Die approbierte Originalversion dieser Diplom-/Masterarbeit ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.
http://www.ub.tuwien.ac.at

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology.
http://www.ub.tuwien.ac.at/eng

DIPLOMARBEIT

GPS-gestützte Routenverfolgung mit Odometrie und Fahrbahnranderkennung

ausgeführt zur Erlangung des akademischen Grades
eines Diplom-Ingenieurs unter der Leitung von

Univ.Prof. Dipl.-Ing. Dr.techn. Hermann Kaindl
Univ.Ass. Dipl.-Ing. BSc Ralph Hoch

am

Institut für Computertechnik (E384)
der Technischen Universität Wien

durch

Thomas Karner, BSc.
Matr.Nr. 0925228
Otto-Herschmanngasse 4/2/15
1110 Wien

Wien, am 6. Juni 2018 _____

Kurzfassung

Diese Arbeit präsentiert einen Ansatz zur GPS-gestützten Steuerung eines Modellfahrzeugs im Maßstab 1:10. Das Fahrzeug kann einer zuvor angefertigten Route folgen und durch die implementierte Odometrie auch nach Ausfallen des GPS Signals die eigene Position bestimmen. Zur Berechnung aller nötigen Algorithmen wird ein Raspberry Pi 3 in Verbindung mit einem Arduino Uno, welcher alle Echtzeitanwendungen übernimmt, eingesetzt.

Die Odometrie des Fahrzeugs basiert auf einer konstant geregelten Geschwindigkeit und der Orientierungsinformation der Fahrtrichtung durch ein Magnetometer. Sie dient vor allem zur Überbrückung der Zeitspanne zwischen den Positionsdaten des verwendeten GPS-Moduls, kann aber auch längere Zeitspannen kompensieren.

Um etwaige Ungenauigkeiten der GPS-Route zu korrigieren ist das Modellauto zusätzlich mit einem Kamerasystem ausgestattet. Das Kamerasystem dient zur Fahrbahnranderkennung, welche mittels Canny Edge Detection zur Kantenerkennung und Hough Transformation zur Linienfindung realisiert wurde.

Abstract

This thesis presents an approach to GPS-based control of a model vehicle with the scale of 1:10. The vehicle can follow a previously prepared route and determine its own position by the implemented odometry even after failure of the GPS signal. A Raspberry Pi 3 is used to execute all algorithms necessary and all real-time applications are handled by an Arduino Uno.

The Odometry is based on the constant speed provided by a implemented speed control and the orientation given the alignment to the global magnetic field measured by a magnetometer. The Odometry is used to fill the time gap between the positioning data given by a GPS-module, but can compensate longer time gaps aswell.

The model car is also equipped with a camera system to compensate for any inaccuracies of the previously prepared route. This camera system is basically used for lane detection, which was implemented with canny edge detection and Hough transformation for line detection.

Danksagung

An dieser Stelle möchte ich mich bei meiner Familie, meiner Freundin und meinen Freunden für die aufgebrachte Geduld und Unterstützung während meines Studiums und der Erstellung dieser Arbeit bedanken. Sie haben es geschafft mich in schwierigen Phasen zu motivieren und mir gleichzeitig einen guten Ausgleich zu bieten.

Ein weiterer Dank gilt meinem Betreuer Univ.Ass. Dipl.-Ing. BSc Ralph Hoch, der mir stets mit Rat und Tat bei der Erstellung dieser Arbeit zur Seite stand.

Abschließend möchte ich mich bei Herrn Stefan Müllner von Hutchinson Drei Österreich GmbH für die Leihgabe eines mobilen Routers bedanken.

Inhaltsverzeichnis

Abkürzungen

CAN	Controller Area Network
CSI	Camera Serial Interface
GPS	Global Positioning System
HSI	Hue, Saturation, Intensity (Farbwert, Sättigung, Intensität)
I^2C	Inter-Integrated Circuit
NMEA	National Marine Electronics Association
OpenCV	Open Computer Vision
PID	Proportional Integral Differential
PKW	Personenkraftwagen
PWM	Pulsweitenmodulation
RANSAC	Random Sample Consensus
SSH	Secure Shell
UART	Universal Asynchronous Receiver Transmitter
WGS84	World Geodetic System 1984

1 Einleitung

Fahrassistenzsysteme für Kraftfahrzeuge werden nicht nur zur Sicherheit der Insassen, sondern auch zur Erhöhung des Fahrkomforts entwickelt. Manche dieser Assistenten, wie das Antiblockiersystem (ABS) oder das elektronische Stabilitätsprogramm (ESP), kommen erst in Notsituationen zum Einsatz, wohingegen die Antriebsschlupfregelung (ASR) die Traktion beim Anfahren mit zu viel Gas oder auf schlechtem Untergrund verbessert. Das Gegenteil hierzu ist die Motor-Schleppmoment-Regelung (MSR) als Teil des ESP, die ein abruptes Einsetzen der Motorbremswirkung, die die Stabilität des Fahrverhaltens negativ beeinflussen würde, verhindert.

Andere Systeme können vom Fahrer je nach Bedarf aktiviert bzw. deaktiviert werden. Ein Beispiel hierfür ist die Geschwindigkeitsregelanlage, auch Tempomat genannt, die die Geschwindigkeit des Fahrzeuges je nach Möglichkeit auf dem vom Fahrer vorgegebenen Wert hält. Eine Weiterentwicklung ist der Abstandsregeltempomat, der als Rückführ- bzw. Stellgröße den Abstand zum vorausfahrenden Fahrzeug miteinbezieht.

Weitere Assistenzsysteme greifen zwar nicht direkt in das Fahrverhalten des Fahrzeugs ein, helfen aber dem Fahrer sich besser auf das Fahrgeschehen zu konzentrieren. Als Beispiele hierfür seien die Scheibenwischautomatik, die Leuchtweitenregulierung und der Fernlichtassistent erwähnt. Falls sich der Fahrer durch Müdigkeit nicht mehr ausreichend dem Fahrgeschehen widmen kann, kann er durch einen Aufmerksamkeitsassistent zur Müdigkeitserkennung darauf aufmerksam gemacht werden.

1.1 Motivation

Die bisher genannten Assistenten können den Fahrer entweder durch das Eingreifen in den Bremsvorgang, die Regelung der Motordrehzahl oder andere Bedienelemente, die das Fahrverhalten nicht beeinflussen, unterstützen. Die Lenkung wird, abgesehen von der bisher noch nicht erwähnten Servolenkung, nicht aktiv beeinflusst.

Assistenzsysteme, die aktiv in die Lenkung eingreifen, um die Trajektorie des Fahrzeugs zu beeinflussen, werden ebenfalls bereits eingesetzt. Angefangen von Parkassistenzsystemen, bei denen das Fahrzeug die Lenkung übernimmt und der Fahrer nur das Gaspedal bedienen muss, bis hin zu vollautomatischen Parkassistenten, bei denen das Fahrzeug die komplette Kontrolle während des Parkvorgangs übernimmt.

In neueren Fahrzeugen kommen auch schon optische Sensoren zur Fahrspurerkennung zum Einsatz. Diese werden zur grundlegenden Fahrspurerkennung bis hin zum Spurhalteassistent eingesetzt, wobei letzterer auch Eingriffe in die Lenkung vornehmen kann.

Diese Vielzahl an Assistenzsystemen sind aber nur als Assistenten des Fahrers gedacht und nicht um den Fahrer als solchen zu ersetzen. Dieser Schritt wird erst durch Autopilotsysteme und autonome Fahrmodi realisiert.

Autopilotsysteme und autonome Fahrmodi von Kraftfahrzeugen sind ein wachsender Bereich der mobilen Fortbewegung. Dadurch werden zahlreiche Möglichkeiten geschaffen. Der Verkehrsfluss könnte nicht nur effizienter und zunehmend sicherer gestaltet werden, sondern auch die Art der Mobilität würde sich nachhaltig verändern. Zusätzlich würden viele Menschen, die aus unterschiedlichen Gründen nicht in der Lage sind einen PKW zu steuern, diesen Teil ihrer verlorenen Möglichkeiten zurückgewinnen.

Aus diesen Gründen beschäftigt sich diese Arbeit mit der Entwicklung eines Systems zur GPS-gestützten Routenverfolgung mit Odometrie und Fahrbahnranderkennung auf Basis eines Modellautos.

1.2 Problemstellung

In dieser Arbeit wird ein elektrisches Modellauto so aufgebaut, dass ein teilautonomer Fahrbetrieb möglich ist. Dieser Fahrbetrieb fährt eine vorgegebene GPS-Route und darf dabei einen als Fahrbahn markierten Bereich nicht verlassen. Die Route soll vom Fahrzeug autonom anhand der GPS-Daten des Fahrzeugs abgefahren werden, wobei auch im Falle einer größeren Abweichung der GPS-Daten die Fahrbahnränder nicht überfahren werden sollen.

Im Falle eines Ausfalls der GPS-Positionsbestimmung soll sichergestellt sein, dass die Position des Fahrzeugs weiterhin durch Odometrie bestimmt werden kann. Hierfür musste ein mathematisches Modell erstellt werden, anhand dessen die Position mittels der Bewegungsdaten des Fahrzeuges bestimmt wird.

1.3 Lösungsansatz

Autonome Steuerungsansätze für Fahrzeuge verwenden häufig Fahrbahnerkennungsalgorithmen zur Orientierung. Gemeinsam mit einer Fahrzeugsteuerung kann dadurch eine Route entlang der Fahrbahn automatisiert abgefahren werden. Durch die zunehmende Genauigkeit dieser Navigationssysteme, bezogen auf die Richtigkeit und Präzision des Kartenmaterials, kann die Herangehensweise durch eine Steuerung basierend auf GPS-Koordinaten geändert werden.

Der Ansatz, der in dieser Arbeit verfolgt wird, basiert auf der Steuerung des Fahrzeuges entlang einer vorgefertigten GPS-Route. Hierbei dienen die GPS-Koordinaten, aus denen sich die GPS-Route zusammensetzt, als Grundlage für die Steuerung. Die Routenpunkte entlang einer Route werden dabei so aneinandergereiht, dass komplexe Straßenverläufe zu Polygonzügen werden. Dabei kann die Teilstrecke zwischen zwei Routenpunkten auf gerader Strecke zurückgelegt werden, ohne die Fahrbahn zu verlassen. Falls jedoch die Genauigkeit der GPS-Standortbestimmung, bezogen auf die Richtigkeit, sinkt, sichert ein Kamerasystem das Fahrzeug zusätzlich gegen das Verlassen der Fahrbahn ab.

Das Fahrzeug wird in seiner Orientierung anhand des Erdmagnetfeldes so ausgerichtet, dass es den nächsten Punkt auf kürzestem Weg erreicht. Falls dieser Pfad eine Fahrbahnbegrenzung kreuzt, wird die Orientierung des Fahrzeugs entsprechend geändert. Die Umfahrung bis zum aktuellen Routenpunkt wird solange fortgesetzt, bis sich die Fahrbahngrenze nicht mehr auf dem Pfad befindet. Sollte dadurch die Erreichbarkeit des Routenpunktes beeinflusst werden, dann wird, sobald die Differenz der Orientierung des Fahrzeuges und der gewünschten Orientierung einen bestimmten Schwellwert übersteigt, der nächste Routenpunkt angesteuert.

Zur Absicherung gegen eine zu langsame oder ausfallende Standortaktualisierung wurde ein Algorithmus implementiert, der den aktuellen GPS-Standort des Fahrzeugs durch Odometrie extrapoliert.

1.4 Aufbau

Diese Arbeit ist so aufgebaut, dass in den Kapiteln 2 und 3 zunächst die zugrunde liegende Theorie der verwendeten Methoden erläutert und danach ein Einblick in ähnliche Aufgabenstellungen in der Literatur gegeben wird.

Ab Kapitel 4 werden der Aufbau des Modellautos sowie die implementierten Regelalgorithmen dargestellt, um das Konzept der hier verwendeten Routenführung erklären zu können.

Die Routenführung erschließt sich über die Kapitel 5 und 6 und wird iterativ aufgebaut, um bei einer reinen GPS-gestützten Routenführung zu beginnen und bei einer Routenführung mit Kameraunterstützung und Odometrie zu enden.

Die Evaluierung des während dieser Arbeit entstandenen Modellautos ist in Kapitel 7 beschrieben.

Abschließend werden in Kapitel 8 die Erkenntnisse dieser Arbeit zusammengefasst.

2 Background und Theorie

Dieses Kapitel beschreibt die Sensoren, Methoden und die zugrundeliegende Theorie, die für diese Arbeit relevant sind.

2.1 Sensoren

In diesem Abschnitt wird auf die Funktion der Sensorik hinter der Positionsbestimmung und der Orientierung des Modellautos eingegangen.

2.1.1 Positionsbestimmung

Zur Positionsbestimmung wird ein GPS-Modul mit einem U-blox NEO-6M Chipsatz eingesetzt. Es gibt alle Informationen, unter anderem auch die Positionsdaten, im NMEA 0183 Standard[1] über dessen UART-Schnittstelle aus. Neben den Positionsdaten in Form von Längen- und Breitengraden sind noch folgende Informationen erwähnenswert:

- Anzahl der Satelliten, anhand derer die aktuelle Position bestimmt wurde, sowie deren Signalstärke
- Geschwindigkeit der Empfangsantenne in Kilometer pro Stunde sowie in Knoten
- Seehöhe der Empfangsantenne
- Abweichung zum geodätischen Referenzsystem des WGS-84-Ellipsoid

2.1.2 Orientierung

Als digitaler Kompass wird ein Magnetometer mit dem HMC5883L Chipsatz verwendet, welches die magnetische Flussdichte in drei Achsen messen kann. Es verfügt über einen Einzel- sowie einen Dauermessmodus und eine individuell einstellbare Genauigkeit und Messfrequenz. Die Messwerte werden immer in einer 12 Bit Auflösung zur Verfügung gestellt und der Messbereich bei diesem Sensor erstreckt sich von Milligauß bis zu acht Gauß[2].

[1] Der NMEA 0183 Standard ist in [4] verfügbar

[2] Gauß (Gs) ist die Einheit der magnetischen Flussdichte im CGS-System, wobei 1Gs $\hat{=}10^{-4}$mT

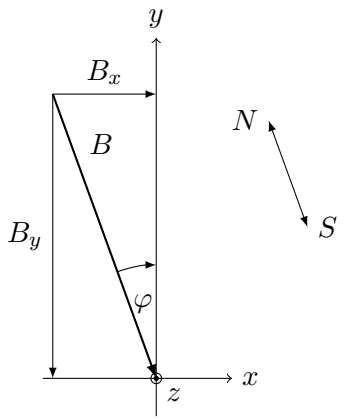


Abbildung 2.1: Geometrische Darstellung der gemessenen magnetischen Flussdichten

Zur Bestimmung der magnetischen Orientierung, ähnlich einem Kompass, müssen nicht nur die Magnetfelddaten für alle Achsen bekannt sein, sondern auch die räumliche Lage des Magnetometers zum Zeitpunkt der Messung. Da aber das Magentometer, bedingt durch die Art der Befestigung am Modellfahrzeug, nur um die z-Achse gedreht werden kann, reichen die Messwerte der x- und y-Achse aus um die Orientierung φ gemäß Abbildung 2.1 zu bestimmen. Das Magnetometer ist in diesem Fall so ausgerichtet, dass die y-Achse parallel zur Längsachse des Modellautos ist.

Jedoch sind alle Magnetfeldmessungen durch externe Einflüsse mit einem Offset behaftet, der vor der Verwendung der Daten kompensiert werden muss. Der Vorgang zur Ermittlung dieser Offsetwerte beinhaltet mindestens eine vollständige Umdrehung des Magnetometers um die z-Achse, während laufend Messwerte der beiden anderen Achsen aufgenommen werden. Aus diesen Daten wird je Achse der minimale und maximale Messwert ermittelt, denn der Mittelwert dieser beiden Werte entspricht dem Offset dieser Achse. Um die Genauigkeit der aus den Messwerten berechneten Orientierung zu verbessern, sollte diese Kalibrierung zu Beginn jeder Inbetriebnahme des Sensors durchgeführt werden.

2.2 Bildverarbeitung

Im Nachfolgenden wird die Theorie der verwendeten Bildverarbeitungswerkzeuge und -methoden erklärt.

2.2.1 OpenCV

OpenCV [8] ist eine frei zugängliche Programmbibliothek, die vor allem die Implementierung von Algorithmen für die Bildverarbeitung und Analyse zur Verfügung stellt. Die Bilder werden als Pixelmatrix dargestellt, deren Einträge die Farbwerte des jeweiligen Pixels repräsentieren. Die in dieser Arbeit verwendeten Algorithmen werden in den nachfolgenden Unterabschnitten kurz beschrieben.

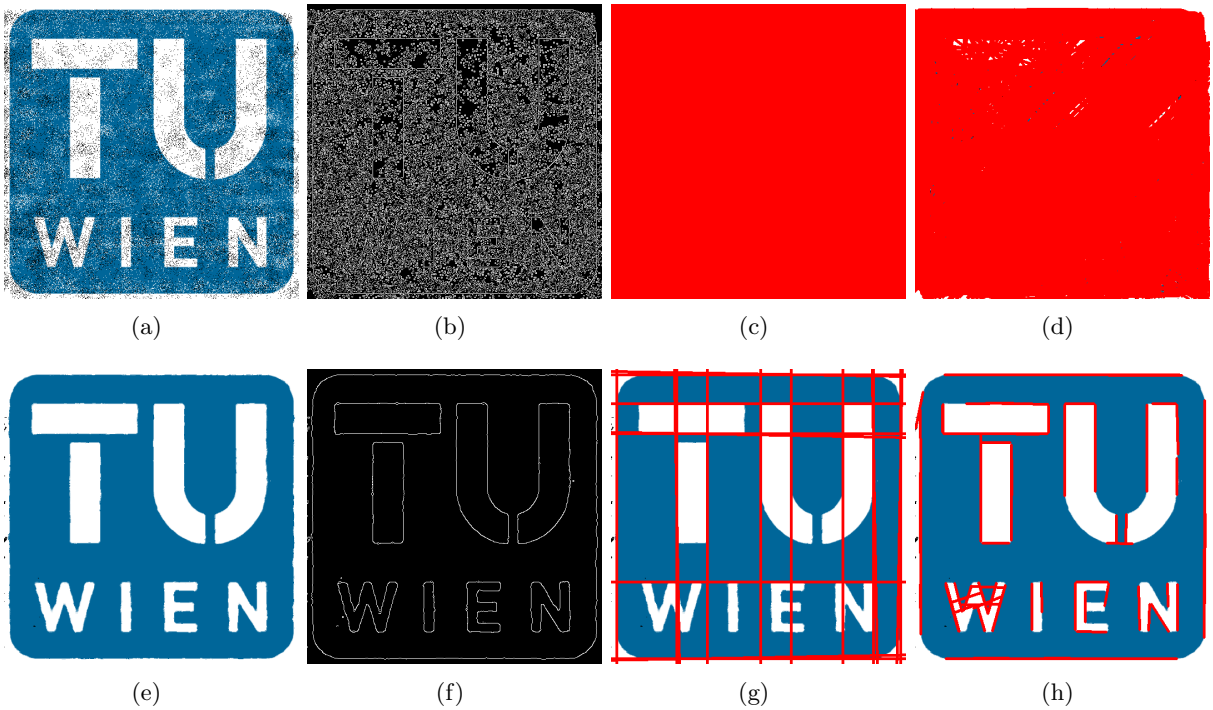


Abbildung 2.2: Beispiele der einzelnen Bildbearbeitungsverfahren
(a) Originalbild, (b) Canny Edge Detection des Originalbildes, (c) Hough Transformation des Originalbildes, (d) Progressive Probabilistic Hough Transformation des Originalbildes, (e) Blurring des Originalbildes, (f) Canny Edge Detection des geblurrten Bildes, (g) Hough Transformation des geblurrten Bildes, (h) Progressive Probabilistic Hough Transformation des geblurrten Bildes

2.2.2 Blurring

Unter Blurring oder auch Weichzeichnen versteht man das bewusste Verringern des Kontrasts einer Aufnahme. Ein wesentlicher Vorteil, den Blurring mit sich bringt ist, dass man Störungen des Bildes kompensieren kann und so unerwünschten Effekten in den folgenden beiden Schritten vorbeugen kann.

Das in dieser Arbeit verwendete Blurringverfahren ist ein Medianfilter, ein sogenannter Rangordnungsfilter, bei dem jedes Pixel gemeinsam mit einer zuvor bestimmten Anzahl an umliegenden Nachbarpixel betrachtet wird. Die einzelnen Pixel werden nach deren Farbwert sortiert, wobei anschließend der mittlere Wert dem betrachteten Pixel zugeordnet wird. In Abbildung 2.2(e) ist die weichgezeichnete Darstellung von Abbildung 2.2(a), welche eine deutliche Verbesserung bezüglich der Bildstörungen im Originalbild zeigt.

Weitere Blurringfilter, basierend auf einem Rangordnungsfilter, sind ein Minimum- und Maximumfilter. Bei diesen Filtern werden die benachbarten Pixel ebenfalls nach ihrem jeweiligen Farbwert sortiert, aber im Gegensatz zum Medianfilter wird das betrachtete Pixel dann mit dem minimalen bzw. maximalen Farbwert seiner Nachbarn überschrieben. Eine Alternative zu den Rangordnungsfiltern wäre ein Gaußfilter, bei dem die benachbarten und das momentan betrachtete Pixel mit einer glockenkurvenartigen Verteilung gewichtet werden, um dann den Farbwert des aktuellen Pixels neu zu berechnen.

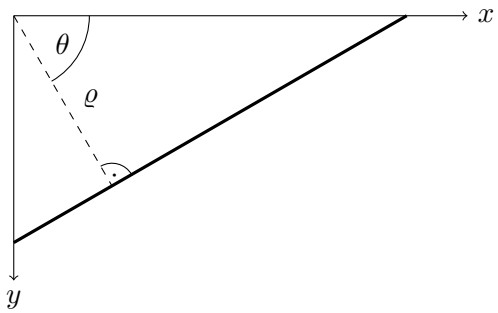


Abbildung 2.3: Parameterdarstellung der Hough Transformation

Die genaue Verwendung dieser Funktion mit OpenCV kann in [9] nachgelesen werden.

2.2.3 Canny Edge Detection

Bei der Canny Edge Detection [Can86] handelt es sich um einen Kantenerkennungsalgorithmus basierend auf der Kontrastveränderung zwischen einzelnen Pixel. Der Algorithmus ermittelt den Farbgradientenverlauf über das gesamte Bild, der im nächsten Schritt mit einem Schwellwertfilter ausgewertet wird. Je nach Über- oder Unterschreiten des Schwellwerts wird das aktuell betrachtete Bildpunkt mit dem logischen Maximum oder Minimum des Farbwerts überschrieben. Das Resultat ist ein schwarzes Bild, bei dem alle detektierten Kantenpixel weiß sind.

Ein Beispiel zur Veranschaulichung ist in Abbildung 2.2 dargestellt, wobei die Kantenbilder von Abbildung 2.2(b) und 2.2(f) jeweils von Abbildung 2.2(a) und 2.2(e) erstellt wurden. Um falschen Detektionen durch Bildstörungen wie in Abbildung 2.2(a) vorzubeugen, ist es ratsam, vor der Kantenerkennung ein Blurringverfahren wie aus Abschnitt 2.2.2 anzuwenden.

Die genaue Verwendung dieser Funktion mit OpenCV kann in [6] nachgelesen werden.

2.2.4 Hough Transformation

Der Hough Transformation Algorithmus [HP62] dient dazu, nach erfolgreicher Kantenerkennung (siehe Abschnitt 2.2.3) Linien im Bild zu finden.

Das Prinzip dieser Linienfindung besteht darin, dass eine definierte Linienschar vorgegeben wird, die sich in diesem Bild befinden darf. Diese Linien werden mit der Anzahl der auf ihr liegenden Kantenpunkte gewichtet und durch einen Schwellwertfilter als Linien detektiert. Die Linienschar wird, wie in Abbildung 2.3 gezeigt, durch den Normalabstand ϱ zum Nullpunkt und den Winkel θ des Normalabstandes mittels

$$\varrho = x \cos\theta + y \sin\theta \tag{2.1}$$

definiert, wobei die Einheiten von ϱ Pixel und von θ Grad sind.

Da der Algorithmus alle Kombinationen dieser beiden Komponenten für das betreffende Bild testet, ist es aus Gründen der Performance sinnvoll, geeignete Schrittweiten beider Komponenten zu wählen.

2.2.4.1 Probabilistic Hough Transformation

Die Probabilistic Hough Transformation ist eine Weiterentwicklung der Hough Transformation, die nicht alle Punkte der vorangegangenen Kantenerkennung in Betracht zieht, sondern nur eine zufällig gewählte Untermenge. Der Vorteil dieser Methode besteht darin, dass bei verbesserter Performance in Bezug auf den nötigen Berechnungsaufwand gleich gute Ergebnisse wie mit herkömmlicher Hough Transformation erzielt werden können.

Progressive Probabilistic Hough Transformation

Aufbauend auf der Probabilistic Hough Transformation wurde in [MGK00] die Progressive Probabilistic Hough Transformation entwickelt, welche ebenfalls in OpenCV implementiert ist. Die Implementierung in OpenCV bringt gegenüber der Implementierung der herkömmlichen Hough Transformation mit sich, dass nicht die Werte für ϱ und θ zurückgegeben werden, sondern die Start- und Endpunkte der erkannten Linien.

Der direkte Vergleich der Implementierung der Hough Transformation zu jener der Progressive Probabilistic Hough Transformation ist in den Abbildungen 2.2(c) und 2.2(d) sowie 2.2(g) und 2.2(h) dargestellt. Hierbei ist vor allem in den Abbildungen 2.2(c) und 2.2(d) zu erkennen, wie sich ungefiltertes Rauschen auf die Linienfindung auswirkt.

Anhand des Vergleichs aus Abbildung 2.2, speziell in den Abbildungen 2.2(c), 2.2(d), 2.2(g) und 2.2(h), in denen die detektierten Linien der unterschiedlichen Hough Transformationen eingezeichnet sind, wird in Abschnitt 6.1 erkennbar sein, warum die Methode der Progressive Probabilistic Hough Transformation für die Problemstellung besser geeignet ist als jene der herkömmlichen Hough Transformation. Grundsätzlich ist hierbei aber der Unterschied der Implementierung der beiden Methoden in OpenCV ausschlaggebend. Bei der Methode der Progressive Probabilistic Hough Transformation werden nämlich die Endpunkte der erkannten Linien als deren Parameter ausgegeben.

Die genaue Verwendung dieser OpenCV Funktionen kann in [7] nachgelesen werden.

3 Bisherige Arbeiten in der Literatur

In diesem Kapitel werden Arbeiten mit ähnlichen Aufgabenstellungen aus der Literatur vorgestellt und deren Ansätze beleuchtet.

3.1 Odometrie von Fahrzeugen

In der Arbeit [MDVS14] wird das Problem behandelt, dass die Positionsbestimmung durch GPS gerade im innerstädtischen Bereich ungenau sein kann. Der Ansatz ist die Position des Fahrzeugs, in diesem Fall eines PKW, anhand der Drehzahl der Reifen einer Achse zu bestimmen. Die Drehzahlen der einzelnen Reifen werden hierbei durch den CAN Bus ausgelesen. Durch die Differenz dieser Drehzahlen wurde die gefahrene Winkelgeschwindigkeit des Fahrzeugs berechnet. Das Resultat war, dass sie auf einer gefahrenen Strecke von etwa 12 km eine Abweichung von 5 m hatten.

Die Arbeit [SPAJ15] versucht anhand bereits verfügbarer Sensoren im Fahrzeug die Position zu berechnen. Als Testfahrzeug wurde ein Mitsubishi i-MiEV verwendet. Die Autoren weisen zunächst darauf hin, dass man zwar mit den verfügbaren Sensoren eine Positionsbestimmung durchführen kann, diese ihren Anforderungen aber nicht gerecht wird und sie somit die Daten von zwei Sensoren, welche über den CAN Bus ausgelesen werden, fusionieren. Das Resultat dieser Arbeit ist, dass empfohlen wird, bei unterschiedlichen Geschwindigkeiten auch die Daten verschiedener Sensoren zu verwenden.

Zur Odometrie wurde in [MSR16] ein optimaler Zustandsschätzer in Verbindung mit zwei Kameras und dem Einspurmodell nach Ackermann verwendet. Die beiden Kameras wurden mit Blickrichtung auf den Untergrund auf beiden Seiten des fahrzeugähnlichen Roboters montiert. Dadurch ergab sich die Möglichkeit über den optischen Fluss des Kamerabildes die jeweilige Geschwindigkeit und somit durch die Differenzgeschwindigkeit die Winkelgeschwindigkeit des Robotors zu errechnen.

In [NVRS09] wurde zur Odometrie nur eine Kamera mit Blickrichtung auf den Untergrund montiert. Hierbei wurde dann durch den optischen Fluss des Kamerabildes die Positionsänderung bestimmt. Zur Evaluierung wurde das System mit herkömmlicher radbasierter Odometrie verglichen und eine Verbesserung bezüglich der errechneten Position sowie Orientierung gezeigt.

3.2 Fahrbahnranderkennung

Die Fahrspurerkennung von unbemannten Fahrzeugen wurde in der Arbeit [WMKW14] mittels „Inverse Perspective Mapping“ durchgeführt. Dazu wird zuerst ein binäres Bild mit einer Schwellwertmethode, die den optimalen Schwellwert des gesamten Bildes verwendet, erstellt, welches auf den relevanten Bildbereich zugeschnitten wird. Dann wird mittels „Inverse Perspective Mapping“ das Bild in die Vogelperspektive transformiert. Durch „k-means Clustering“ werden die Fahrbahnmarkierungen gruppiert und angepasst. Der entwickelte Algorithmus ist den Autoren zufolge gegenüber Störeungen nicht anfällig.

In der Arbeit [AKIK08] wurde dem Ansatz der Kantenerkennung mit dem „Canny Edge Detection“ Algorithmus und einer Hough Transformation nachgegangen. Dieser Ansatz erwies sich für gerade Strecken und auch leichte Kurven als robust und schnell genug für Echtzeitanwendungen. Einige Probleme durch Schatten und den Horizont blieben aber bestehen.

In [STC06] gingen die Autoren dem Ansatz der Fahrstreifenerkennung über das HSI-Farbmodell nach. Dafür wurde die Intensitätsverteilung der Pixelreihen im relevanten Bildausschnitt betrachtet und entsprechend gruppiert. Durch weitere Schwellwerte im Bezug auf Sättigung und Intensität konnten die Bodenmarkierungen erkannt werden. Der Algorithmus funktioniert für Bodenmarkierungen unterschiedlicher Farben und auch bei Nacht.

Die Arbeit [BHS09] verwendet zur Fahrspurerkennung ebenfalls „Inverse Perspective Mapping“, sowie RANSAC zur Unterdrückung von Störungen durch Bildrauschen. Zusätzlich wird zur Glättung der Spurverfolgung ein Kalman-Filter verwendet.

4 Aufbau des Gesamtsystems

Dieses Kapitel widmet sich dem Aufbau des Gesamtsystems, der in drei Teilbereiche gegliedert ist. Die Gliederung umfasst hierbei den logischen Aufbau durch Funktionsblöcke und deren Zusammenspiel, den Aufbau der Hardware und die Struktur der Software.

4.1 Logischer Aufbau des Gesamtsystems

Um die Problemstellung aus Abschnitt 1.2 lösen zu können, muss das System unterschiedliche Funktionalitäten bieten. Einerseits muss ein vorgegebener Routenverlauf bereitgestellt werden um diesem mittels Positionsbestimmung folgen zu können, wobei die detaillierte Umsetzung in Kapitel 5 ab Abschnitt 5.2 beschrieben ist. Andererseits soll die Trajektorie innerhalb eines als Fahrbahn markierten Bereichs liegen. Zur Bestimmung der Trajektorie wird außerdem der aktuelle Standort des Fahrzeugs benötigt. Abbildung 4.1 zeigt einen Überblick der Funktionalitäten des Gesamtsystems. Die in blau gehaltene Blöcke stellen die Features zur Bereitstellung der Route,

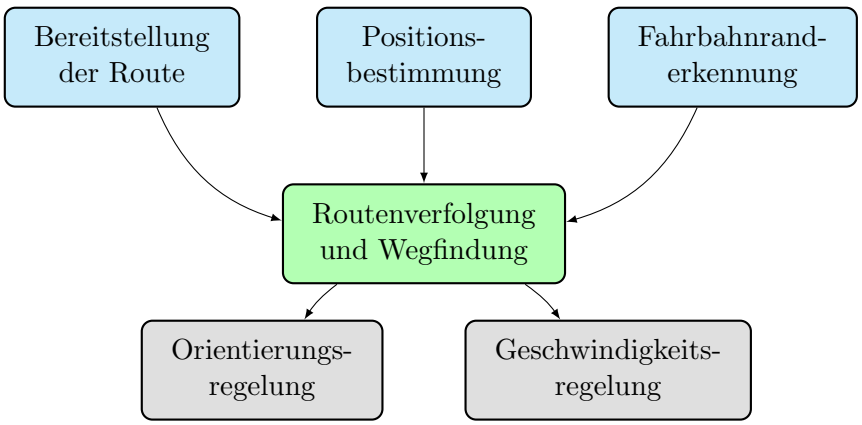


Abbildung 4.1: Überblick der Funktionalitäten des Gesamtsystems

zur Positionsbestimmung sowie der Fahrbahnranderkennung dar.

Die oben genannten Features sind die Ausgangsbasis, anhand dessen die Routenverfolgung und Wegfindung die Sollorientierung des Fahrzeugs bestimmt. Diese liefern dabei zwei Orientierungsvorgaben, die sich voneinander unterscheiden und zueinander in Konflikt stehen können. Eine

Orientierungsvorgabe ergibt sich aus der Berechnung der Orientierung von der aktuellen Position zum nächsten Punkt der GPS-Route. Ausgehend von der Fahrbahnranderkennung wird die zweite Orientierungsvorgabe anhand der aktuelle Orientierung des Fahrzeugs und der erkannten Fahrbahnmarkierungen bestimmt. Durch diese zwei Orientierungsvorgaben kann es bei der Routenverfolgung und Wegfindung dazu kommen, dass diese in Konflikt zueinander stehen. Diese in Konflikt stehenden Orientierungsvorgaben sind eine Feature Interaction und müssen durch einen Koordinator – die Routenverfolgung und Wegfindung – aufgelöst werden. Beispielsweise kann diese Feature Interaction bei der Bestimmung der Sollorientierung während eines Kurvenverlaufs auftreten. Ist der Verlauf der GPS-Route in Konflikt zu den Begrenzungen des Fahrbahnrandes, so muss die Sollorientierung anhand des Fahrbahnrandes angepasst werden um das Verlassen des Fahrbahnbereichs zu vermeiden. Im Falle eines solchen Konfliktes hat grundsätzlich die Orientierungsvorgabe aufgrund der Fahrbahnranderkennung Priorität. In Abschnitt 6.2 wird die Auflösung dieser Feature Interaction bei der Routenverfolgung und Wegfindung im Detail beschrieben.

Der Block Routenverfolgung und Wegfindung liefert als Ausgangsgrößen die Sollgrößen für die Features Orientierungs- und Geschwindigkeitsregelung. Die Geschwindigkeits- und Orientierungsregelung werden in den Abschnitten 5.1.1 und 5.1.2 beschrieben.

4.2 Hardwareaufbau

Aufbauend auf den beschriebenen Features folgt nun die Identifizierung der benötigten Hardware. Die Anforderungen für diese Arbeit beschränken sich jedoch nicht ausschließlich auf das Modellauto, sondern betreffen auch die erforderlichen Hardwarekomponenten.

Bei der Wahl des Modellautos war vor allem wichtig, dass es ausreichend Platz und gute Möglichkeiten zur Anbringung aller erforderlichen Hardwarekomponenten und der Geschwindigkeitsmessung bietet. Die Wahl ist hierbei auf ein handelsübliches Modellauto in der Bauform eines Monstertrucks im Maßstab 1:10 gefallen. Durch den Allradantrieb mittels Mittelkardanwelle, welche die beiden Differentiale starr mit dem Antriebsmotor verbindet, bietet sich eine gute Möglichkeit die Geschwindigkeit zu messen.

Abbildung 4.2 zeigt das Blockdiagramm aller Hardwarekomponenten des Fahrzeugs und deren elektrische Verknüpfung miteinander. Die Grün gehaltenen Blöcke symbolisieren die program-

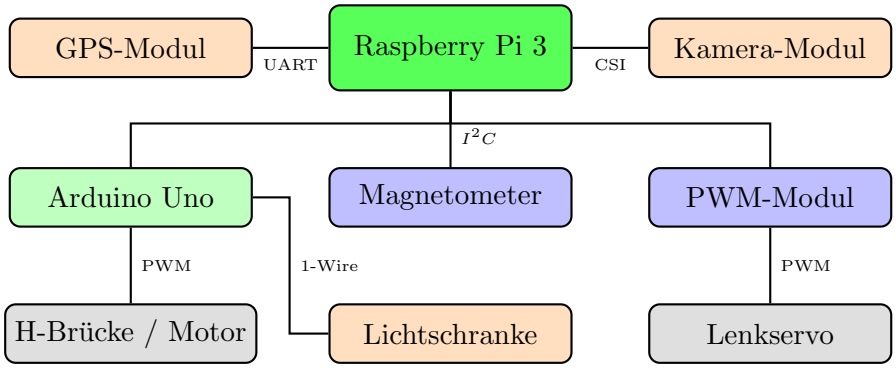


Abbildung 4.2: Blockdiagramm aller Hardwarekomponenten

mierbaren Hardwarekomponenten, die violetten Blöcke sind I^{2}C-Bus Teilnehmer, die beige farbigen Blöcke sind Sensoren und die grauen Blöcke repräsentieren Aktuatoren zur Steuerung des Fahrzeugs.

Zur Verarbeitung der Daten dient ein Raspberry Pi 3 B$^{\text{TM}}$, der mit einem Vierkernprozessor bei einer Taktfrequenz von 1,2 GHz ausreichend Rechenleistung bietet [10]. Der Raspberry Pi übernimmt, wie auch in Abbildung 4.2 dargestellt, als zentraler Block die Steuerung der restlichen dargestellten Hardwarekomponenten. Für alle Echtzeitoperationen, wie beispielsweise die Messung und Regelung der Geschwindigkeit, wird ein Arduino Uno Mikrocontrollerboard mit einem ATmega328 Mikrocontroller, der mit 16 MHz getaktet ist, eingesetzt [2]. Diese Komponente ist über den I^{2}C-Bus mit dem Raspberry Pi verbunden. Die Ansteuerung des Motors wird durch einen PWM-Ausgang des Arduino Uno und einer H-Brücke zur Leistungsverstärkung umgesetzt. Die Lichtschranke, welche über eine einzige Leitung (1-Wire) mit dem Arduino Uno verbunden ist, dient zur Geschwindigkeitsmessung und wird in Abschnitt 4.2.1 näher behandelt.

Die Ansteuerung des Lenkservos wird von einem individuell einstellbaren PWM-Modul, welches via I^2C-Bus gesteuert werden kann, übernommen. Zur Positionsbestimmung über GPS wird ein GPS-Modul mit serieller Schnittstelle (UART) verwendet. Zur Aufnahme der nötigen Bilder wird das Kamera-Modul der Raspberry Pi Foundation eingesetzt und über ein Camera Serial Interface (CSI) mit dem Raspberry Pi 3 verbunden.

Die individuellen Befestigungen der Baugruppen am Modellauto wurden zuerst mit einem 3D-Konstruktionsprogramm entworfen und dann mit einem 3D-Drucker angefertigt. Der Aufbau des Kamerasystems am Modellauto wird in Abschnitt 4.2.2 erläutert.

Die geometrischen Abmessungen des verwendeten Modellautos sind in Abbildung 4.3 dargestellt, wobei der Radstand $l_{\text{RS}} = 275$ mm, der Abstand des GPS-Moduls zur Hinterachse $l_{\text{GPS}} = 215$ mm, die Spurweite $l_{\text{SW}} = 252$ mm und der maximaler Lenkwinkel $\delta = \pm 19,8$ Grad sind.

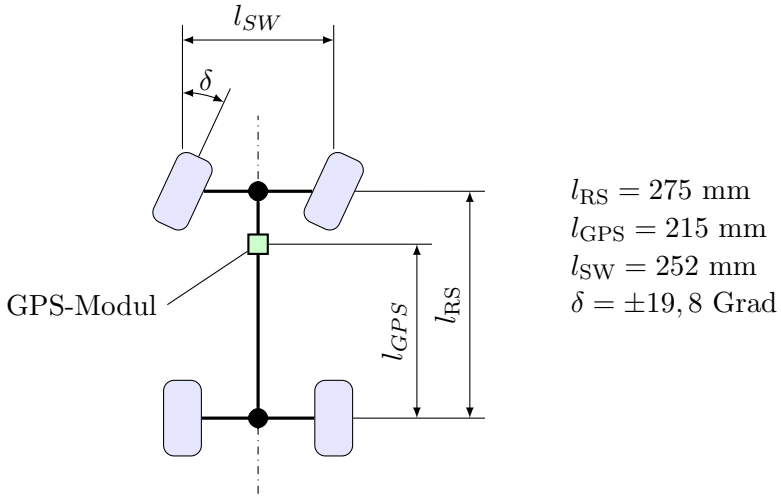


Abbildung 4.3: Geometrische Abmessungen des Modellautos
Radstand $l_{\text{RS}} = 275$ mm, Abstand des GPS-Moduls zur Hinterachse $l_{\text{GPS}} = 215$ mm, Spurweite $l_{\text{SW}} = 252$ mm und maximaler Lenkwinkel $\delta = \pm 19,8$ Grad

Ein Foto des vollständig aufgebauten Modellautos ist in Abbildung 4.4 zu sehen.

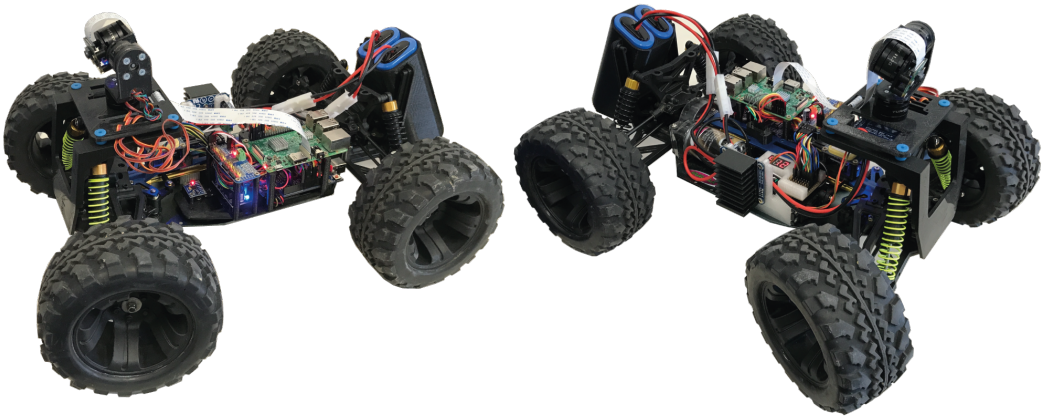


Abbildung 4.4: Vollständig aufgebautes Modellauto

4.2.1 Geschwindigkeitsmessung

Wie in Abbildung 4.2 zu erkennen ist, ist der Arduino Uno Mikrocontroller für die Messung und Regelung der Geschwindigkeit des Fahrzeuges vorgesehen. Die Geschwindigkeitsregelung wird im Abschnitt 5.1.1 behandelt. Die Messung erfolgt durch einen auf der Mittelkardanwelle aufgebrachten Fächer, der bei einer vollständigen Umdrehung mehrmals eine Lichtschranke unterbricht. Die Zeit der Unterbrechung der Lichtschranke wird durch den Mikrocontroller gemessen, der über die Übersetzungsverhältnisse der Differentiale und den Radumfang, die Strecke und somit die gefahrene Geschwindigkeit berechnet.

4.2.2 Kamerasystem

Das Kamerasystem, wie in Abbildung 4.5 gezeigt, ist so aufgebaut, dass sich die Kamera in einer Höhe von 23 cm direkt über der Vorderachse befindet. Zur Stabilisierung der Kamera dient ein 3-achsiges Gimbalsystem, welches in die 3D-gedruckte Halterung integriert wurde. Die Regelung des Gimbalsystems wird über ein Controller-Board [5] realisiert. Diese Regelung hält die Kameraposition relativ zum Modellauto, um etwaige Störungen zu glätten.

Wie im Blockdiagramm in Abbildung 4.2 dargestellt, ist die Kamera über das CSI direkt mit dem Raspberry Pi verbunden.

4.3 Software

Dieser Abschnitt beschreibt den Aufbau der Software. Dabei wird auf die strukturelle Sicht der Software, die Aufteilung in Prozesse und die Kommunikation mit den Hardwarekomponenten eingegangen.

Abbildung 4.5: Aufbau des Kamerasystems

4.3.1 Strukturelle Sicht der Software

Die Struktur der Software wurde basierend auf dem Überblick des Gesamtsystems aus Abbildung 4.1 aufgebaut und um unterstützende Softwareklassen erweitert. Abbildung 4.6 zeigt den strukturellen Aufbau in Form eines Klassendiagramms. Die durchgezogene Pfeile repräsentieren hierbei

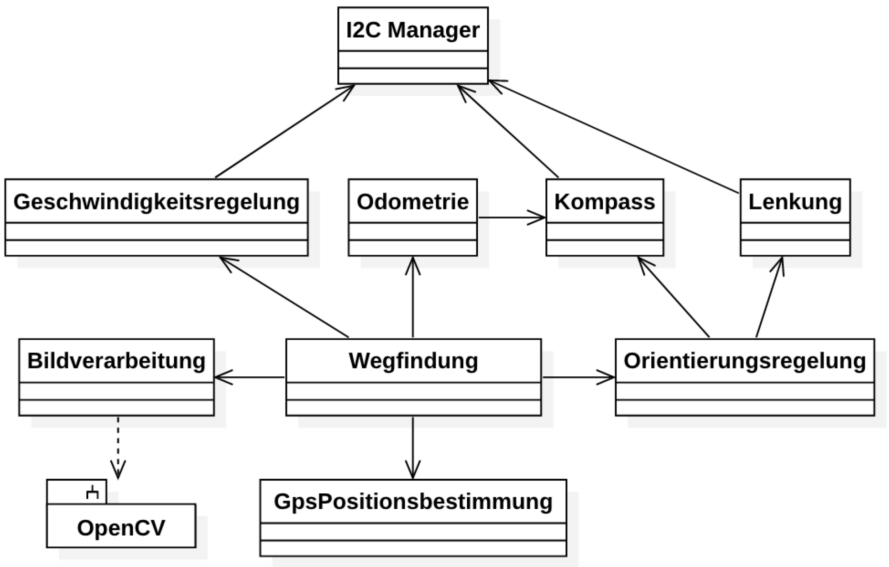


Abbildung 4.6: Klassendiagramm der Software zur Fahrzeugsteuerung

eine gerichtete Assoziation, die ausdrückt, dass die Instanz einer Klasse am Ursprung des Pfeils eine Referenz auf die Instanz der jeweils anderen Klasse besitzt. Der strichlierte Pfeil stellt eine Abhängigkeit einer zusätzlichen Bibliothek, in diesem Fall OpenCV, dar.

Die zentrale Klasse der Software ist die Klasse „Wegfindung“, in der die Funktion des Features

„Routenverfolgung und Wegfindung“ aus Abbildung 4.1 implementiert ist. Diese setzt auch das Feature „Bereitstellung der Route“ um. Der Routenverlauf wird dabei aus einer Datei geladen.

Das Feature „Positionsbestimmung“ aus Abbildung 4.1 wird durch die Klassen „GpsPositionsbestimmung“ und „Odometrie“ umgesetzt. Die Klasse „GpsPositionsbestimmung“ dient als Schnittstelle für die Kommunikation mit dem GPS-Modul aus Abbildung 4.2. In der Klasse „Odometrie“ wird die aktuelle Position des Fahrzeugs anhand der Fahrdaten berechnet. Die detaillierte Beschreibung der Odometrie befindet sich in Abschnitt 5.3.

Die Verwaltung des I^{2}C-Bus Zugriffs befindet sich in der Klasse „I2C Manager“, dessen Implementierung im Abschnitt 4.3.2.2 beschrieben wird.

Zur Steuerung der Hardwarekomponenten dienen die beiden Klassen „Orientierungsregelung“ und „Geschwindigkeitsregelung“. Die Geschwindigkeitsregelung kommuniziert die Sollgeschwindigkeit über den I^{2}C Manager an den Arudino Uno, der die Messung der Geschwindigkeit sowie die Ansteuerung des Motors übernimmt. Die Orientierungsregelung nimmt, ausgehend von einer Sollorientierung, anhand der aktuellen Orientierung eine Änderung des Lenkwinkels vor. Das Magnetometer wird über die Klasse „Kompass“ angesprochen und stellt die aktuelle Orientierung bereit. In der Klasse „Lenkung“ erfolgt die Steuerung des Lenkwinkels. Dabei wird, entsprechend des vorgegebenen Lenkwinkels, mittels des PWM-Moduls die PWM zur Positionierung des Lenkservos eingestellt.

Die Klasse „Bildverarbeitung“ kapselt die Kommunikation mit dem Kamera-Modul und verarbeitet die aufgenommenen Bilder. Aus diesen Bildern wird der Fahrbahnrand mit Hilfe der Bibliothek OpenCV (siehe Abschnitt 2.2) extrahiert. Auf Basis dieser Information wird eine Orientierungsänderung berechnet und für die Wegfindung bereitgestellt. Der detaillierte Ablauf der Fahrbahnranderkennung wird in Kapitel 6 behandelt.

4.3.2 Aufteilung in Prozesse

Um eine effizientere Auslastung des Prozessors zu erreichen, wurden Teilaufgaben des Programms in eigene Prozesse ausgegliedert. Mit Hilfe der von Python bereitgestellten Interprozesskommunikation in Form blockierender Warteschlangen wurde ein Mechanismus entworfen, der die Performance einzelner Prozesse zusätzlich steigert. Hierfür wurde für jeden zusätzlichen Prozess ein Thread im Hauptprozess erstellt, der ebenfalls nur nach erfolgtem Informationsaustausch mit seinem Prozess aktiv wird.

4.3.2.1 Sensordatenaktualisierung

Im Falle der Sensordatenaktualisierung, also dem GPS-Modul, dem Magnetometer und der Kamera, werden die erforderlichen Daten direkt nach deren Verfügbarkeit im jeweils eigenen Prozess bearbeitet und dann das Ergebnis über den Warteschlangenmechanismus an den Hauptprozess übergeben. Der prinzipielle Ablauf zur multiprozessbasierten Aktualisierung ist in Abbildung 4.7 dargestellt.

Andere Anwendungsfälle dieses Mechanismus sind das Management des I^{2}C Bus und die Berechnung der Odometrie, welche in Abschnitt 5.3 behandelt wird.

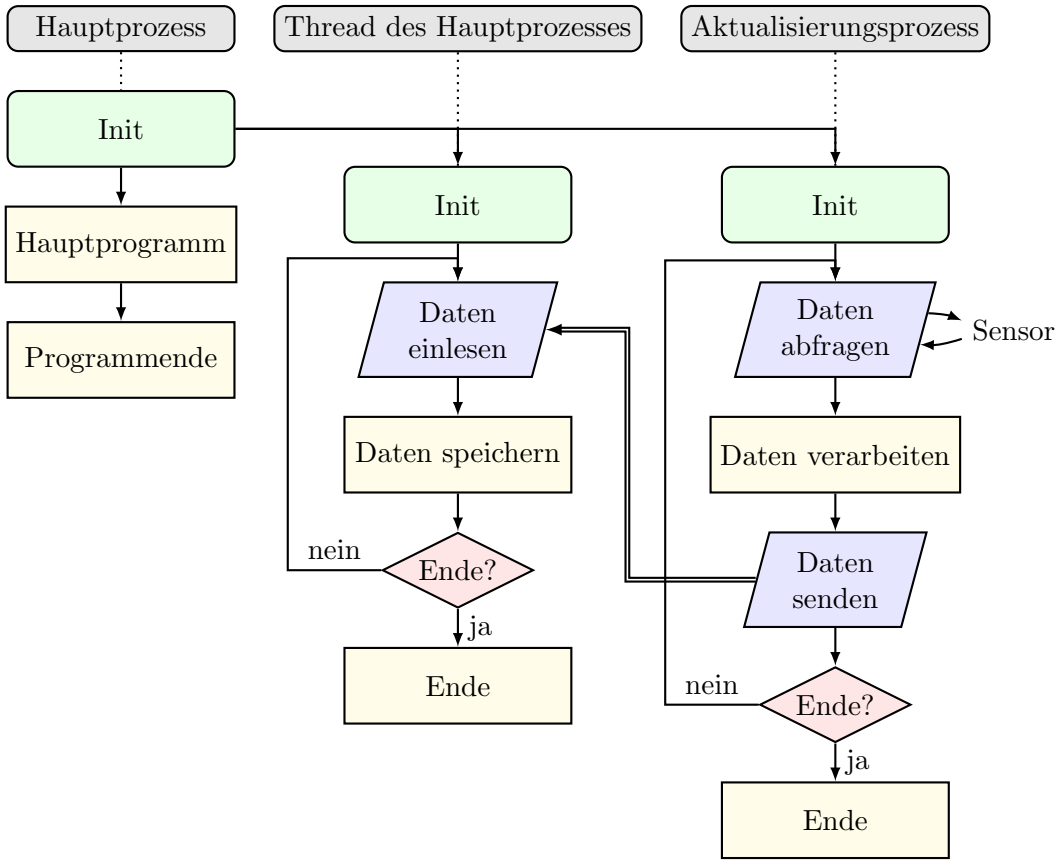


Abbildung 4.7: Kommunikationsablauf zur Sensoraktualisierung

4.3.2.2 I²C-Bus Management

Im Falle des I²C-Bus Managements kann es vorkommen, dass mehrere Prozesse mitunter gleichzeitig auf denselben physikalischen Bus zugreifen müssen, was ohne einem Management unweigerlich zu Kollisionen und Datenverlust führen würde. Dieser spezielle Prozess hat also die Aufgabe, die komplette I²C-Kommunikation so zu verwalten, sodass alle Lese- und auch Schreiboperationen ordnungsgemäß durchgeführt werden können.

Bei der Instanziierung des I²C-Bus Managers startet dieser sofort einen Prozess über den andere Softwarekomponenten Zugriffsanfragen auf den I²C-Bus durchführen können. Zugriffsanfragen auf den I²C-Bus (Lese- und Schreibzugriffe) werden nacheinander abgearbeitet und können von einer weiteren Zugriffsanfragen nicht unterbrochen werden. Da die Abarbeitung nacheinander erfolgt, kann es zu keinen Kollisionen von Zugriffsanfragen auf den I²C-Bus kommen. Durch zwei Methoden des I²C-Bus Managers können andere Prozesse Lese- und Schreibanfragen auf den I²C-Bus eintragen. Beim Aufruf einer dieser Methoden werden dann die Adresse des gewünschten I²C-Bus Teilnehmers, sowie das zu beschreibende oder zu lesende Register und im Falle eines Schreibvorgangs die zu schreibenden Daten übergeben. Alle Prozesse, die eine Zugriffsanfrage stellen, blockieren solange, bis die sich in Bearbeitung befindende Zugriffsanfrage abgearbeitet ist. Anschließend wird die nächste Zugriffsanfrage bearbeitet. Die Abbildung 4.8 zeigt den Ablauf des I²C-Bus Management Prozesses. Die hier verwendete Warteschlange ist eine Sonderform der, in Python integrierten, Interprozesskommunikation und stellt sicher, dass Anfragen an den I²C-

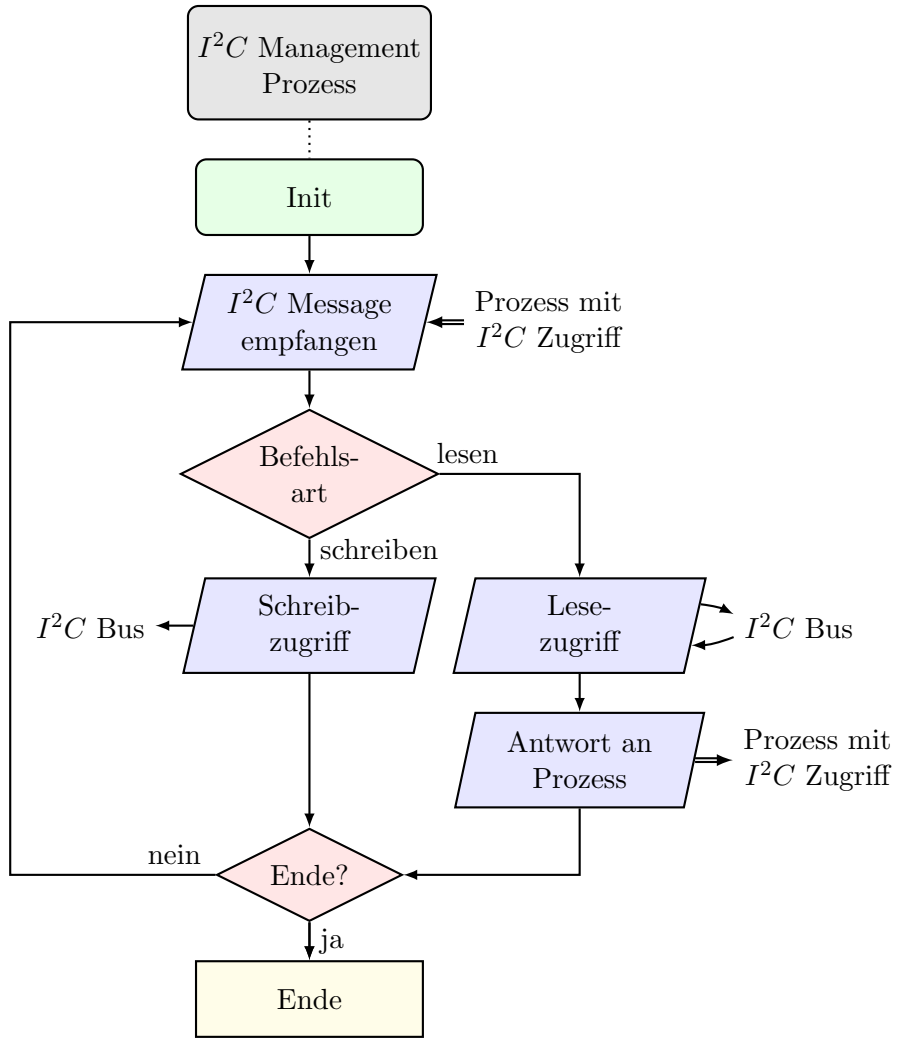


Abbildung 4.8: Ablaufdiagramm des I^{2}C Bus Management Prozess

Bus, wie zuvor beschrieben, nacheinander abgearbeitet werden.

4.3.3 Kommunikation zwischen Raspberry Pi und Uno

Die Kommunikation zwischen dem Raspberry Pi und dem Arduino Uno basiert auf dem I^{2}C-Bus und einer direkten Verbindung zweier Pins. Dadurch, dass der Arduino Uno und auch seine I^{2}C-Schnittstelle frei programmierbar sind, konnten alle nötigen Steuerbefehle frei gewählt werden und wurden wie in Tabelle 4.1 festgelegt.

Beginnend mit dem gelesenen Byte des Status-Lesebefehls lässt sich dessen Inhalt wie folgt beschreiben. Die Zustände der einzelnen Bits sind so zu verstehen, dass, falls die zugehörige Be-

Befehl	Register	R/W
Status	0x00	R
Geschwindigkeit	0x01	W
Motor Status	0x02	W

Tabelle 4.1: I²C-Befehle des Arduino Uno

schreibung nicht zutrifft, das Bit den jeweils anderen logischen Wert annimmt.

`Bit 0:` 1: Fahrzeugstatus OK
0: Fahrzeug außer Betrieb
`Bit 1:` x:
`Bit 2:` x:
`Bit 3:` 1: Geschwindigkeitsregler ist aktiv.
0: Geschwindigkeit ist auf 0 km/h gesetzt.
`Bit 4:` 1: Richtung rückwärts
0: Richtung vorwärts
`Bit 5:` x:
`Bit 6:` x:
`Bit 7:` x:
`Bit 8:` 1: Notaus Funktion wurde ausgelöst.

Da zu Beginn dieser Arbeit geplant war, dass das GPS-Modul durch den Arduino Uno ausgelesen wird, waren im Status Byte auch Informationen über den Zustand des GPS-Signals enthalten.

Zur Übertragung der Geschwindigkeit für die Geschwindigkeitsregelung, welche in Abschnitt 5.1.1 behandelt wird, wird der zweite Befehl aus Tabelle 4.1 eingesetzt. Der Wert der Geschwindigkeit selbst wird mit einem Byte als Sollgeschwindigkeit v_d in km/h übertragen und kann durch

$$v_{\text{bit}} = \lfloor 10 v_d \rfloor \tag{4.1}$$

ausgedrückt werden. Die Auflösung der Sollgeschwindigkeit ist demnach $\Delta v_{d,min} = 0,1$ km/h und die maximale Sollgeschwindigkeit beträgt $v_{d,max} = 25,5$ km/h.

Der Fahrzeugzustand bezogen auf die Inbetriebnahme des Motors durch den Arudino Uno lässt sich mit dem Schreiben von unterschiedlichen Steuerbytes einstellen. Hierfür wurden die folgenden Steuerbytes festgelegt:

- `0x00` - Startzustand
  Dieser Befehl muss in jedem Fall gesendet werden, bevor die Geschwindigkeitsregelung aktiviert werden kann.
- `0x01` - Motor betriebsbereit
  Das erste Bit des Steuerbytes muss für den betriebsbereiten Zustand immer gesetzt sein
- `0x03` - Motordrehrichtung vorwärts
- `0x05` - Motordrehrichtung rückwärts
- `0x80` - Notaus
  Nach dem Senden dieses Steuerbefehls, kann das Fahrzeug nur durch das Senden des Bytes für den Startzustand reaktiviert werden.

5 Modellerklärung

Dieses Kapitel behandelt zunächst die Regelung der Geschwindigkeit sowie der Orientierung und den Algorithmus zum Fahren einer definierten GPS-Route. Im Anschluss wird auf das mathematische Modell eingegangen, welches die Grundlage bildet, um bei ausgefallenem GPS-Signal die Fahrt fortsetzen zu können.

5.1 Regler

Um in den überlagerten Vorgängen, wie der Routenverfolgung oder dem Verhalten bei ausgefallenem GPS-Signal, konstante Werte der Geschwindigkeit zu garantieren und keine Fehler durch die Bauform des Modellautos in der Lenkung zu erhalten, wurde eine Regelung zur Erfüllung dieser Anforderungen entworfen.

5.1.1 Geschwindigkeitsregelung

Die Geschwindigkeitsregelung erfolgt durch Kombination einer Vorsteuerung und eines PID-Reglers. Eine solche Kombination bietet den Vorteil, dass bei einer Änderung der Sollgröße die Stellgröße durch die Vorsteuerung bereits in die Nähe des Arbeitspunktes gebracht wird. Der durch den Regler auszugleichende Fehleranteil wird damit verringert, wodurch der Arbeitspunkt schneller erreicht werden kann. Eine spezielle Form dieser Kombination ist die exakte Linearisierung, bei der die Vorsteuerung durch ein inverses Modell des Systems gebildet wird [Röb17]. Für diese Regelung wurde allerdings nur eine proportionale Vorsteuerung verwendet, was vor allem dadurch begründet ist, dass hier eine genauere Modellierung des Antriebsstranges zu viel Zeit in Anspruch genommen, aber keine signifikante Verbesserung gebracht hätte. Die wesentlichen Probleme bei dieser Modellierung sind die Auswirkungen der Akkuspannung auf die Drehzahl des Motors, die momentane Steigung der Fahrbahn und das Zahnradspiel im Antriebsstrang.

Im Blockschaltbild in Abbildung 5.1 sind alle wesentlichen Elemente der Regelstrecke zu finden, die im folgenden beschrieben werden. Die Abtastung erfolgt mit $T = 100$ ms und ist durch

$$v_k = v(kT), \text{ mit } k \in \mathbb{Z} \tag{5.1}$$

definiert. Gleiches gilt auch für alle anderen diskreten Größen dieses Reglers. Die Vorsteuerungsgröße $u_{\mathrm{FF},k}$ wird durch

$$u_{\mathrm{FF},k} = K_{\mathrm{FF}} v_{d,k} \tag{5.2}$$

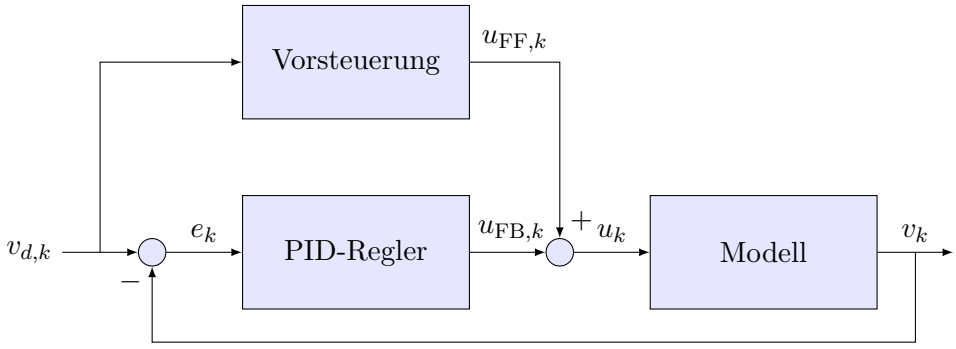


Abbildung 5.1: Blockschaltbild der Geschwindigkeitsregelung

definiert, wobei $v_{d,k}$ die Sollgröße und $K_{\mathrm{FF}} = 33$ der proportionale Vorsteuerungskoeffizient sind. Die Stellgröße u_k erfolgt durch Addition der Vorsteuerstellgröße $u_{\mathrm{FF},k}$ und der Reglerstellgröße $u_{\mathrm{FB},k}$. Die Reglerstellgröße ergibt sich mit dem Fehlersystem

$$e_k = v_{d,k} - v_k. \tag{5.3}$$

zu

$$u_{\mathrm{FB},k} = K_{\mathrm{P}} e_k + K_{\mathrm{I}} \sum_{i=0}^{k} e_i T + K_{\mathrm{D}} \frac{e_k - e_{k-1}}{T}, \tag{5.4}$$

mit den Reglerkoeffizienten $K_{\mathrm{P}} = 42$, $K_{\mathrm{I}} = 84$ und $K_{\mathrm{D}} = 2$. Der Integralanteil wird mit Hilfe eines Anti-Wind-Up Filters[1] softwaremäßig begrenzt.

5.1.2 Orientierungsregelung

Der prinzipielle Zusammenhang der Fahrzeuggeschwindigkeit v, des Lenkwinkels δ und der Fahrzeugorientierung φ lässt sich durch

$$\dot{\varphi} = \frac{v}{l_{\mathrm{RS}}} \sin \delta \tag{5.5}$$

definieren, wobei l_{RS} der Radstand aus Abbildung 4.3 und $\dot{\varphi}$ die zeitliche Ableitung der Fahrzeugorientierung φ sind. Theoretisch wäre es nun möglich mittels geeigneter Vorsteuerung in der Form von

$$\delta_{\mathrm{FF}} = \arcsin\left(\frac{l_{\mathrm{RS}}}{v} \dot{\varphi}_d\right) \tag{5.6}$$

und einer PID-Regelung nicht nur das Fahrzeug der Sollorientierung nach auszurichten, sondern auch eine fast beliebige Trajektorie zu fahren.

Durch das große Lenkspiel des für diese Arbeit verwendeten Modellautos ist eine zuverlässige Steuerung über den Lenkwinkel δ nicht ohne weiteres möglich. Die Lösung dieses Problems liegt in der Verwendung einer Orientierungsregelung auf Basis der Daten des Magnetometers aus Abschnitt 2.1.2. Ein Vorteil gegenüber einer Vorsteuerung aus Gleichung (5.6) ist, dass keine zeitliche

[1] Ein Anti-Wind-Up Filter begrenzt den Wert der Summe oder des Integrals eines Integralreglers auf ein definiertes Maximum

Ableitung der Orientierung nötig ist. Ebenfalls von Vorteil ist das integrierende Verhalten des Modells bezüglich des Lenkwinkels und der Orientierung, welches durch Integration von Gleichung (5.5) mit

$$\varphi(t) = \varphi_0 + \int_0^t \frac{v(\tau)}{l_{\mathrm{RS}}} \sin\delta(\tau) d\tau, \text{ mit } \varphi_0 = \varphi(0) \tag{5.7}$$

gezeigt werden kann. Durch Nutzung dieser Eigenschaften kann eine Regelung, gemäß dem Blockschaltbild in Abbildung 5.2, der Orientierung bei einer Geschwindigkeit $v \neq 0$ über den Lenkwinkel

$$\delta_k = K_{\mathrm{P}} e_{\varphi,k}, \text{ mit } e_{\varphi,k} = \varphi_{d,k} - \varphi_k \tag{5.8}$$

definiert werden. Für diese Regelung wurde eine Abtastzeit von 40ms verwendet.

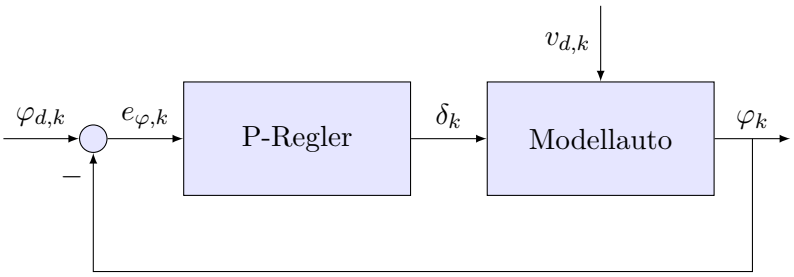


Abbildung 5.2: Blockschaltbild der Orientierungsregelung

5.2 Algorithmus zur Routenverfolgung und Wegfindung

Um das Modellauto gezielt zum nächsten Routenpunkt zu steuern, sind neben den Zielkoordinaten auch der aktuelle Standort und die aktuelle Orientierung des Modellautos nötig. Die Orientierung des Fahrzeugs kann durch die zurückgelegte Strecke berechnet werden. In diesem Fall könnte man auch auf die Regelung aus Abschnitt 5.1.2 verzichten. Allerdings ist das Verhältnis der zurückgelegten Strecke zwischen den einzelnen GPS-Punkten für den Fahrzeugstandort und die Genauigkeit des verwendeten GPS-Moduls zu klein um hier eine zuverlässige Aussage über die tatsächliche Fahrzeugorientierung treffen zu können. Erschwerend kommt noch das bereits besprochene Problem der unzuverlässigen Lenkwinkelsteuerung hinzu. Unter solchen Umständen führt dieser Ansatz also zu keinem zufriedenstellenden Ergebnis.

Eine Verbesserung des zuvor besprochenen Ansatzes kann die Hinzunahme der Orientierungsinformation durch ein Magnetometer, wie in Abschnitt 2.1.2 beschrieben, bringen. Dadurch reicht es aus die nötige Sollorientierung vom aktuellen Standort zum Zielstandort zu berechnen und das Fahrzeug über den Orientierungsregler aus Abschnitt 5.1.2 auszurichten. So verläuft die Fahrzeugtrajektorie direkt zu den Zielkoordinaten. Eine Zielkoordinate gilt genau dann als erreicht, sobald sich das Fahrzeug in einem Abstand von maximal zwei Metern befindet.

Durch die verhältnismäßig kleinen Abstände der GPS-Koordinatenpunkte zueinander kann bei der Berechnung der Sollorientierung von einer zweidimensionalen Ebene ausgegangen und die kugelartige Geometrie der Erdoberfläche vernachlässigt werden. Abbildung 5.3 veranschaulicht die geometrischen Umstände. In dieser Abbildung sind die gewünschte Orientierung $\varphi_{d,k}$, bezogen

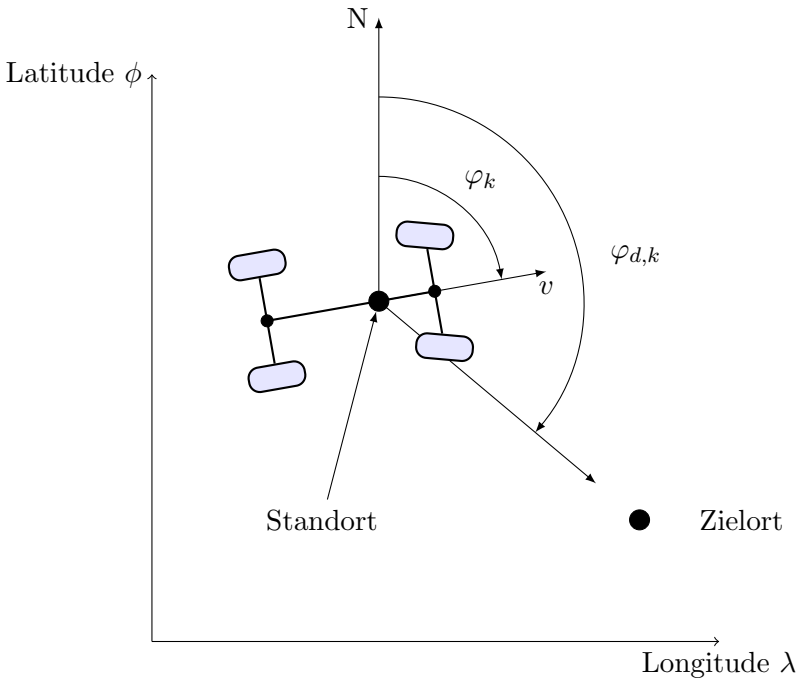


Abbildung 5.3: Orientierung des Modellautos im vereinfachten Weltkoordinatensystem

auf die Ordinate, vom Standort des Modellautos zum Zielort, sowie die aktuelle Orientierung φ_k eingezeichnet. Anhand dieser beiden Koordinatenpunkte kann der Winkel der Sollorientierung $\varphi_{d,k}$ quadrantenrichtig bestimmt und das Modellauto durch die Orientierungsregelung ausrichtet werden.

Der komplette Ablauf ist in Algorithmus 5.1 als Pseudocode dargestellt.

5.3 Odometrie

Der Standort des Fahrzeuges kann durch das GPS-Modul nur maximal einmal pro Sekunde bestimmt werden. Um während dieser Zeitspanne weiterhin die Position des Fahrzeuges zu kennen, wurde ein mathematisches Modell für Odometrie entwickelt. Dieses Modell errechnet anhand der Fahrzeugtrajektorie seit dem letzten Standort mit Hilfe des GPS-Moduls den aktuellen Standort mit einer Approximation.

Der Planet Erde ist in seiner Form zwar ein Rotationsellipsoid, doch kann durch die differentiell kleinen Distanzen, die für dieses Modell betrachtet werden, als Vereinfachung die Geometrie einer Kugel verwendet werden. Das verwendete GPS-Modul liefert, wie in Abschnitt 2.1.1 beschrieben, die Abweichung zum WGS84-Referenzellipsiod, wodurch der aktuelle Erdradius errechnet werden kann.

Die Länge eines differentiellen Wegstücks ds entlang einer Kugeloberfläche kann in Kugelkoordinaten (r, θ, φ) durch

$$\mathrm{d}s_{\mathrm{K}}^2 = \mathrm{d}r^2 + r^2\mathrm{d}\theta^2 + r^2 \sin^2\theta \mathrm{d}\varphi^2 \tag{5.9}$$

Eingabe :
v_{soll} ... Fahrgeschwindigkeit
r_{Liste} ... Liste der Routenpunkte

```
1  Fahrzeug initialisieren;
2  p = Position durch GPS Modul;
3  r = Erster Punkt in r_Liste;
4  wenn Abstand von p zu r > 50 m dann
5  |  Abbruch;
6  sonst
7  |  ψ_soll = Orientierung von p zu r;
8  |  Geschwindigkeitsregelung mit v_soll starten;
9  |  Orientierungsregelung für ψ_soll starten;
10 |  für alle Routenpunkte r in r_Liste tue
11 |  |  solange Abstand von p zu r > 2 m tue
12 |  |  |  ψ_soll = Orientierung von p zu r;
13 |  |  |  p = Position durch GPS Modul;
14 |  |  Routenpunkt r erreicht;
15 |  |  nächster Routenpunkt;
16 |  v_soll = 0;
17 |  Ziel erreicht;
```

Algorithmus 5.1 : Algorithmus zur Routenverfolgung und Wegfindung

ausgedrückt werden. Durch die einzelnen Komponenten

$$\begin{aligned} \mathrm{d}s_{\mathrm{K},r}^2 &= \mathrm{d}r^2 \\ \mathrm{d}s_{\mathrm{K},\theta}^2 &= r^2\mathrm{d}\theta^2 \\ \mathrm{d}s_{\mathrm{K},\varphi}^2 &= r^2\sin^2\theta\mathrm{d}\varphi^2 \end{aligned} \tag{5.10}$$

kann die Gleichung (5.9) auch durch

$$\mathrm{d}s_{\mathrm{K}}^2 = \mathrm{d}s_{\mathrm{K},r}^2 + \mathrm{d}s_{\mathrm{K},\theta}^2 + \mathrm{d}s_{\mathrm{K},\varphi}^2 \tag{5.11}$$

angeschrieben werden, wobei die Komponente $\mathrm{d}s_{\mathrm{K},r}$ durch $\mathrm{d}r = 0$ entfällt. Die Länge eines differentiellen Wegstücks entlang einer Kugeloberfläche mit konstantem Radius ist somit definiert durch

$$\mathrm{d}s_{\mathrm{K}}^2 = \mathrm{d}s_{\mathrm{K},\theta}^2 + \mathrm{d}s_{\mathrm{K},\varphi}^2. \tag{5.12}$$

Um die Komponenten des Kugelkoordinatensystems durch das Weltkoordinatensystem mit der Latitude ϕ und der Longitude λ darstellen zu können, müssen diese erst mit

$$\begin{aligned} \theta &= \pi - \phi \\ \varphi &= \lambda \end{aligned} \tag{5.13}$$

und deren differentiellen Änderungen mit

$$\begin{aligned} \mathrm{d}\theta &= -\mathrm{d}\phi \\ \mathrm{d}\varphi &= \mathrm{d}\lambda \end{aligned} \tag{5.14}$$

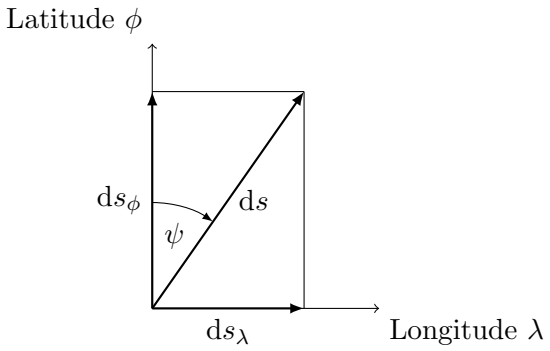


Abbildung 5.4: Berechnung eines differentiellen Wegstücks $\mathrm{d}s$

transformiert werden. Diese Transformation angewandt auf die Gleichungen aus (5.10) ergibt

$$\begin{aligned} \mathrm{d}s_\phi^2 &= r^2 \mathrm{d}\phi^2 \\ \mathrm{d}s_\lambda^2 &= r^2 \sin^2\left(\frac{\pi}{2} - \phi\right) \mathrm{d}\lambda^2 = r^2 \cos^2\phi \mathrm{d}\lambda^2 \end{aligned} \tag{5.15}$$

Damit ist ein differentielles Wegstück mit

$$\mathrm{d}s^2 = \mathrm{d}s_\phi^2 + \mathrm{d}s_\lambda^2 \tag{5.16}$$

berechenbar. Das vom Fahrzeug zurückgelegte differentielle Wegstück $\mathrm{d}s_v$ lässt sich durch

$$\mathrm{d}s_v = v\mathrm{d}t \tag{5.17}$$

definieren. Wie in Abbildung 5.4 gezeigt, lässt sich das zurückgelegte Wegstück aus Gleichung (5.17) auch als

$$\mathrm{d}s_v^2 = \mathrm{d}s_\phi^2 + \mathrm{d}s_\lambda^2 \tag{5.18}$$

darstellen. Diese Komponenten sind durch deren Abhängigkeit der Fahrzeugorientierung ψ mit

$$\begin{aligned} \mathrm{d}s_\phi^2 &= v^2 \cos^2\psi \mathrm{d}t^2 \\ \mathrm{d}s_\lambda^2 &= v^2 \sin^2\psi \mathrm{d}t^2 \end{aligned} \tag{5.19}$$

definiert. Eingesetzt in die Gleichungen aus (5.10) ergibt sich nach Umformung für die differentielle Änderung der Winkelkoordinaten

$$\begin{aligned} \mathrm{d}\phi &= \frac{v}{r} \cos\psi \mathrm{d}t \\ \mathrm{d}\lambda &= \frac{v}{r\cos\phi} \sin\psi \mathrm{d}t. \end{aligned} \tag{5.20}$$

Die Position kann dann durch

$$\begin{aligned} \phi_{k+1} &= \phi_k + \mathrm{d}\phi_k \\ \lambda_{k+1} &= \lambda_k + \mathrm{d}\lambda_k \end{aligned} \tag{5.21}$$

berechnet werden.

In Algorithmus 5.2 befindet sich die implementierte Version in Pseudocode, wobei die Positionsänderung in zwei Schritten berechnet wird. Solange das Fahrzeug in Betrieb ist, fordert der

```
    Thread der Odometrie im Hauptprozess
 1  solange Fahrzeug in Betrieb tue
 2      Warte auf Kommunikationsanfrage;
 3      unterscheide Kommunikationsanfrage tue
 4          Fall Statusanfrage tue
 5              wenn 300 ms seit letzter Positionsänderung dann
 6                  Sende Fahrzeugdaten und Anfrage um Positionsänderung;
 7              sonst
 8                  Sende Fahrzeugdaten;
 9          Fall Positionsänderung tue
10              Empfange Positionsänderungsdaten;

    Prozess der Odometrie
 1  solange Fahrzeug in Betrieb tue
 2      Sende Statusanfrage;
 3      Berechne Positionänderung;
 4      wenn Anfrage der Positionsänderung dann
 5          Sende Positionsänderung;
 6      Warte 75 ms;
```

Algorithmus 5.2 : Algorithmus der Odometrie

Prozess der Odometrie in einem Abstand von 75 ms die aktuellen Fahrzeugdaten vom Hauptprozess an. Im Thread des Hauptprozesses werden nicht nur die angeforderten Daten bereitgestellt, sondern auch die Entscheidung getroffen, ob die letzte Aktualisierung der Position länger als 300 ms zurückliegt. Ist dies der Fall, dann fordert der Thread die Positionsänderung an, um diese dann zur absoluten Position des Fahrzeuges zu addieren. Zu den angeforderten Fahrzeugdaten des Prozesses zählen die Geschwindigkeit, die Orientierung, der zuletzt berechnete Erdradius und der Latitudenwert ϕ der letzten Position. Anhand dieser Daten wird vom Prozess, durch die Gleichungen in (5.20) die Positionsänderung berechnet. Im Thread werden dann die Gleichungen aus (5.21) angewandt um den neuen Standort des Fahrzeuges zu approximieren.

Durch die dadurch gewonnene höhere Aktualisierungsgeschwindigkeit der Position des Fahrzeuges, welche zuvor noch durch das GPS-Modul auf etwa 1Hz limitiert war, kann auch der Algorithmus zur Routenverfolgung entsprechend adaptiert werden. Hierfür muss der Inhalt der Schleife beginnend in Zeile 11 des Algorithmus 5.1 um ein zusätzliches Warteelement erweitert werden. Algorithmus 5.3 ist die adaptierte Version von Algorithmus 5.1.

Eingabe :
v_{soll} ... Fahrgeschwindigkeit
r_{Liste} ... Liste der Routenpunkte

```
1  Fahrzeug initialisieren;
2  p = Position durch GPS Modul;
3  r = Erster Punkt in r_Liste;
4  wenn Abstand von p zu r > 50 m dann
5  |  Abbruch;
6  sonst
7  |  ψ_soll = Orientierung von p zu r;
8  |  Geschwindigkeitsregelung mit v_soll starten;
9  |  Orientierungsregelung für ψ_soll starten;
10 |  für alle Routenpunkte r in r_Liste tue
11 |  |  solange Abstand von p zu r > 2 m tue
12 |  |  |  ψ_soll = Orientierung von p zu r;
13 |  |  |  400 ms warten;
14 |  |  |  p = Position durch Odometrie;
15 |  |  Routenpunkt r erreicht;
16 |  |  nächster Routenpunkt;
17 |  v_soll = 0;
18 |  Ziel erreicht;
```

Algorithmus 5.3 : Algorithmus zur Routenverfolgung und Wegfindung mit Odometrie

6 Fahrbahnranderkennung und Integration in die Routenverfolgung und Wegfindung

Dieses Kapitel behandelt die Fahrbahnranderkennung und die Integration in den Algorithmus zur Routenverfolgung und Wegfindung.

6.1 Fahrbahnranderkennung

Zur Erkennung des Fahrbahnrandes wurde das Raspberry Pi Kamera Modul Version 2.1 [11] verwendet. Dieses Modul ist mittels CSI mit dem Raspberry Pi 3 verbunden und bietet eine einstellbare Auflösung von maximal 8 Megapixel (3280 x 2464). Das Kameramodul befindet sich wie bereits in Abschnitt 4.2.2 erläutert auf einer Höhe von 23 cm über der Fahrbahn und ist um etwa 5 Grad zur Fahrbahn geneigt. Diese Kameraposition wird relativ zum Fahrzeug stabilisiert und etwaige Störungen durch Fahrbahnunebenheiten werden geglättet.

Zur schnelleren Verarbeitung der Bilddaten wurden für diese Arbeit allerdings nur Bilder mit einer Auflösung von 320 x 240 Pixel aufgenommen. Da diese Aufnahmen aber auch Information enthalten, die nicht für eine Fahrbahnranderkennung verwendet werden kann, werden die Aufnahmen zugeschnitten, um den Bildbereich zu verkleinern und auf das Wesentliche zu reduzieren. Nach diesem Bearbeitungsschritt betragen die Abmessungen des relevanten Bildbereichs nur noch 300 x 128 Pixel.

6.1.1 Linienfindung

Die Linienfindung basiert auf den Methoden aus den Abschnitten 2.2.2, 2.2.3 und 2.2.4.1, wobei auf die genaue Anwendung im Folgenden eingegangen wird.

Im ersten Schritt wird nach dem Aufnehmen und Zuschneiden das Bild durch den veränderbaren Blurringfilter weichgezeichnet, um etwaige Störungen für die nachfolgende Kantenerkennung zu glätten. Nach der Kantenerkennung wird mit der Progressive Probabilistic Hough Transformation die eigentliche Linienfindung gestartet. Die Implementierung dieses Algorithmus in OpenCV (siehe Abschnitt 2.2.1) retourniert als Ergebnis sämtliche Anfangs- und Endpunkte der gefundenen Linien als Koordinatenpunkte. Da aber durch die Weichzeichnung zusammenhängende Linien zum Teil nicht als solche erkannt und mit Unterbrechungen interpretiert werden, müssen diese

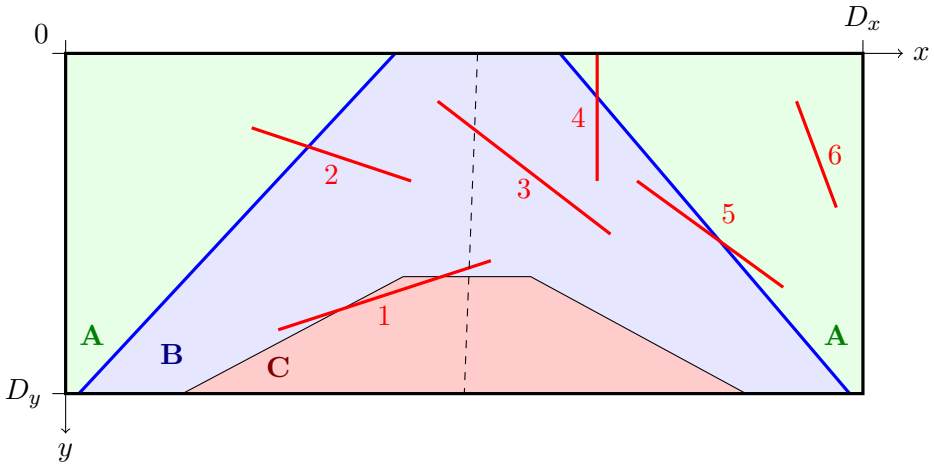


Abbildung 6.1: Bereichseinteilung des Sichtfeldes und Beispiele für auftretende Linien
Bereich A ist nicht Teil der Fahrbahn, Bereich B ist Teil der Fahrbahn, Bereich C ist jener Teil der Fahrbahn, in dem eine Anpassung der Trajektorie nicht mehr möglich ist

Linien nachträglich verbunden werden. Hierzu werden alle erkannten Linien hinsichtlich ihrer Geradengleichung

$$y = kx + d, \tag{6.1}$$

mit

$$\begin{aligned} k &= \frac{y_1 - y_2}{x_1 - x_2} \\ d &= y_1 - kx_1 \end{aligned} \tag{6.2}$$

und (x_1/y_1) und (x_2/y_2) als Anfangs- und Endpunkt, verglichen. Senkrecht verlaufende Linien, mit $x_1 = x_2$, werden hierbei nicht berücksichtigt. Für das eingesetzte Verfahren gelten zwei Linien i und j als ähnlich, wenn die Bedingungen

$$\begin{aligned} |k_i - k_j| &< 0,0175 \\ |d_i - d_j| &< 1,75 \end{aligned} \tag{6.3}$$

erfüllt sind. Diese Linien werden entsprechend ihrer Endpunkte zusammengefügt, indem für die neuen Start- bzw. Endpunkte der minimale und maximale x-Wert und, je nach Vorzeichen der Steigung, der zugehörige y-Wert verwendet werden.

6.1.2 Linienklassifizierung

Der nächste Schritt nach der Linienfilterung ist deren Klassifizierung anhand der Bildbereiche, in denen sie auftreten. Wie in Abbildung 6.1 gezeigt, wurde das Sichtfeld des Modellautos in drei Bereiche A, B und C unterteilt. Bereich A ist nicht Teil der Fahrbahn und besitzt keine Priorität. Bereich B ist jener Teil der Fahrbahn, in dem im Nah- und Fernbereich eine Anpassung der Trajektorie möglich ist und hat mittlere Priorität. Bereich C definiert jenen Teil der Fahrbahn, in dem eine Anpassung der Trajektorie des Modellautos nicht mehr möglich ist und hat hohe Priorität.

Zur Klassifizierung einer Linie sind nicht nur deren Anfangs- und Endpunkte zu betrachten, sondern auch der Verlauf der Linie, da diese, wie durch Linie 1 in Abbildung 6.1 gezeigt, unterschiedliche Bereiche überschneiden kann. Zur Detektion solcher Linien wird die Geradengleichung, siehe Gleichung (6.1) und (6.2), bestimmt und vom Anfangspunkt der x-Wert in zehn äquidistanten Schritten zum Endpunkt inkrementiert und der zugehörige y-Wert berechnet. Befindet sich einer dieser errechneten Punkte in einem unterschiedlich priorisierten Bereich, dann wird dieser Linie immer der höchste Prioritätsbereich zugeordnet. Im Fall von Linie 1 aus Abbildung 6.1 würde diese Linie die Priorität von Bereich C erhalten. Bei senkrechten Linien, wie Linie 4, wird nur der Bereich für den Randpunkt mit dem höchsten y-Wert bestimmt.

6.1.3 Veränderung des Blurringfilters

Wie eingangs in Abschnitt 6.1.1 erwähnt, ist das Blurringfilter veränderbar. Der Grund dafür ist, dass durch den geringen Abstand der Kamera zur Fahrbahn die Rauheit der Fahrbahn ähnlich wie starkes Rauschen wirkt. Das Resultat dieses Rauschens ist eine ähnliche Kantenstruktur wie in Abbildung 2.2(b) aus Abschnitt 2.2.2 im vorderen Bereich des Bildes. Dadurch werden fälschlicherweise Linien vor allem im kritischen Bereich C erkannt, die zum Stillstand des Fahrzeuges führen. Deshalb wurde für die Anzahl der erkannten Linien ein Schwellwert eingeführt, ab dem das Blurringfilter die Größe des Filterfensters erhöht. Falls dieser Schwellwert nicht erreicht wird, wird das Filterfenster wieder verkleinert. Da dies ein iterativer Prozess ist, sind die Auswirkungen dieser Filterveränderung erst mit dem nächsten aufgenommenen Bild erkennbar.

6.1.4 Liniengestützte Wegfindung

Nach erfolgreicher Klassifizierung aller erkannten Linien wird entschieden, ob und wie die Trajektorie des Fahrzeugs beeinflusst werden muss. Grundlage hierfür ist im ersten Schritt die jeweilige Priorität der Linie und im zweiten Schritt ihre Lage innerhalb des Sichtfeldes.

Alle Linien, die, wie Linie 6 in Abbildung 6.1, vollständig in Bereich A zu liegen kommen, können verworfen werden, da dieser Bereich nicht Teil der Fahrbahn ist. Im Fall, dass eine oder mehr Linien den Bereich C, wie Linie 1, zumindest teilweise durchlaufen, wird das Fahrzeug sofort durch Setzen der Sollgeschwindigkeit (siehe 5.1.1) auf 0 km/h angehalten. Die Fahrt wird erst fortgesetzt, sobald keine Linien im Bereich C erkannt werden.

Die übrigen Linien, die zumindest teilweise in Bereich B detektiert wurden, erfordern eine Veränderung der Trajektorie des Fahrzeuges. Die Richtung der Änderung der Trajektorie wird dadurch bestimmt, in welcher Bildhälfte sich die betrachtete Linie befindet. Da sich die Linie 2 aus Abbildung 6.1 auf der linken Bildhälfte befindet, muss hier auf jeden Fall die Änderung der Trajektorie nach rechts erfolgen. Bei den Linien 4 und 5 würde eine Änderung der Trajektorie nach links erfolgen. Einen Sonderfall repräsentiert Linie 3, da sie in beiden Bildhälften vorkommt und somit die Steigung der ausschlaggebende Faktor ist. In diesem Fall wird die Trajektorie entlang der Linie ausgerichtet.

Für den Fall, dass mehr als eine Linie im Bildbereich vorkommt, wird die nötige Änderung der Trajektorie für jede Linie berechnet und die Trajektorie dann so verändert, wie es für die Linie mit der größten Änderung der Trajektorie nötig ist. Wie in Abbildung 6.2 anhand der Linie mit der Nummer 2 aus Abbildung 6.1 gezeigt, wird die Änderung der Trajektorie berechnet. Der eingezeichnete Winkel ψ_k zum rechten Endpunkt der Linie 2 wird bezüglich des Fahrbahnrandes

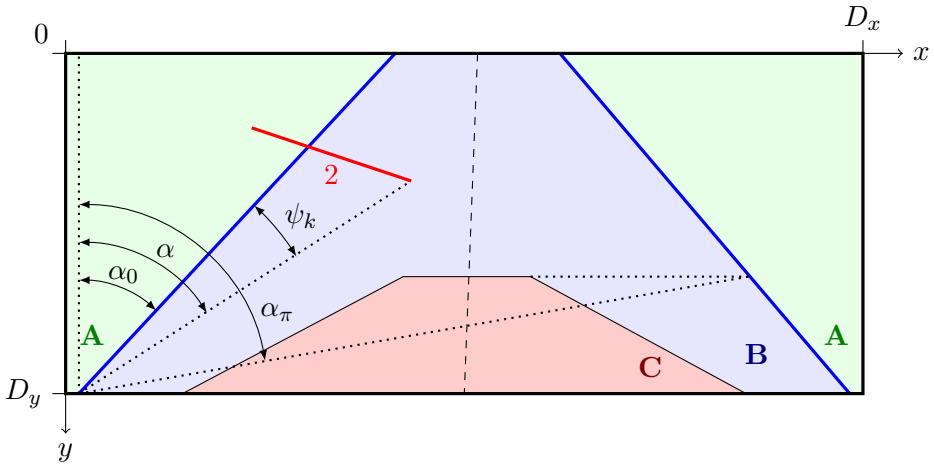


Abbildung 6.2: Beispiel zur Bewertung einer Linie

und dem linken oberen Eckpunkt von Bereich C skaliert. Der Fahrbahnrand repräsentiert hierbei Null Grad und der Endpunkt von Bereich C 90 Grad. Durch die Transformationsvorschrift

$$\psi_k = \frac{\alpha - \alpha_0}{\alpha_\pi - \alpha_0} \tag{6.4}$$

und eine Skalierung wird der Orientierungsänderungswinkel

$$\psi_{Kamera} = 90\psi_k \tag{6.5}$$

berechnet. Diese Skalierung hat ihre Begründung vor allem dadurch, dass der Lenkwinkel nur indirekt über die Vorgabe einer Sollorientierung der Orientierungsregelung, siehe Abschnitt 5.1.2, beeinflussbar ist. Die eigentliche Bedeutung des Winkels ψ_{Kamera} wird in Abschnitt 6.2 erläutert.

6.1.5 Implementierung der Linienfindung

Die Implementierung wurde ähnlich wie die Berechnung der Odometrie in Abschnitt 5.3 durch eine Kombination eines Threads im Hauptprozess und einem Prozess durchgeführt. Algorithmus 6.1 zeigt die Implementierung in Form von Pseudocode. Hierbei ist erkennbar, dass die rechenintensiven Operationen zur Fahrbahnranderkennung innerhalb des zusätzlichen Prozesses durchgeführt werden, um eine schnellere Abarbeitung zu erzielen. Das Kameramodul selbst wird ebenfalls erst im Prozess initialisiert, da die Peripherie nur jeweils von einem Prozess verwendet werden kann.

6.2 Integration in den Algorithmus zur Routenverfolgung und Wegfindung

Wie bereits in den vorigen Abschnitten erwähnt, kann eine Änderung der Trajektorie des Fahrzeugs durch dessen Orientierungsregelung nicht durch einen direkten Eingriff in den Lenkwinkel erfolgen. Grund dafür ist, wie in Abschnitt 5.2 erläutert, das Spiel der Lenkung, wodurch keine

```
Thread zur Fahrbahnranderkennung im Hauptprozess
1  solange Fahrzeug in Betrieb tue
2      Ergebnis aus Fahrbahnranderkennungsprozess abwarten;
3      unterscheide Ergebnis tue
4          Fall keine Trajektorienanpassung möglich tue
5              Kamerastatus = keine Trajektorienanpassung möglich;
6          Fall Fahrbahnrand erkannt tue
7              Kamerastatus = Fahrbahnrand erkannt;
8              ψ_Kamera aus Ergebnis setzen;
9          Fall kein Fahrbahnrand gefunden tue
10             Kamerastatus = kein Fahrbahnrand gefunden;

Prozess zur Fahrbahnranderkennung
1  Kamera Modul initialisieren;
2  solange Fahrzeug in Betrieb tue
3      Kamerabild aufnehmen;
4      Kamerabild zuschneiden;
5      wenn Anzahl zuletzt gefundener Linien > Maximale Anzahl gefundener Linien
        dann
6          Blurringfilterstufe erhöhen;
7      sonst
8          Blurringfilterstufe reduzieren;
9      Kamerabild mit Blurringfilter weichzeichnen;
10     Kantenbild = Kantenerkennung im Kamerabild;
11     Linien = Linienerkennung im Kantenbild;
12     Linienfilterung;
13     Linienklassifizierung;
14     Ergebnis an Thread senden;
```

Algorithmus 6.1 : Algorithmus zur Fahrbahnranderkennung

zuverlässige Lenkwinkelsteuerung möglich ist. Aus diesem Grund musste die Änderung der Trajektorie infolge des Kamerabilds in den bereits bestehenden Algorithmus, siehe Algorithmus 5.3, eingebettet werden.

Für die Integration in den bestehenden Algorithmus müssen die Ergebnisse der Fahrbahnranderkennung so weiterverarbeitet werden, dass sich daraus die erforderliche Orientierungsänderung des Fahrzeugs ergibt. Wie in Abschnitt 6.1.4 bereits erörtert, werden die erkannten Linien relativ zur Fahrbahn so skaliert, sodass daraus direkt die erforderliche Orientierungsänderung des Fahrzeugs abgeleitet werden kann. Der resultierende Algorithmus ist in Algorithmus 6.2 in Form von Pseudocode dargestellt. Dieser Algorithmus spiegelt somit die Feature Interaction der dargestellten Elemente aus Abbildung 4.1 in Abschnitt 4.1 wieder.

Falls durch die Kamera kein Fahrbahnrand entlang der Fahrbahn erkannt wird, ist der Ablauf genau so wie in Algorithmus 5.3 und die Sollorientierung entspricht der Orientierung zum nächsten Routenpunkt. Für den Fall, dass eine Fahrbahnrandlinie im Bereich C, siehe Abbildung 6.1,

Eingabe :
v_{soll} ... Fahrgeschwindigkeit
r_{Liste} ... Liste der Routenpunkte

```
1  Fahrzeug initialisieren;
2  p = Position durch GPS Modul;
3  r = Erster Punkt in r_Liste;
4  wenn Abstand von p zu r > 50 m dann
5  |  Abbruch;
6  sonst
7  |  ψ_soll = Orientierung von p zu r;
8  |  Geschwindigkeitsregelung mit v_soll starten;
9  |  Orientierungsregelung für ψ_soll starten;
10 |  für alle Routenpunkte r in r_Liste tue
11 |  |  solange Abstand von p zu r > 2 m tue
12 |  |  |  ψ_GPS = Orientierung von p zu r;
13 |  |  |  für 200 ms tue
14 |  |  |  |  unterscheide Kamerastatus tue
15 |  |  |  |  |  Fall keine Trajektorienanpassung möglich tue
16 |  |  |  |  |  |  Fahrzeug fährt erst bei möglicher Trajektorienanpassung weiter;
17 |  |  |  |  |  Fall Fahrbahnrand erkannt tue
18 |  |  |  |  |  |  ψ_ist = Orientierung des Fahrzeugs;
19 |  |  |  |  |  |  ψ_soll = ψ_ist + ψ_Kamera;
20 |  |  |  |  |  Fall Rückkehr zu Routenorientierung tue
21 |  |  |  |  |  |  Δψ = (ψ_GPS - ψ_ist) / 60;
22 |  |  |  |  |  |  ψ_soll = ψ_ist + Δψ;
23 |  |  |  |  |  sonst tue
24 |  |  |  |  |  |  ψ_soll = ψ_GPS;
25 |  |  |  |  p = Position durch Odometrie;
26 |  |  |  Routenpunkt r erreicht;
27 |  |  |  nächster Routenpunkt;
28 |  v_soll = 0;
29 |  Ziel erreicht;
```

Algorithmus 6.2 : Algorithmus zur Routenverfolgung und Wegfindung mit Odometrie und Fahrbahnranderkennung

erkannt wird, wird das Fahrzeug sofort zum Stillstand gebracht und setzt seine Fahrt nicht eher fort bis eine Änderung der Trajektorie möglich ist und die Fahrt fortgesetzt werden kann. Dieser Fall kann beispielsweise bei fälschlich erkannten Linien durch hohe Kontrastunterschiede der Fahrbahnoberfläche entstehen, die mit Erhöhung der Blurringfilterstufe, siehe Abschnitt 6.1.3, ausgeblendet werden.

Wenn eine Linie im Bereich B der Abbildung 6.1 erkannt wird, dann wird zunächst die nötige Orientierungsänderung ψ_{Kamera} berechnet, um diese vorzeichenrichtig zur aktuellen Orientierung

ψ_{ist} zu addieren. Die Summe wird dann als Sollorientierung

$$\psi_{soll} = \psi_{ist} + \psi_{Kamera} \tag{6.6}$$

an den Orientierungsregler übergeben.

Sobald die Trajektorie des Fahrzeugs so weit geändert wurde, dass die zuvor erkannte Fahrbahnrandlinie in Bereich A aus Abbildung 6.1 verschoben wurde, kann die Fahrt zum nächsten Routenpunkt in Richtung ψ_{GPS} fortgesetzt werden. Da hierdurch aber eine eine sehr abrupte Orientierungsänderung erfolgen würde, würde die zuvor erkannte Fahrbahnrandlinie erneut einen starken Einfluss auf die Trajektorie des Fahrzeugs haben. Zudem könnte dadurch das Fahrzeug die Fahrbahnrandlinie zu spät detektieren und überfahren. Hierfür wurde eine softwaremäßige Dämpfung der Orientierungsänderung eingeführt. Diese Dämpfung wurde im Kamerastatus „Rückkehr zu Routenorientierung“ realisiert. Dieser Status bewirkt, dass die nötige Orientierung aufgrund des nächsten Koordinatenpunktes nicht sofort als Sollorientierung eingestellt wird, sondern die Differenz zwischen ψ_{GPS} und ψ_{ist} in 60 äquidistante Schritte

$$\Delta\psi = \frac{\psi_{\text{GPS}} - \psi_{ist}}{60} \tag{6.7}$$

unterteilt wird. Die Sollorientierung wird dann über die Vorschrift

$$\psi_{soll} = \psi_{ist} + \Delta\psi \tag{6.8}$$

in 60 Schritten so verändert, bis die Fahrbahnbereich keine Fahrbahnrandmarkierungen erkannt werden und ψ_{GPS} als Sollorientierung eingestellt werden kann.

7 Evaluierung

Dieses Kapitel beinhaltet die Evaluierung des Verhaltens des Fahrzeugs und dessen Komponenten. Besonderes Augenmerk wurde dabei auf das Zusammenspiel der einzelnen Komponenten untereinander gelegt.

7.1 Geschwindigkeitsregelung

Die Geschwindigkeitsregelung, siehe Abschnitt 5.1.1, regelt die Geschwindigkeit des Fahrzeugs konstant und kann auch bei Steigung oder Gefälle den Sollwert halten. Kurzfristige Steigungen auf grobem Untergrund hingegen führen dazu, dass das Fahrzeug stehen bleibt bis der Regler genug Spannung für den Motor bereitstellt. Nach Überwindung einer solchen kurzfristigen Steigung ist die Geschwindigkeit kurzfristig erhöht, da erst der Integralanteil des Reglers bis zum eigentlichen Arbeitspunkt abgebaut werden muss.

Belastungsfrei, also ohne Bodenkontakt der Reifen, schwingt der Regler bei Sollgeschwindigkeiten $v_d < 2$ km/h, wobei das Festhalten eines Reifens und der damit verbundene Widerstand des Differentials als Dämpfung ausreicht, um eine konstante Umdrehungszahl der Reifen zu erhalten. Bei höheren Geschwindigkeiten wird die Drehzahl konstant gehalten und auch Arbeitspunktwechsel verlaufen gedämpft.

Die tatsächlich gefahrene Geschwindigkeit hängt jedoch stark vom Übersetzungsverhältnis des Getriebes ab. Für die exakte Bestimmung dieses Verhältnisses hätte das Differential geöffnet werden müssen, was aber aufgrund des Aufbaus nicht möglich war. Das Übersetzungsverhältnis im Differential wurde durch Vergleich der Umdrehungszahl der Kardanwelle und der Reifendrehzahl ermittelt. Dadurch weicht die Fahrzeuggeschwindigkeit von der Sollgeschwindigkeit geringfügig ab.

7.2 Orientierungsregelung

Die Orientierungsregelung, siehe Abschnitt 5.1.2, entspricht den gestellten Anforderungen für dieses System. Sobald die Regelung aktiviert und eine Sollorientierung eingestellt ist, richtet sich das Fahrzeug entsprechend der Sollorientierung aus und behält diese bei. Das Problem der unzuverlässigen Lenkwinkelsteuerung durch das große mechanische Spiel des Lenkmechanismus

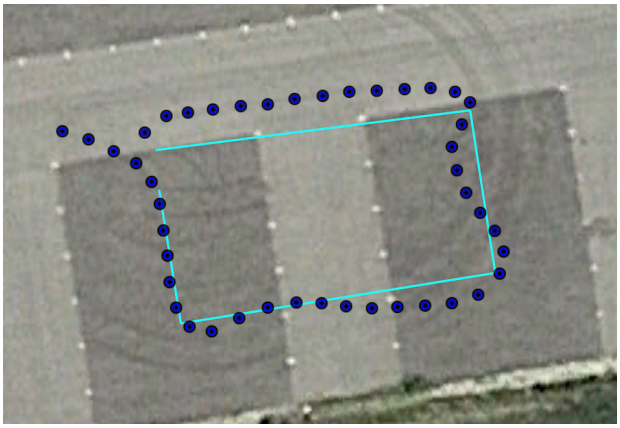


Abbildung 7.1: Fahrt entlang einer etwa 50 Meter langen Route bestehend aus insgesamt fünf Routenpunkten gegen den Uhrzeigersinn mit 4 km/h. Die durchgezogene Linie in Cyan ist die erstellte Route, die dunkelblauen Punkte stellen die GPS-Positionspunkte der gefahrenen Trajektorie des Fahrzeugs dar[3]

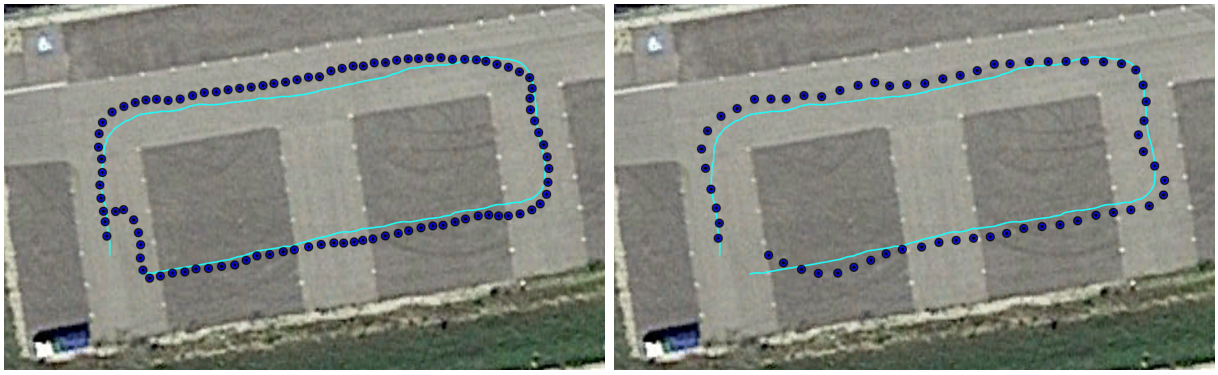
wurde damit behoben. Voraussetzung ist jedoch, dass eine Kalibrierung des Kompassmoduls durchgeführt wurde.

In der Nähe größerer Objekte aus Metall wird jedoch das Erdmagnetfeld so stark verzerrt, dass das Kompassmodul falsche Orientierungswerte ausgibt und somit auch die Orientierungsregelung negativ beeinflusst. Ähnlich negativ beeinflussbar ist auch die Kalibrierung in der Nähe solcher metallischer Objekte. Ein PKW in etwa 3 m Entfernung reicht bei der Kalibrierung aus, um die Orientierung in einer Entfernung von etwa 20-30 Meter vom Kalibrierungsort, durch verfälschte Offset-Werte, zu stören. Im Fahrverhalten äußert sich diese verfälschte Orientierung dadurch, dass das Modellfahrzeug trotz fixer Sollorientierung sichtlich seine Orientierung ändert.

Ein weitere idealisierte Annahme für diese Arbeit war, dass die befahrene Ebene eine ideale plane Ebene darstellt. Durch diese Annahme wurde auf die dritte Achse des Magnetometers sowie einen Lagesensor zur Bestimmung der räumlichen Lage verzichtet. Sobald das Fahrzeug jedoch gegenüber der Kalibrierungsebene geneigt wird, verändert sich auch der magnetische Fluss durch den Sensor, wodurch wiederum eine verfälschte Orientierung berechnet wird.

7.3 Routenfolge

Die Qualität der Trajektorie des Fahrzeugs entlang der vorgegebenen Route korreliert mit der Qualität der vorgegebenen Route. Wenn die vorgegebene Route nur aus vereinzelten Eckpunkten aufgebaut ist, dann kann die gefahrene Trajektorie zwischen den einzelnen Punkten durch GPS-bedingte Ungenauigkeiten in der Positionsbestimmung, beispielsweise durch verminderte Signalqualität bei Bewölkung oder ungenaues Kartenmaterial mit dem die Route erstellt wurde, von der direkten Verbindung zweier Routenpunkte abweichen. Abbildung 7.1 zeigt die Fahrt entlang einer Route mit einer Länge von etwa 50 Meter gegen den Uhrzeigersinn. Diese Route besteht aus nur fünf Routenpunkten, welche durch die Eckpunkte der durchgezogenen Linie in cyan dargestellt sind. Zwischen den einzelnen Routenpunkten ist eine Art Einschwingverhalten



(a) 4 km/h (b) 8 km/h

Abbildung 7.2: Fahrt entlang einer etwa 90 Meter langen Route gegen den Uhrzeigersinn mit unterschiedlichen Geschwindigkeiten. Die durchgezogene Linie in Cyan ist die erstellte Route, die dunkelblauen Punkte stellen die GPS-Positionspunkte der gefahrenen Trajektorie des Fahrzeugs dar[3]

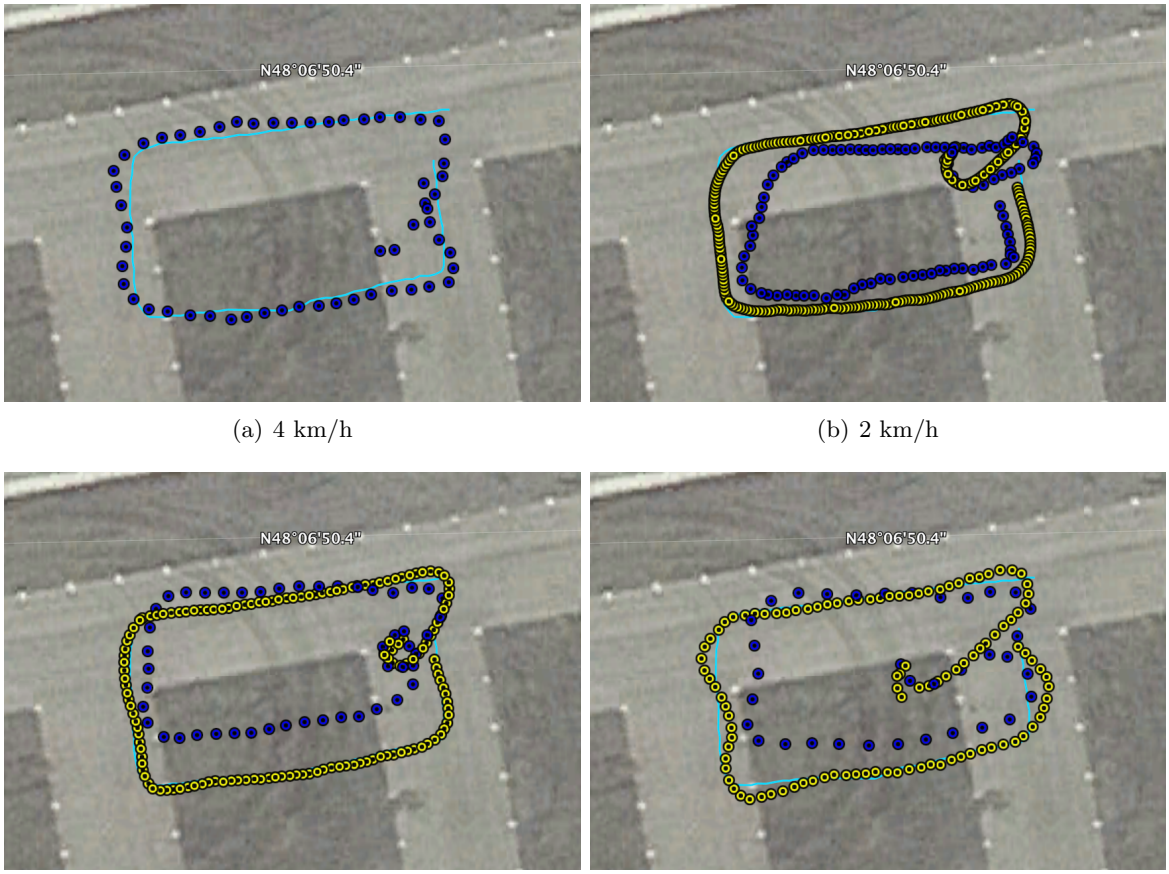
erkennbar, das sowohl auf das veränderte Erdmagnetfeld entlang der gefahrenen Strecke und damit auf eine leicht verfälschte Orientierung als auch auf die größere Distanz zwischen den Punkten zurückzuführen ist.

Bei einer Route, die aus wesentlich mehr Punkten besteht als zu ihrer Beschreibung nötig wären, verläuft die gefahrene Trajektorie mit dem Versatz von etwa einem meter entlang der vorgegebenen Route. Der Grund dafür ist, dass das Fahrzeug jeden einzelnen Punkt, auch entlang einer Geraden, erfolgreich passieren und somit zwingend die vorgegebene Route verfolgen muss. In Abbildungen 7.2(a) ist wiederum die Route als cyan-farbige Linie dargestellt und die tatsächlich gefahrene Trajektorie als dunkelblaue Punkte dargestellt. Erkennbar ist, dass das Einschwingverhalten entlang der Route durch die gegebene Mehrinformation deutlich unterdrückt wird. Zu beachten ist außerdem dass die Route in Abbildung 7.2(a) mit einer Länge von 90 Meter wesentlich länger ist als jene aus Abbildung 7.1. Abbildung 7.2(b) hingegen zeigt durch die verdoppelte Geschwindigkeit wiederum Einschwingverhalten. Begründet werden kann dies vor allem damit, dass sich kleine Lenkwinkeländerungen, siehe auch Gleichung (5.7), bei höheren Geschwindigkeiten wesentlich stärker auf die Orientierungsänderung auswirken. Im Zusammenspiel mit Traktionsproblemen bei eben diesen höheren Geschwindigkeiten und engen Kurvenradien ist das Einschwingverhalten durch Vernachlässigungen von Unter- bzw. Übersteuerung bei der Modellierung des Orientierungsreglers begründet.

7.4 Odometrie

Die Odometrie des Modellautos wurde in dieser Arbeit grundsätzlich dafür entwickelt, um einzelne ausgefallene GPS-Punkte des GPS-Moduls zu überbrücken. Beispielsweise bei schlechtem Empfang oder um die Zeitspanne der Positionsaktualisierung zwischen zwei Koordinatenpunkten des GPS-Moduls zu verkürzen.

Das Einfügen von berechneten Positionspunkten zwischen jenen des GPS-Moduls äußert sich während der Fahrt durch einen sanfteren Verlauf der Trajektorie des Fahrzeugs. Zudem fällt es



(a) 4 km/h (b) 2 km/h

(c) 4 km/h (d) 8 km/h

Abbildung 7.3: Fahrt entlang einer etwa 45 Meter langen Route gegen den Uhrzeigersinn. In (a) handelt es sich um eine rein GPS-gestützte Fahrt, wobei bei (b) bis (d) nach dem fünften erfolgreichen GPS-Punkt durch das GPS-Modul die Positionsbestimmung rein durch Odometrie erfolgt ist. Die durchgezogene Linie in Cyan ist die erstellte Route, die dunkelblauen Punkte stellen die GPS-Positionspunkte der gefahrenen Trajektorie des Fahrzeugs dar, die gelben Punkte sind jene Positionspunkte, die mittels Odometrie berechnet wurden [3]

dadurch nicht weiter auf, wenn ein Koordinatenpunkt nicht zum erwarteten Zeitpunkt vom Modul bereitgestellt wird. Falls das Fahrzeug nämlich den Zielpunkt bereits erreicht hätte, jedoch noch keine Positionsaktualisierung des GPS-Moduls bekommen hat, würde es am Zielpunkt vorbei fahren und müsste bei der nächsten Positionsaktualisierung umkehren.

Um die Odometrie über eine längere Fahrzeit bzw. Strecke zu evaluieren, wurde eine softwaremäßige Ausblendung der Information des GPS-Moduls programmiert. Diese Ausblendung erlaubt im Routenverfolgungsmodus fünf Koordinatenpunkte des GPS-Moduls und unterdrückt alle weiteren bis zur Ankunft im Ziel. Zu diesem Zweck wurde zudem eine Route mit einer Länge von etwa 45 Meter erstellt. In Abbildung 7.3(a) ist die erstellte Route als durchgezogene dünne Linie in türkis dargestellt. Die blauen Punkte sind jene GPS-Koordinaten, die während der Fahrt ohne Ausblendung der Daten des GPS-Moduls entstanden sind. Abbildung 7.3(b) zeigt wiederum die echten Positionsdaten des GPS-Moduls als dunkelblaue Punkte, wobei diese Daten für die

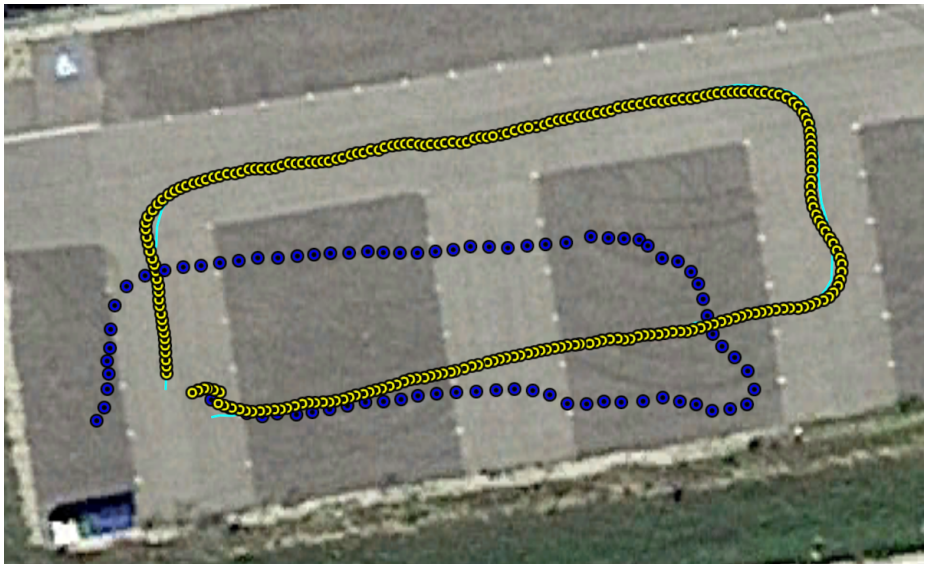


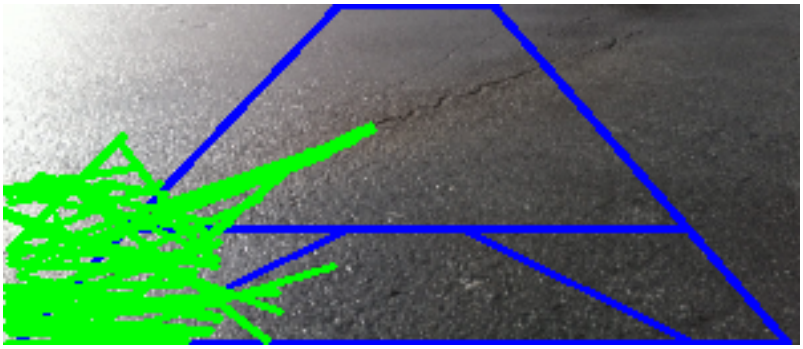
Abbildung 7.4: Testfahrt der Odometrie entlang einer vorgegebenen Route gegen den Uhrzeigersinn mit gestörter Kompasskalibrierung durch einen PKW. Die durchgezogene Linie in Cyan ist die erstellte Route, die dunkelblauen Punkte stellen die GPS-Positionspunkte der gefahrenen Trajektorie des Fahrzeugs dar, die gelben Punkte sind jene Positionspunkte, die mittels Odometrie berechnet wurden [3]

Routenverfolgung ausgeblendet wurden. Die gelben Punkte repräsentieren jene Positionspunkte der Odometrie, anhand derer das Fahrzeug dieselbe vorgegebene Route wie in Abbildung 7.3(a) abfährt. Die Fahrten in den Abbildungen 7.3(a) und 7.3(c) wurden mit einer Geschwindigkeit von 4 km/h durchgeführt. Die Abbildungen 7.3(b) und 7.3(d) zeigen die Daten der Routenverfolgungsfahrten mit einer Geschwindigkeit von 2 und 8 km/h.

Wie bereits in Abschnitt 7.2 erläutert ist auch für die Qualität der Odometrie die Kalibrierung des Kompassmoduls essentiell. Abbildung 7.4 zeigt anhand derselben Route, die in Abbildung 7.2 verwendet wurde, wie stark sich eine verfälschte Kalibrierung auf den weiteren gefahrenen Routenverlauf auswirken kann. In diesem Fall wurde die Kalibrierung nur wenige Meter von einem parkenden PKW entfernt durchgeführt, wodurch die errechnete Orientierung des Modellfahrzeugs deutlich beeinflusst wurde. Anhand der gelb markierten Punkte ist aber erkennbar, dass die Odometrie des Modells der zuvor geplanten Route, welche direkt unter diesen Punkten verläuft, folgt. Die dunkelblau markierten Punkte, die die tatsächlich gefahrene Strecke repräsentieren, zeigen aber, dass aufgrund dieses Orientierungsfehlers ein deutliche Abweichung entlang der geplanten Route aufgetreten ist.

7.5 Fahrbahnranderkennung

Die Fahrbahnranderkennung konnte in der verwendeten Auflösung von 320 x 240 Pixel mit der niedrigsten Einstellung des Blurringfilters bis zu 30 Bilder pro Sekunde aufnehmen und verarbeiten. Bei einer höheren Auflösung von 640 x 480 Pixel ist diese Zahl auf etwa 10 Bilder pro Sekunde gesunken. Sobald die Maske des Blurringfilters vergrößert wird, sinkt die Anzahl der verarbeiteten Bilder auf unter 15 Bilder pro Sekunde.



(a) geringe Blurringfiltereinstellung

(b) erhöhte Blurringfiltereinstellung

Abbildung 7.5: Aufnahmen der Fahrzeugkamera mit eingezeichnetem Fahrbahnbereich und erkannten Linien

Der adaptive Blurringfilter unterdrückt Linien, die durch Bildrauschen fälschlich erkannt werden. Der Unterschied wird bei starker Belichtung bzw. im Falle der Abbildungen 7.5(a) und 7.5(b) auf nasser Fahrbahn mit kleinem und vergrößertem Blurringfilter deutlich. Der dunkelblaue Rahmen im Bild markiert die Grenzen der Fahrbahn, die nachträglich eingefügt wurden, um die einzelnen Fahrbahnbereiche abzugrenzen. Die rot gefärbte Linie in Abbildung 7.5(b) ist in diesem Fall jene Linie, die zur Orientierungsänderung herangezogen wurde. In Abbildung 7.5(a) sind alle erkannten Linien in der gleichen Farbe dargestellt, da Linien in Bereich C, siehe Abschnitt 6.1.2 bzw. Abbildung 6.1, erkannt wurden und das Fahrzeug somit zum Stillstand gebracht wurde.

Wie bereits in den Abbildungen 7.5(a) und 7.5(b) erkennbar, ist die Belichtung der aufgenommenen Bilder für die Qualität der erkannten Linien ebenfalls maßgebend. Durch die zu helle linke Bildhälfte in Abbildung 7.5(a) kommt es ohne Blurring zu falsch erkannten Linien. Dieses Problem könnte ebenfalls mit einer Steuerung der Belichtungszeit des Kameramoduls selbst behandelt werden, um helle Aufnahmen durch verkürzte Belichtungszeit des Photosensors dunkler erscheinend zu lassen. Hier wurde aber die Einstellung für eine automatische Belichtungszeit des Kameramoduls verwendet.

Wegen der relativ niedrigen Position der Kamera ist auch das effektive Blickfeld auf eine maximale Distanz von etwa 1,5 Meter begrenzt. Die minimal erkennbare Distanz zu einer Linie direkt vor dem Fahrzeug ist 30 cm. Dadurch ergibt sich eine maximale Geschwindigkeit, bis zu der Fahrbahnränder rechtzeitig erkannt werden können. Bei Geschwindigkeiten von etwa 4 km/h konnte das Fahrzeug den Fahrbahnrändern noch rechtzeitig die Orientierung ändern. Bei höheren

Geschwindigkeiten wurde zwar ein kurzer Stillstand des Fahrzeugs eingeleitet, jedoch wurde der Stillstand erst in einer Entfernung von weniger als 30 cm erreicht. Somit wurde der Fahrbahnrand nicht mehr in Bereich C erkannt und die Fahrt fortgesetzt.

Die Rückkehr nach einer kamerabedingten Trajektorienänderung zur ursprünglichen Sollorientierung verläuft auf Grund der gedämpften Orientierungsänderung langsam genug, um ein erneutes Erkennen des Fahrbahnrandes zu ermöglichen. Im Falle einer ungedämpften Rückkehr würde der Fahrbahnrand überfahren werden. Durch die gedämpfte Rückkehr wird das Fahrzeug entlang des Fahrbahnrandes ausgerichtet. Das erneute Erkennen des Fahrbahnrandes und die gedämpfte Rückkehr resultieren auf diese Weise in einer gedämpften Schwingung entlang der Trajektorie parallel zum Fahrbahnrand.

8 Zusammenfassung

Ziel dieser Arbeit war es, ein System zur GPS-gesteuerten Routenverfolgung mit Unterstützung der Positionsbestimmung durch Odometrie zu erstellen. Zusätzlich war gefordert, dass ein Kamerasystem sicherstellt, dass das Modellauto während der Routenverfolgung einen als Fahrbahn markierten Bereich nicht verlässt. Um diese Aufgabenstellung zu lösen, wurde ein Modellauto um die nötige Peripherie, wie einen Raspberry Pi 3, einen Arduino Uno sowie ein Kamerasystem, erweitert.

Zur Routenverfolgung wurden unterschiedliche Routen mit einer Länge von bis zu 100 Metern in Google Earth gezeichnet und in ein geeignetes Format, bestehend aus den GPS Koordinaten der Wegpunkte, transformiert. Diese Routenpunkte werden bei einer Fahrt der Reihe nach abgefahren, wobei ein Punkt als erreicht gilt, sobald das Fahrzeug sich in einem Abstand von maximal zwei Metern befindet. Um einen Routenpunkt zu erreichen bestimmt das Fahrzeug die eigene Position mittels eines GPS-Moduls bzw. Odometrie und berechnet eine geografische Orientierung zum Zielpunkt. Diese nötige Orientierung wird dem implementierten Orientierungsregler als Sollgröße vorgegeben, wodurch das Fahrzeug, solange die gefahrene Geschwindigkeit ungleich Null ist, diese Orientierung einnimmt. Da die Aktualisierungsfrequenz des verwendeten GPS-Moduls nur bei maximal 1 Hz liegt, wird die Position zwischen den Aktualisierungen durch die intern berechnete Odometrie bereitgestellt. Die Routenverfolgung der Evaluierung war wiederholbar und das erreichte Ziel lag in einem Umkreis von etwa zwei Metern um den Endpunkt.

Die implementierte Odometrie wurde entworfen, um während der Zeitspanne zwischen den Aktualisierungen des GPS-Moduls ebenfalls Positionsdaten für die Routenverfolgung bereitzustellen. Die Qualität der Odometrie korreliert stark mit der Qualität der Kalibrierung der Orientierungsberechnung durch das Magnetometer. Zudem können Metallobjekte in unmittelbarer Umgebung das gemessene Magnetfeld stören und zu einer Verzerrung der berechneten Orientierung führen. Während der Evaluierung hat sich gezeigt, dass die Odometrie, basierend auf der gefahrenen Geschwindigkeit und der Orientierung des Fahrzeugs, genau genug arbeitet, um das Ziel bei einer 45 Meter langen Strecke in einem Radius von 5 Metern zu erreichen. Diese Tests lieferten auch mit unterschiedlichen Fahrzeuggeschwindigkeiten wiederholbare Ergebnisse in diesem Bereich.

Um auf etwaige Ungenauigkeiten der Route oder Abweichungen der Positionsbestimmung zu reagieren, wurde ein Algorithmus zur Erkennung des Fahrbahnrandes auf Basis der Kantenerkennung durch Canny Edge Detection sowie Linienfindung durch Hough Transformation entwickelt. Dieser Algorithmus teilt das Kamerabild in drei Bereiche ein. Die erkannten Linien werden abhängig von ihrem Auftreten im Bildbereich, kategorisiert um geeignet reagieren zu können. Für den Fall, dass Lichtverhältnisse oder aber auch die Rauheit des Untergrunds dazu führen, dass zu viele

falsche Linien erkannt werden, wurde ein adaptives Blurringfilter implementiert. Dieser Mechanismus verändert die Größe des Fensters des Blurringfilters, sobald ein maximaler Schwellwert an erkannten Linien überschritten wird. Hiermit konnte erreicht werden, dass das Fahrzeug auch auf Asphalt mit grober Rauheit den Fahrbahnrand erkennen konnte.

Die Änderung der Trajektorie des Fahrzeugs wird als Orientierungsänderung durch den Orientierungsregler aufgrund der erkannten Linien durchgeführt. Sobald keine Fahrbahnränder im relevanten Bereich des Kamerabildes erkannt werden, wird wieder jene Orientierung eingestellt, die zum nächsten Routenpunkt führt. Diese Orientierungsänderung erfolgt jedoch gedämpft, da das Fahrzeug entlang eines Fahrbahnrandes fahren könnte und diesen bei zu rascher Orientierungsänderung überfährt.

Die Resultate des implementierten Ansatzes sind vielversprechend und könnten weiterführenden Arbeiten als Ausgangspunkt dienen.

Abbildungsverzeichnis

Liste der Algorithmen

Wissenschaftliche Literatur

[AKIK08] Abdulhakam AM Assidiq, Othman O Khalifa, Md Rafiqul Islam, and Sheroz Khan. Real time lane detection for autonomous vehicles. In *Computer and Communication Engineering, 2008. ICCCE 2008. International Conference on*, pages 82–88. IEEE, 2008.

[BHS09] Amol Borkar, Monson Hayes, and Mark T Smith. Robust lane detection and tracking with ransac and kalman filter. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 3261–3264. IEEE, 2009.

[Can86] J Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, Jun 1986.

[HP62] V. Hough and C. Paul. Method and means for recognizing complex patterns. *United States Patent 3069654*, Dec 1962.

[MDVS14] Pierre Merriaux, Yohan Dupuis, Pascal Vasseur, and Xavier Savatier. Wheel odometry-based car localization and tracking on vectorial map. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pages 1890–1891. IEEE, 2014.

[MGK00] J. Matas, C. Galambos, and J. Kittler. Robust detection of lines using the progressive probabilistic hough transform. *Computer Vision and Image Understanding, April 2000, Vol.78(1), pp.119-137*, 2000.

[MSR16] Stefano Mafrica, Alain Servel, and Franck Ruffier. Minimalistic optic flow sensors applied to indoor and outdoor visual guidance and odometry on a car-like robot. *Bioinspiration & biomimetics*, 11(6):066007, 2016.

[NVRS09] N. Nourani-Vatani, J. Roberts, and M. V. Srinivasan. Practical visual odometry for car-like vehicles. In *2009 IEEE International Conference on Robotics and Automation*, pages 3551–3557, May 2009.

[Röb17] Klaus Röbenack. *Nichtlineare Regelungssysteme: Theorie und Anwendung der exakten Linearisierung*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2017.

[SPAJ15] Xiaotong Shen, Scott Pendleton, and Marcelo H Ang Jr. Accurate odometry for autonomous vehicles using in-car sensors. In *International Conference on Electronics, Information, and Communication (ICEIC)*, January 2015.

[STC06] Tsung-Ying Sun, Shang-Jeng Tsai, and Vincent Chan. Hsi color model based lane-marking detection. In *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*, pages 1168–1172. IEEE, 2006.

[WMKW14] Jun Wang, Tao Mei, Bin Kong, and Hu Wei. An approach of lane detection based on inverse perspective mapping. In *17th International Conference on Intelligent Transportation Systems (ITSC)*, pages 35–38. IEEE, Oct 2014.

Internet-Referenzen

[1] Arduino AG. *Arduino Entwicklungsumgebung*, Feb 2018. https://www.arduino.cc/en/Main/Software.

[2] Arduino AG. *Arduino Uno Rev 3*, Feb 2018. https://store.arduino.cc/arduino-uno-rev3.

[3] Google LLC. *Google Earth*, Mai 2018. https://www.google.com/earth/.

[4] National Marine Electronics Association. *NMEA*, Feb 2018. https://www.nmea.org/.

[5] OlliWs Bastelseiten. *STorM32 BGC: 3-Achsen STM32 Brushless Gimbal Controller*, Feb 2018. http://www.olliw.eu/2013/storm32bgc/.

[6] OpenCV. *Canny Edge Detection*, Feb 2018. https://docs.opencv.org/3.1.0/da/d22/tutorial_py_canny.html.

[7] OpenCV. *Hough Line Transform*, Feb 2018. https://docs.opencv.org/3.1.0/d6/d10/tutorial_py_houghlines.html.

[8] OpenCV. *OpenCV Documentation*, Feb 2018. https://opencv.org/.

[9] OpenCV. *Smoothing Images*, Feb 2018. https://docs.opencv.org/3.1.0/d4/d13/tutorial_py_filtering.html.

[10] Raspberry Pi Foundation. *Raspberry Pi 3 Model B*, Feb 2018. https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

[11] Raspberry Pi Foundation. *Raspberry Pi Camera Module V2.1*, Feb 2018. https://www.raspberrypi.org/products/camera-module-v2/.

Eidesstattliche Erklärung

Hiermit erkläre ich, dass die vorliegende Arbeit gemäß dem Code of Conduct, insbesondere ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel, angefertigt wurde. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Wien, am 6. Juni 2018

Thomas Karner