



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 folgen kann. Es ist eine praktische Umsetzung von Systemarchitektur, Sensorik und der Nutzung von den Werkzeugen der Bildverarbeitung der Library 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 Fehleranfälligkeit abfangen, um so eine kontinuierliche Fahrt zu ermöglichen.

# Contents

<b>1 Einleitung</b>	<b>1</b>
<b>2 Ziel des Projekts</b>	<b>2</b>
<b>3 Prinzip der Steuerung</b>	<b>3</b>
<b>4 Hardware</b>	<b>3</b>
4.1 Raspberry Pi 3 . . . . .	3
4.2 Motorcontroller . . . . .	4
4.3 Ultraschallsensor . . . . .	4
4.4 RC Fahrzeug . . . . .	5
<b>5 Software</b>	<b>5</b>
5.1 Aufbau . . . . .	5
5.2 Externe Module . . . . .	6
5.3 Eigene Module . . . . .	6
<b>6 Werkzeuge der Bildverarbeitung</b>	<b>6</b>
6.1 Digitalisierung . . . . .	7
6.2 Vorverarbeitung- Canny-Kantenoperator . . . . .	7
6.3 Segmentierung . . . . .	7
6.4 Merkmalsextraktion zur Linienerkennung . . . . .	7
<b>7 Iterative Auswertung des Kamerabildes</b>	<b>8</b>
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 Fahrzeuge 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 Fortsetzen 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 geradeaus als auch in Kurven von  $90^\circ$  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**



Figure 1: Raspberry Pi 3 Model B

## 4.2 Motorcontroller

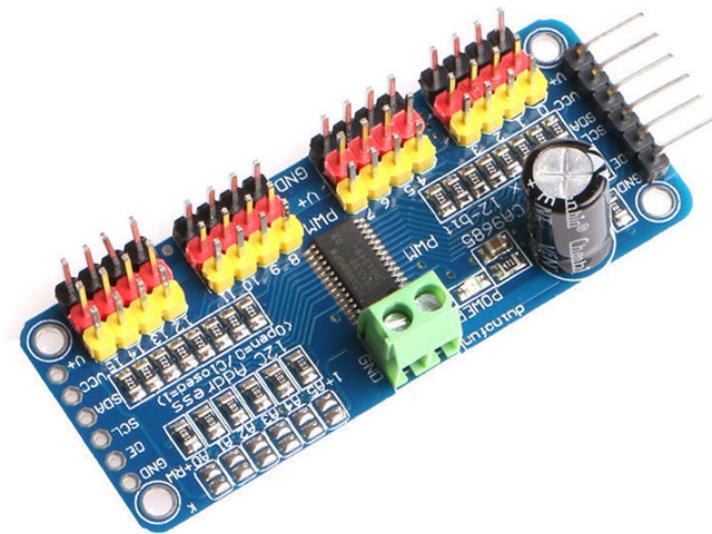


Figure 2: PCA9685 Controller Modul

## 4.3 Ultraschallsensor

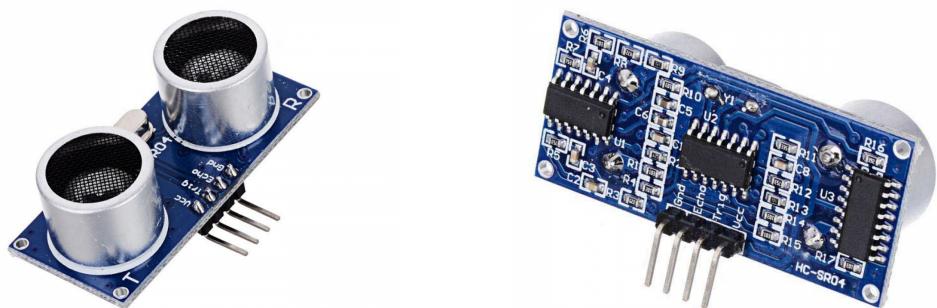


Figure 3: HC-SR04 Ultrasonic Sensor Module

## 4.4 RC Fahrzeug

# 5 Software

## 5.1 Aufbau

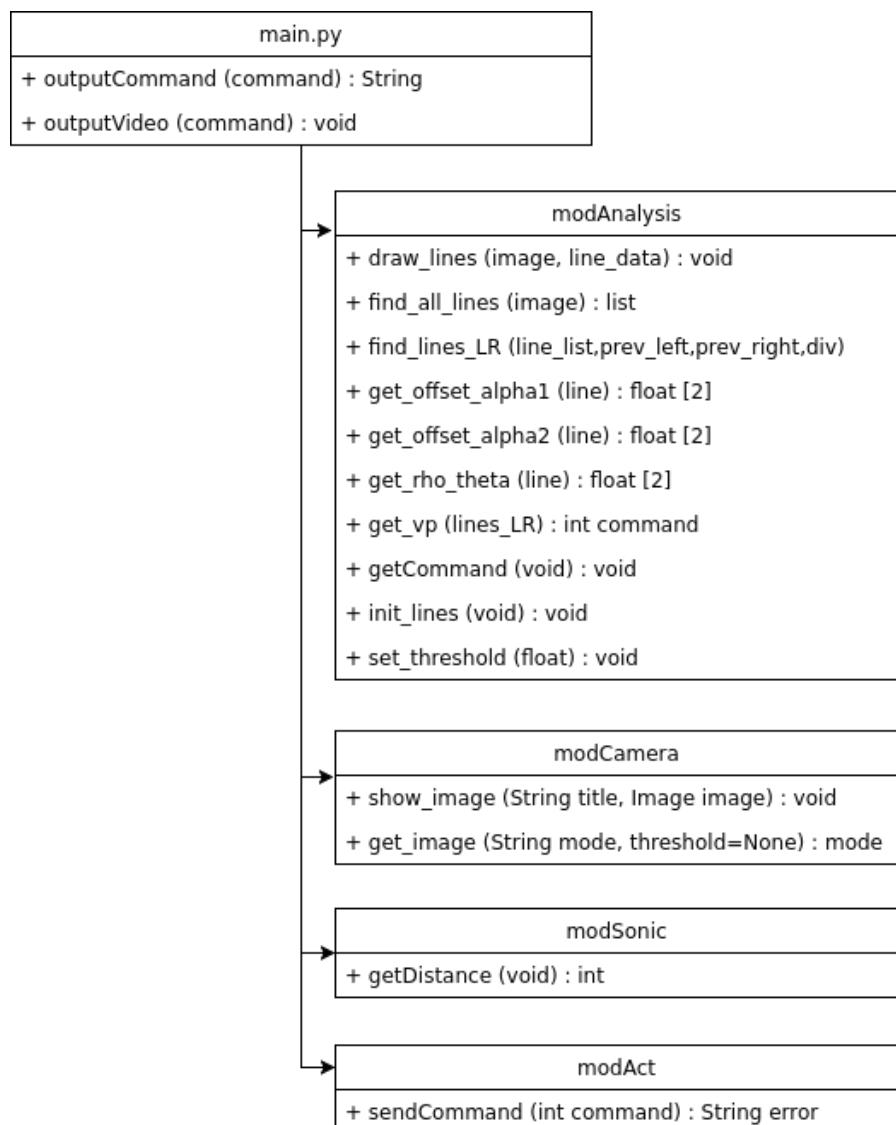


Figure 4: Aufbau des Python Codes

## 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	...
pivideoostream	...
picamera	...
threading	...
RPi.GPIO	Bibliothek zur Ansteuerung der GPIO ports des Raspberry 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

## 6 Werkzeuge der Bildverarbeitung

Die folgende Einleitung in die Thematik basiert auf der Lektüre des Buches "Digitale Bildverarbeitung" von Burger, Burge (?), das als Literatur für den Einstieg empfohlen wird. Eine zusammenfassende Wiedergabe für das Projekt relevanter Themen ist im Folgenden nachzulesen.

Ein Bestandteil der Bildverarbeitung ist die Bildanalyse, bei der es darum geht, sinnvolle Informationen aus Bildern zu extrahieren. Genauer ist der Bereich der Computer Vision gefragt, die Sehvorgänge des Menschen in der dreidimensionalen Welt zu mechanisieren.

Die Steuerung des Wagens soll durch Informationen aus den Kamerabildern erfolgen: Durch Erkennen der Fahrbanhlinien soll die Lenkung innerhalb der Spur geregelt werden.

Dieser Prozess lässt sich beschreiben mit:

1. Digitalisierung mit Hilfe der Kamera

2. Vorverarbeitung (Bildverbesserung bzw. Anpassung an den Zweck) durch Umwandlung in ein binäres Canny-Edges-Bild
3. Segmentierung eine Vorauswahl des Bildausschnittes 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(ergänzen Fabian?). Um ein optimales Bild zu erhalten sind die Beleuchtungsumstände zu beachten, da hierdurch die Differenzierung von Linien im Bild beeinflusst wird. Beachte timing?

In der Bildverarbeitung ist der Nullpunkt der x- und y-Achse in der linken oberen Ecke des Bildes definiert, was für ein eindeutiges Anwenden von Prozessen und daraus generierten Informationen relevant ist.

## 6.2 Vorverarbeitung- 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 erfolgt, wie z. B. bei einem weißen Farbahnstreifen zwei Kannten erkannt werden sollen.

Die Canny-Edges-Funktion aus der openCV Library ist ein bewährter Algorithmus zur Kantenerkennung, da drei Ziel gleichzeitig erreicht werden drei Zile: ein zuverlässiges Detektieren vorhandener Kanten, die Position der Kante präzise zu bestimmen, und Farbsprünge, die nicht als Kante interoretiert werden sollen, auszulassen.

## 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 Erkennung 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.

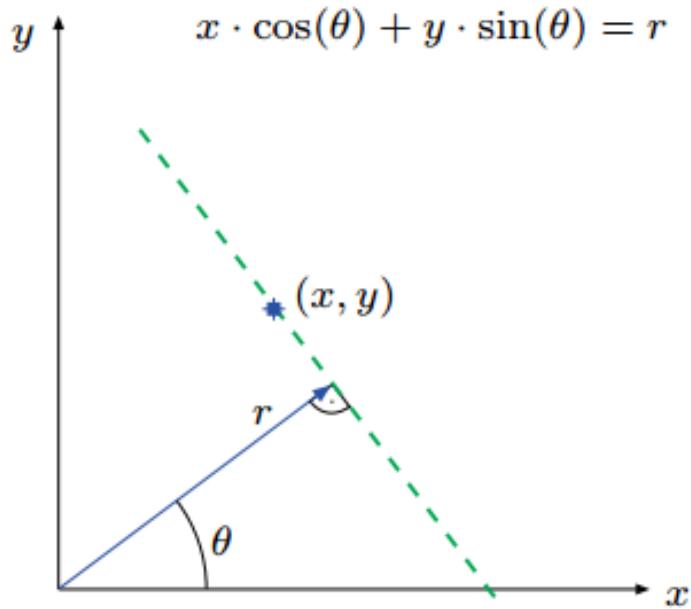


Figure 5: gradeThetaR.png

Dabei kommt zur Anwendung, dass eine Gerade mathematisch sowohl mit  $y = mx + b$  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 Ursprung, 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 Hough-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 Programm 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,

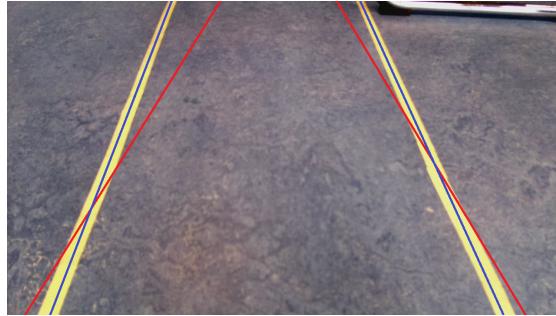


Figure 6: Beispiel für die erste Form der Geradenrepräsentation

die dann korrigiert und an die erkannten Linien angepasst werden. In Abbildung 6 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 Durchschnitt 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 7 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.

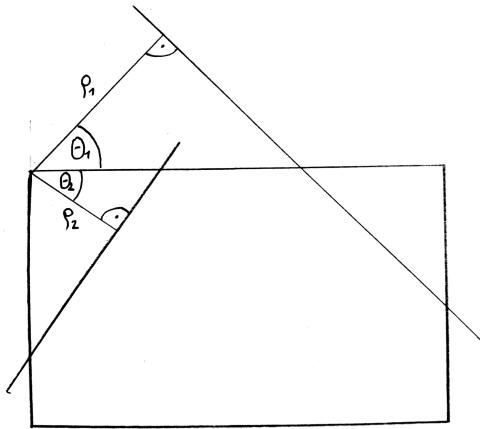


Figure 7: Beispiel für die erste Form der Geradenrepräsentation

Eine Schwäche dieser Repräsentationsform wird darin liegen, dass zwei sehr ähnliche Geraden komplett verschiedene Winkel  $\theta$  haben können. Dies wird in Abbildung 8 deutlich.  $\theta_1$  beträgt hier ca.  $70^\circ$ ,  $\theta_2$  ca.  $-100^\circ$ .

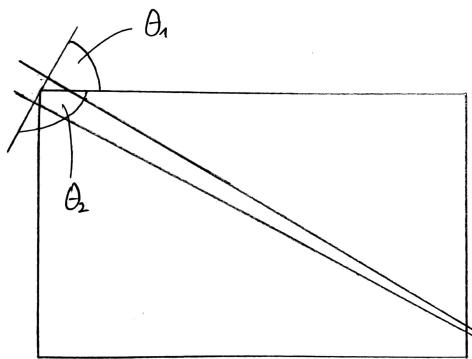


Figure 8: Beispiel für die erste Form der Geradenrepräsentation

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  $\theta < 90^\circ$ :

$$\begin{aligned}\omega &= -\frac{\rho}{\cos 90^\circ - \theta} \\ \alpha &= 90^\circ - \theta\end{aligned}$$

Für Winkel  $\theta > 90^\circ$ :

$$\begin{aligned}\omega &= -\frac{\rho}{\cos \theta - 90^\circ} \\ \alpha &= \theta - 90^\circ\end{aligned}$$

Abbildung 9 veranschaulicht die Repräsentation der beiden Fahrbahnlinien in dieser Form.

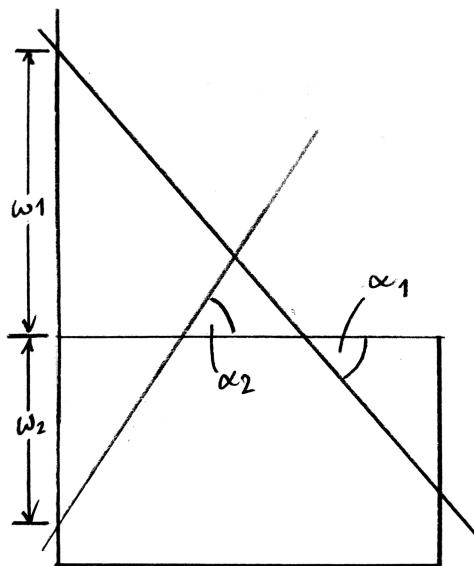


Figure 9: Beispiel für die erste Form der Geradenrepräsentation

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 10 erkennen lässt. Hier sieht man, dass sich der Offset  $\omega$  stark ändert, wenn sich der Winkel der Geraden leicht ändert.

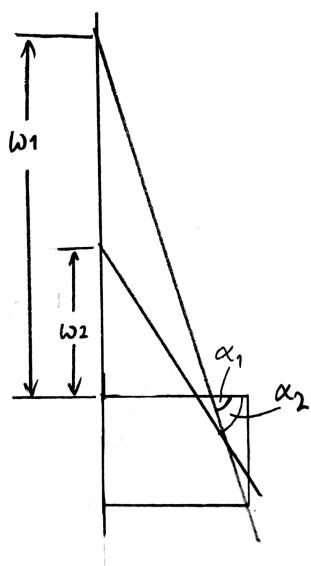


Figure 10: Beispiel für die erste Form der Geradenrepräsentation