Abschlussbericht
Projektseminar Echtzeitsysteme

Team Das AuTU
Projektseminar eingereicht von
Hendrik Göttmann, Dominik Lorych, Florian Netzer, Raphael Wenzel, Lars Wolf
am 17. April 2017

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachgebiet Echtzeitsysteme

Elektrotechnik und
Informationstechnik (FB18)

Zweitmitglied Informatik (FB20)

Prof. Dr. rer. nat. A. Schürr
Merckstraße 25
64283 Darmstadt

www.es.tu-darmstadt.de

Gutachter: Géza Kulcsár
Betreuer: Géza Kulcsár

Erklärung zum Projektseminar

Hiermit versichere ich, das vorliegende Projektseminar selbstständig und ohne Hilfe Dritter angefertigt zu haben. Gedanken und Zitate, die ich aus fremden Quellen direkt oder indirekt übernommen habe, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen und wurde bisher nicht veröffentlicht.

Ich erkläre mich damit einverstanden, dass die Arbeit auch durch das Fachgebiet Echtzeitsysteme der Öffentlichkeit zugänglich gemacht werden kann.

Darmstadt, den 17. April 2017

_____

(Hendrik Göttmann, Dominik Lorych, Florian Netzer, Raphael Wenzel, Lars Wolf)

Inhaltsverzeichnis

1 Vorwort

Dies ist der Abschlussbericht der Gruppe „Das AuTU“ im Projektseminar Echtzeitsysteme im WS 16/17 der TU Darmstadt.

1.1 Aufgabenstellung

Als grobe Vorlage wurden 5 Aufgaben gestellt, von denen 3 erfüllt werden sollten. Wir haben uns für folgende Aufgaben entschieden:

- Rundkurs ohne Hindernisse: Abfahren eines Ganges in einem Gebäude, der aus vier Geraden und vier 90° Ecken besteht. Dies war Pflicht für alle Gruppen.
- Rundkurs mit Hindernissen: Abfahren des selben Ganges, nur dass sich diesmal Hindernisse in dem Gang befinden, die umfahren werden müssen.
- Einparken: Einparken in eine Parklücke, die sich vor dem Auto befindet.

2 Rundkurs ohne Hindernisse

Bei dieser Aufgabe muss das Auto einen bekannten Rundkurs im Flur des Instituts in möglichst kurzer Zeit bewältigen.

2.1 Grundidee

Der Rundkurs besteht aus Geraden, an deren Ende sich jeweils eine Kurve befindet. Demnach existieren in diesem System zwei Hauptzustände: „Fahrzeug auf der Geraden“ oder „Fahrzeug in Kurve“. Die Modellierung des Ganzen als Zustandsautomat erscheint also sinnvoll.

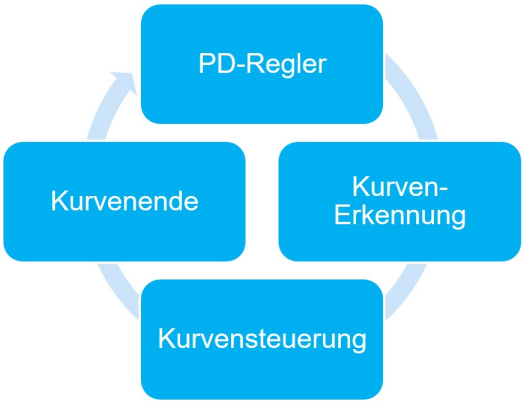

Abbildung 2.1: Zustandsdiagramm

2.2 Regelung der Fahrt auf den Geraden

Jede Gerade hat den Vorteil, dass sie durch eine Wand begrenzt ist, an welcher sich das Fahrzeug orientieren kann. Der aktuelle Abstand zur Wand kann nämlich näherungsweise mit dem Ultraschall-Sensor, der auf der linken Seite des Autos montiert ist, bestimmt werden. Mit Hilfe eines PD-Reglers wird sichergestellt, dass sich das Auto parallel zur Wand fortbewegt.
Die dazu benötigten Parameter wurden bei Testfahrten empirisch ermittelt. Außerdem werden die Messwerte des Ultraschall-Sensors geglättet, um nervöse Lenkbewegungen zu vermeiden, indem aus den letzten 10 Messwerten ein Mittelwert gebildet wird.

2.3 Steuerung der Fahrt in den Kurven

Zu Beginn wurde auch hier versucht, das Problem mittels einer Regelung zu lösen. Mit der Zeit sollte sich jedoch herausstellen, dass eine simple Steuerung bessere Ergebnisse erzielt. In der Kurve wird also einfach ein fester Lenkwinkel eingestellt, der während der gesamten Kurvenfahrt konstant bleibt.

2.4 Kurvenerkennung

Die Kurvenerkennung entspricht im Zustandsautomaten dem Übergang von der Geraden hin zur Kurve. Wie erkennt man also eine Kurve? Verlässt man sich ausschließlich auf den Ultraschall-Sensor, kann man eine Kurve erst detektieren, wenn links des Fahrzeugs keine Wand mehr vorhanden ist. Dann befindet sich das Auto allerdings schon mitten im Kurvenbereich und hat somit den optimalen Punkt zum Einlenken bereits weit hinter sich gelassen.

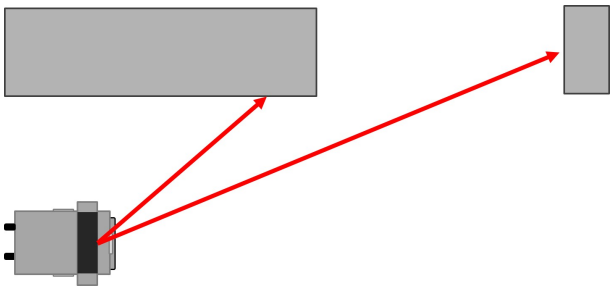

Abbildung 2.2: Kurvenerkennung

An dieser Stelle kommt die Kinect-Kamera ins Spiel. Durch sie ist es möglich, bereits deutlich vor einer anstehenden Kurve selbige zu erkennen. Somit wird es überhaupt erst möglich eine optimale Kurvenlinie zu fahren.
Auf diesem Rundkurs zeichnet sich eine Kurve dadurch aus, dass am Kurvenbeginn die Wand abrupt endet. Die gemessenen Distanzen im von der Kinect generierten Laserscan steigen an dieser Stelle also sprunghaft an, was detektiert werden kann.

2.5 Kurvenende

Der vorgegebene Rundkurs enthält ausschließlich 90°-Kurven. Am Kurvenausgang ist demnach der Gierwinkel des Fahrzeugs etwa 90° größer als am Kurveneingang. Steigt der Gierwinkel also über einen gewissen Schwellwert, übernimmt der PD-Regler wieder die Kontrolle.

2.6 Probleme

2.6.1 Kurvenerkennung am Kurvenausgang

Am Kurvenausgang, wenn gerade wieder auf den PD-Regler umgeschaltet wurde, befindet sich die linke Wand meist nicht im Sichtfeld der Kinect. Dadurch kommt die weiter oben beschriebene Kurvenerkennung aus dem Tritt. Es kommt zu einem False-Positive: Eine Kurve wird in der Punktmenge des Laserscans erkannt, die gar nicht existiert. In Folge dessen lenkt das Auto direkt in die Wand.
Durch Einführung einer Totzeit kann dieses Problem gelöst werden. Nachdem eine Kurve beendet wurde, muss erst eine gewisse Zeitspanne vergehen, bis die Kurvenerkennung reaktiviert wird. Dann ist die Wand schon wieder im Sichtfeld und die Kurvenerkennung arbeitet somit wieder korrekt.

2.6.2 Glaserkennung

Auf dem Rundkurs befindet sich ein Besprechungsraum mit mehreren Glasscheiben. Durch diese schaut die Kinect einfach hindurch. In dem daraus resultierenden Laserscan wird auch hier wieder eine Kurve erkannt.
In einer Eigenschaft unterscheiden sich allerdings Glasscheiben von echten Kurven: Die breiteste Glasscheibe misst etwa 80cm in der Breite, der schmalste Flur 160cm. Für den Fall, dass eine Ecke erkannt wird, wird in einem gewissen Bereich nach der Ecke überprüft, ob sich dort Objekte befinden. Durch geeignete Wahl dieses Bereichs kann zwischen „echten“ und „falschen“ Kurven unterschieden werden.

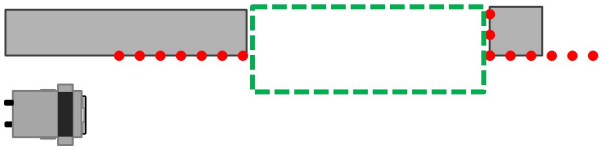

Abbildung 2.3: Glaserkennung an Kurven

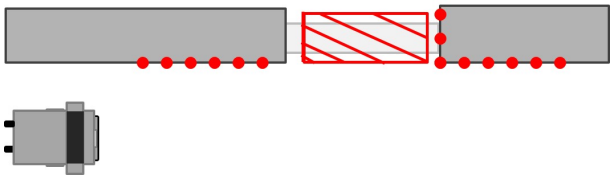

Abbildung 2.4: Glaserkennung an Glasscheiben

Darüber hinaus ist die Glaserkennung nicht nur für Glasscheiben nützlich. Wegen wahrscheinlich von LED-Lampen verursachten Störungen treten sporadisch Lücken in der Wand auf. Da sich diese Lücken hinsichtlich der Glaserkennung allerdings exakt gleich verhalten, wurde dieses Problem hiermit bereits adressiert.

2.6.3 Begrenztes Sichtfeld der Kinect

Der Winkel des Sichtfeldes der Kinect ist stark begrenzt. In Folge dessen kann es passieren, dass eine Ecke, der das Fahrzeug zu nahe kommt, aus dem sichtbaren Bereich verschwindet.
Veranschaulichen lässt sich diese Problematik am besten an einem Beispiel: Die optimale Stelle zum Einlenken in die Kurve liege 75cm vor der Ecke. Unglücklicherweise verschwindet die Ecke bereits aus dem Sichtfeld der Kamera bei einer Entfernung von 100cm. Die Position der Ecke muss also gespeichert werden solange die Kinect sie noch sieht. Anschließend kann mit Hilfe der Odometrie der korrekte Einlenkzeitpunkt bestimmt werden. Im gegebenen Beispiel müsste das Auto also noch 25cm blind geradeaus fahren bevor es mit dem Einlenken beginnt.

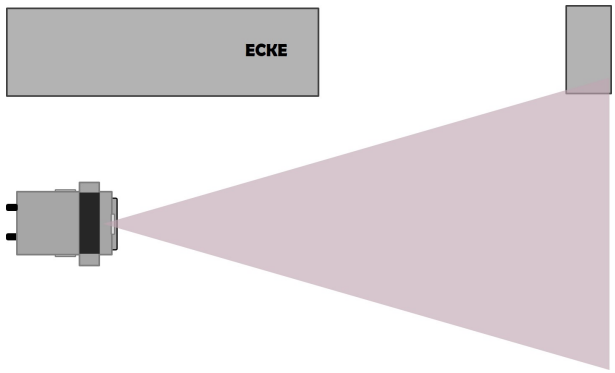

Abbildung 2.5: Sichtfeld der Kinect

2.6.4 Entkopplung von Geraden- und Kurvengeschwindigkeit

Bei höheren Geschwindigkeiten kann es sinnvoll sein, in den Kurven langsamer zu fahren als auf den Geraden. Ein normales Auto würde dazu sicherlich vor den Kurven abbremsen, bei diesem Modell hingegen ist es ausreichend, kurz vor den Kurven den Motor abzuschalten. Aus einigen Experimenten mit dem Auto ging hervor, dass die Verzögerung beim Ausrollen näherungsweise als konstant angenommen werden kann. Eine gute Schätzung für diesen Wert ist $1 m/s^2$.
Auf Basis dieser Überlegungen lässt sich die Länge der Strecke zum Ausrollen berechnen:

a: Verzögerung des Fahrzeugs bei Motorlevel 0 in m/s^2

v_0: Geschwindigkeit auf der Geraden in m/s

v_k: Gewünschte Geschwindigkeit in der Kurve in m/s

Zu jedem betrachteten Zeitpunkt t beträgt die Geschwindigkeit

$$v(t) = \dot{x}(t) = v_0 - at$$

Mit der Anfangsbedingung $x(0) = 0$ erhält man durch Integration die zurückgelegte Strecke

$$x(t) = v_0 t - \frac{1}{2} a t^2$$

Nun wird die benötigte Dauer zum Ausrollen bestimmt, um das Auto auf Kurventempo abzubremsen

$$v(t_a) = v_k \Rightarrow t_a = \frac{v_0 - v_k}{a}$$

Daraus ergibt sich die benötigte Strecke zum Ausrollen

$$x(t_a) = v_0 \frac{v_0 - v_k}{a} - \frac{1}{2} \frac{(v_0 - v_k)^2}{a} = \frac{1}{2a}(v_0^2 - v_k^2)$$

3 Rundkurs mit Hindernissen

Bei dieser Aufgabe muss das Auto einen bekannten Rundkurs im Flur des Instituts in möglichst kurzer Zeit bewältigen. Allerdings befinden sich auf der Strecke nicht verzeichnete Hindernisse, die dynamisch erkannt und umfahren werden müssen.

3.1 Grundidee

Da unser Ursprünglicher Ansatz der Erkennung der weitesten Distanz und der entsprechenden Reglung (wie unten beschreiben) in einer so kurzen Zeit nicht zufriedenstellend umsetzbar war, entschied sich unsere Gruppe für den Abschlusspräsentation, bestehende Bibliotheken zu verwenden. Unsere gewählte Methode besteht aus:

Positionierung
Hierfür haben wir „Adaptive Monte Carlo Localization“ (nachfolgend AMCL) gewählt. Dieser benötigt eine Karte und Laserscans, und kann anhand dessen die Position des Fahrzeugs annähern und diese als Odometrie-Daten veröffentlichen.

Costmaps
Um einen Globalen Plan zu erstellen, muss beschrieben werden, welche Zonen auf der Karte vermieden werden sollten. Hierzu kann „costmap_2d“ Costmaps erstellen. Dafür wird eine Hinderniskarte verwendet und die Hindernisse zusätzlich „aufgepustet“ (inflation), um zu vermeiden, dass das Fahrzeug ihnen zu nahe kommt.

Globaler Planer
Auf dem Navigation Stack aufbauend, können wir mit Hilfe des globalen Planers einen Pfad zu einem Navigationsziel erstellen. Dies beachtet keine lokalen Hindernisse, sondern nur die globale Costmap der vorgegebenen Karte.

Lokaler Planer
Dabei haben wir den „Timed Elastic Band Local Planner“ (nachfolgend teb) gewählt. Damit dieser richtig funktioniert, müssen einige Parameter angepasst werden. Teb ist insbesondere für Auto-ähnliche Roboter eine sinnvolle Wahl, da bei den meisten anderen Planern der Roboter als holonom angenommen wird – wir jedoch das Fahrzeug nicht um die eigene Achse drehen können. Auch Rückwärts-Bewegungen können so sinnvoll umgesetzt werden.

3.2 Umsetzung

Anhand dieser Grundeinrichtung können wir uns bereits auf einer Karte positionieren und anhand von Kriterien, die im globalen und lokalen Planer beschrieben werden, Routen erstellen. Teb sendet hierbei Steuerbefehle, die von einem von uns erstellten Controller in „pses_basis“-Befehle übersetzt werden. Wenn das Fahrzeug sich dann in einem von uns bestimmten Abstand zum Navigationsziel befindet, published der Controller das nächste Ziel aus einem Array von Navigationszielen, und der globale

Planer passt die Route entsprechend an. Wenn das letzte Ziel im Array erreicht wurde, beginnt die Prozedur von vorne. Das Array von Navigationszielen kann der Einfachheit halber in einem eigenen Modus erstellt werden. Hierfür bietet das AuTU-Dashbord die Option „Create Route“. Nach auswählen des Modus können auf der Karte nacheinander Navigationsziele gesetzt werden. Diese werden beim Verlassen des Modus (Controller-Zerstörung) in eine XML-Datei gespeichert und können anschließend vom Controller des Modus „Rundkurs mit Hindernissen“ eingelesen und abgefahren werden.

3.3 Fazit

Die gewählte Variante des Speicherns von Navigationspunkten in einer XML-Datei und dem Abfahren mit dem teb-Local-Planner bietet eine solide Variante, den Rundkurs mit Hindernissen zu bewältigen. Die XML-Datei kann auch nach Neustarten des Fahrzeuges wiederverwendet werden, und der Modus „Create Route“ bietet eine einfache Möglichkeit, Richtung, Ort und Anfangspunkt zu ändern. Die Positionierung auf einer Karte mit AMCL war besonders bei geringen Geschwindigkeiten und kleinen Lenkwinkeln problematisch, sodass andere Gruppen auf einen zusätzlichen Drehgeber oder Magnetkompass zurückgreifen mussten. Für einen einfachen Rundkurs ist die vorhandene Odometrie jedoch völlig ausreichend. Zusätzlich ist die Pfadplanung mit globalem und lokalem Pfad nicht immer optimal.

Abbildung 3.1: Zustandsdiagramm

Es wird oft sehr dicht zu einem Hindernis gefahren, bis diesem dann ausgewichen wird, da der lokale Planer versucht dem globalen Planer möglichst nahe zu kommen. Deshalb kam uns ursprünglich die Idee, nicht möglichst genau dem globalen Pfad zu folgen, sondern, vereinfacht ausgedrückt, dorthin zu fahren, wo am meisten Platz ist.

3.4 Alternative (verworfene) Lösung

Im Laufe des Projekts haben wir auch eine andere Lösung für den Rundkurs mit Hindernissen programmiert, die jedoch später wieder verworfen wurde. Der Algorithmus besteht generell aus zwei Teilen: Beim ersten Teil sucht sich das Auto mithilfe der Kinect Kamera den Punkt, der am weitesten von dem Auto entfernt ist.

Dann steuert das Auto gerade auf diesen Punkt zu. Um jedoch um Kurven fahren zu können, braucht das Auto einen Linksdrall, da es ansonsten einfach an der Kurve geradeaus den Gang herunter fahren würde. Dazu wird eine Funktion auf alle

Abbildung 3.2: Zustandsdiagramm

Laserscan-Punkte angewendet, die in Abhängigkeit zum Winkel zur Fahrtrichtung den gemessenen Abstand verringert, je weiter rechts desto mehr. Außerdem wurde die Funktion bei großen Ultraschall-Abstandswerten nach links verstärkt, um nochmals besser um Kurven herum zu kommen.

Dieser Ansatz hat beim Hindernis umfahren und Kurven fahren gut funktioniert, jedoch führte der Linksdrall auf Geraden zu stark schwankender Fahrt, da das Auto immer wieder auf die Wand zusteuerte und dann wieder ausweichen musste. Aus diesem Grund haben wir den Algorithmus um den zweiten Teil, einem PD-Regler, erweitert. Dieser funktionierte im Prinzip bis auf leicht andere Parameter genauso wie bei dem Rundkurs ohne Hindernisse: Der Regler hält den Abstand zur linken Wand auf einem konstanten Niveau, solange kein Hindernis von der Kinect Kamera vor dem Auto in einem bestimmten Abstand erkannt wurde. Diese Hinderniserkennung suchte ein etwa 1,2 Meter langes und 0,4 Meter breites Rechteck vor dem Auto nach Laserscan Punkten ab, die ein Hindernis andeuten. Sollte solch ein Hindernis erkannt werden, wurde der Regler deaktiviert und der weiteste-Distanz-Controller benutzt. Der Regler war allerdings nicht stark genug, um um enge Kurven herum zu fahren. Darum haben wir wiederum die Kurvenerkennung aus dem Rundkurs ohne Hindernisse benutzt, um den Regler für einige Sekunden zu deaktivieren und die Kurve mit dem weiteste-Distanz-Algorithmus zu fahren. Die Kurvenerkennung war jedoch aufgrund der unregelmäßigen Positionierung von Hindernissen und dem unregelmäßigen Abstand zur Wand nicht mehr ganz zuverlässig, insbesondere false-positive-Fehler traten immer wieder auf. Dies war jedoch noch verschmerzbar, da der weiteste-Distanz-Controller generell auch gut auf der Geraden fahren kann. Letztendlich wurde der Algorithmus aber aufgrund verschiedener kleiner Probleme verworfen, insbesondere das fehlende Gedächtnis war ein Problem (also dass das Auto sich bereits vorher erkannte Hindernisse nicht merkt und es auch keinerlei Wissen über die aktuelle Position besitzt).

4 Paralleles Einparken

Bei dieser Aufgabe muss das Auto in eine möglichst kleine Parklücke hinter einem anderen bereits geparkten Fahrzeug rückwärts parallel einparken. Die theoretischen Grundlagen der in diesem Abschnitt gemachten Annahmen entstammen dem wissenschaftlichen Artikel [Roj10], in dem sich eingehend mit der Geometrie des Einparkens auseinandergesetzt wird. Die Herleitung der Formeln kann der Quelle entnommen werden.

4.1 Grundidee

Das Fahrzeug fährt zu einer zuvor berechneten Startposition. Dort angekommen wird nach rechts eingelenkt und rückwärts gefahren. Wird nun ein gewisser Gierwinkel erreicht, wird nach links gelenkt bis das Fahrzeug in der Parklücke steht.

4.2 Verwendete Variablen

Folgende Variablen werden in den folgenden Abschnitten analog zu [Roj10] verwendet:

- R: minimaler Kurvenradius
- W: Breite des Autos
- b: Distanz zwischen Hinterrad und vordere Ecke des Autos
- θ : Gierwinkel

4.3 Ackermann-Lenkung

Als Erstes ist es wichtig zu verstehen, wie sich das Auto bei Lenkbewegungen verhält. Dazu greift man auf das Modell der Ackermann-Lenkung zurück. Es sagt aus, dass das Auto sich beim Kurvenfahren auf einer Kreisbahn um ein Rotationszentrum herum bewegt.

4.4 Minimale Parklücke

In der Aufgabenstellung ist gefordert, dass die Parklücke möglichst klein sein soll. Was ist also die minimal benötigte Parklückengröße bei gegebenem Kurvenradius des Fahrzeugs? Besagte Größe lässt sich mit einem Gedankenexperiment recht leicht ermitteln. Hierfür wird angenommen, dass sich das Auto bereits in der Parklücke befindet. Die optimale Lücke ist nun die kleinstmögliche Lücke, aus welcher das Fahrzeug durch volles Einschlagen der Räder nach links noch ausparken kann, ohne das vor ihm geparkte Auto zu berühren.
Mit Hilfe der Formel $h = \sqrt{b^2 - W^2 + 2RW}$ aus [Roj10] lässt sich die Größe der Lücke wie folgt ermitteln: Länge des Autos $+ h - b$
Setzt man für den Kurvenradius des Autos $1m$ an, ergibt sich daraus eine Parklücke von ca. 70cm. Wählt man den in Abschnitt 4.7 definierten Sicherheitsabstand zu 10cm, wächst die minimale Parklücke auf ca. 80cm an. Aufgrund von diversen Ungenauigkeiten des Systems sollten reale Parklücken mindestens 90cm groß sein, um Parkrempler jegliche Art mit Sicherheit ausschließen zu können.

4.5 Parklückenerkennung

Unsere Annahme ist, dass sich im Sichtbereich der Kinect-Kamera ein geparktes Auto befindet, an dem sich beim Parken orientiert werden kann. Die hintere linke Ecke des geparkten Autos wird erkannt, indem die Punkte des Laserscans mittels Linienerkennung nach 2 Linien abgesucht werden, die einen gemeinsamen Punkte besitzen und somit eine Ecke darstellen.

4.5.1 Linienerkennung mittels RANSAC

Mit Hilfe eines **Ra**ndom **Sa**mple **C**onsensus-Algorithmus (RANSAC) lassen sich Linien im Laserscan erkennen. Die einzelnen Schritte sind im Folgenden erklärt:

1. Wähle zwei Punkte zufällig aus.
2. Erzeuge mit Hilfe dieser beiden Punkte eine Linie.
3. Iteriere über alle Punkte und zähle, wie viele Punkte sich in der unmittelbaren Umgebung der Linie befinden.
4. Unterstützen genug Punkte das Modell auf diese Weise, wird die Lösung gespeichert. Die Menge der unterstützenden Punkte wird als *Consensus Set* bezeichnet.
5. Wiederhole die ersten vier Schritte. Wurden genügend Iterationen durchgeführt, wird das Modell mit dem größten Consensus Set ausgewählt.

Die Qualität und Zuverlässigkeit der Lösung hängen dabei stark ab von der Anzahl der durchgeführten Iterationen und der größtmöglichen Entfernung eines Punktes von der Linie, für welche er noch im Consensus Set liegt, ab.

4.5.2 Linienerkennung mittels RLF

Um in einer Menge von Punkten Linien zu erkennen, geht man beim **R**ecursive **L**ine **F**itting (RLF) wie folgt vor: Man verbindet die beiden äußeren Punkte im Laserscan mit einer Linie. Nun wird der am weitesten von der Linie entfernte Punkt bestimmt. Ist diese Abweichung kleiner als ein gewisser Schwellwert, wird die Linie als solche akzeptiert. Ansonsten wird die Punktmenge an dieser Stelle aufgetrennt und der Algorithmus rekursiv auf die beiden Teilmengen angewendet. Zusätzlich bricht der Algorithmus ab, wenn die Punktmenge nicht eine gewisse Mindestzahl an Elementen aufweist.

Aufgrund des deterministischen Verhaltens und des deutlich geringeren Rechenaufwandes wird RLF zur Linienerkennung eingesetzt und nicht der oben beschriebene RANSAC.

4.6 Berechnung des maximalen Gierwinkels

Die folgenden Formeln stammen aus [Roj10]. Durch sie erhält man den Gierwinkel θ.

$$R_e = \sqrt{R^2 + b^2}$$

$$\alpha = \arccos(1 - \frac{b^2}{4R^2})$$

$$\beta = \arcsin(\frac{R}{R_e} * \sin(\alpha))$$

$$\theta = \arccos(\frac{R - W}{R_e}) - \beta$$

4.7 Sicherheitsabstand

Die bisherigen Annahmen wurden unter idealen Bedingungen gemacht, sodass das Auto etwa mit Abstand 0 an anderen Objekten vorbei fährt. Da dies in echt nicht praktikabel ist, berechnen wir alle Werte mit einem zusätzlichen Sicherheitsabstand. Dies wird erreicht, indem wir sowohl unser Auto breiter annehmen als es eigentlich ist und noch die Startposition für die Einlenkbewegung etwas verschieben.

5 Projektstruktur

5.1 Hauptprogramm

Um einen einfachen Übergang von einem Modus in einen anderen zu ermöglichen, entschieden wir uns dafür, nicht jeden Modus als eigenständige Node zu schreiben, sondern eine Hauptnode zu erstellen, die im jeweiligen Modus dann nur den dazugehörigen Controller erstellt.

Die Controller implementieren hierbei jeweils eine Run-Methode und kann direkt auf den Command-Publisher der Hauptnode zugreifen. Kommandos werden nur von der Run-Methode gesendet. Dies bietet einerseits den Vorteil, dass Kommandos nur mit einer in der Hauptnode definierten, bestimmten Frequenz an das Microcontroller Board gesendet werden und andererseits die Möglichkeit, Controller für Zeiträume die Publish-Befugnis zu entziehen. Dadurch ist es zum Beispiel möglich, eine globale Notbremse zu implementieren oder global einen Controller anzuschließen, der immer die Daten des jeweiligen Modus überschreibt.

Fazit

Das Prinzip des übergeordneten Hauptprogramms ermöglichte uns in der Praxis, sehr simpel globale Änderungen an z.B. der Notbremse vorzunehmen. Auch eine Controllerübergreifende Nutzung von Daten wurde dadurch möglich, in dem beispielsweise die Hauptnode Laserscans abboniert, und der Pointer an die Controller übergeben wird. Neue Controller müssen so nicht erst warten bis nächster Laserscan gesendet wird, sondern können bereits beim Moduswechsel auf die in der Hauptmethode gespeicherten Daten zugreifen.

6 Boostmode

6.1 Motivation der Leistungssteigerung

Durch den in Abschnitt 2 präsentierten Ansatz konnten Rundenzeiten im Bereich von 21 Sekunden ermöglicht werden. Der eingesetzte PD-Regler ermöglicht eine robuste Geradenfahrt parallel zur Wand und die Kurvenerkennung verbesserte sowohl Einlenkzeitpunkt sowie -verhalten. Durch die Entkopplung von Geraden- und Kurvenfahrt, die ebenfalls eine wichtige Funktionalität unseres Steuerungskonzepts ist, konnten beide Bereiche unabhängig voneinander optimiert werden. Das resultierte in einem Steuerungskonzept, deren Trajektorien nur noch unwesentlich verbessert werden konnte. Die limitierende Größe war nun die Leistung des Motors. Diese ist bei maximaler PWM vom anliegenden Strom abhängig.

6.2 Technische Umsetzung der Leistungssteigerung

Elektromotoren, wie der im Fahrzeug eingebaute Gleichstrommotor Tamiya Typ 540 können prinzipbedingt für eine kurze Zeit mehr Leistung abgeben, als dass im Nennbetrieb möglich wäre[1] . Beim sogenannten Kurzzeitbetrieb wird durch einen Strom, der größer ist als der Nennstrom des Motors eine größere Leistungsabgabe ermöglicht. Die erhöhte Leistungsabgabe führt aufgrund der Verluste im Motor zu dessen Erwärmung, weshalb bei stationären Maschinen der Motor nur eine kurze Zeit überlastet werden darf. Man spricht dabei der Überlastbarkeit eines Elektromotors. In unserem speziellen Anwendungsfall geht jedoch mit einer höheren Motorleistung eine höhere Kühlwirkung durch den Fahrtwind einher, weshalb eine moderate Überlastung des Motors vertretbar ist. Bei gegebener Impedanz des Motors hängt die Leistungsaufnahme des Motors nach der Drehmomentformel für Gleichstromantriebe (1) vom zur Verfügung gestellten Strom I_a ab.

$$M_e = k_2 * I_a * \phi \tag{1}$$

Der Strom wiederrum ist bei gegebenem Widerstand R_m des Motors durch die anliegende Spannung gegeben. Um die Motorleistung zu erhöhen muss somit die Motorspannung erhöht werden. Der Motor ist somit nicht das limitierende Bauteil. Der Speedcontroller, der die PWM für den Motor erzeugt besitzt jedoch eine maximale Betriebsspannung. Der im Fahrzeug verbaute TEU-101BK ist dabei auf $U_{max} = 7,2V$ limitiert, welche auch der Akkuspannung entspricht. Um die Rundenzeiten wiedergehend zu optimieren ist also eine höhere Akkuspannung nötig. Zu diesem Zweck ist der ursprüngliche Speedcontroller durch ein Modell mit einer Maximalspannung von $U_{max} = 12V$ ersetzt worden und der ursprüngliche Akku durch Lithium-Polymer-Akkus mit einer Nennspannung von ebenfalls 12 V. Dabei war die besondere Handhabung von LiPo-Akkus zu beachten, deren zulässiges Spannungsfenster nicht verlassen werden darf, da sonst eine Gefährdung der

[1] Vgl. Praktikum Aktoren für mechatronische Systeme, Versuch 4: Geschaltete Reluktanzmaschine, Seite 22

Seminarteilnehmer entstehen kann. Zu diesem Zweck wurde der Akku ausschließlich mit einem Ladungs-Überwacher betrieben, der bei zu niedriger Spannung ein Warnsignal ausgibt, um eine Tiefenentladung der Akkus zu verhindern. Im Rahmen dieses Seminars wurde der LiPo Alarm (LED & Buzzer) von Tarot verwendet.

6.3 Potential und Herausforderungen

Im Folgenden soll das Potential des leistungsgesteigerten Fahrzeugs abgeschätzt werden. Dabei wird die Annahme getroffen, dass die Fahrwiderstände des Fahrzeugs durch einen geschwindigkeitsproportionalen Reibwiderstand hinreichend genau abgebildet werden. Diese Annahme kann getroffen werden, da der Luftwiderstand quadratisch mit der Geschwindigkeit eingeht, wodurch er bei niedrigen Geschwindigkeiten vernachlässigt werden kann. Nach Formel (1) besteht eine direkte Proportionalität $M_e \sim I_a$. Bei gegebenem Widerstand des Motors R_m ist das Motormoment somit proportional zur anliegenden Spannung U. Aus dieser Proportionalität lässt sich folgende Gleichung (2) ableiten:

$$\frac{M_{neu}}{M_{alt}} = \frac{U_{neu}}{U_{alt}} \tag{2}$$

Somit konnte das Antriebsmoment des Fahrzeugs um 67% gesteigert werden. Durch die getroffene Annahme ist es nun möglich die neue Höchstgeschwindigkeit abzuschätzen:

$$\frac{v_{max,neu}}{v_{max,alt}} = \frac{M_{neu}}{M_{alt}} \tag{3}$$

Somit ist eine neue Höchstgeschwindigkeit von $v_{max,neu} = 3,3m/s$ möglich. Diesem Potential stehen eine Reihe technischer Herausforderungen entgegen, die das Ausschöpfen des Potentials erschweren. Regelungstechnisch führt eine höhere Geschwindigkeit zu einem tendenziell schwerer regelbarem Systemverhalten. Für die Analyse des Systemverhaltens ist im Rahmen dieses Seminars das Ackermann-Modell verwendet worden. Daraus konnte eine Führungsübertragungsfunktion für das Querverhalten abgeleitet werden:

$$G(s) = \frac{\gamma}{\varphi_L} = \frac{v * l_H}{l} * \frac{\frac{v}{l_H} + s}{s^2} \tag{4}$$

Wie in der Gleichung zu sehen ist, hat die Geschwindigkeit wesentlichen Einfluss auf die Lage der Nullstelle des Systems. Durch eine höhere Geschwindigkeit entfernt sich diese Nullstelle immer weiter vom Nullpunkt. Der zur Regelung des Systems mit einem PD-Regler notwendige Einfluss der Nullstelle wird somit schwächer, was somit zu einer höheren Instabilität des Gesamtsystems führt. Die zweite Herausforderung bei Hochgeschwindigkeitsfahrten ist die wachsende Fehleranfälligkeit der visuellen Sensorik. Zum einen existiert eine nicht vernachlässigbare Latenz zwischen der Kinect und dem Empfangen der Information nach Transformation in Laserscan-Daten. Diese Latenz wurde in der Praxis zu durchschnittlich $t_L = 300ms$ ermittelt. Zum anderen sinkt die Qualität der visuellen Informationen. Die Anzahl der Fehldetektionen, sowohl

falsch-positiver als auch falsch-negativer Natur nehmen zu, wodurch eine akzeptable Steuerung auf Basis der Kinect-Tiefeninformationen nicht mehr realisierbar ist. Bei praktischen Fahrversuchen konnten mit leicht gesteigerter Höchstgeschwindigkeit Rundenzeiten bis zu 19 s realisiert werden, was das Potential des Motors verdeutlicht.

6.4 Ausblick

Um die höhere Maximalgeschwindigkeit beherrschbar zu machen existieren einige Ansätze, die es nicht in die finale Implementierung geschafft haben, allerdings an dieser Stelle kurz erläutert werden sollen. Das von uns gewählte Konzept beruht auf einem Zustandsautomaten, der die Kurvenfahrt von der Geradenfahrt trennt. Durch Einführen eines weiteren Zustands für Hochgeschwindigkeiten auf Geraden kann mit einfachen Mitteln eine signifikante Verbesserung erzeugt werden. Um die Stabilität zu gewährleisten, könnte als Eingangsbedingung in die Transition eine minimale Zeitdauer festgelegt werden, in der das Fahrzeug eine maximale Gierrate nicht mehr überschritten hat. Dadurch könnte eine Gradeausfahrt erkannt werden, bei der ein Geschwindigkeitsschub beherrschbar bliebe. Des Weiteren führt das hohe Moment des Motors dazu, dass die Ableitung der Beschleunigung, der Ruck, bei Zustandsübergängen besonders hohe Werte annimmt. Das führt zu einer plötzlichen Zustandsänderung, die nur schwer auszuregeln ist. Um dieses Phänomen zu kontrollieren könnte eine kontinuierliche Erhöhung der Beschleunigung, eine sogenannte Ruckbegrenzung, implementiert werden. Nicht zuletzt könnte ein robusteres und effektiveres Regelkonzept die Schwingungen unterbinden, die bei höheren Geschwindigkeiten auftreten. Die Anfälligkeit der visuellen Sensorik bei hohen Geschwindigkeiten kann umgangen werden, wenn die Robustheit durch eine Sensordatenfusion erhöht wird. Dadurch können plausible Werte der Kinect herausgefiltert werden. Die dadurch gewonnene Positionierung macht zudem eine Trajektorienfolgereglung möglich, die, nach Auslegen einer zeitoptimalen Trajektorie, sogar das optimale Rundenzeit erreichen könnte.

7 Fazit

Das Projektseminar hat eine gute Plattform vorgegeben, die viele Möglichkeiten zur Bewältigung der Aufgabenstellung bietet. Insbesondere die Kinect-Kamera stellt eine sehr starke Erweiterung der Hardware dar und macht das Navigieren mit vorhandenen ROS Paketen relativ einfach. Wir konnten sie aber auch gut in unsere eigenen Pakete einbinden. Die Simulation ist ein hilfreiches Werkzeug, um zu testen, ob Code grundsätzlich funktioniert, allerdings gab es eine starke Abweichung zum realen Verhalten des Fahrzeugs.

7.1 Probleme

Wie einige andere Gruppen hatten auch wir das Problem, dass der Mikrocontroller häufig abgestürzt ist. Bei einem Absturz konnten Motor- und Lenkbefehle nicht mehr versendet werden, was zu Unfällen führen konnte und das Testen erschwert hat. Die Ursache blieb bis zum Schluss unklar. Allerdings konnten wir das Problem durch das Verwenden einer älteren Firmware des Mikrocontroller-Boards lösen.

7.2 Ausblick und mögliche Verbesserungen

Unsere Gruppe musste viel Zeit investieren, um sich in die Grundlagen von der Verwendung von ROS und des Basis Pakets einzuarbeiten. Durch eine ausführlichere Einführung in ROS und Dokumentation des Basis Pakets könnten die Gruppen sich mehr auf die eigentliche Aufgabenstellung konzentrieren.

Um das Projektseminar noch weiter voranzubringen, halten wir es für sinnvoll, mehr auf den Lösungen der Vorjahresgruppen aufzubauen. Da sich durch die Verwendung der Kinect eine Navigation mithilfe der Pakete AMCL und teb local planner anbietet, welche auch von vielen Gruppen verwendet wurden, könnte man dies in das Basis Paket aufnehmen. So könnte man in Zukunft neue und komplexere Aufgaben bewältigen. Außerdem könnte man Gruppen mehr dazu auffordern, sich eigene Aufgabenstellungen auszudenken, um noch mehr einzigartige Ideen im Projektseminar zu gestalten.

8 Quellen

Quellenverzeichnis vom 14.04.2017:

(1) http://wiki.ros.org/amcl

(2) http://wiki.ros.org/depthimage_to_laserscan

(3) http://eigen.tuxfamily.org

(4) http://wiki.ros.org/costmap_2d?distro=indigo

(5) http://wiki.ros.org/base_local_planner?distro=indigo

(6) http://wiki.ros.org/teb_local_planner

Literatur

[Roj10] Rojas, Raul: Die Weltformel: Wie man ein Auto einparkt. (2010)

[XPC+05] Xavier, João ; Pacheco, Marco ; Castro, Daniel ; Ruano, António ; Nunes, Urbano: Fast line, arc/circle and leg detection from laser scan data in a player driver. In: *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on* IEEE, 2005, S. 3930–3935

[Zhe16] Zheng, Kaiyu: ROS Navigation Tuning Guide. (2016)