Hochschule für Angewandte Wissenschaften Hamburg
*Hamburg University of Applied Sciences*

Fakultät Technik und Informatik

Department Informations- und Elektrotechnik

**Bachelorprojekt**

# Automated Driving

## RC Car Control with Open Source Image Processing

Prof. Dr.-Ing. Marc Hensel
**Projektgruppe:** Fabian Huber, Enzo Morino, Markus Trockel
**Abgabe:** 11.02.2019

# Kurzübersicht

Das Ziel des Projekts ist ein Modellauto, dass mit Kamera ,Abstandssensor und Rapsberry Pi ausgestattet, durch Fahrbahnlinienerkennung teilautonom einem Straßenverlauf foglen kann. Es ist eine pratkitsche Umsetzung von Systemarchitektur, Sensorik und der Nutzung von den Werkzeugen der Bildverabreitung der Librry openCV in Python.
Dafür ist ein Prozess entwickelt worden, bei dem aus Sensordaten Steuerungsinformationen für den Wagen generiert werden. Die Durchführung hat gezeigt, dass die Umsetzung möglich ist, dass für eine stabile Funktionsweise weitere Entwicklungen von Funktionen nötig sind, die die zeitkritischen Anforderungen erfüllen und hardwarebedingte Fehleranfälligkeiten abfangen, um so eine rkontinuierliche Fahrt zu ermöglichen.

# Contents

**1 Einleitung** **1**

**2 Ziel des Projekts** **2**

**3 Prinzip der Steuerung** **3**

**4 Hardware** **3**
4.1 Raspberry Pi 3 . . . 3
4.2 Motorcontroller . . . 3
4.3 Ultraschallsensor . . . 3
4.4 RC Fahrzeug . . . 3

**5 Software** **3**
5.1 Aufbau . . . 4
5.2 Externe Module . . . 5
5.3 Eigene Module . . . 5
5.4 Eingesetzte Funktionen aus der openCV Library . . . 5
5.4.1 Canny-Edges Filter . . . 5
5.4.2 Hough-Transformation . . . 6

**6 Werkzeuge der Bildverabreitung** **6**
6.1 Digitalisierung . . . 7
6.2 Vorverarbeitung - Der Canny-Kantenoperator . . . 7
6.3 Segmentierung . . . 7
6.4 Merkmalsextraktion zur Linienerkennung . . . 8

**7 Iterative Auswertung des Kamerabildes** **8**
7.1 Verwendete Repräsentationsformen von Geraden . . . 9

# 1 Einleitung

Im Jahr 2005 ging das Projektauto Stanley der Standford University an den Start der DARPA Grand Challenge, einem Rennen autonomer Fahrezeuge über eine definierte Strecke in der Mojave-Wüste. Nach 212,76km durch unwegsames Gelände ging Stanley als Erster über die Ziellinie und das Team um Professor Sebastian Thrun war Gwinner des mit 2 Millionen Dollar dotierten Rennens.
Die Entwicklung des Autonomen Fahrens für Fahrzeuge des Straßenverkehrs ist ein aktuelles Thema, das immer mehr in die Praxis umgesetzt wird. Dabei gibt es verschiedenen Stufen des teilautonomen Fahrens. So gehört zur Stufe 2 die Funktion, dass Lenkvorgänge vom Fahrassistenten ausgeführt werden.
Durch dieses Projekts soll ein Einblick in die Materie geschaffen werden und die Umsetzbarkeit im Rahmen studentischer Möglichkeiten erfahren werden .
Die Autoren wollen mit ihrem Auto Hawey einen Startschuß setzen und mit diesem Bericht ein nachfolgendes Team inhaltlich in der Sache abholen, dass ein Einstieg in das Projekt und so ein konstruktives Fortsetzten und Verbessern möglich ist.

Fabian Huber, Enzo Morino, Markus Trockel

# 2 Ziel des Projekts

Das Projekt hat als Ziel, ein Modellauto mit Elektroantrieb durch Fahrbahnlinienerkennung eine vorgegebene Strecke fahren zu lassen. Als technische Bestandteile wird ein Raspberry Pi 3 mit dazugehöriger Kamera gewählt, dazu ein Sensor zur Abstandsmessung, um etwaige Kollisionen zu vermeiden. Implementiert wird der Code in Python unter der Nutzung der Bildverarbeitungslibrary openbCV.
Das Auto soll sowohl ein 4m lange Strecke gradeaus als auch in Kurven von 90° zwischen zwei Fahrbahnlinien die Spur halten. Idealerweise ist es möglich, dass der Wagen auch mit Linienunterbrechungen weitersteuern und so eine acht-förmige Strecke abfahren kann wie sie beim Carolo-Cup, einem Wettbewerb für autonome Modellfahrzeuge, befahren wird.

# 3 Prinzip der Steuerung

Über ein Python-Programm, welches auf dem Raspberry Pi läuft, wird der Video-Stream der angeschlossenen Kamera iterativ ausgewertet. Es werden die beiden Fahrbahnlinien erkannt, durch Geraden angenähert und deren Fluchtpunkt berechnet. Auf Grundlage der x-Koordinate dieses Fluchtpunktes wird ein Ausgangssignal berechnet, welches über das PWM-Modul den Servo ansteuert, und damit den Lenkwinkel festlegt.

# 4 Hardware

## 4.1 Raspberry Pi 3

## 4.2 Motorcontroller

## 4.3 Ultraschallsensor

## 4.4 RC Fahrzeug

# 5 Software

## 5.1 Aufbau

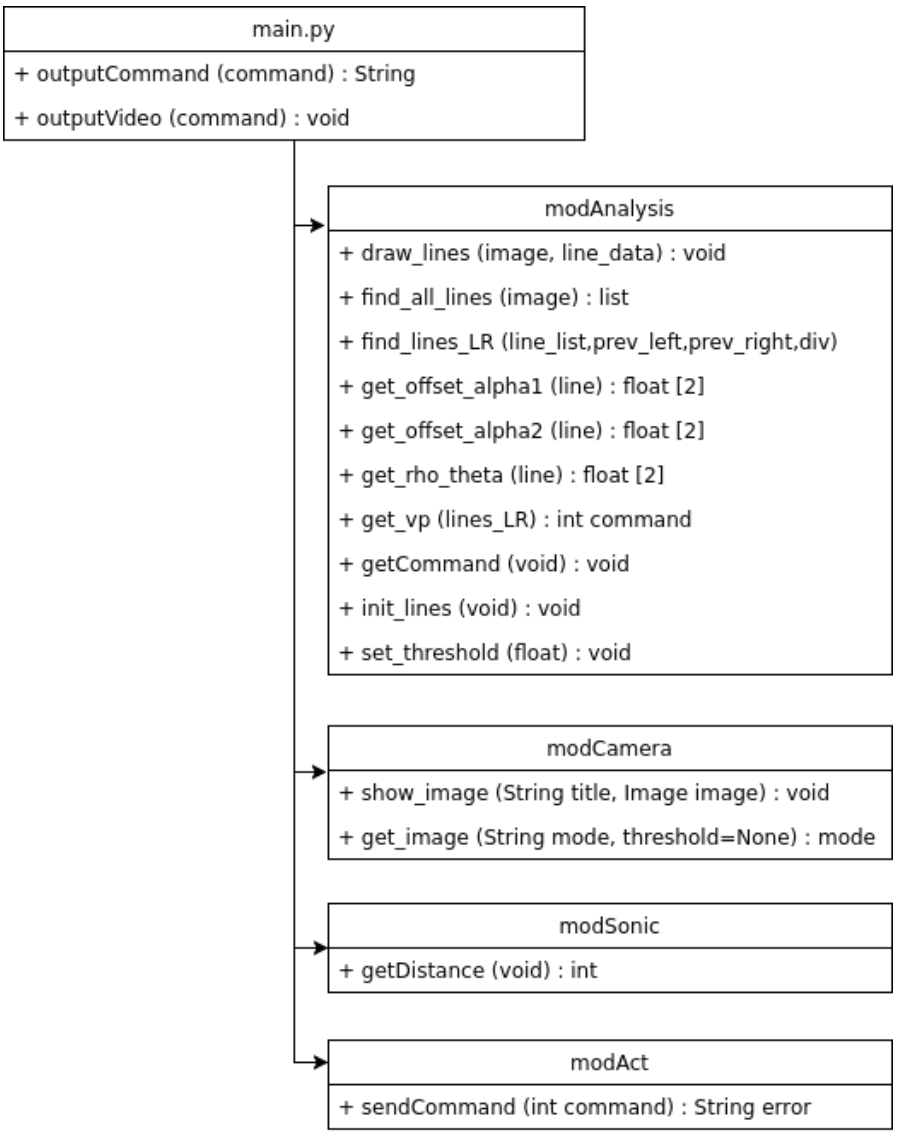


Figure 1: Aufbau des Python Codes

## 5.2 Externe Module

| Name | Beschreibung |
|---|---|
| tkinter | ... |
| Adafruit_PCA9685 | Bibliothek zur Ansteuerung des Motorcontrollers |
| numpy | Bibliothek zur Verwendung von Matlab Funktionen |
| cv2 | OpenCV 2 bietet Algorithmen zur Bildverarbeitung |
| io | ... |
| time | ... |
| importlib | ... |
| argparse | ... |
| pivideostream | ... |
| picamera | ... |
| threading | ... |
| RPi.GPIO | Bibliothek zur Ansteuerung der GPIO ports des Raspbery Pi |

Table 1: verwendete externe Python Module

## 5.3 Eigene Module

| Name | Beschreibung |
|---|---|
| modAnalysis | Verantwortlich für die eigentliche Verarbeitung der visuellen Informationen |
| modAct | Verantwortlich für die Ansteuerung des Motors und der Lenkung |
| modCamera | Bereitet das Kamerabild für die Verarbeitung und Anzeige vor. |
| modSonic | Kommuniziert mit dem Ultraschallsensor und liefert Distanz zum Hindernis. |

Table 2: verwendete eigene Python Module

## 5.4 Eingesetzte Funktionen aus der openCV Library

"OpenCV ist eine freie Programmbibliothek mit Algorithmen für die Bildverarbeitung und maschinelles Sehen." Sie ist u.a. für die Programmiersprache Python geschrieben und beinhaltet Funktionen, die für die Fahrbahnlinienerkennung eingesetzt werden.

### 5.4.1 Canny-Edges Filter

[BILD Canny] Der Canny-Edges-Filter wird im Kamera-Modul angewendet und über openCV.Canny("image-file", int untererTreshold, int obererTreshold) aufgerufen. Die Funktion liefert ein binäres Bild, das die Umrisse von Bildobjekten in weiß auf schwarzem Hintergrund liefert.
Nach einem Rauschfilter, der im Hintergrund läuft, wird der Gradient für jedes Pixel bestimmt. Liegt ein Gradient unterhalb des Lower Thresholds, wird ein Pixel schwarz im Zielbild, ist der Gradient oberhalb des Upper Thresholds, wird ein weißes Pixel ins Zielbild geschrieben. Gradientwerte im Zwischenbereich werden

als schwarz geschrieben, es sei denn, sie sind mit Gradienten im Bild verbunden, die oberhalb des Thresholds liegen.
Die Einstellungen des Thresholds sind ein sensibler Bereich, da hier eine Voreinscheidung getroffen wird, welche Bildinformationen erhalten bleiben bzw. verworfen werden. Idealerweise würden hier nur die Umrisse der Fahrbanlinien erhalten bleiben.

### 5.4.2 Hough-Transformation

[BILD zwei Koordinatensysteme: Kartesisch, Hough] Die Houghtransformation wird mit opencv.HoughLines(image, rho, theta, threshold) aufgerufen und gibt einen Vektor zurück mit allen im Bild erkannten Lininen, die jeweils mit dem Wertepaar($\rho$, $\theta$) beschrieben werden.Für $\rho$ und $\theta$ wird durch den Wert ein Toleranzwert beschrieben. "threshold" bezeichnet einen Wert, der die Mindestanzahl von Punkten bestimmt, die vorliegen müssen, damit eine Grade als beschrieben gilt.
Die Funktion wird zweimal angewendet: zum Einen beim Start zur Initialisierung, daß im gegeben Kamerabild die Fahrbahnlinien erkannt werden. Hierbei wird mit einem größeren Toleranzbereich gearbeitet, um eine Erkennung sicherzustellen. Zum anderen wird für jedes Bild des Kamerastreams die Linienerkennung gemacht, und mit einem strengeren Toleranzbereich mit den aus dem Vorbild erkannten Linienpaar verglichen. Dadurch soll verhindert werden, dass weitere erkannte Linien, die nicht Teil der Fahrbahn sind, fehlerhaft in die erkannten Linien mit einbezogen werden.
Die so noch erkannten Linien werden anhand von rho nd theta entweder der linken oder der rechten Fahrbahnlinie zugeordnet, und für beide Seite eine Linie gemittelt und als neue Orientierungslinen zur Richtungsbestimmung genutzt.

# 6 Werkzeuge der Bildverabreitung

Die folgende Zusammenfassung der Thematik basiert auf dem Buch "Digitale Bildverarbeitung" von Burger, Burge (**?**), das als Standartliteratur zu empfehlen ist. Eine zusammenfassende Wiedergabe der für das Projekt relevanter Themen ist im Folgenden nachzulesen.
Ein Bestandteil der Bildverarbeitung ist die Bildanalyse, bei der sinnvolle Informationen aus Bildern extrahiert werden. Genauer ist der Themenbereich *Computer Vision* gemeint, bei dem es um das Mechanisieren von Sehvorgängen des Menschen in der dreidimensionalen Welt geht.
Die Steuerung des Wagens soll durch Informationen aus den Kamerabildern erfolgen: Durch Erkennen der Fahrbahnlinien wird der Lenkungsservo geregelt, der das Auto innerhalb der Spur zwischen den Fahrpahnlinien hält.
Dieser Prozess lässt sich in Einzelschritten beschreiben mit:

1. *Digitalisierung* der dreidimensionalen Welt mit Hilfe der Kamera

2. *Vorverarbeitung* (Bildverbesserung bzw. Anpassung an den Zweck) durch Umwandlung in ein binäres Canny-Edges-Bild

3. *Segmentierung* durch Vorauswahl des Bildauschnittes mit den relevante Informationen

4. *Merkmalsextraktion* zur Linienerkennung durch die Hough-Transformation

5. *Parametrisierung* als mathematische Beschreibung der Fahrbahnlinien zur weiteren Informationsverarbeitung

## 6.1 Digitalisierung

Der von der Kamera aufgenommen Bilder-Stream liegt in digitaler Form vor. Dabei lassen sich benötigte Parameter eintellen. Um ein optimales Bild zu erhalten sind die Beleuchtungsumstände zu beachten, da hierdurch die Differenzierung von Linien im Bild erheblich beeinflusst wird.
Anzumerken ist hier, dass in der Bildverabreitung der Ursprung des Koordinatensystems bzw. der x- und y-Achse in der linken oberen Ecke des Bildes definiert ist.

## 6.2 Vorverarbeitung - Der Canny-Kantenoperator

Zur späteren Erkennung der Linien werden die dafür relevanten Informationen aus dem Bild gefiltert: sogenannte Bildkannten. Bildkanten sind Übergangsstellen, wo ein hoher Grauwertsprung von einem Pixel zum Nachbarpixel vorliegt, wie z. B. bei einem weißen Farbahnstreifen auf dunkler Fahrbahn, hier werden zwei Kanten analysiert. Kanten werden dem entsprechend an diversen Stellen im Bild detektiert, deshalb sind die Parameter entsprechend zu wählen.
Die Canny-Edges-Funktion ist ein bewährter Algorithmus zur Kantenerkennung, da drei Ziel gleichzeitig erreicht werden drei Ziele: ein zuverlässiges Detektieren vorhandener Kanten, die Position der Kante präzise zu bestimmen, und Farbsprünge, die nicht als Kante interoretiert werden sollen, auszulassen. Diese Funktion ist in der OpenCV-Library enthalten.

## 6.3 Segmentierung

[BILD Kamera mit Fahrbahnlinien] Der für die Erkennung der Fahrbahnlinien relevante Bereich liegt in einem unteren Dreieck des Kamerabildes. Dieser Teil wird ausgewählt, bzw. davon ausserhalb liegende Bereiche werden nicht bei der Erkennenung der Farbahnlinien berücksichtigt.

## 6.4 Merkmalsextraktion zur Linienerkennung

Das Canny-Kantenbild wird mit Hilfe der Hough-Transformation aus der openCV-Library in den Hough-Raum transformiert, wo dann die Erkennung der Linien erfolgt.

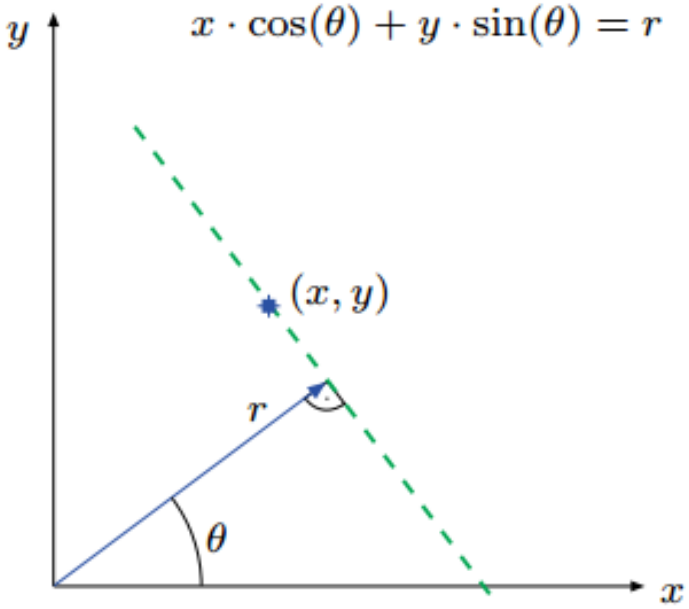


Figure 2: Beschreibung einer Graden durch Winkel und Abstand vom Ursprung

Dabei kommt zur Anwendung, dass eine Gerade mathematisch sowohl mit $y = m \cdot x + b$ als auch mit $r(\theta) = x \cdot cos(\theta) + y \cdot sin(\theta)$ beschrieben werden kann. Dabei ist in der zweiten Beschreibung $\theta$ der Winkel des Radius $r$ zum Urspung, an dessen Ende senkrecht die beschriebene Grade verläuft. Zu Bedenken ist, dass wie schon erwähnt der Koordinatenursprung des Bildes oben links definiert ist.
Das rechenaufwendige Verfahren der Hough-Transformation generiert für jeden Kantenbildpunkt des Canny-Edges-Bildes Bildpunkte im Hough-Raum, dessen Achsen ein $\theta$ / $r$ - Koordinatensystem bilden. Linien im kartesisches System sind als Punkthäufungen im Hough-Raum erkennbar. Liegt eine Anzahl von Punkten oberhalb eines zu definierenden Schwellwertes, wird eine Linie erkannt und die Houh-Transformationsfunktion gibt ein Wertpaar $\theta$, $r$ zurück.

# 7 Iterative Auswertung des Kamerabildes

Das Bild der Kamera werden iterativ ausgewertet um die beiden Fahrbahnlinien zu erkennen und daraus ein Ausgangssignal für die Steuerung zu generieren. Der gewählte Ansatz war es, die Fahrbahnlinien durch zwei Geraden anzunähern. Die Funktion HoughLines wird verwendet um die Fahrbahnlinien zu erkennen und die zugehörigen Geraden zu berechnen. Da die Funktion HoughLines in den meisten Fällen nicht nur die beiden Fahrbahnlinien erkennt, sondern alle möglichen Geraden, bestand die erste Aufgabe darin, nur die Geraden herauszufiltern, welche die Fahrbahnlinien repräsentieren, und alle anderen Geraden zu verwerfen. Im Pro-

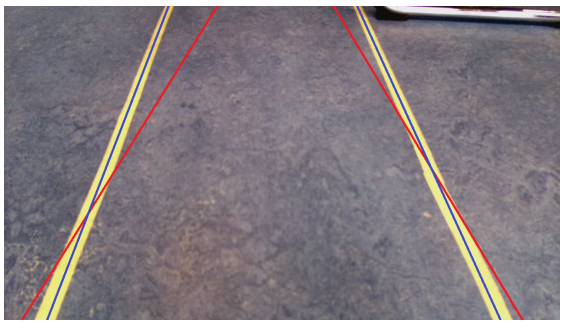

Figure 3: Beispiel für die erste Form der Geradenrepräsentation

gramm wird dies erreicht, indem alle Geraden, die durch die Funktion HoughLines gefunden wurden mit dem Geradenpaar aus dem vorherigen Programmdurchlauf verglichen werden, und nur ähnliche Geraden beibehalten werden.

Beim Start des Programm werden für die beiden Geraden feste Werte vorgegeben, die dann korrigiert und an die erkannten Linien angepasst werden. In Abbildung 3 ist dies bildlich dargestellt. Die roten Linien sind zu Beginn des Programms fest vorgegeben. Die blauen Linien sind das Ergebnis der Korrektur durch das Programm.

Der Vergleich der Geraden wird durchgeführt, indem um die beiden Geraden aus dem letzten Durchlauf ein Toleranz-Fenster gelegt wird, und von allen gefundenen Geraden nur diejenigen herausgefiltert werden, die innerhalb dieses Toleranz-Fensters liegen. Von diesen Geraden wird dann der Durschnitt berechnet, sodass nur noch zwei Geraden für die beiden Fahrbahnlinien übrig bleiben.

Eine Schwierigkeit besteht darin die Geraden so zu repräsentieren, dass diese gut miteinander vergleichbar sind. Bei einer geringen Änderung einer Geraden von einem zum nächsten Zyklus sollen sich deren Parameter dabei auch nur geringfügig ändern. Ansonsten werden ähnliche Geraden beim filtern verworfen, da deren Parameter nicht im Toleranzfenster liegen. Im Programm wurden daher je nach Anwendungsfall verschiedene Repräsentationsformen verwendet, welche im folgenden beschrieben werden.

## 7.1 Verwendete Repräsentationsformen von Geraden

Als erstes wurde die $\rho$-$\theta$-Form verwendet (der Name ist frei gewählt). Dies ist die Form, die von der Funktion HoughLines verwendet wird. Sie besteht aus einem Radius $\rho$ und einem Winkel $\theta$. $\theta$ ist der Winkel zwischen der Normalen, die senkrecht auf der Geraden steht, und der x-Achse. Der Winkel $\rho$ ist die Länge dieser Normalen, d.h. der Abstand zwischen der Geraden und dem Ursprung des Koordinatensystems. Der Winkel Abbildung 4 zeigt ein Beispiel für die Repräsentation zweier Fahrbahnlinien in der $\rho$-$\theta$-Darstellung, für den Fall, dass der Wagen relativ mittig und gerade auf der Fahrbahn steht. Das Rechteck symbolisiert die Ränder des Kamerabildes. Der Koordinatenursprung liegt in der oberen linken Ecke. $\theta 1$ ist positiv, $\theta 2$ negativ. $\rho 1$ und $\rho 2$ sind jeweils die Abstände der Geraden vom

Koordinatenursprung.

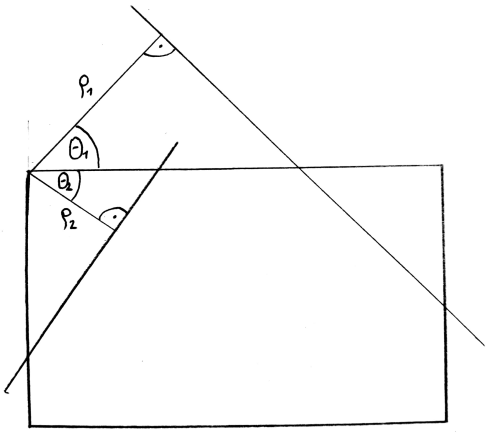


Figure 4: Beispiel für die erste Form der Geradenrepräsentation

Eine Schwäche dieser Repräsentationsform wird liegt darin, dass zwei sehr ähnliche Geraden komplett verschiedene Winkel $\theta$ haben können. Dies wird in Abbildung 5 deutlich. $\theta 1$ beträgt hier ca. $70°$, $\theta 2$ ca. $-100°$.

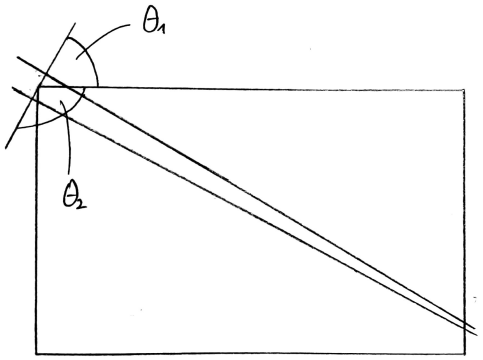


Figure 5: Beispiel für die erste Form der Geradenrepräsentation

Um dieses Problem zu umgehen wurden die Geraden mithilfe der Funktion offset_alpha1 in eine andere Form umgerechnet, bei der diese durch den Offset $\omega$ auf der y-Achse und den Winkel zur x-Achse beschrieben werden. Die Umrechnung erfolgt folgendermaßen:

Für Winkel $\theta < 90°$:

$$\omega = -\frac{\rho}{\cos 90° - \theta}$$
$$\alpha = 90° - \theta$$

Für Winkel $\theta > 90°$:

$$\omega = -\frac{\rho}{\cos\theta - 90^\circ}$$
$$\alpha = \theta - 90^\circ$$

Abbildung 6 veranschaulicht die Repräsentation der beiden Fahrbahnlinien in dieser Form.

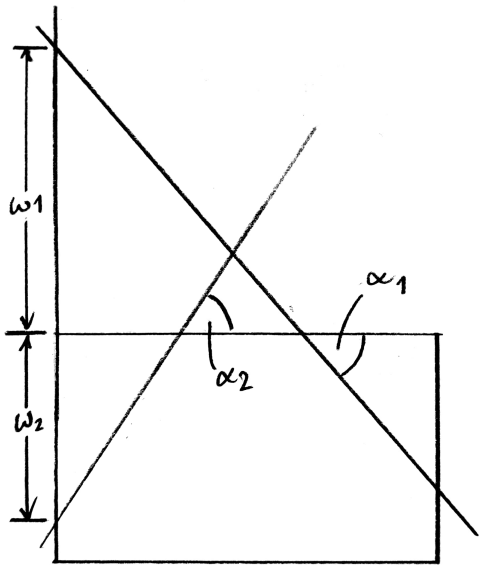


Figure 6: Beispiel für die erste Form der Geradenrepräsentation

Auch bei dieser Form ergibt sich das Problem, dass sich bei geringfügiger Veränderung der Geraden deren Parameter in bestimmten Situationen stark ändern. Dies ist besonders für die rechte Fahrbahnlinie der Fall, wie sich in Abbildung 7 erkennen lässt. Hier sieht man, dass sich der Offset $\omega$ stark ändert, wenn sich der Winkel der Geraden leicht ändert.

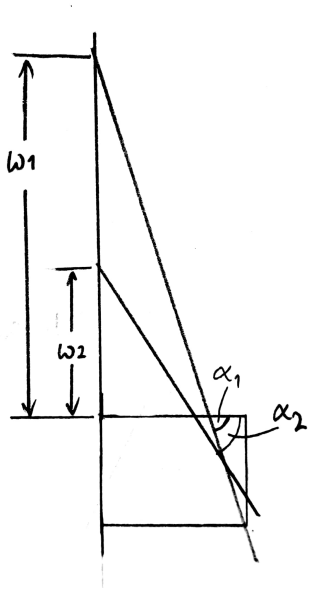


Figure 7: Beispiel für die erste Form der Geradenrepräsentation