

Aufgabe 2

Gruppe 4

Jonas Eckhoff Anton Jabs Florian Brach Felix Kieckhäuser

10. Dezember 2018

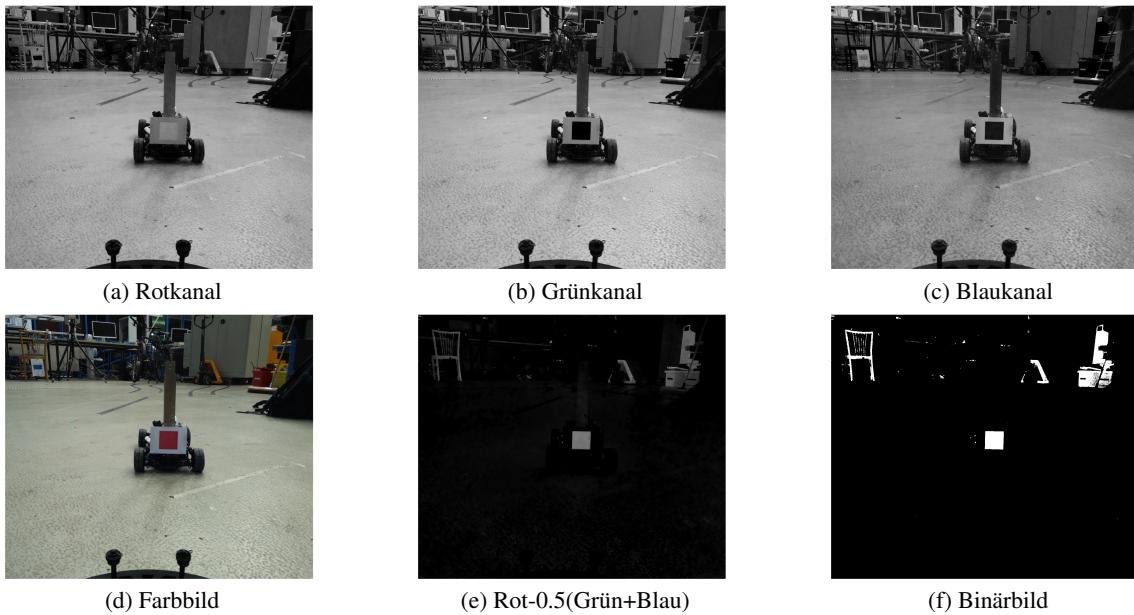
Es soll ein Abstandsregler für das Modellfahrzeug implementiert werden. Um einen Abstand messen zu können und sich in der Welt orientieren zu können, benötigt das Fahrzeug eine Mustererkennung. Außerdem wird ein Geschwindigkeitsregler benötigt, um den Abstand halten zu können. Im Folgenden beschreiben wir unsere Lösung für dieses Problem.

1 Die Bilderkennung

Das Ziel der Bilderkennung ist es, zwei Punkte zu identifizieren, von denen man den Abstand in den Weltkoordinaten kennt. Anschließend kann man dann durch eine Koordinatentransformation von den Bildkoordinaten zu den Weltkoordinaten die Position der Kamera in der Welt in Relation zum bekannten Objekt bestimmen. Für dieses Objekt haben wir ein rotes Rechteck gewählt.

1.1 Bildvorverarbeitung

Zuerst soll das Bild so angepasst werden, dass das rote Rechteck einfacher identifizierbar wird. Normalerweise würde man damit beginnen, das Bild erstmal vorzglätten (zum Beispiel mit einem Gauss-Filter) um Rauschen zu minimieren. Dies hat sich aber aus Performancegründen als nicht so hilfreich herausgestellt. Stattdessen kann hier gut die Eigenschaft, dass das Rechteck rot ist, verwendet werden. Wenn man sich die RGB-Kanäle (a, b und c) anschaut, fällt auf, dass das rote Rechteck sehr hell im roten Kanal ist, allerdings fast gar nicht in dem blauen und grünen Kanal sichtbar ist. Alle anderen Farben haben auch einen Blau- oder Grünanteil. Aus diesem Grund wird im ersten Schritt ein Graustufenbild (e) erzeugt, in dem der blaue und grüne Kanal vom roten Kanal abgezogen werden.



In diesem Graustufenbild (e) sind nun hauptsächlich helle Rottöne sichtbar. Nun kann ein Binärbild (f) erstellt werden, indem alle Grautöne über einem Schwellenwert weiß dargestellt werden und alle anderen schwarz.

1.2 Rechteckserkennung

Nun sollen die Eckpunkte des Rechteckes gefunden werden. Im ersten Schritt wird ein Gitter über das Bild gelegt, das so fein gewählt wird, dass ein Gitterpunkt auf jeden Fall im Rechteck liegen muss. Dafür wird angenommen, dass das Rechteck eine Mindestgröße im Bild hat und nicht zu weit weg ist. Ob ein Gitterpunkt (x_1, y_1) nun wahrscheinlich in einem Rechteck liegt, wird mithilfe des Algorithmus 1 festgestellt.

Dies bestimmt den Punkt (x_1, y_2) und (x_1, y_3) . Anschließend lässt sich der Mittelpunkt $(x_1, y_m) = (x_1, \frac{y_2+y_3}{2})$ bestimmen. Nun wird der gleiche Algorithmus nochmal in x-Richtung durchgeführt mit Startwert (x_1, y_m) um x_l und x_r zu finden. Um leicht schräge Seiten auszugleichen, wird nun der Algorithmus auf $(x_l + a, y_m)$ und $(x_r - a, y_m)$ angewandt, um die die vier Rechteckecken $(x_{ol}, y_{ol}), (x_{or}, y_{or}), (x_{ul}, y_{ul})$ und (x_{ur}, y_{ur}) zu finden.

Nun wird noch überprüft, ob die gefundenen Ecken ein sinnvolles Rechteck bilden. Dafür darf $y_{ol} - y_{ul}$ nicht mehr als 30% größer oder kleiner als $y_{or} - y_{ur}$ oder $x_r - x_l$ sein, da das Rechteck leicht nach hinten gekippt sein kann (vorausfahrendes Auto lenkt). Außerdem soll noch $\frac{y_m - y_{ol}}{y_m - y_{or}}$ ungefähr $\frac{y_{ul} - y_m}{y_{ur} - y_m}$ sein. Wenn diese Bedingungen eingehalten werden, handelt es sich wahrscheinlich um ein Rechteck.

Algorithmus 1 Pseudo-Code

```

1:  $(x, y) = (x_1, y_1)$ 
2: while  $I(x, y) = \text{white}$  do
3:    $y = y - 1$ 
4: end while
5:  $y_2 = y$ 
6:  $(x, y) = (x_1, y_1)$ 
7: while  $I(x, y) = \text{white}$  do
8:    $y = y + 1$ 
9: end while
10:  $y_3 = y$ 

```

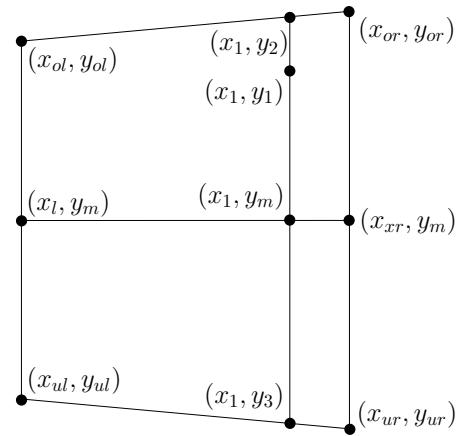


Abbildung 1: Beispiel-Rechteck

Um möglichst schnell den richtigen Startpunkt (x_1, y_1) zu finden, wird spiralförmig um den Mittelpunkt des im letzten Bild gefundenen Rechtecks gesucht.

2 Umrechnung der Pixelkoordinaten in Weltkoordinaten

Nachdem wir die Eckpunkte des gesuchten Rechtecks im Bild der Kamera gefunden haben, möchten wir nun die Orientierung ${}^A\xi_B$ des vorausfahrenden Autos B gegenüber dem Auto A bestimmen. Die Koordinatensysteme $\{A\}/\{B\}$ seien dafür mittig zwischen den Hinterrädern auf Höhe des Bodens plaziert, wobei die x-Achse in Fahrtrichtung und die z-Achse in den Himmel zeigt. Die Orientierung des Kamerakoordinatensystems $\{C\}$ zum Auto A ist bekannt und wird mit ${}^A\xi_C$ bezeichnet. Außerdem ist die intrinsische Matrix K der Kamera bekannt und hat folgende Form:

$$K = \begin{pmatrix} f/\rho_\omega & 0 & u_0 \\ 0 & f/\rho_h & v_0 \\ 0 & 0 & 1 \end{pmatrix}$$

Sei nun die homogene Pixelkoordinate $\tilde{p} = (u, v, 1)^T$ eines Punktes gegeben und dessen homogene Koordinate ${}^A\tilde{P} = \lambda(X, Y, Z, 1)^T$, $\lambda \in \mathbb{R}$ bezüglich des Autos A gesucht. Durch K und homogener Koordinatentransformationsmatrix CT_A von ${}^C\xi_A$ lässt sich folgende Gleichung aufstellen:

$$\tilde{p} = K \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} {}^CT_A {}^A\tilde{P}$$

Durch die Rechtecksmatrix ist die Bestimmung von ${}^A\tilde{P}$ wegen dem unbekannten Faktor λ nicht eindeutig möglich:

$${}^AT_C \begin{pmatrix} K^{-1}\tilde{p} \\ \lambda \end{pmatrix} = {}^A\tilde{P} \quad (1)$$

Deswegen benutzen wir zusätzlich die Annahme, dass die Höhe Z des Punktes bekannt ist und können die dritte Zeile der Gleichung nach λ umformen:

$$\lambda = \frac{e_3^T {}^AT_C K^{-1} \tilde{p}}{Z - ({}^AT_C)_{3,4}}$$

Dies setzte vorraus, dass die Höhe Z des Punktes nicht mit der Höhe der Kamera $({}^AT_C)_{3,4}$ übereinstimmt. Nachdem λ errechnet worden ist, kann ${}^A\tilde{P}$ durch Gleichung 1 berechnet werden.

Unter der Annahme, dass das Auto A in der gleichen Ebene wie das Auto B liegt und wir zwei Punkte \tilde{p}_1 und \tilde{p}_2 mit der Kamera erkannt haben, von denen wir die Koordinaten ${}^B\tilde{P}_1$ und ${}^B\tilde{P}_2$ bezüglich $\{B\}$ kennen, kann nun ${}^A\xi_B$ bestimmt werden. In unserer Implementierung benutzen wir hierfür die unteren beiden Eckpunkte des roten Rechtecks.

Um die Annahme der gleichen Ebene zu vermeiden, kann auch ein Gleichungssystem für mehrere Punkte durch Gleichung 2 und deren Koordinaten bezüglich $\{B\}$ ausgestellt werden. Dies erfordert aber eine aufwändige Betrachtung

der eindeutigen Lösbarkeit.

3 Abstandsregler

3.1 Simulink-Modell

Für die Abstandsregelung werden die zuvor beschrieben Objekterkennung und die Matrixtransformation genutzt. Aus dem RGB-Eingangsvideosignal wird der Rotkanal durch Subtraktion der beiden anderen Kanäle extrahiert und anschließend in ein Schwarzweißbild konvertiert. In der *finderechteck*-Funktion werden die vier Eckpunkte des Quadrats ermittelt. Dabei wird nur ein Signal weitergeleitet, wenn ein Quadrat gefunden wurde. Mit der vorher definierten Höhe des Quadrats über dem Boden, den Kameraparametern und der Transformationsmatrix können die Weltkoordinaten der beiden unteren getrackten Punkte bestimmt werden. Nach der Mittlung und Trennung der Signale wird die x -, und y -Position ausgegeben.

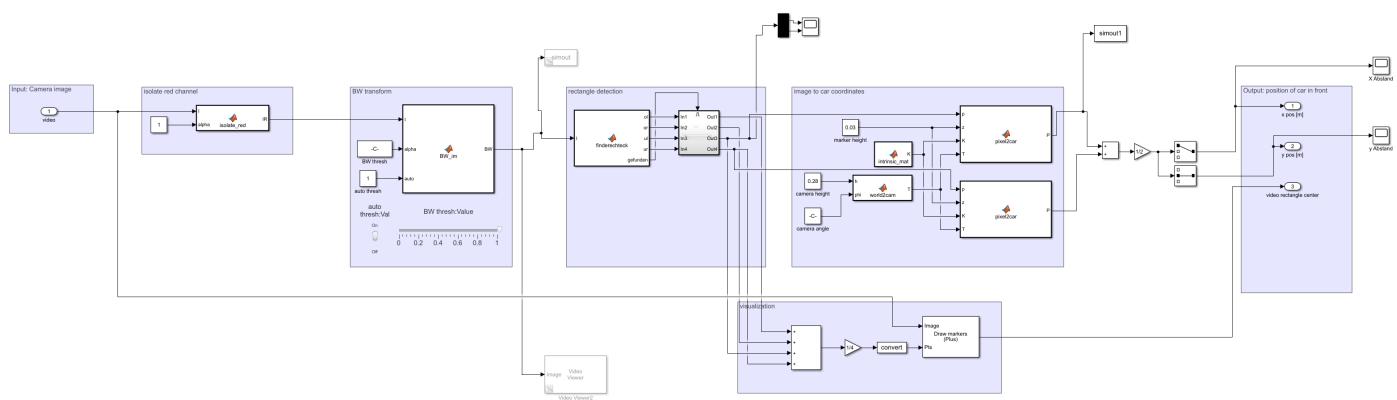


Abbildung 2: Abstandsbestimmung Kamera-Objekt in Simulink

3.2 Abstandsregler

Der implementierte Abstandsregler nutzt die zuvor berechneten Abstände zum getrackten Objekt. Der Abstand in Fahrtrichtung wird durch $x pos$ beschrieben, der Abstand quer zur Fahrtrichtung durch $y pos$. Für die Abstandsregelung wird der aktuelle Abstand mit einer Zieldistanz verglichen, mittels eines P-Regler geregelt und auf Werte zwischen -0.5 und 0.5 begrenzt. Um ein Rückwärtsfahren bei Unterschreitung des geforderten Abstands zu gewährleisten, wird bei einem negativen Wert die Drehrichtung des Motors umgedreht.

Für die Regelung des Lenkungswinkels wird der gemessene Querabstand durch eine Verstärkung auf einen Wert zwischen -1 und 1 gebracht.

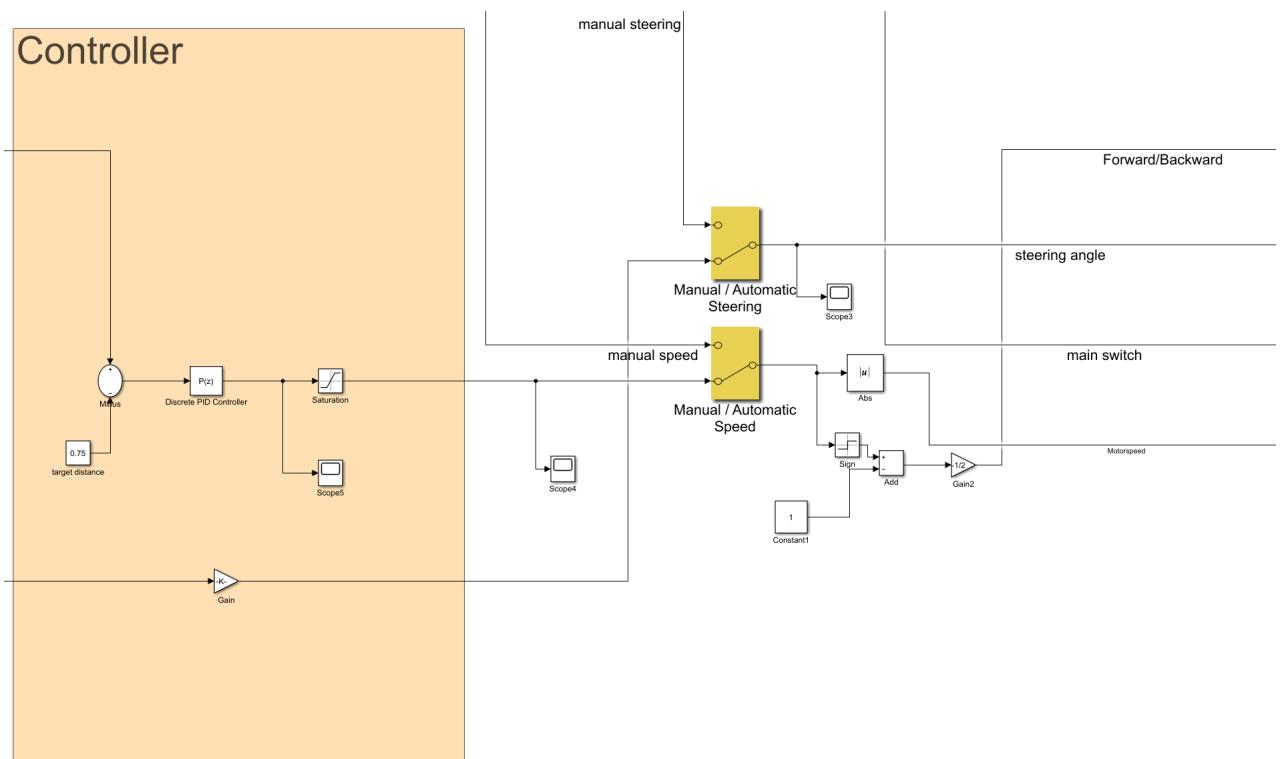


Abbildung 3: Abstandsregler in Simulink