# Autonomes Parken mit Parklückenerkennung

Jannes Bützer, Elektromobilität
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Anlässlich des Projektseminars Elektrotechnik/Informationstechnik an der Otto-von-Guericke-Universität Magdeburg sollte ein Roboter entwickelt werden, der die Funktionsweise eines Parkassistenten simuliert. In nachfolgender Arbeit wird veranschaulicht, wie sich dies mithilfe von Lego Mindstorms realisieren ließ. Dabei wird auch grafisch auf die Umsetzung des Roboters eingegangen. Neben Konstruktion und Programmierung erfolgt zudem eine Analyse der dabei aufgetretenen Schwierigkeiten und Probleme. Im Zuge des Projekts ist dabei ein Fahrzeug entstanden, das seine Parklücke selbstständig erkennt und sowohl ein- als auch ausparken kann. Hierbei wird zwischen Längs- und Querparken, sowie zwischen verschiedenen Größen der Parklücke differenziert.**

***Schlagwörter*—Autonomes Fahren, Assistenzsysteme, Parken, Parklückenerkennung, LEGO® MINDSTORMS®, MATLAB**

## I. Motivation

DIE Mobilität der Zukunft hat längst begonnen. Auch wenn das vollautonome Fahren gesetzlich noch nicht erlaubt ist, werden moderne Automobile immer eigenständiger. Doch bis sich der Mensch seiner Aufgaben als Fahrzeugführer voll und ganz entledigen und zum reinen Passagier werden kann, sollen unzählige Fahrerassistenzsysteme für Entlastung und Unterstützung sorgen. So verringert beispielsweise der Notbremsassistent, falls ein Hindernis naht, die Geschwindigkeit des Fahrzeuges bis hin zum Stillstand. Der Fernlichtassistent blendet je nach Lichtverhältnissen selbstständig auf oder ab. Neben der Sicherheit im Straßenverkehr bietet die Entwicklung des autonomen Fahrens auch noch einige andere Vorteile. Zum einen lassen sich während der Fahrt wichtige Dinge erledigen, da die Steuerung des Autos nicht mehr übernommen werden muss. Zum anderen können auch Menschen mobil sein, die aufgrund ihres körperlichen oder psychischen Gesundheitszustandes nicht in der Lage sind, ein Fahrzeug eigenständig zu führen.

Aus genannten Gründen war es ersichtlich, sich des Themas Mobilität im Rahmen des zweiwöchigen Projekts anzunehmen und einen Teil des autonomen Fahrens, das autonome Parken, näher zu betrachten. Die Idee war es dementsprechend, ein Fahrzeug zu konstruieren, das eigenständig eine geeignete Parklücke erkennt und nach dessen Begebenheiten autonom ein- und ausparkt. Zur Verfügung standen dabei die Komponenten von Lego Mindstorms mit Motoren und Sensoren, wie z. B. Licht-, Tast- oder Ultraschall-Abstandssensoren. Außerdem der „NXT-Brick“, der die Schnittstelle der Komponenten bildet und mithilfe der Software MATLAB programmiert werden konnte. Eine eigens von der RWTH Aachen entwickelte Toolbox mit NXT-Befehlen diente zur Ansteuerung der Aktorik und Sensorik.

## II. Vorbetrachtungen

### A. Stand der Technik

Parkassistenzsysteme gibt es schon seit mehreren Jahren. So stellte Volkswagen mit dem „Park Assist“ im Jahre 2006 den weltweit ersten Parklenkassistenten mit integrierter Parklückensuche vor. Nach weiteren Entwicklungen beherrscht dieser, neben dem ursprünglichen Längsparken, heute auch das Quer- und Vorwärtsparken [1]. Benötigt werden dafür, zusätzlich zu zwei seitlichen Sensoren, jeweils vier Ultraschallsensoren an Front und Heck des Fahrzeuges [2]. Beschleunigen, bremsen und zwischen den Zügen schalten, muss der Fahrer hingegen noch selbst übernehmen. Ein Modell, in dem das Auto sogar komplett ohne Fahrer auskommt, wurde in Zusammenarbeit von Bosch und Daimler entwickelt [3]. Hierbei fährt das Fahrzeug autonom durch ein Parkhaus und sucht unterdessen einen Parkplatz, während sich der Fahrer schon mit anderen Dingen beschäftigen kann. In gleicher Weise wird das Auto auch wieder ausgeparkt. Den Ein- oder Ausparkvorgang startet der Fahrer per Smartphone-App.

### B. Eigene Lösung

Da der NXT nur Anschlussmöglichkeiten für vier Sensoren bietet, konnten lediglich drei Ultraschallsensoren für Abstandsmessungen und ein Farbsensor zur Lenkzeitpunkterkennung zum Einsatz kommen. Die Punkte, die dabei im Vorfeld geklärt werden mussten, waren zum einen die Ermittlung der Größe der Parklücke und zum anderen die unterschiedlichen Parkabläufe bei unterschiedlich großen Parklücken.

Zur Parklückenerkennung bot es sich an, aus der Geschwindigkeit des Fahrzeuges und der zum Passieren der Lücke benötigten Zeit, die Länge der Lücke zu bestimmen. An einen Motor kann allerdings nur die sogenannte Power $p$ übergeben werden. Diese entspricht der am Motor anliegenden Batteriespannung und wird in Prozent angegeben. Um den Zusammenhang zwischen Power und der daraus resultierenden Geschwindigkeit $v$ (in $\frac{\mathrm{cm}}{\mathrm{s}}$) herzustellen, führte man einen Versuch durch. Dazu wurde eine Strecke von einem Meter mit schwarzen Linien abgeklebt und ein Programm zur Zeitmessung zwischen den Linien geschrieben. Die Ergebnisse dieses Versuchs veranschaulicht Tabelle I. Anhand des ermittelten Faktors $c$ und der Gleichung (1) kann folglich aus jeder beliebigen Power (in %) eine Geschwindigkeit (in $\frac{\mathrm{cm}}{\mathrm{s}}$) berechnet werden.

$$v = p \cdot c \tag{1}$$

$p =$ Power (in %)
$c =$ konstanter Faktor (in $\frac{\mathrm{cm}}{\mathrm{s} \cdot \%}$)

DOI: 10.24352/UB.OVGU-2020-028 

Tabelle I
MESSDATEN ZUR GESCHWINDIGKEIT

| Power (in %) | Zeit (in s) | Geschwindigkeit (in $\frac{\text{cm}}{\text{s}}$) | Faktor (in $\frac{\text{cm}}{\text{s}\cdot\%}$) |
|---|---|---|---|
| 10 | 21,190 | 4,71 | 0,471 |
| 20 | 10,616 | 9,41 | 0,470 |
| 30 | 7,066 | 14,15 | 0,471 |
| 40 | 5,271 | 18,97 | 0,474 |
| 60 | 3,461 | 28,89 | 0,481 |
| 80 | 2,642 | 37,85 | 0,473 |
| 100 | 2,591 | 38,59 | 0,385 |

Eine weitere Grundlage war die Berücksichtigung unterschiedlicher Längen der Parklücke. In einer großen Lücke kann nicht mit dem gleichem Ablauf eingeparkt werden wie in einer kleinen. In einer sehr engen Parkbox muss außerdem noch ein Zug zum Korrigieren vorgenommen werden. Demnach müssen beim Längsparken drei Größen in Betracht gezogen werden, die nach Konstruktion und Erprobung des Fahrzeuges in das Programm aufgenommen werden konnten. In Folge dieser Betrachtungen lassen sich nun die Algorithmen mithilfe von MATLAB formulieren.

## III. REALISIERUNG

Um das autonom parkende Fahrzeug realisieren zu können, müssen zwei Komponenten in Betracht gezogen werden. Zum einen die konstruktive Seite, welche die effektive Anordnung von Aktorik und Sensorik umfasst und dem Vehikel Stabilität verleiht, und zum anderen die programmiertechnische Seite, die das Fahrzeug letztendlich in Bewegung versetzt und steuert. In folgenden Abschnitten wird deshalb genauer auf diese beiden Bestandteile eingegangen.

### A. Konstruktion

Bei der Konstruktion des Roboters stehen neben Stabilität und Funktionalität vor allem auch die realitätsnahe Gestaltung des Vehikels im Fokus. Die Wahl fällt daher auf ein Fahrzeug mit vier Rädern, zwei Motoren für die Aktorik, sowie drei Ultraschallsensoren und einem Farbsensor für die Sensorik. Alle Komponenten des Fahrzeuges sind an den Lego-NXT-Brick angeschlossen, der die Steuereinheit des Roboters bildet und gut zugänglich oben auf dem Fahrzeug angebracht ist.

Einer der Motoren fungiert zur Fortbewegung und ist über ein Differentialgetriebe mit den Hinterrädern verbunden. Dieses dient dem Ausgleich der Drehzahlen der beiden angetriebenen Räder in Kurvenfahrten. Dabei dreht sich das kurveninnere Rad, aufgrund seines kleineren Radius zum Kurvenmittelpunkt, langsamer als das äußere Rad. Somit entsteht ein verbessertes Fahrverhalten in engen Kurven, wie es z. B. bei Parkvorgängen der Fall ist. Ein weiterer Motor befindet sich im vorderen Teil des Fahrgestells und betreibt die Lenkung der Vorderräder. Diese ist realitätsgetreu nach dem Ackermann-Lenkprinzip [4] konstruiert. Hierbei sind die Räder nicht über eine Achse miteinander verbunden, sondern besitzen jeweils eine eigene Schwenkachse. Bei Kurvenfahrten wird so das kurveninnere Rad weiter eingeschlagen als das kurvenäußere. Besonders bei großen Lenkwinkeln, wie es auch beim Parken der Fall ist, wird so eine gute Fahrdynamik aufrecht erhalten. Die großen Lenkwinkel werden durch eine optimierte Konstruktion erreicht, die es ermöglicht, von ursprünglich $100°$ auf $160°$ Drehwinkel am Motor zu gelangen.

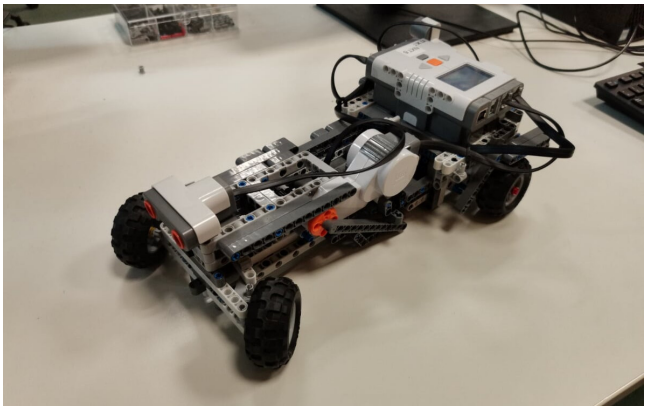

Abbildung 1. Konstruktion des Fahrzeuges

Bei der Anordnung der Sensorik war es das Ziel, alle Komponenten so kompakt wie möglich anzuordnen, die Funktionalität dabei jedoch nicht zu beeinträchtigen. Um die Konstruktion realitätsnah zu gestalten und auch das Einparken in kleinen Parklücken zu ermöglichen, sollte keiner der Sensoren über das Vehikel hinausragen. Aus diesem Grund endet das Fahrgestell, wie in Abbildung 1 dargestellt, genau mit dem vorderen Ultraschallsensor. Dieses Prinzip wird auch auf den hinteren und seitlichen Ultraschallsensor angewandt. Beim seitlichen Sensor wurde zudem noch darauf geachtet, die Befestigung am hinteren Ende vorzunehmen, damit das Heck des Fahrzeugs nach Erkennung einer Parklücke mit dem Ende der Lücke abschließt. Um die Farbmarkierungen zum richtigen Zeitpunkt erkennen zu können, musste auch der Farbsensor im hinteren Teil untergebracht werden. Dieser ist daher an der Innenseite des rechten Hinterrads positioniert.

### B. Programmierung

Um das Programm zu beginnen, muss der Nutzer über die grafische Benutzeroberfläche (siehe Abschnitt: III-C) vorab den Wert der Power eingeben und den gewünschten Vorgang starten. Wie bereits in den Vorbetrachtungen erwähnt, wird diese mithilfe der Gleichung (1) in eine Geschwindigkeit (in $\frac{\text{cm}}{\text{s}}$) umgewandelt, damit später die Größe der Parklücke bestimmt werden kann. Die weiteren Programmierungen lassen sich in drei Vorgänge unterteilen. Soll das Fahrzeug einparken, muss es zunächst eine Parklücke finden. Danach folgt der eigentliche Parkvorgang, während zum Schluss noch ausgeparkt werden kann. Alle Abläufe werden im Folgenden näher erläutert.

*1) Parklückenerkennung:* Während sich das Fahrzeug vorwärts bewegt, misst der seitlich angebrachte Ultraschallsensor dauerhaft den Abstand zur Wand, welche die bereits parkenden Autos darstellt. Tritt hierbei eine große Differenz der Abstandswerte nach oben auf, so wird dieser Zeitpunkt gespeichert. Werden die Abstände sprunghaft kleiner, wird erneut ein Zeitpunkt im Speicher hinterlegt. Aus der Differenz beider Zeitpunkte wird nun die zum Passieren der Lücke

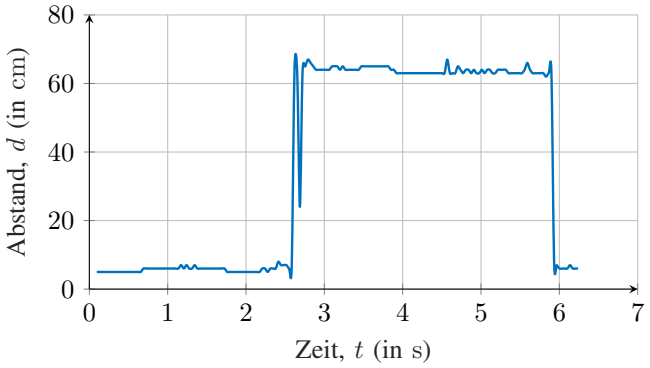


Abbildung 2. Zeitverlauf des Abstandes des Fahrzeuges zur Wand, gemessen durch den seitlichen Ultraschallsensor

benötigte Zeit bestimmt. Erkennbar ist dieser Zusammenhang in Abbildung 2. Die Zeitpunkte des sprunghaften Verhaltens des Graphen signalisieren jeweils den Anfang und das Ende der Lücke.

Aus Geschwindigkeit und Zeit wird folglich die Länge der Lücke ermittelt. Sollte diese größer sein als die Eigenlänge bzw. Eigenbreite des Fahrzeugs (zzgl. einer Toleranz von 10 cm), erkennt das Fahrzeug diese Lücke als Parklücke an und stoppt zunächst seine Bewegung nach vorn.

*2) Einparken:* Unmittelbar nach der Parklückenerkennung wird mit dem gewählten Einparkvorgang (längs bzw. quer) begonnen. Der Algorithmus des Querparkens ist in Abbildung 3 dargestellt. Im Folgenden wird nur der Ablauf des Längsparkens erläutert.

Hier entscheidet das Fahrzeug zwischen drei unterschiedlichen Parklücken. Alle Einparkvorgänge beginnen mit dem Rückwärtsfahren auf vollem Lenkeinschlag nach rechts. Sobald der Farbsensor die rote Markierung erkennt, erfolgt der Volleinschlag der Lenkung nach links. Für die beiden kleinen Parklücken wird der hintere Ultraschallsensor verwendet, um soweit wie möglich nach hinten zu fahren und zu stoppen, bevor die hintere Wand erreicht wird. Sollte die Parklücke sehr eng sein, wird noch ein Zug zum Korrigieren benötigt. Dabei wird erneut rechts eingeschlagen und vorwärts gefahren. Damit das Fahrzeug zum Schluss ordnungsgemäß in der Lücke steht, werden die beiden Ultraschallsensoren an Front und Heck verwendet. Diese werten die Abstände nach vorn und hinten aus und es wird gegebenenfalls vor- bzw. zurückgefahren, bis die beiden Abstände übereinstimmen. Bei großen Parklücken wird der vordere Ultraschallsensor genutzt, um die Bewegung zu stoppen, sobald das Fahrzeug gerade in der Lücke steht. Letzten Endes wird so weit wie möglich vorgefahren und der Einparkvorgang ist beendet.

*3) Ausparken:* Das Ausparken erfolgt auf Basis der Einparkvorgänge, deren Abläufe umgekehrt und teilweise spezifiziert sind. Auch hier werden die Farbmarkierungen genutzt, um den genauen Zeitpunkt der Lenkeinschläge zu bestimmen. Das Fahrzeug wurde so programmiert, dass es seine momentane Parksituation selbstständig erkennt und so den korrekten Ausparkvorgang autonom einleitet. Die Unterscheidung der Fälle erfolgt durch Auswertung von Abständen, die mittels Ultraschallsensoren an Front und Heck gemessen werden.

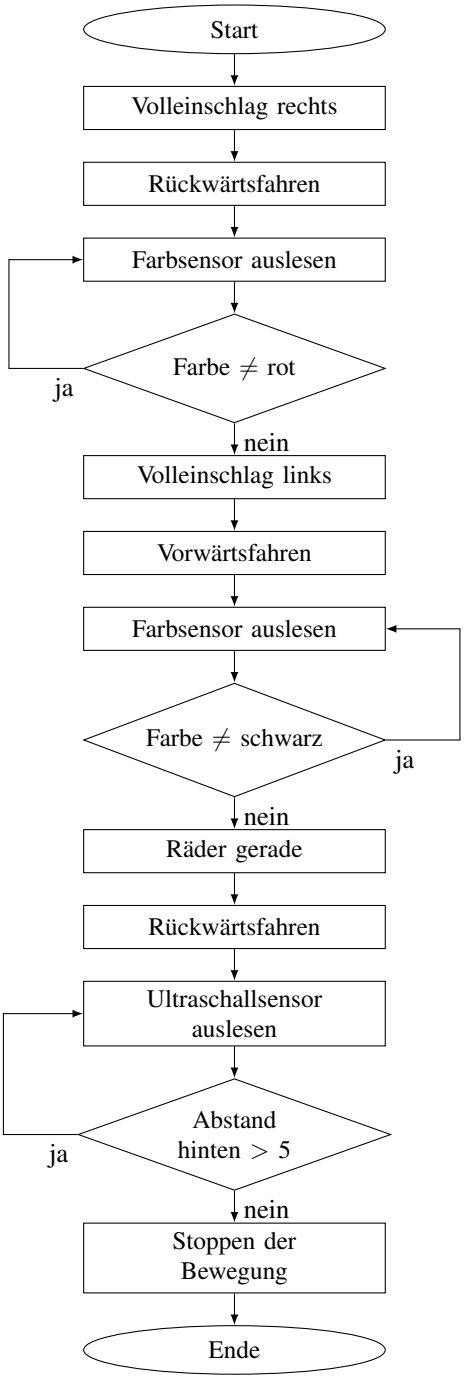


Abbildung 3. Programmablaufplan des Einparkalgorithmus zum Quereinparken

### C. Grafische Benutzeroberfläche

Zur Interaktion des Fahrzeuges mit seiner Umwelt wurde eine grafische Benutzeroberfläche (GUI, englisch: graphical user interface) entwickelt, die es dem Anwender ermöglicht, durch einfache Bedienung mit dem Fahrzeug zu kommunizieren. Durch die Anordnung zweier Knöpfe auf der Bedienoberfläche kann der Nutzer entscheiden, ob er das Fahrzeug ein- oder ausparken möchte. Soll das Fahrzeug eine Parklücke suchen und einparken, ist es mittels eines PopUp-Menüs möglich, zwischen einem der beiden Einparkvorgänge auszuwählen. Für den Ausparkvorgang muss allerdings keine Auswahl im PopUp-

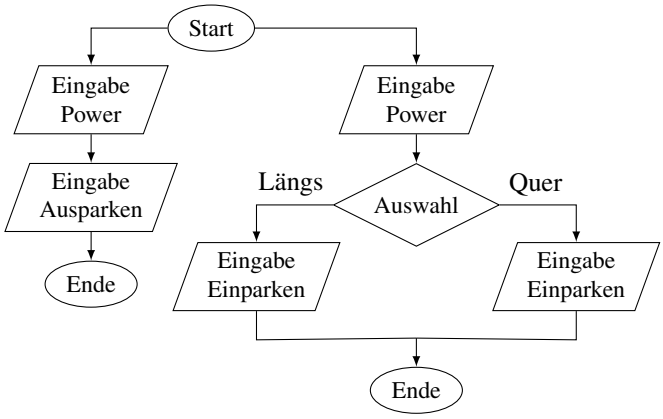


Abbildung 4. Programmablaufplan der grafischen Benutzeroberfläche

Menü getroffen werden, da das Fahrzeug selbstständig erkennt in welcher Parklücke es sich augenblicklich befindet. Durch einen Schieberegler kann zudem noch die Geschwindigkeit eingestellt werden, mit der das Auto eine Parklücke sucht. Hierbei ist darauf zu achten, dass sowohl das eigentliche Ein- als auch Ausparken mit halber Geschwindigkeit ablaufen. Zum Schluss muss noch der entsprechende Knopf zum Ein- oder Ausparken betätigt werden und das Fahrzeug beginnt mit dem gewählten Ablauf. Der Programmablauf der Benutzeroberfläche ist in Abbildung 4 dargestellt.

## IV. Ergebnisdiskussion

Das Ergebnis des Projekts ist ein Roboter, der eigenständig Parklücken erkennt und autonom ein- sowie ausparkt. Die Ermittlung der Größe der Parklücke funktioniert mit einer geringen Abweichung von $1\,\mathrm{cm}$ bis $2\,\mathrm{cm}$ sehr genau und wurde daher ohne weitere Verbesserungen umgesetzt. Die Parkvorgänge stellen allerdings nur eine Simulation des eingangs erwähnten Parkassistenten dar. Die Gründe dafür werden im Folgenden aufgezeigt.

Aufgrund der teilweise instabilen Legoteile erwies sich die Befestigung der Räder als sehr schwierig und die Lenkung bekam somit zu viel Bewegungsfreiraum. Des Weiteren zeigten sich die Ultraschallsensoren in der Abstandsmessung ziemlich ungenau und ermittelten keine Abstände unter $5\,\mathrm{cm}$. Diese beiden Faktoren führten dazu, dass ein Algorithmus, der den Roboter auf einem gleichmäßigen Abstand zur Wand hält, nicht umsetzbar war. Ein gerades Vorwärtsfahren des Fahrzeuges konnte daher nicht realisiert werden. Der Roboter beginnt jedes erneute Parken mit einer unterschiedlichen, seitlichen Entfernung zur Wand, was dazu führt, dass er manchmal schief in der Parklücke steht oder beim Parken sogar an der Wand anstößt. Außerdem besteht das Problem, dass die Parkvorgänge immer noch an einen speziellen Modellaufbau gebunden sind. Es werden Farbmarkierungen benötigt, die den richtigen Zeitpunkt zum Lenkeinschlag signalisieren und mit festen Abständen am Boden befestigt sind. Dementsprechend ist das Ergebnis eine unter den Möglichkeiten von Lego Mindstorms sinnvolle Lösung, die aber nicht der Realität entspricht.

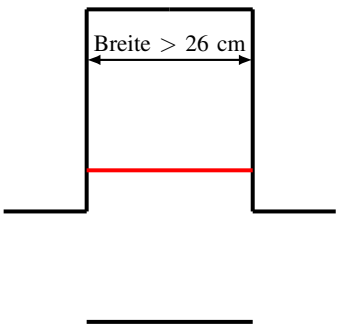


Abbildung 5. Darstellung einer Parklücke zum Quereinparken inkl. der benötigten Farbmarkierungen

## V. Zusammenfassung und Fazit

Das anfangs erwähnte Ziel, einen Roboter zu konstruieren, der eine geeignete Parklücke erkennt und autonom ein- sowie ausparkt, wurde vollständig in die Realität umgesetzt. Dabei wurden neben zwei Motoren für Lenkung und Antrieb vier Sensoren genutzt, die die Umgebung des Fahrzeuges scannen. Drei Ultraschallsensoren sind für Abstandsmessungen verantwortlich, wobei ein seitlich angebrachter Sensor zur Vermessung der Parklücke dient. Zudem wird ein Farbsensor verwendet, um den exakten Zeitpunkt der Lenkbewegungen zu erfassen.

Zur zukünftigen Entwicklung gehören die Verbesserung der genannten Probleme und weitere Optimierungen. So ließe sich zusätzlich noch das Vorwärtsparken, sowohl parallel als auch quer zur Fahrbahn, realisieren. Weiterhin muss speziell beim Ausparken auf sich nahende Hindernisse, wie z. B. den fließenden Verkehr, geachtet werden. Das Fahrzeug wäre dementsprechend so zu programmieren, dass seine momentane Bewegung unterbrochen wird, wenn Hindernisse erkennbar sind.

## Anhang

Die Abbildung 5 zeigt die Gestaltung einer Parklücke zum Quereinparken inklusive der benötigten Farbmarkierungen zur Lenkzeitpunkterkennung. Des Weiteren wurden im Rahmen der Abschlusspräsentationen des Projektseminars Kurzfilme der diesjährigen Projekte erstellt. Die Playlist [5] enthält mehrere Videos, die den Ablauf der Parkvorgänge darstellen.

## Literaturverzeichnis

[1] *Einparken leicht gemacht: ein Jahrzehnt „Park Assist*. https://www.volkswagen-newsroom.com/de/pressemitteilungen/einparken-leicht-gemacht-ein-jahrzehnt-park-assist-1574. Version: 5/3/2020

[2] Volkswagen Service Training VSQ-1: SSP 389 Der Parklenkassistent. (2007). https://www.motor-talk.de/forum/aktion/Attachment.html?attachmentId=677825

[3] *Fahrerlos geparkt. Automated Valet Parking*. https://www.daimler.com/innovation/case/autonomous/fahrerlos-geparkt.html. Version: 5/3/2020

[4] *VolksBot*. https://www.volksbot.de/ackermann-de.php. Version: 1/5/2020

[5] Magdowski, Mathias: *Lego-Praktikum 2020*. https://www.youtube.com/playlist?list=PLWCaO6Bpqy-5xCwNJblRE4hnuXa2PTu0M. Version: 5/3/2020