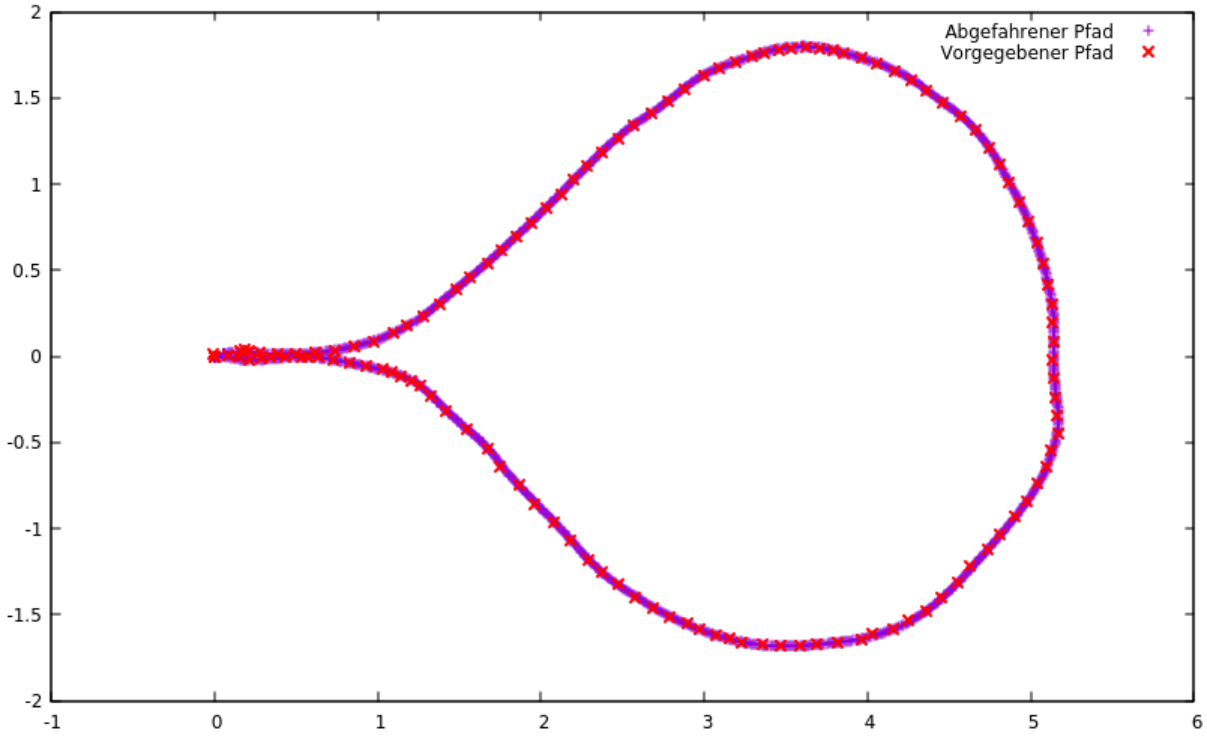

Abbildung 9: Veranschaulichung von geringen Pfadabweichungen

Literatur

[1] C Canudas De Wit, H Khennouf, C Samson, and O Jl Sordalen. Nonlinear control design for mobile robots. *Recent trends in mobile robots*, 11:121–156, 1993.

[2] Giovanni Indiveri. Kinematic time-invariant control of a 2D nonholonomic vehicle. In *Proceedings of the 38th IEEE Conference on Decision and Control*, pages 2112–2117. IEEE, December 1999.

[3] Giovanni Indiveri and Maria Letizia Corradini. Switching linear path following for bounded curvature car-like vehicles. In *Proceedings of the IFAC Symposium on Intelligent Autonomous Vehicles*. IFAC, September 2004.

[4] Andreas Nüchter, Kai Lingemann, Joachim Hertzberg, and Hartmut Surmann. About the Control of High Speed Mobile Indoor Robots. In *Proceedings of the Second European Conference on Mobile Robotics*, pages 218–223. ECMR, September 2005.