Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Waldemar Heck

Hindernis- und Kreuzungserkennung auf autonomen Fahrzeugen durch Kamera und Infrarotsensorik

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Waldemar Heck

Hindernis- und Kreuzungserkennung auf autonomen Fahrzeugen durch Kamera und Infrarotsensorik

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. rer. nat. Stephan Pareigis
Zweitgutachter : Prof. Dr. Zhen Ru Dai

Abgegeben am 27.05.2013

Waldemar Heck

Thema der Bachelorarbeit

Hindernis- und Kreuzungserkennung auf autonomen Fahrzeugen durch Kamera und Infrarotsensorik

Stichworte

FAUST, autonomes Fahrzeug, Hinderniserkennung, Kreuzungserkennung, Bildverarbeitung, Schwellwertbestimmung, Binarisierung, Region Of Interest, Infrarotsensoren, Triangulation, Carolo-Cup

Kurzzusammenfassung

In dieser Arbeit werden Verfahren zur Hindernis- und Kreuzungserkennung für ein autonomes Modelfahrzeug entwickelt. Dies geschieht innerhalb des Forschungsprojektes FAUST der HAW Hamburg. Eingesetzt wird das Fahrzeug im Carolo-Cup Wettbewerb der Technischen Universität Braunschweig. Als Sensoren werden eine monochrome 0,36 Megapixel Kamera und Infrarot-Entfernungsmesser verwendet. Die statischen und dynamischen Hindernisse auf dem Rundkurs werden mittels Infrarotsensoren erkannt und mit der Fahrzeugkamera verifiziert. Eine Straßenkreuzung wird anhand der Haltelinie erkannt. Hindernisse und Haltelinie unterscheiden sich in ihren Grauwerten signifikant von der Fahrbahn. Um diese im Kamerabild zu erfassen wird eine ROI (*Region Of Interest* = Bereich von Interesse) in das Bild gelegt. Die ROI passt sich während der Fahrt dynamisch der Fahrbahn an. Innerhalb der ROI werden Bildpixel dann entsprechend ihrem Grauwert klassifiziert und entweder den Hindernissen, der Haltelinie oder der Fahrbahn zugewiesen.

Title of the paper

Obstacle and crossing detection for autonomous vehicle by camera and infrared sensors

Keywords

FAUST, autonomous vehicle, obstacle detection, crossing detection, image processing, thresholding, binarization, region of interest, infrared sensors, triangulation, Carolo-Cup

Abstract

For the Carolo-Cup competition of the Technical University of Braunschweig methods for obstacle and crossing detection on an autonomous vehicle have been developed in this thesis. The autonomous vehicle is being developed as part of the FAUST research project (FAUST stands for Driver Assistance and Autonomous Systems) at the University of Applied Sciences in Hamburg. As sensors, a 0.36 megapixel monochrome camera and infrared rangefinders are used. The static and dynamic obstacles on the road are detected by infrared sensors and verified with the vehicle camera. A crossing is detected by stop line. The grayscale values of obstacles and stop line differ significantly from the road surface. To recognize them a region of interest (ROI) is placed into the camera image. The ROI adapts dynamically the road while the vehicle is driving. Inside the ROI pixels are classified according their grayscale value and assigned to the obstacles, the stop line or the road.

Inhaltsverzeichnis

1 Einleitung

Der Einsatz autonomer Systeme gewinnt in der heutigen industrialisierten Gesellschaft zunehmend an Bedeutung. Der Wunsch nach Maschinen, Robotern oder Fahrzeugen, die sich in bekannter oder unbekannter Umgebung autonom bewegen können wird immer größer. Zur kollisionsfreien und selbständigen Navigation ist eine funktionierende Positionsbestimmung und Hinderniserkennung in Echtzeit durch das jeweilige autonome System unerlässlich. Dazu benötigt das Fahrzeug eine Sensorik (analog dem menschlichen Auge), die es ihm ermöglicht zu "sehen". Um diesen Anspruch zu erfüllen, gibt es verschiedene Ansätze bezüglich der Ausstattung der Systeme mit Sensoren. So werden Radarsysteme, Laserscanner, Ultraschallsystem und Kameras (Bildverarbeitung) zur Umgebungswahrnehmung eingesetzt [RIES 2005].

Zu den bereits erhältlichen oder in naher Zukunft verfügbaren Systemen in autonomen Fahrzeugen gehören unter anderem das Spurhalten des Fahrzeugs, langsames Stop-and-Go-Fahren in Staus sowie das Lesen und Interpretieren von Verkehrsschildern und Verkehrssituationen [Wirtschaftsspiegel 2006] [Vacek 2009].

Gefördert wird die Forschung an autonomen Fahrzeugen unter anderem von der DARPA [DARPA], einer Behörde des Verteidigungsministeriums der Vereinigten Staaten. Diese sponsert verschiedene Wettbewerbe, um die Entwicklung von autonomen Fahrzeugen voran zu treiben. Einer der Halbfinalteilnehmer der DARPA Urban Challenge 2007 war das autonome Fahrzeug "Caroline" von der TU Braunschweig [Caroline 2007].

Auch hierzulande setzt sich die TU Braunschweig mit dem jährlich stattfindenden Hochschulwettbewerb "Carolo-Cup" für die Forschung an autonomen Fahrzeugen ein. Bei diesem Wettbewerb treten Studierendenteams verschiedener Hochschulen mit autonom fahrenden Modellfahrzeugen in den Disziplinen *Rundkurs ohne Hindernisse*, *Rundkurs mit Hindernissen* und *paralleles Einparken* gegeneinander an [Carolo-Cup Regelwerk (2013)].

Gegenstand dieser Arbeit ist die Entwicklung einer Methode zur Hindernis- und Kreuzungserkennung auf dem autonomen Modelfahrzeug "Saphir", das im Rahmen des FAUST-Projektes (Fahrerassistenz- und Autonome Systeme) der Hochschule für Angewandte Wissenschaften Hamburg entwickelt wurde. Ziel ist es dynamische und statische Hindernisse auf dem Rundkurs des Carolo-Cup Wettbewerbs zu erkennen und diese berührungslos zu überholen. Ein weiteres Ziel ist die Erkennung von Kreuzungen und derer Vorfahrtssituation. Als Sensoren werden eine monochrome 0,36 Megapixel Kamera von IDS und Infrarot-Entfernungsmesser der Sharp Electronics GmbH verwendet. Die Sharp Infrarotsensoren der GP2-Serie verfügen über das Messprinzip der aktiven Triangulation, weshalb die Entfernungsmessung nahezu unabhängig von der Farbe und Helligkeit des gemessenen Objektes erfolgt und nur gering von der Intensität des reflektierten Infrarotstrahls abhängt (vgl. Abschnitt 5.3).

Durch dieses Messprinzip werden die in [Robotik 2008 Seite 36-39] aufgezählten Nachteile der herkömmlichen Infrarot -und Ultraschallsensoren kompensiert.

Kapitel 2 behandelt die Grundlagen, wie das zugreifen auf die Fahrzeugkamera und das Auslesen von Grauwerten mit Hilfe von OpenCV (Open Source Computer Vision), sowie die Benutzung der FAUST Web-Oberfläche mit der auf dem Fahrzeug laufende Tasks und deren Parameter eingestellt werden können.

Kapitel 3 zeigt wie eine Region Of Interest (ROI) in das Bild der Fahrzeugkamera gelegt wird und mit Hilfe von Lines Of Interest (LOIs) und einem geeigneten Schwellwert ressourcenschonend Hindernisse von der Fahrbahn getrennt werden und die Steigung erkannter Kanten innerhalb einer ROI berechnet werden können.

Kapitel 4 beinhaltet den Aufbau und die Funktionsweise des hierarchischen Zustandsautomaten für die Hindernis -und Kreuzungserkennung, sowie der beiden dazugehörenden Subautomaten.

Kapitel 5 beschreibt wie mit Hilfe einer dynamischen ROI im Kamerabild, sowie den vorderen Infrarotsensoren, statische und dynamische Hindernisse auf der Fahrbahn erkannt werden können. Im Abschnitt 5.6 werden die Einstellparameter für die Hinderniserkennung beschrieben und erläutert.

Kapitel 6 behandelt das Erkennen der Haltelinie mit Hilfe einer dynamischen ROI, die Differenzierung zwischen Haltelinie und Startlinie, sowie das Erkennen der Vorfahrtssituation an Kreuzungen. Die Einstellparameter für die Kreuzungserkennung werden im Abschnitt 6.5 beschrieben und erläutert.

Kapitel 7 beinhaltet Test und Evaluation der in Kapitel drei bis sechs entwickelten Algorithmen. Abschließend wird diese Arbeit durch ein Fazit in **Kapitel 8** abgerundet.

2 Grundlagen

In diesem Kapitel sind die Grundlagen zusammengefasst die das Anwenden der in Kapiteln drei bis sechs vorgestellten Methoden erleichtern sollen.

Zunächst wird im Abschnitt 2.1 das zugreifen auf die Fahrzeugkamera und das Auslesen von Grauwerten mit Hilfe von OpenCV (Open Source Computer Vision) erläutert. Im Anschluss folgt die Definition eines Histogramms Abschnitt 2.2. Abschließend wird im Abschnitt 2.3 die Benutzung der FAUST Web-Oberfläche erläutert, mit der auf dem Fahrzeug laufende Tasks und deren Parameter eingestellt werden können.

2.1 OpenCV (Open Source Computer Vision)

OpenCV (Open Source Computer Vision) ist eine freie Programmbibliothek mit Algorithmen für die Bildverarbeitung und maschinelles Sehen. Es ist eine in C/C++ geschriebene Library, welche 1999 als Projekt der Firma Intel gestartet wurde um die bis dahin vorhandenen Bildverarbeitungs- und Erkennungsinfrastrukturen zu ordnen. Inzwischen wird OpenCV quelloffen unter einer BSD-Lizenz von Willow Garage weiterentwickelt. Das OpenCV Wiki beschreibt die Zielsetzung folgendermaßen: *"OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real time computer vision. Example applications of the OpenCV library are Human-Computer Interaction (HCI); Object Identification, Segmentation and Recognition; Face Recognition; Gesture Recognition; Motion Tracking, Ego Motion, Motion Understanding; Structure From Motion (SFM); and Mobile Robotics."* [OpenCV Wiki]

Die Echtzeitanforderungen spiegeln sich besonders darin wider, dass OpenCV insbesondere auf die Benutzung von Intel-Prozessoren mit Integrated Performance Primitives (IPP) abgestimmt wurde. Darüber hinaus bietet OpenCV Funktionalitäten auf mehreren Ebenen. In der Abbildung 2.0.1 werden diese Pakete und ihre Zusammenhänge beschrieben. Das Paket `CxCore` enthält die grundsätzlichen Datenstrukturen, die für die Arbeit mit OpenCV benötigt werden, insbesondere Punkte, Skalare, Matrizen und Bilder. Es werden aber auch komplexere Datenstrukturen wie Bäume oder Sequenzen angeboten. Des Weiteren enthält `CxCore` die grundlegenden Operationen, um auf diesen Datenstrukturen zu arbeiten. Dies sind u. a. auch Methoden der linearen Algebra und Zeichenfunktionen auf Bildern. Auf höherer Ebene stehen die Pakete `Cv` und `CvAux`, sowie `Highgui`. `Highgui` bietet Plattform- und Window-Manager unabhängige Möglichkeiten zum Erzeugen einfacher GUIs und zum Laden und Speichern von Bildern und Videos. `Cv` und `CvAux` enthalten die Algorithmen der Bildbearbeitung und -ver-

arbeitung. Cv enthält dabei vor allem Standardalgorithmen, deren Implementierung sehr stabil und effizient ist [Wienke 2008].

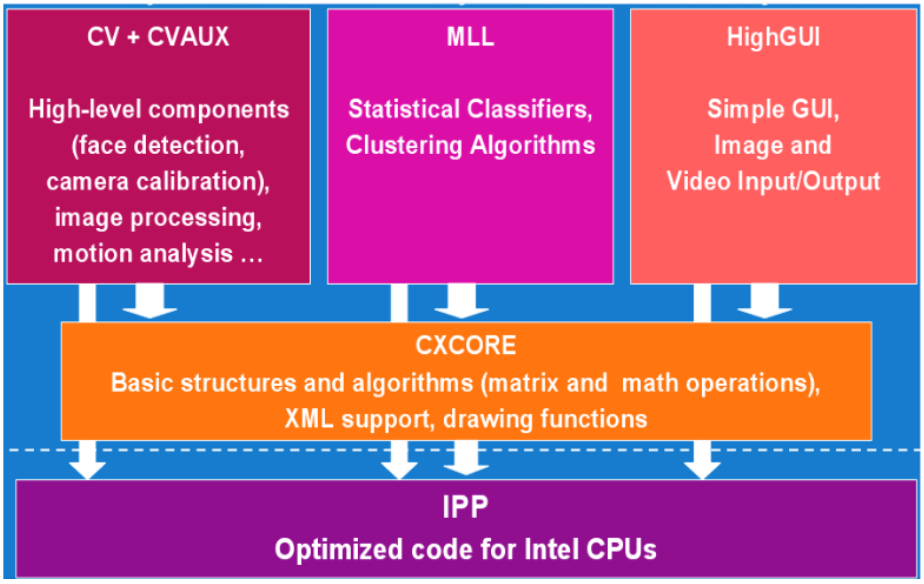


Abbildung 2.0.1: Aufteilung von OpenCV in Aufgabengebiete [Wienke 2008]

Auf dem autonomen Fahrzeug "Saphir" ist die OpenCV Version 2.1.0 installiert. Der Zugriff auf das aktuelle Bild der Fahrzeugkamera erfolgt über den DataContainer:

```
CameraImagePtr camImg = DataContainer<CameraImage>::instance().getData();

IplImage* img = camImg->getImage();
```

Das nachfolgende Beispielprogramm zeigt wie ein 8 bit Graustufen Bild (vgl. Einleitung Kapitel 3) mit OpenCV geladen, auf Grauwerte zugegriffen und wie der Grauwert einzelner Pixel verändert werden kann:

```
#include <iostream>
#include <cv.h>
#include <highgui.h>
#include <stdio.h>
using namespace std;

int main() {

    /* data structure for the image */
    IplImage *img = 0;
    cout<<"lese image ein!"<<endl;
    /* load the image, use CV_LOAD_IMAGE_GRAYSCALE to load image in grayscale */
    img = cvLoadImage( "/home/developer/img.bmp", CV_LOAD_IMAGE_GRAYSCALE);
    if(!img) {
       cerr <<"Could not load image!\n"<<endl;
       exit(EXIT_FAILURE);
    }
    /*For a single-channel byte image:*/
    CvScalar s;
```

```
    s=cvGet2D(img,y,x); // get the (y,x) pixel value
    printf("Grauwert = %f\n",s.val[0]);
    s.val[0]=111;
    cvSet2D(img,i,j,s); // set the (i,j) pixel value

    /* create a window */
    cvNamedWindow( "image", CV_WINDOW_AUTOSIZE );
    /* display the image */
    cvShowImage( "image", img );
    /* wait until user press a key */
    cvWaitKey(0);
    /* free memory */
    cvDestroyWindow( "image" );
    cvReleaseImage( &img );

    return 0;
}
```

2.2 Histogramm

Das Histogramm repräsentiert die Häufigkeitsverteilung der Grauwerte in einem Bild. Bei einem 2-bit-tief digitalisierten Bild (vgl. Abbildung 2.2.1) können die Grauwerte nur die vier Zahlenwerte von 0 bis 3 annehmen. Bei der Ermittlung der Häufigkeitsverteilung wird ein Vektor mit vier Spalten erstellt, eine für jeden der Grauwerte 0 bis 3, und in jeder Spalte mit dem Eintrag n = 0 initialisiert. Dann wird das gesamte Bild Pixel für Pixel durchlaufen. Wenn ein Pixel den Grauwert g hat, wird der Eintrag in der Spalte g der Tabelle von n auf n+1 erhöht. Wenn das Bild vollständig durchlaufen ist, steht in jeder Spalte des Vektors an der Stelle g die absolute Häufigkeit n(g), mit der dieser Grauwert im Bild vorkommt. Diese diskrete Funktion n(g) ist die Häufigkeitsverteilung der Grauwerte dieses Bildes [Hackenkamp 2001].

$h(g) = n_g$ mit g : *Grauwert*

n_g : *Anzahl der Bildpunkte, die diesen Grauwert haben*

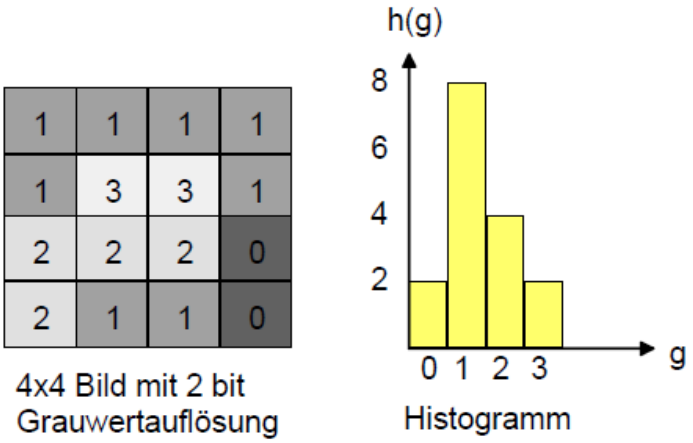


Abbildung 2.2.1: Histogramm eines 4x4 Bildes mit 2 bit Grauwertauflösung [Meisel RV02 2012 S. 15]

2.3 FAUST Web-Oberfläche

Die FAUST Web-Oberfläche dient dazu die Einstellparameter der einzelnen Tasks auf dem autonomen Fahrzeug "Saphir" anzupassen. Zugegriffen wird auf die FAUST Web-Oberfläche mit einem Web-Browser und der IP-Adresse des jeweiligen autonomen Fahrzeugs (vgl. Abbildung 2.3.1 *a.*). Der Saphir hat die IP-Adresse: `192.168.1.33` somit wird auf die FAUST Web-Oberfläche des Saphirs folgendermaßen zugegriffen:

`http://192.168.1.33:8080/`

Auf der geladenen Seite erscheinen die Treiber (`Driver`) die dazugehörige Beschreibung (`Description`) und der Status des jeweiligen Treibers (`active`). Hier können die Parameter für die Kameratreiber (`UEyeDriver`) eingestellt werden. Mit einem Klick auf das Plus Zeichen neben `UEyeDriver` öffnet sich eine Liste mit den in alphabetischer Reihenfolge sortierten Parametern für die Kameratreiber. Es muss drauf geachtet werden, dass vor jedem Start des autonomen Fahrzeugs die Treiber `RS232CaroloDriver` und `UEyeDriver` auf `active` geschaltet sind (vgl. Abbildung 2.3.1 *b.*).

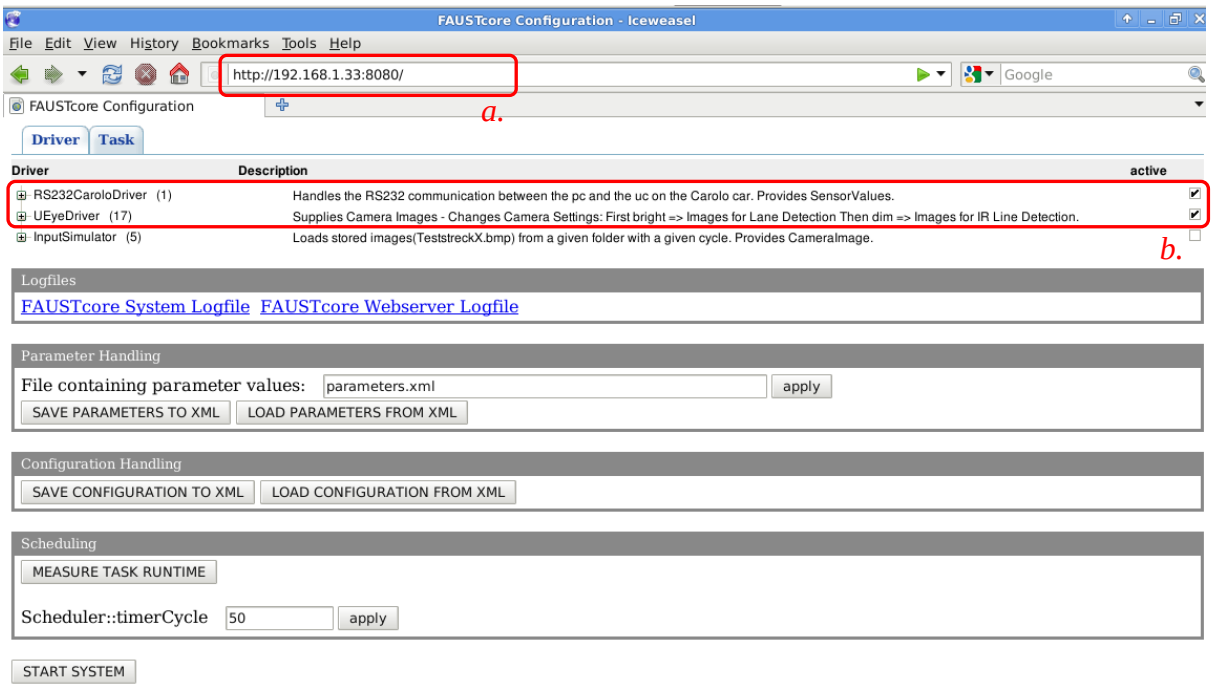


Abbildung 2.3.1: FAUST Web-Oberfläche mit Registerkarte für die Treibereinstellungen

Mit einem Klick auf Task wird eine Liste mit den zurzeit auf dem Fahrzeug befindlichen Tasks geladen (vgl. Abbildung 2.3.2 *a.*). Diese Liste ist alphabetisch geordnet. Hier können die Parameter der einzelnen Tasks angepasst werden. Die Parameter für die Hindernis -und Kreuzungserkennung befinden sich unter dem Unterpunkt "OAandCR" (Obstacle Avoidance and Crossing Recognition). Mit einem Klick auf das Plus Zeichen öffnet sich eine Liste mit den in alphabetischer Reihenfolge sortierten Parametern (vgl. Abbildung 2.3.2 *b.*).

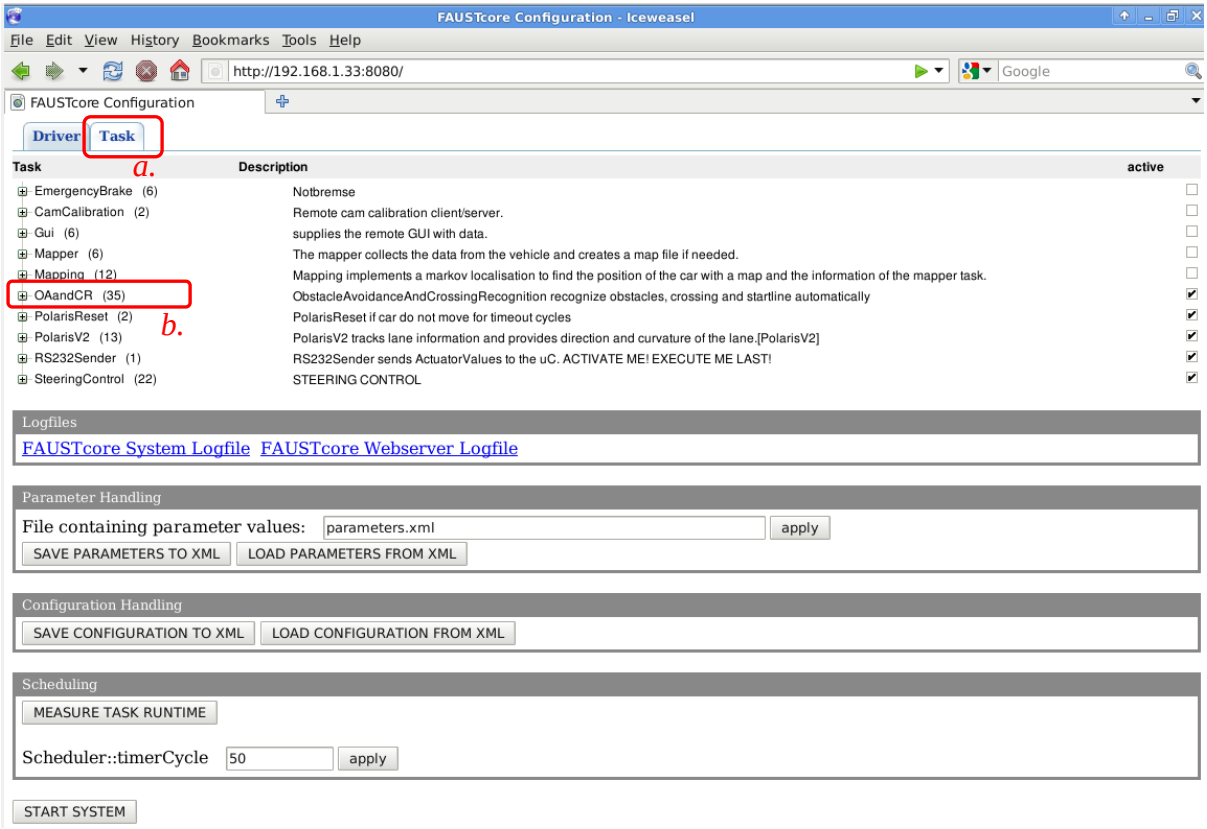


Abbildung 2.3.2: FAUST Web-Oberfläche mit Registerkarte für die Taskeinstellungen

In diesem Abschnitt werden nur die zwei allgemeinen Parameter für die Hindernis -und Kreuzungserkennung erläutert. Die speziellen Parameter für die Hinderniserkennung befinden sich im Abschnitt 5.6, die speziellen Parameter für die Kreuzungserkennung befinden sich im Abschnitt 6.5.

Allgemeine Einstellparameter der Hindernis -und Kreuzungserkennung:

Mit `enableCOUT` wird die Menge der debug-Ausgaben im Terminal eingestellt:

2 = alle debug-Ausgaben für die Hindernis -und Kreuzungserkennung werden angezeigt
1 = Kreuzung erkannt und Startlinie erkannt werden im Terminal angezeigt
0 = es werden keine debug-Ausgaben im Terminal angezeigt

Mit `enableOAandCRreaction` wird die Reaktion des autonomen Fahrzeugs auf Hindernisse und die Haltelinie an Kreuzungen Ein -bzw. Ausgeschaltet (wird für Mapping verwendet).

1 = schaltet die Reaktionen (weicht Hindernissen aus, hält an Kreuzungen an) ein
0 = schaltet die Reaktionen aus

3 Auslesen von Informationen aus einem Kamerabild

Bevor ein Fahrzeug Hindernissen ausweichen oder an einer Kreuzung anhalten kann, muss es diese sensortechnisch erfassen. Dazu wird der Bereich vor dem Fahrzeug über eine Kamera mit Weitwinkelobjektiv überwacht. Diese monochrome 8 bit Kamera liefert Bilder mit einer Auflösung von 752 x 480 Pixel (vgl. Kapitel 7 Einleitung). Monochrom bedeutet einfarbig. In unserem Fall ist es ein Schwarz-Weiß-Bild mit einer Farbtiefe von einem Byte (8 bit). Das bedeutet pro Bildpunkt stehen $2^8 = 256$ Grauwerte zu Verfügung. Das ergibt eine Grauskala die mit dem Wert '0' für schwarz beginnt und mit dem Wert '255' für weiß endet. Ein Bild mit einer Farbtiefe von 1 bit wird als Binärbild bezeichnet. Jeder Pixel des Bildes entspricht einem Bit und kann den Wert '0' für schwarz oder '1' für weiß annehmen. (vgl. Abbildung 3.0.1)

Abbildung 3.0.1: Anzahl der Grauwerte in Abhängigkeit von der Farbtiefe [vhs] [Meisel RV02 2012]

Die Auflösung der Fahrzeugkamera beträgt 752 x 480 Pixel. Das Kamerabild besteht demnach aus 752 Bildspalten und 480 Bildzeilen und enthält 360.960 Bildpunkte. Jedes dieser Bildpunkte ist mit einem Grauwert versehen. Das Bild lässt sich somit als Matrix beschreiben:

f(x, y) : Grauwert an der Position (x, y)
M : Spaltenzahl (culumns)
N : Zeilenzahl (rows)

Der Bildursprung des Bildes befindet sich oben links [Meisel RV02 2012]. In der Bildverarbeitung ist es üblich bei Bildkoordinaten mit der y-Koordinate zu beginnen. Der Grauwert des Bildpunktes f(y, x) = f(2, 1) in Abbildung 3.0.2 beträgt 152.

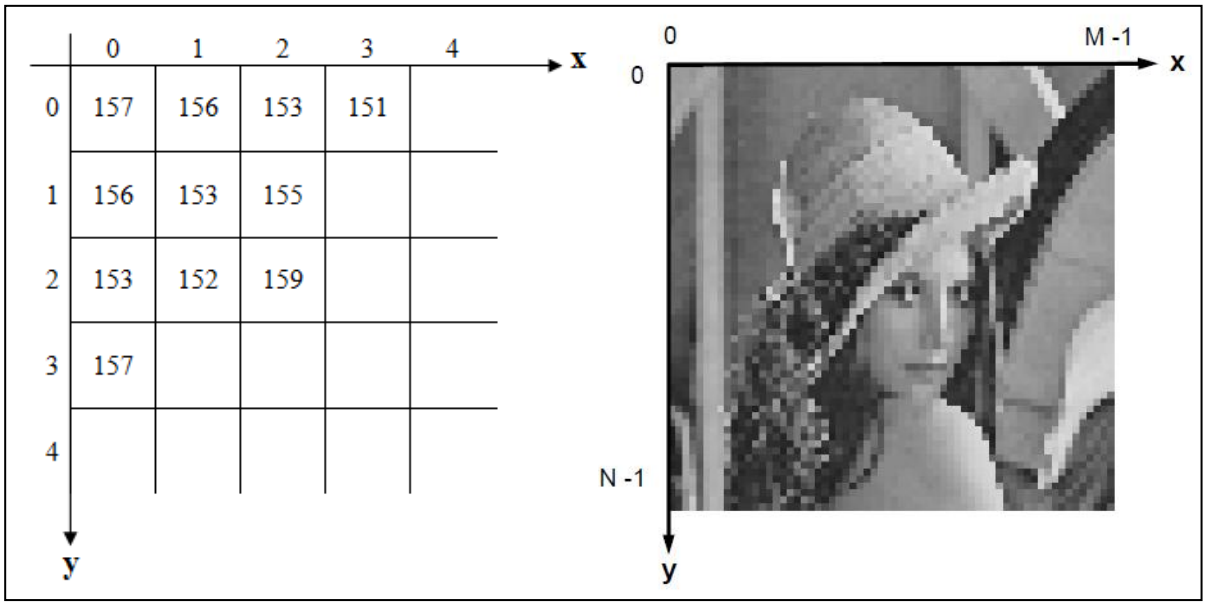


Abbildung 3.0.2: Aufbau eines Bildes als Matrixform [Meisel RV02 2012]

In den nachfolgenden Abschnitten wird zunächst gezeigt wie ein geeigneter Schwellwert für ein monochromes 8 bit Kamerabild gefunden werden kann (vgl. Abschnitt 3.1). Im Anschluss wird eine Region Of Interest (ROI) in ein Kamerabild gelegt um den Fokus auf relevante Bildbereiche zu setzen (vgl. Abschnitt 3.2). Eine ganze ROI Pixel für Pixel abzusuchen kann bei größeren ROIs viel Rechenzeit und Ressourcen in Anspruch nehmen. Um Ressourcenkosten zu minimieren und Rechenzeit einzusparen werden einzelne Spalten, die von besonderem Interesse sind, in den ROIs betrachtet. Diese Spalten werden nachfolgend als Lines Of Interest (LOIs) bezeichnet. Diese LOIs werden in die ROI gelegt. Ihre Anzahl beträgt nur einen Bruchteil der gesamten Spalten einer ROI und hilft dabei Rechenzeit einzusparen (vgl. Abschnitt 3.3). Eine LOI besteht aus mehreren aneinander gereihten Pixeln. Ein Pixel dessen Grauwert über dem Schwellwert liegt wird nachfolgend als Pixel Of Interest (POI) bezeichnet. Das auslesen von POIs innerhalb einer LOI wird in Abschnitt 3.4 erläutert. Anschließend wird im Abschnitt 3.5 die Berechnung der Steigung erkannter Kanten innerhalb einer ROI beschrieben.

3.1 Bestimmung eines geeigneten Schwellwertes für ein 8 bit Kamerabild

In der industriellen Bildverarbeitung ist der nächste Schritt nach der Bildaufnahme oft die Trennung der Objekte, die geprüft werden sollen, vom Untergrund. In vielen Fällen unterscheiden sich die Objekte durch ihren Grauwert signifikant vom Untergrund. Die Häufigkeitsverteilung der Grauwerte in einem Bild, das Histogramm (vgl. Abschnitt 2.2), kann dann Auf-

schluss darüber geben welche Grauwertbereiche den Objekten und welche dem Untergrund zugeordnet werden können. Im nächsten Schritt werden die Bildpixel dann entsprechend ihrem Grauwert klassifiziert und entweder den Objekten oder dem Untergrund zugewiesen [Hackenkamp 2001].

Ein Histogramm des gesamten Kamerabildes (vgl. Diagramm 3.1) liefert nur wenig Informationen darüber ob sich ein Hindernis vor dem Fahrzeug befindet. Zwar unterscheidet sich das Hindernis in seinen Grauwerten deutlich von der Fahrbahn, ist jedoch nicht das einzige Objekt im Bild mit diesen Grauwerten. Die Wände im Hintergrund, die Straßenmarkierungen sowie vordere Aufhängung des Fahrzeugs haben dieselben Grauwerte wie das Hindernis. Die Anzahl der Pixel mit dem Grauwert 255 in der Abbildung 3.1.0 beträgt 64.970 und ist somit der am meisten vorkommende Grauwert. Dies erschwert zusätzlich das Auslesen von Informationen aus dem Histogramm.

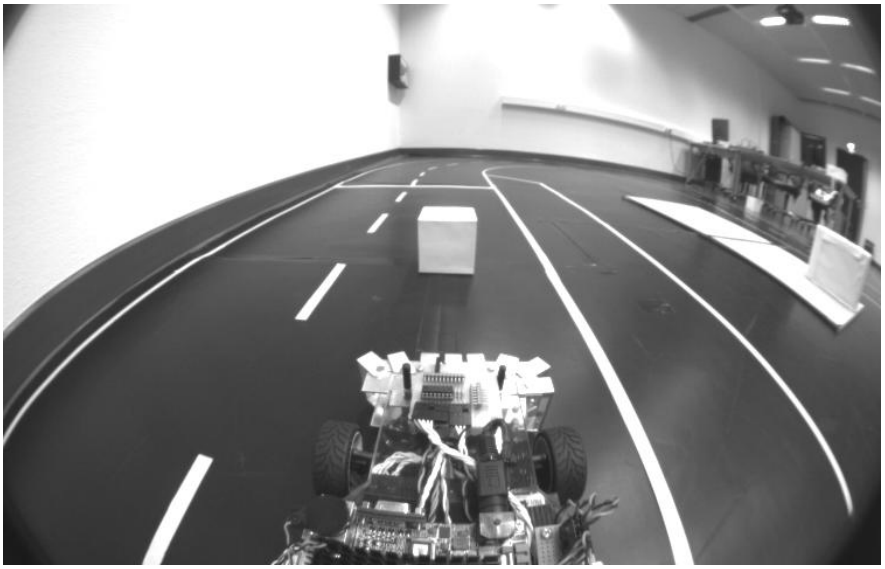


Abbildung 3.1.0: Hindernis vor dem Fahrzeug

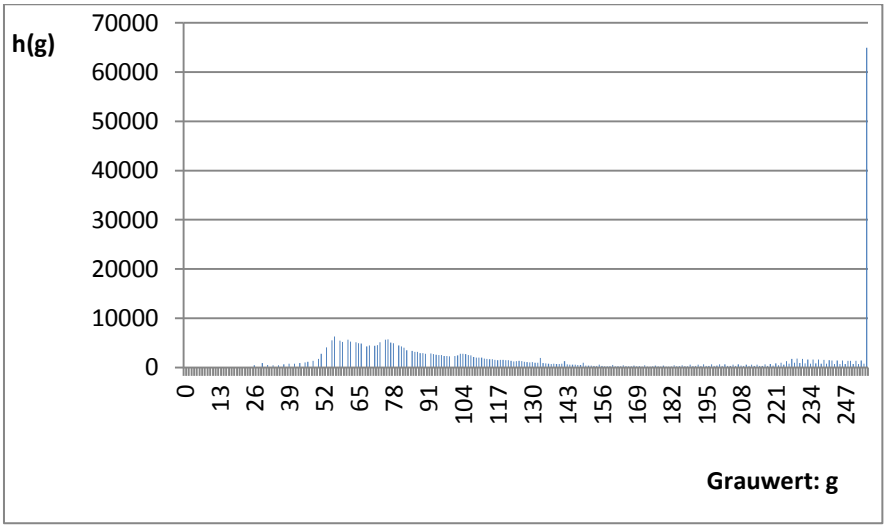


Diagramm 3.1: Histogramm h(g) der Abbildung 3.1.0

Wird das Histogramm nicht für das ganze Kamerabild, sondern nur für einen Ausschnitt des Bildes gebildet (vgl. Abbildung 3.1.1), können die Grauwerte in drei Bereiche unterteil werden. Die Grauwerte aus dehnen die Fahrbahn besteht, die Hindernisse und die Fahrbahnmarkierung. Diese Grauwerte sind abhängig von den Lichtverhältnissen bei denen das Kamerabild entstand. Im ersten Bereich befinden sich die Grauwerte der Fahrbahn. Diese beginnen mit dem Wert 58 und enden mit einem Wert von 150 (vgl. Diagramm 3.2).

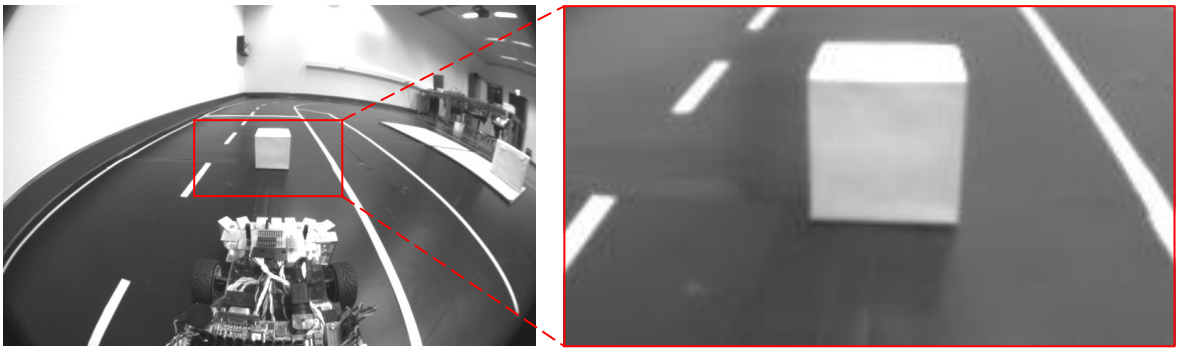


Abbildung 3.1.1: Bildausschnitt Hindernis vor dem Fahrzeug

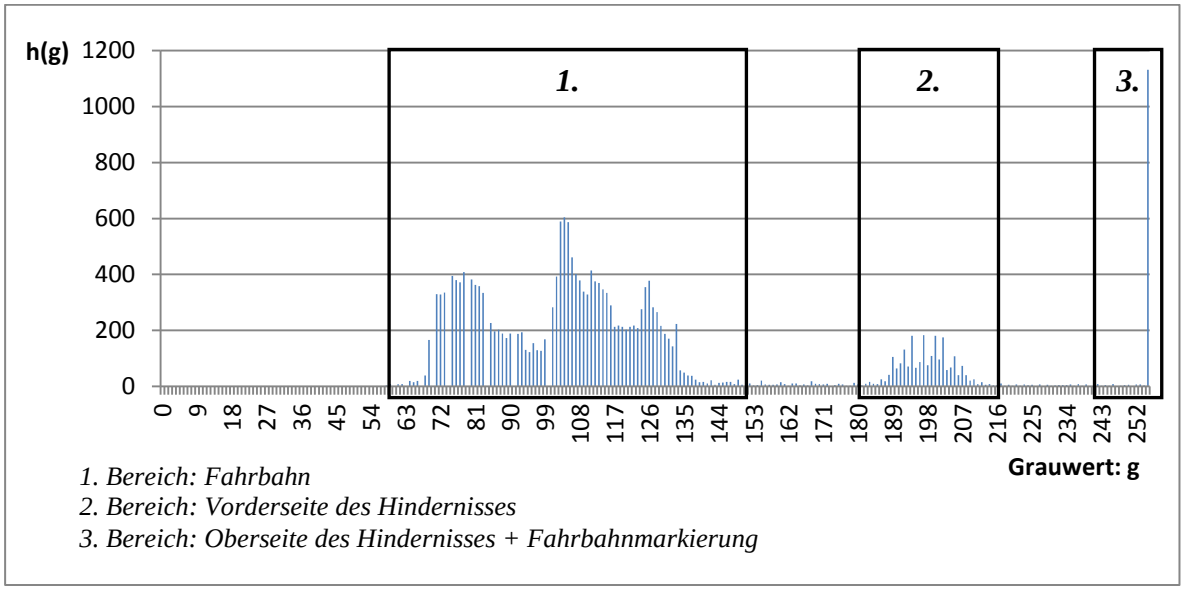


Diagramm 3.2: Histogramm h(g) des Bildausschnittes aus der Abbildung 3.1.1

Diese Werte sind aus folgenden Gründen so breit gefächert:

1. Die Fahrbahn wird nicht gleichmäßig ausgeleuchtet, dadurch entstehen Reflektionen deren Grauwert höher ist als der Grauwert der restlichen Fahrbahn. Dieses muss besonders bei der Schwellwert Bestimmung berücksichtigt werden.
2. Der Schatten vor dem Hindernis hat einen niedrigeren Grauwert als der Rest der Fahrbahn. Schatten können aber auch durch Personen oder Gegenstände entstehen, die sich in der Nähe der Fahrbahn befinden und einen Schatten auf diese werfen. Diese Schat-

ten beeinflussen nicht nur die Grauwerte der Fahrbahn, sondern auch die Grauwerte der Hindernisse und der Fahrbahnmarkierung in ihrem Bereich. Dies muss bei der Bestimmung des Schwellwertes ebenfalls berücksichtigt werden.

3. Der Grauwertverlauf im Kantenbereich ist nicht abrupt, sondern geht über mehrere Grauwerte die stetig ansteigen [Meisel RV03 2012] (vgl. Abbildung 3.1.2)

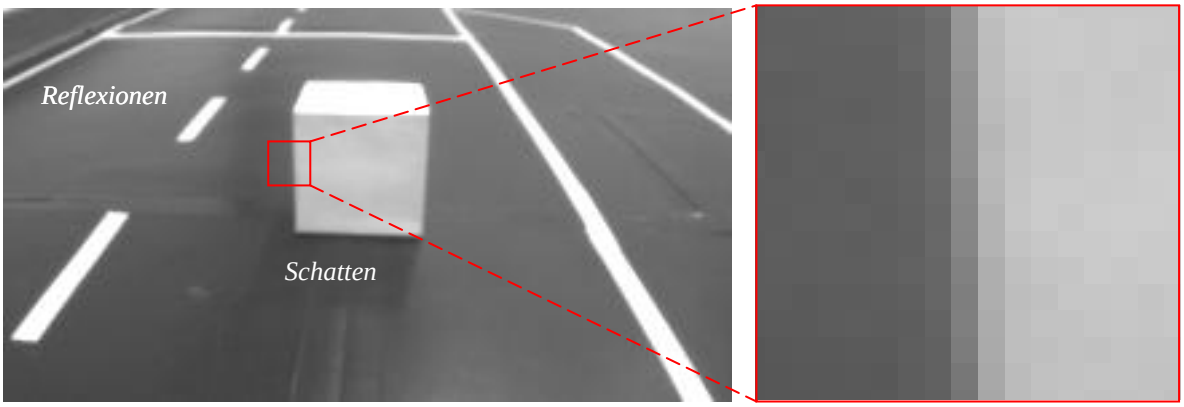


Abbildung 3.1.2: Das Spektrum der Fahrbahngrauwerte wird vergrößert durch Reflexionen, Schatten und Grauwertverlauf im Kantenbereich

Der zweite Bereich ist die Vorderseite des Hindernisses. Dieser beginnt mit einem Grauwert von ca. 180 und enden mit einem Grauwert von ca. 215. Somit unterscheiden sie sich deutlich von den Grauwerten der Fahrbahn.
Der dritte Bereich ist die Oberseite des Hindernisses zusammen mit der Fahrbahnmarkierung. Sie reflektieren das meiste Licht, erscheinen weiß und besitzen einen Grauwert von 240 bis 255, wobei der Grauwert 255 am stärksten vertreten ist (vgl. Diagramm 3.2).

Um die Hindernisse und die Fahrbahnmarkierung von der Fahrbahn zu trennen muss ein Schwellwert bestimmt werden. So kann jeder Pixel anhand seines Grauwertes der Fahrbahn oder dem Hindernis bzw. der Fahrbahnmarkierung zugeordnet werden. So ein Verfahren wird als Schwellwertverfahren bezeichnet. Das Ergebnis eines Schwellwertverfahrens ist ein Binärbild in dem Objekt und Hintergrund getrennt sind. Die Voraussetzung dafür ist ein bimodales Histogramm [Heinrich & Biehl]. Ein Histogramm wird als bimodal bezeichnet, wenn es zwei klar getrennte Maxima aufweist [Risse].
Ein Schwellwert kann manuell festgelegt oder berechnet werden. Zur Berechnung eines Schwellwertes stehen mehrere Methoden zur Verfügung:

1. **Mitte zwischen den beiden Maxima:** der Schwellwert wird in die Mitte zweier Maxima gelegt. Hier muss allerdings sichergestellt sein, dass ein bimodales Histogramm vorliegt [Heinrich & Biehl].
2. **Minima zwischen den beiden Maxima:** der Schwelwert wird auf die kleinste Grauwerthäufigkeit zwischen den Maxima gelegt. Auch in diesem Fall muss ein bimodales Histogramm vorliegen.

3. **Iterative Bestimmung eines Schwellwertes:** der Schwellwert wird zunächst mit der Mitte des Histogramms initialisiert. Im Anschluss wird für jedes der daraus resultierenden Teil-Histogramms der mittlere Grauwert ermittelt. Der Schwellwert wird über das arithmetische Mittel der zuvor bestimmten mittleren Grauwerte neu gesetzt und das Verfahren iteriert. Ändert sich der Schwellwert nicht mehr, wird die Binarisierung mit dem zuletzt berechneten Schwellwert ausgeführt[Heinrich & Biehl].

Weitere Methoden sind das Verfahren von Otsu [Otsu 1979] und die Methode nach Niblack [Niblack 1986]. Das Histogramm von dem Bildausschnitt (vgl. Diagramm 3.2) hat streng genommen drei Maxima und ist somit kein bimodales Histogramm. Das erste Maxima der Fahrbahngrauwerte liegt bei Grauwert 104 mit einer Anzahl von 604 Pixeln. Das zweite Maxima für die Oberseite des Hindernisses liegt bei Grauwert 197 mit einer Anzahl von 183 Pixeln. Das dritte Maxima für die Oberseite des Hindernisses und für die Fahrbahnmarkierung liegt bei Grauwert 255 mit einer Anzahl von 1.132 Pixeln. Das Ziel ist es das Hindernis und die Fahrbahnmarkierung von der Fahrbahn zu trennen, denn im späteren Verlauf sollen nicht nur die Hindernisse erkannt werden, sondern auch die Startlinie und die Haltelinie an Kreuzungen. Somit kann das dritte Maxima mit dem Grauwert 255 vernachlässigt werden. Dadurch erhalten wir ein bimodales Histogramm mit den beiden Maxima für die Grauwerte der Fahrbahn und der Oberseite des Hindernisses. Jetzt wird ein Schwellwert benötigt der die beiden Bereiche (vgl. Diagramm 3.2) teilt. Zuerst wird gezeigt wie ein Schwellwert manuell festgelegt werden kann. Im Anschluss werden Schwellwerte mit den oben aufgelisteten Methoden berechnet. Danach werden die Ergebnisse mit einander verglichen und so der bestmöglichen Schwellwert ermittelt. Um ein Schwellwert manuell festzulegen, betrachten wir die zwei Bereiche (vgl. Diagramm 3.2) und stellen fest das die Grauwerte für die Fahrbahn bei einem Grauwert von ca. 150 enden und die Grauwerte für die Vorderseite des Hindernisses bei ca. 180 beginnen. Es bietet sich an den Schwellwert auf ein Grauwert von 165 zu setzen. Dadurch entsteht eine Pufferzone von 15 Graustufen zu den beiden Bereichen, den Grauwerten der Fahrbahn wie auch zu den Grauwerten der Oberseite des Hindernisses. Durch diese Pufferzone können kleine Lichtschwankungen und Effekte wie Reflektionen und Schatten auf der Fahrbahn aufgefangen werden. Ändern sich allerdings die Lichtverhältnisse gravierend, das Fahrzeug wird in einem hellerem/dunklerem Raum eingesetzt, muss ein neues Histogramm erstellt werden und der Schwellwert neu berechnet oder manuell neu festgelegt werden. Hier muss drauf geachtet werden, dass je schwächer die Beleuchtung desto stärker ist das Bildrauschen und desto geringer ist die Bildqualität.

Es wurden drei Methoden Vorgestellt mit dehnen ein Schwellwert berechnet werden kann. Bei der ersten Methode wird der Schwellwert in die Mitte zweier Maxima gelegt. Für unser Histogramm der Abbildung 3.1.1 entspricht das einem Grauwert von 151. Das ist ein Schwellwert der sich nur um eine Graustufe von der Grenze des ersten Bereichs (Grauwerte der Fahrbahn) unterscheidet (vgl. Diagramm 3.2) hat somit einen zu kleinen Puffer und ist als Schwellwert bei den zum Testzeitpunkt vorhandenen Lichtverhältnissen im Einsatzraum des Fahrzeugs nicht zu empfehlen.

Bei der zweiten Methode wird das Minima zwischen den beiden Maxima ermittelt. Der Schwellwert wird auf die kleinste Grauwerthäufigkeit zwischen den Maxima gelegt. Für unser Histogramm der Abbildung 3.1.1 entspricht das einem Grauwert von 173. Dieser Wert hat zu beiden Bereichen (vgl. Diagramm 3.2) Puffermöglichkeiten und kann als Schwellwert bei den

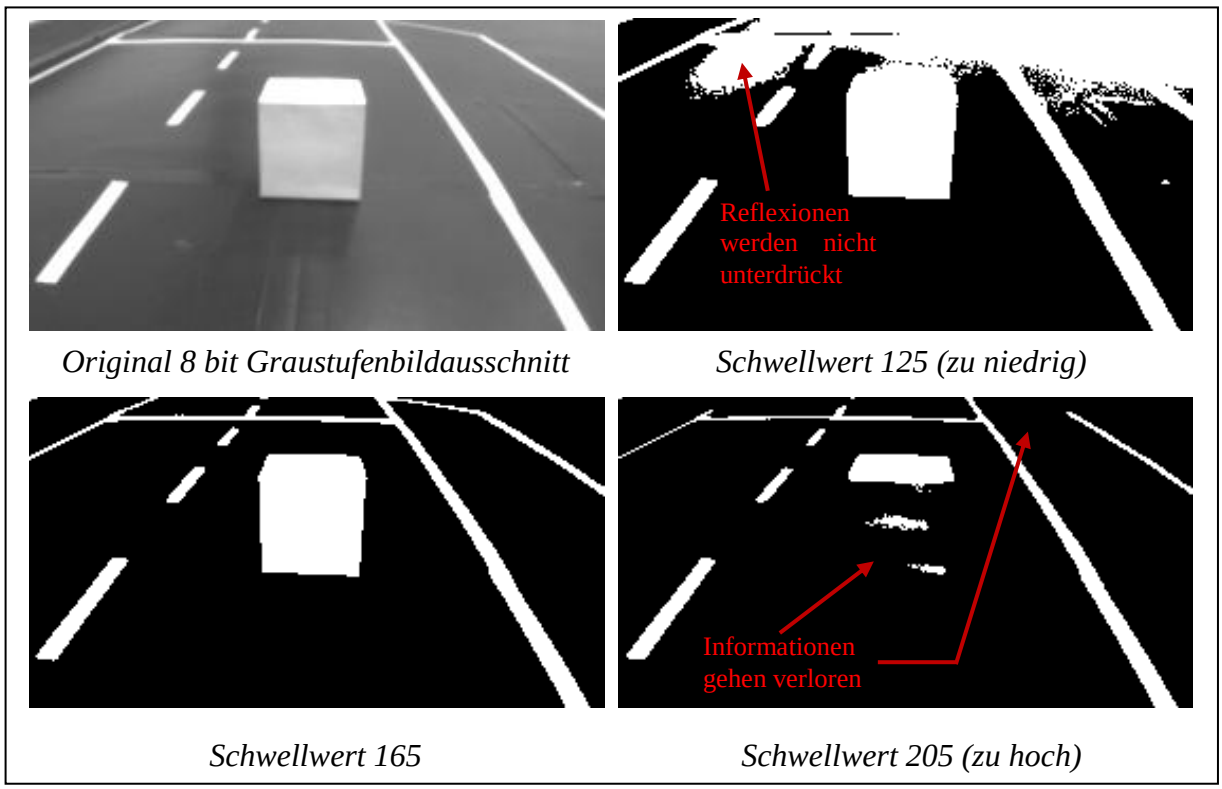
zum Testzeitpunkt vorhandenen Lichtverhältnissen im Einsatzraum des Fahrzeugs verwendet werden.
Die dritte Methode ist die Iterative Bestimmung des Schwellwertes sie liefert als Ergebnis einen Schwellwert von 157. Dieser Wert hat ebenfalls Puffermöglichkeiten und kann als Schwellwert bei den zum Testzeitpunkt vorhandenen Lichtverhältnissen im Einsatzraum des Fahrzeugs verwendet werden.

Durch eine Binarisierung kann die Qualität des berechneten oder manuell festgelegten Schwellwertes überprüft werden. Binarisierung ist die Transformation eines Grauwertbildes $f(y,x)$ in ein Binärbild $b(y,x)$ durch Schwelwertoperation[Jiang 2007].

$$b(y,x) = \begin{cases} 0, & \text{falls } f(y,x) \leq S \\ 1, & \text{falls } f(y,x) > S \end{cases}$$

S = Schwelwert (threshold)
Die Visualisierung als Graustufenbild $g(y,x)$ erfolgt mit:

$$g(y,x) = \begin{cases} 0, & \text{falls } b(y,x) = 0 \\ 255, & \text{falls } b(y,x) = 1 \end{cases}$$



Original 8 bit Graustufenbildausschnitt *Schwellwert 125 (zu niedrig)*

Schwellwert 165 *Schwellwert 205 (zu hoch)*

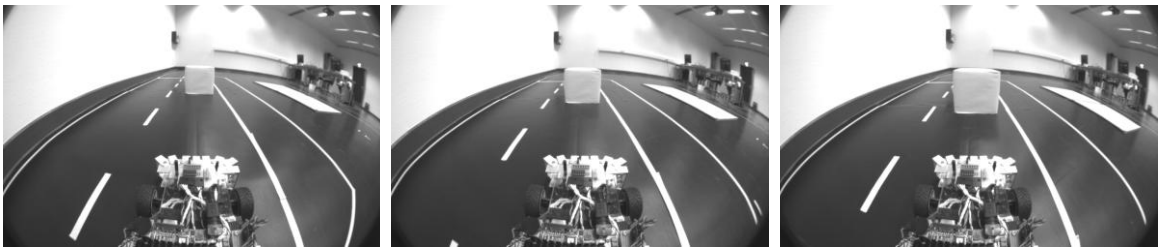
Abbildung 3.1.3: Transformation eines Grauwertbildes in ein Binärbild durch Schwelwertoperation

In der Abbildung 3.1.3 sind Binarisierungen von Graustufenbilder eines Hindernisses vor dem Fahrzeug in Abhängigkeit von unterschiedlichen Schwellwerten aufgezeigt. Wird der Schwellwert zu niedrig gewählt können Reflexionen nicht unterdrückt werden. Das führt zu falschen Informationen z. B. würde eine Haltelinie oder ein Hindernis erkannt werden an Stellen an dehnen keine sind. Wird der Schwellwert zu hoch gewählt gehen Informationen verloren. Hindernisse werden nicht rechtzeitig erkannt. Schlecht ausgeleuchtete Hindernisse werden ganz unterdrückt und somit gar nicht erkannt. Bei Fahrbahnmarkierungen gehen ebenfalls Informationen durch den Grauwertverlauf im Kantenbereich (vgl. Abbildung 3.1.2) verloren. So sind Fahrbahnmarkierungen wie Haltelinien an Kreuzungen im binarisierten Bild dünner. Schlecht ausgeleuchtete Fahrbahnmarkierungen werden vollständig unterdrückt und gehen als Information verloren (vgl. Abbildung 3.1.3). Bei dem manuell festgelegten Schwellwert 165 sowie den berechneten Schwellwerten 173 und 157 gehen keine Informationen verloren. Somit sind das die Schwellwerte die bei den vorhandenen Lichtverhältnissen im Einsatzraum des Fahrzeugs verwendet werden können um die Fahrbahn von den Hindernissen und den Fahrbahnmarkierungen zu trennen.

3.2 Anlegen einer Region Of Interest in einem Kamerabild

ROI ist die Abkürzung des englischen Begriffs Region Of Interest („Bereich von Interesse“), mit dem in der Bildverarbeitung, die für die Bildauswertung und Bildanalyse besonders relevanten Bildbereiche bezeichnet werden [JENOPTIK].

Da die Fahrzeugkamera fest am Fahrzeug montiert ist, wird die Fahrbahn stehst unter demselben Kamerawinkel abgelichtet (vgl. Abbildung 3.2.1).



Entfernung zum Hindernis 0,8m *Entfernung zum Hindernis 0,6m* *Entfernung zum Hindernis 0,4m*

Abbildung 3.2.1: heranfahren an ein Hindernis (Fahrzeugsicht)

Um zu überprüfen ob sich ein Hindernis vor dem Fahrzeug befindet muss nicht das ganze Kamerabild analysiert werden. Es reicht aus eine ROI in das Bild zu legen und diese darauf zu untersuchen, wie viele Pixel der ROI einen höheren Grauwert besitzen als der zuvor im Abschnitt 3.1 berechnete Schwellwert.

Eine ROI kann durch zwei Koordinatenpunkte, die eine Fläche aufspannen, beschrieben werden. Die Koordinaten *leftX, leftY* sind die X -und Y-Koordinaten für die linke obere Ecke der ROI. Die Koordinaten *rightX, rightY* sind die X -und Y-Koordinaten der rechten unteren Ecke der ROI (vgl. Abbildung 3.2.2). Um ein Hindernis in einer bestimmten Entfernung vor

dem Fahrzeug zu erkennen, wird eine ROI in das Kamerabild gelegt. Dazu müssen die Koordinaten: *leftX, leftY, rightX, rightY* aus dem Kamerabild für genau diese Position ermittelt werden. Wurden diese Koordinaten erfolgreich ermittelt, spannen sie eine ROI im Kamerabild auf die den zu Untersuchenden Bereich eingrenzt (vgl. Abbildung 3.2.3).

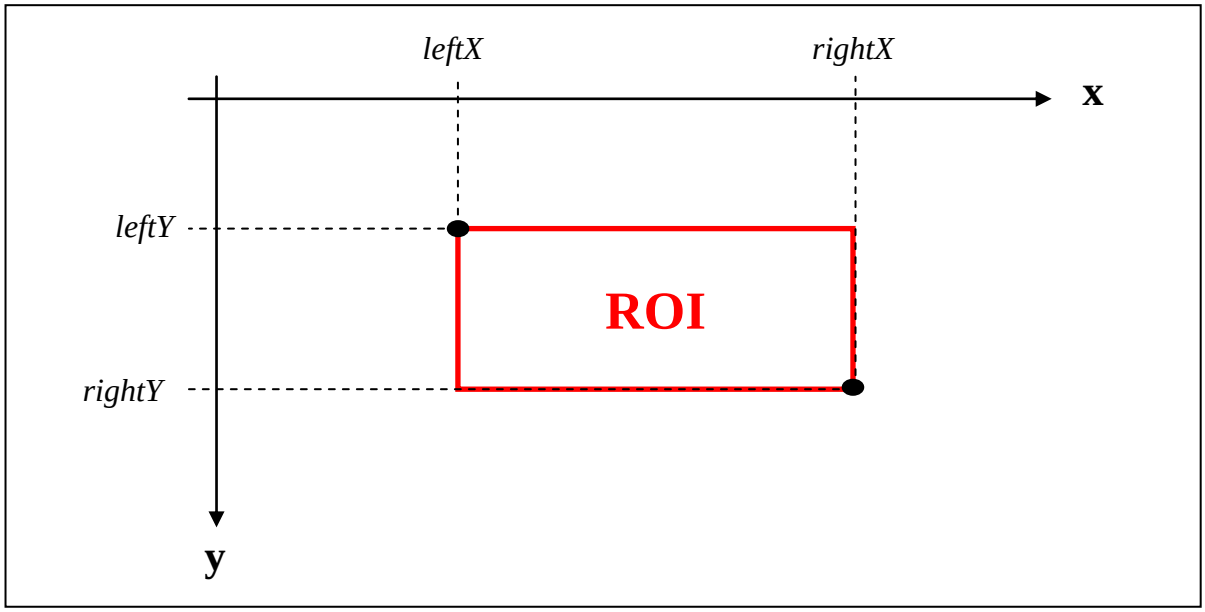


Abbildung 3.2.2: Aufbau einer Region Of Interest (ROI)

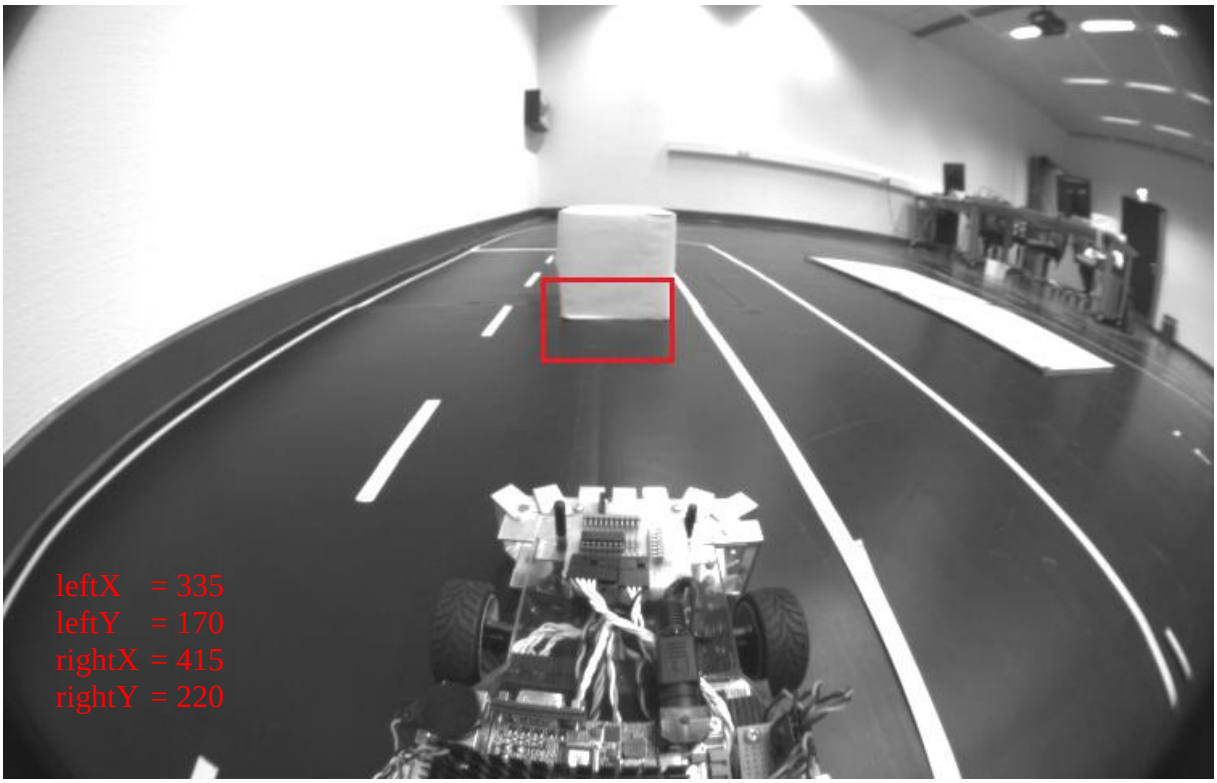


Abbildung 3.2.3: Bild der Fahrzeugkamera mit eingezeichneter (ROI) und den dazugehörigen Koordinaten für leftX, leftY, rightX, rightY

Jeder Pixel der ROI wird mit dem errechneten bzw. manuell festgelegtem Schwellwert aus dem Abschnitt 3.1 verglichen. Ist sein Grauwert größer als der Schwellwert ist dieser Pixel ein Pixel Of Interest (vgl. Abschnitt 3.4) und gehört entweder zu einem Hindernis oder der Fahrbahnmarkierung.

3.3 Positionierung von Lines Of Interest innerhalb einer ROI

Die in Abbildung 3.2.3 verwendete ROI ist 80 Pixel breit und 50 Pixel hoch. Der zu untersuchende Bereich umfasst 4.000 Pixel. Um festzustellen ob sich in diesem Bereich ein Hindernis befindet muss jedes dieser 4.000 Pixel mit dem aus Abschnitt 3.1. ermittelten Schwellwert verglichen werden. So ein Vorgehen würde viel Rechenzeit und Ressourcen in Anspruch nehmen. Um Ressourcenkosten zu minimieren und Rechenzeit einzusparen werden in der ROI einzelne Spalten in gleichen Abständen betrachtet und Pixel für Pixel mit dem Schwellwert verglichen. Diese Spalten werden nachfolgend als Lines Of Interest (LOIs) bezeichnet. In der Abbildung 3.3.1 wurden sechs LOIs in die ROI gelegt 6 * 50 Pixel = 300 Pixel müssen mit dem Schwellwert verglichen werden. Das entspricht 7,5% der vorherigen Pixelanzahl. Mehr Informationen liefert hierbei eine Komplexitätsanalyse.

Komplexitätsanalyse:

Annahme: eine quadratische ROI mit n x n Pixeln wird durchsucht und Pixel für Pixel mit einem Schwellwert verglichen.

$Anz_{P_G} = Anzahl\ Pixel\ deren\ Grauwert$ ü$ber\ dem\ Schwellwert\ liegt$
$P_G \quad = der\ zu\ Untersuchene\ Pixel\ und\ sein\ Grauwert$
$S \quad = Schwellwert$

$$Anz_{P_G} = \sum_{k=1}^{n} \mu\ , \qquad \mu = \sum_{i=1}^{n} (P_G > S)$$

$Das\ entspricht\ einer\ Laufzeit\ von: T(n) = O(n^2)$

Annahme: eine quadratische ROI mit sechs LOIs, jede LOI wird Pixel für Pixel durchsucht und mit einem Schwellwert verglichen.

$$Anz_{P_G} = \sum_{k=1}^{6} \mu\ , \qquad \mu = \sum_{i=1}^{n} (P_G > S)$$

$Das\ entspricht\ einer\ Laufzeit\ von: T(n) = O(6n)$

Um eine ROI ohne LOIs zu durchsuchen und mit einem Schwellwert zu vergleichen wird eine quadratische Laufzeit benötigt. Werden LOIs in eine ROI gelegt so ist die Laufzeit von der Anzahl der LOIs abhängig, denn es werden nur die LOIs durchsucht und mit einem

Schwellwert verglichen. Um die Laufzeit möglichst gering zu halten sollten nicht zu viele LOIs in eine Region Of Interest gelegt werden. Es empfiehlt sich eine Anzahl zwischen fünf und zehn LOIs zu benutzen um die Ressourcenkosten zu minimieren und Rechenzeit einzusparen.

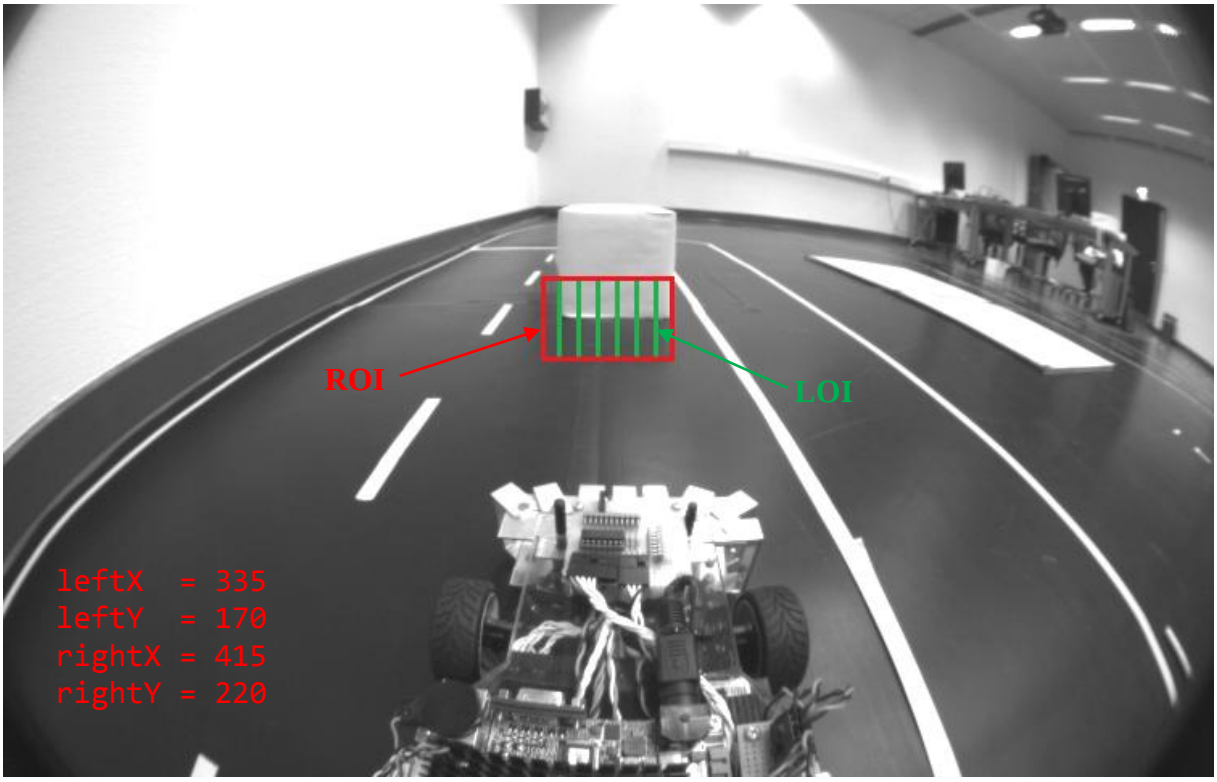


Abbildung 3.3.1: Bild der Fahrzeugkamera mit eingezeichneter (ROI) und den dazugehörigen Lines Of Interests (LOIs)

3.4 Identifikation von Pixel Of Interest innerhalb einer LOI

Eine Line Of Interest (LOI) innerhalb einer Region Of Interest (ROI) besteht aus mehreren an einander gereihten Pixeln. Jedes dieser Pixel besitzt einen Grauwert. Um die Fahrbahn von den Hindernissen zu trennen muss ein Schwellwert bestimmt werden (vgl. Abschnitt 3.1). Ein Pixel dessen Grauwert über dem Schwellwert liegt wird nachfolgend als Pixel Of Interest (POI) bezeichnet. Ein POI ist somit ein Pixel, der sich entweder auf einem Hindernis oder auf der Fahrbahnmarkierung befindet (vgl. Abbildung 3.4.1).

Mit der Klasse `ROIonPic` (vgl. Abbildung 3.4.2), die sich im Verzeichnis:

`FAUSTplugins/Tasks/ObstacleAvoidanceAndCrossingRecognition`

befindet, kann in das Bild der Fahrzeugkamera eine ROI mit den dazugehörigen LOIs gelegt werden. Die ersten vier Parameter sind die Koordinaten der Region Of Interest, mit `numberOfLines` wird die Anzahl der LOIs übergeben, `limit` ist der Schwellwert und mit `enableLOI` werden die Lines Of Interest zu Kalibrations Zwecken in das Kamerabild eingeblendet.

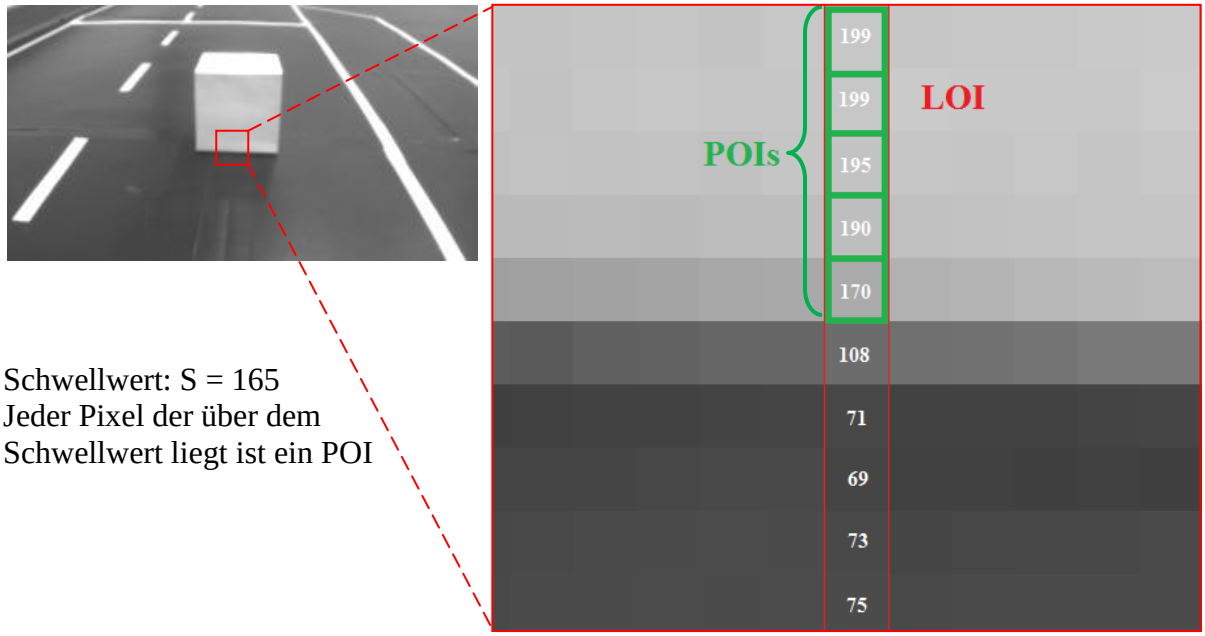


Abbildung 3.4.1: Points Of Interest(POIs) in einer LOI beim Übergang von Fahrbahn zum Hindernis

Mit der Methode `setCamPic(CameraImagePtr camPic)` wird der Klasse ein aktuelles Kamerabild übergeben. Die Methode `getPixelOverLimitInPercent()` gibt den aufgerundeten Prozentsatz der Pixel Of Interest (POIs) der jeweiligen ROI zurück.

Funktionsweise der Methode `getPixelOverLimitInPercent()`:

$Anz_{POIs} = Gesamt\ Anzahl\ Pixel\ Of\ Interest$
$POI = Pixel\ Of\ Interest$
$m = Anzahl\ Lines\ Of\ Interest\ (LOIs)$
$n = breite\ der\ ROI\ (rightY - leftY)$

$$Anz_{POIs} = \sum_{k=1}^{m} \mu\ , \qquad \mu = \sum_{k=1}^{n} POI$$

Der Rückgabewert ist ein gerundeter Integer-Wert, der die Anzahl Pixel über dem Schwellwert (Anzahl der POIs) in Prozent angibt. Aus Gründen der Performance wurde auf die Rück-

gabe des genauen Wertes im Double-Format verzichtet, da sich dadurch keine Vorteile ergeben.

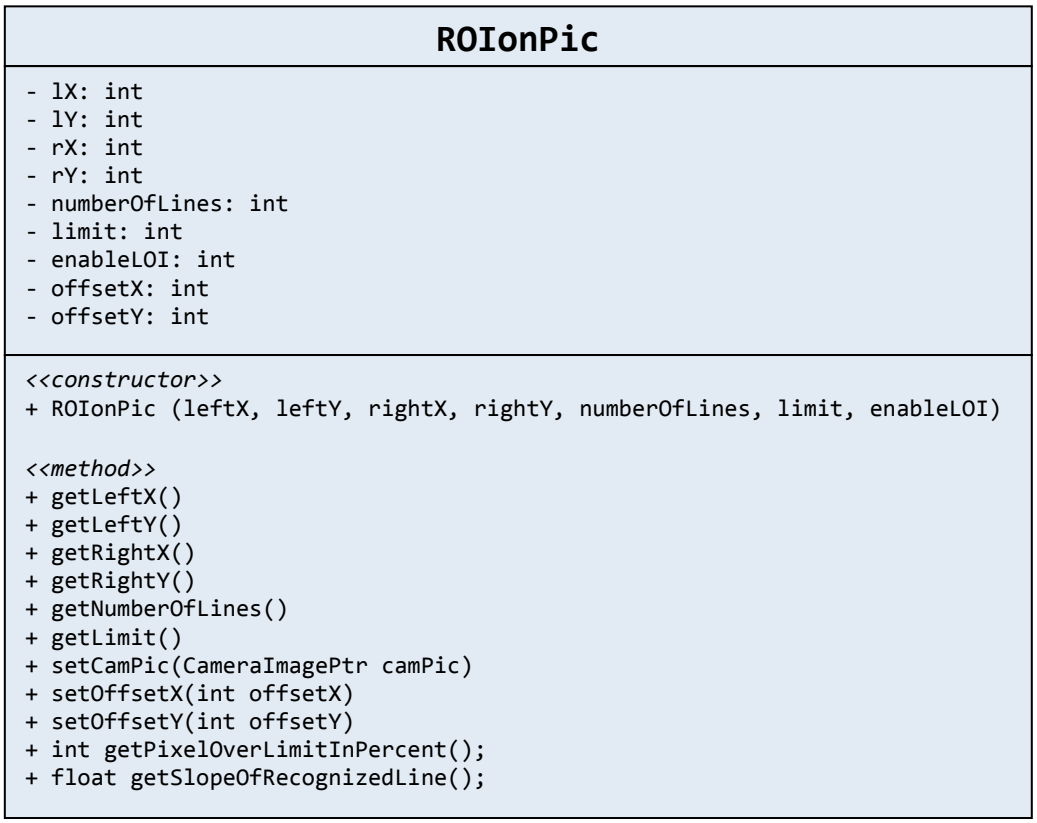


Abbildung 3.4.2: UML Diagramm der Klasse ROIonPic

Die Rückgabewerte der Methode `getPixelOverLimitInPercent()` in Abhängigkeit von der Anzahl der Lines Of Interest können der Abbildung 3.4.3 entnommen werden. Es wurden folgende Parameter verwendet:

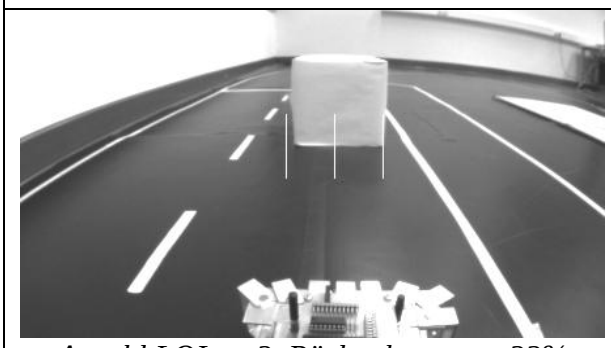
für die ROI:

```
leftX  = 340
leftY  = 170
rightX = 415
rightY = 220
numberOfLines = 3, 5, 7, 9 (Anzahl der LOIs)
limit = 165 (Schwellwert)
```

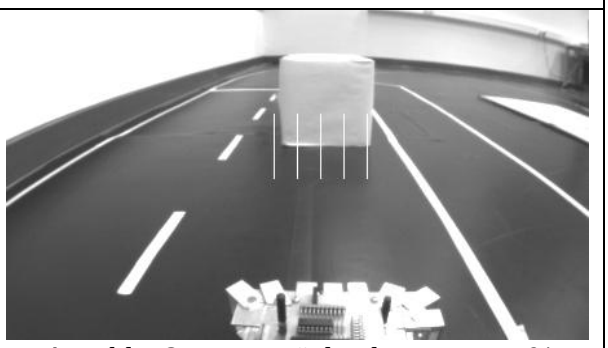
Die gesamte ROI ohne LOIs liefert als Rückgabewert einen Integer Wert von 44, das heißt 44% aller in der ROI befindlichen Pixel liegen über dem Schwellwert von 165. Wie in der Abbildung 3.4.3 zu sehen ist, kann schon mit sieben LOIs in einer ROI ein Rückgabewert erreicht werden der nur um 3% von dem tatsächlichen Wert abweicht. Es ist deshalb nicht

notwendig viele LOIs in die ROI zu legen das verbraucht nur unnötig Ressourcen, liefert jedoch nur geringe Verbessrungen in der Genauigkeit des Rückgabewertes.

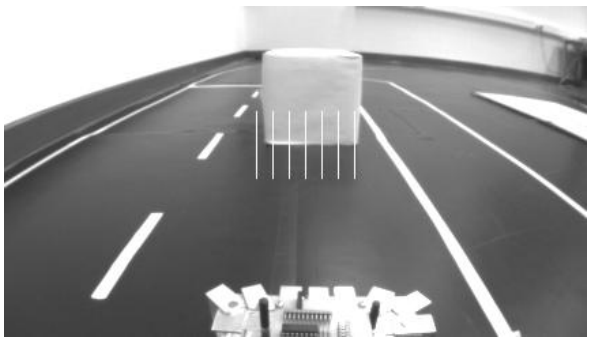
Rückgabewert für die gesamte ROI ohne LOIs = 44%



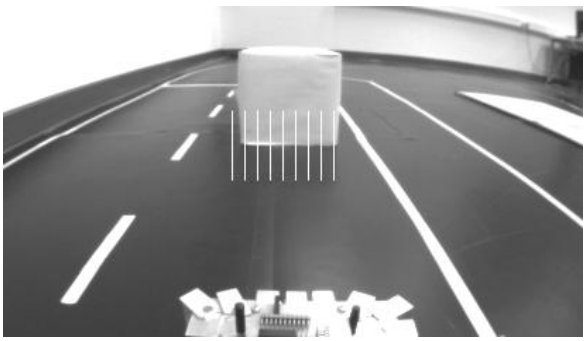
Anzahl LOIs = 3, Rückgabewert = 32%



Anzahl LOIs = 5, Rückgabewert = 39%



Anzahl LOIs = 7, Rückgabewert = 41%



Anzahl LOIs = 9, Rückgabewert = 43%

Abbildung 3.4.3: Rückgabewerte der Methode `getPixelOverLimitInPercent()` in Abhängigkeit von der Anzahl der Lines Of Interest (LOIs)

3.5 Berechnung der Steigung erkannter Kanten innerhalb einer ROI

Um Fahrbahnmarkierungen wie die Startlinie oder die Haltelinie an einer Kreuzung zu erkennen muss ebenfalls eine ROI (vgl. Abschnitt 3.2) in das Kamerabild gelegt werden. Es ist jedoch in manchen Situationen von Vorteil nicht nur eine Linie zu erkennen, sondern auch ihre Steigung zu wissen (vgl. Abschnitt 6.3).

Um die Steigung einer erkannten Haltelinie (an einer Kreuzung) zu bestimmen, muss ihre unterste Kannte erkannt werden. Dazu werden die Koordinaten der Schnittpunkte der LOIs mit der Haltelinie benötigt (vgl. Abbildung 3.5.1). Diese Koordinaten werden ermittelt, indem bei einer erkannten Haltelinie die LOIs von unten nach oben Pixel für Pixel durchsucht werden. Die Durchsuchung startet links bei der ersten LOI. Liegt der Grauwert des zu untersuchenden Pixels über dem Schwellwert so befindet sich dieser Pixel auf der Haltelinie.

Die Koordinaten dieses Pixels werden gespeichert und die nächste LOI wird durchsucht. Wurden alle LOIs durchsucht und die Koordinaten aller Schnittpunkte ermittelt, wird eine Ausgleichsgerade zwischen die Punkte gelegt und ihre Steigung berechnet.

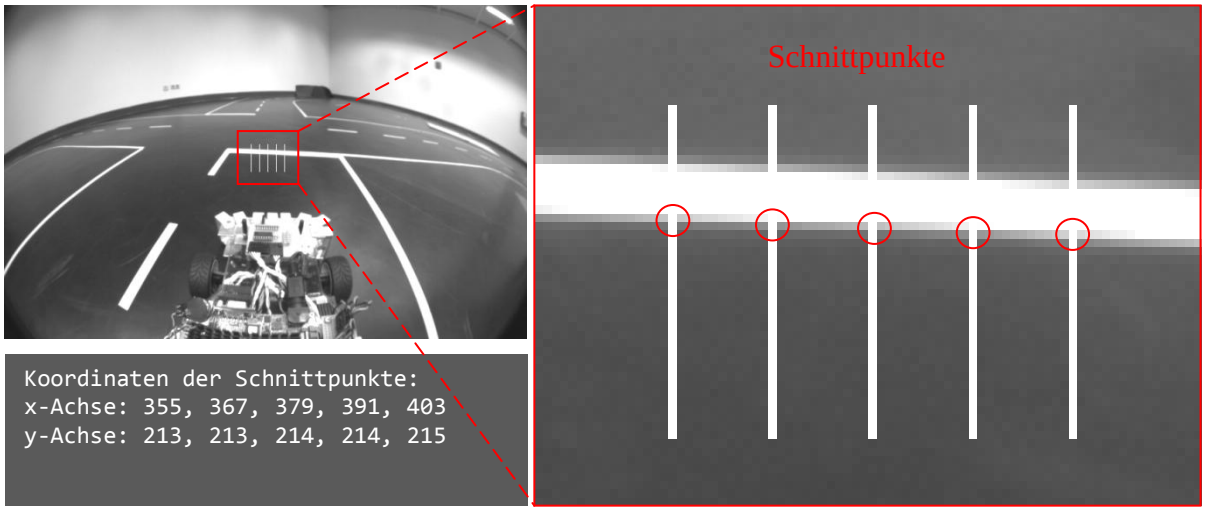


Abbildung 3.5.1: Koordinaten der Schnittpunkte der LOIs mit der Haltelinie

Berechnung der Steigung der in Abbildung 3.5.1 erkannten Haltelinie:

$n = Anzahl\ Schnittpunkte$

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} = \frac{355 + 357 + 379 + 391 + 403}{5} = 377$$

$$\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n} = \frac{213 + 213 + 214 + 214 + 215}{5} = 213{,}8$$

$$slope = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2} = 0{,}041$$

Die Haltelinie hat eine Steigung von 0,041. Obwohl es sich in Abbildung 3.5.1 um eine fallende Gerade handelt, ist ihre Steigung positiv. Das liegt daran das der Bilduhrsprung links oben liegt und das Bildkoordinatensystem von dort aus mit den Koordinaten (y=0, x=0) beginnt (vgl. Einleitung von Kapitel 3). Um die Steigung einer erkannten Kante innerhalb einer ROI zu berechnen steht die Methode `getSlopeOfRecognizedLine()` in der Klasse `ROIonPic` zu verfügung. Der Rückgabewert der Methode ist `float`. Die Steigung wird nur berechnet, wenn mindestens drei Schnittpunkte mit einer Kante erkannt wurden. Wurden weniger als drei Schnittpunkte erkannt, ist der Rückgabewert der Methode 1000.

4 Zustandsautomat zur Hindernis -und Kreuzungserkennung

Für die Hindernis -und Kreuzungserkennung wurde ein hierarchischer Zustandsautomat entwickelt. Dieser basiert auf der sich bereits auf dem Fahrzeug befindliche Implementierung der `StateMachine` die sich im folgenden Verzeichnis befindet:

`FAUSTplugins/Util/StateMachine`

Das Funktionsprinzip der StateMachine findet man in [PEARL-News] beschrieben. Dieses Prinzip behebt einige der typischen Implementationsprobleme hierarchischer Zustandsautomaten und ist besonders geeignet für Echtzeitsysteme.

Der hirarchische Automat für die Hindernis -und Kreuzungserkennung befindet sich in dem Verzeichnis:

`FAUSTplugins/Tasks/ObstacleAvoidanceAndCrossingRecognition`

und besteht aus einem Controller (`ControllerForOAandCR`) und vier Hauptzuständen: `Idle, ObstacleRecognized, CrossingRecognized, StartlineRecognized` (vgl. Abbildung 4.0.1). Die beiden Hauptzustände `ObstacleRecognized` und `Crossing-Recognized` besitzen Subautomaten. Die Funktion des Subautomaten für den Zustand `ObstacleRecognized` wird im Abschnitt 4.1 erläutert. Im Anschluss wird das Funktionsprinzip des Subautomaten für den Zustand `CrossingRecognized` beschrieben (vgl. Abschnitt 4.2).

Die Funktion des Controllers ist das Erstellen und Verwalten der vier Hauptzustände und das Starten derer `exexute()` Methoden. Bei der FAUT-StateMachine gibt es folgendes zu beachten: wird ein Zustand erstellt so wird sofort nach der Erstellung des Zustandes seine `entry()` Methode aufgerufen. In dieser `entry()` Methode können dann für den Zustand wichtige Initialisierungen vorgenommen werden. Somit entspricht die `entry()` Methode eher einer init() Methode und nicht wie vermutet einer Methode die erst beim eintreten in den Zustand aufgerufen wird.

Im `Idle` Zustand wird mit Hilfe der Kamera und der Infrarotsensoren die Fahrbahn vor dem Fahrzeug überwacht. Wird ein Hindernis vor dem Fahrzeug erkannt, so findet ein Zustandswechsel in den Zustand `ObstacleRecognized` statt. In diesem Zustand übernimmt ein Subautomat den Ausweichvorgang. Ist der Ausweichvorgang abgeschlossen, beendet sich der

Subautomat und es findet ein Zustandswechsel zurück in den `Idle` Zustand statt. Wird im `Idle` Zustand eine Kreuzung erkannt, wird in den Zustand `CrossingRecognized` gewechselt. In diesem Zustand übernimmt ein Subautomat die Kreuzungsanalyse. Wird von rechts ein dynamisches Hindernis erkannt, so wird Vorfahrt gewährt. Ist die Kreuzung frei wird sie überquert. Wurde die Kreuzung erfolgreich passiert, beendet sich der Subautomat und es findet ein erneuter Zustandswechsel in den `Idle` Zustand statt. Wird im `Idle` Zustand eine Startlinie erkannt so wird in den Zustand `StartlineRecognized` gewechselt. Hier kann zum Beispiel ein Flag für das Mapping gesetzt werden, um die Position des Fahrzeugs zu bestimmen oder mit den vorhandenen Mapping Ergebnissen abzugleichen.

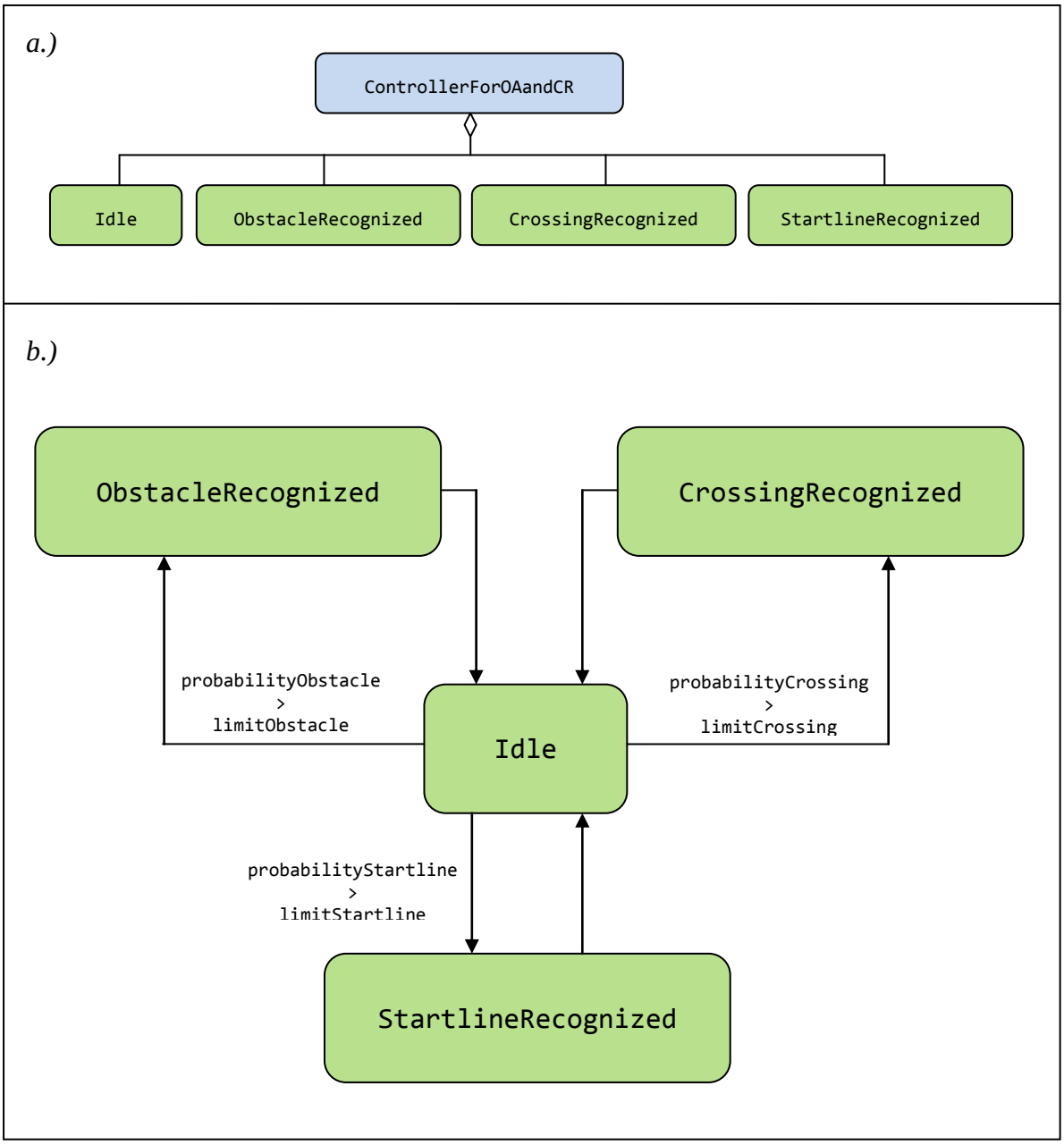


Abbildung 4.0.1: a.) UML Diagramm Hauptautomat b.) Funktionsweise Hauptautomat

4.1 Subautomat Hinderniserkennung

Der Subautomat für die Hinderniserkennung besteht aus vier Zuständen: `DriveToLeftLane`, `LeftToObstacle`, `ParallelToObstacle`, `DriveToRightLane` und steuert den Ausweichvorgang des Fahrzeugs. Der Hauptzustand `ObstacleRecognized` verwaltet die vier Subzustände und startet deren `execute()` Methoden (vgl. Abbildung 4.1.1).

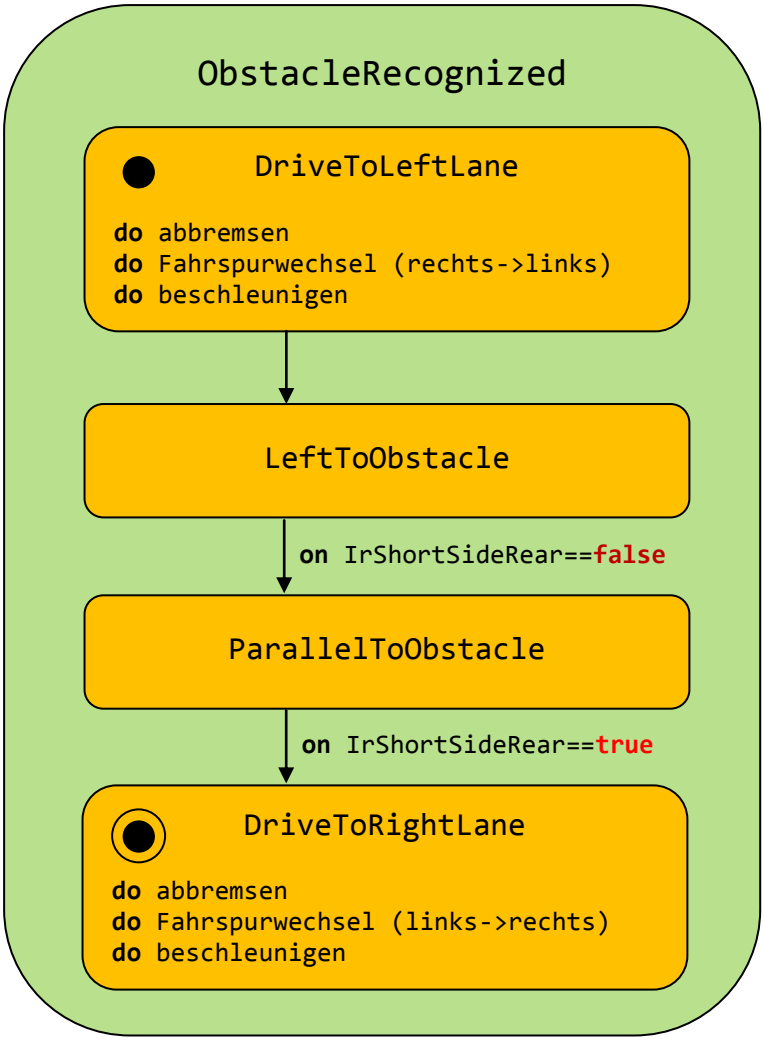


Abbildung 4.1.1: Subautomat Hinderniserkennung

Wird ein Hindernis vor dem Fahrzeug erkannt, befindet sich der Hauptautomat im Hauptzustand `ObstacleRecognized`. Dieser startet die `execute()` Methode des ersten Subzustandes `DriveToLeftLane`. Daraufhin wird das Fahrzeug abgebremst auf 0,86m/s und wechselt die Fahrbahn von rechts nach links. Nach dem Wechsel der Fahrbahn wird das

Fahrzeug auf die ursprüngliche Geschwindigkeit beschleunigt und es findet ein Zustandswechsel statt in den Zustand `LeftToObstacle`. In diesem Zustand ist das Fahrzeug auf der linken Fahrspur hinter dem Hindernis. Befindet sich das Fahrzeug parallel zum Hindernis ändert sich der Wert des digitalen Infrarotsensors GP2Y0D340K der Sharp Electronics GmbH [Datenblatt GP2Y0D340K], der sich hinten auf der rechten Seite des Fahrzeugs befindet, von `true` auf `false`. Daraufhin wird der Zustand von `LeftToObstacle` zu `ParallelToObstacle` gewechselt. Ist das Fahrzeug am Hindernis vorbei wird der Infrarotstrahl des digitalen Infrarotsensors nicht mehr vom Hindernis reflektiert. Der Wert des Sensors ändert sich von `false` auf `true`. Durch diese Änderung wird ein Zustandswechsel nach `DriveToRightLane` vollzogen. Das Hindernis befindet sich nun hinter dem Fahrzeug. Das Fahrzeug wird abgebremst und wechselt zurück auf die rechte Fahrspur. Der Überholvorgang ist zu Ende. Das Fahrzeug wird auf die ursprüngliche Geschwindigkeit beschleunigt und der Subautomat beendet. Anschließend wird in den Hauptzustand `Idle` gewechselt. In der Abbildung 4.1.2 ist der Ausweichvorgang mit den dazugehörenden Subzuständen grafisch dargestellt.

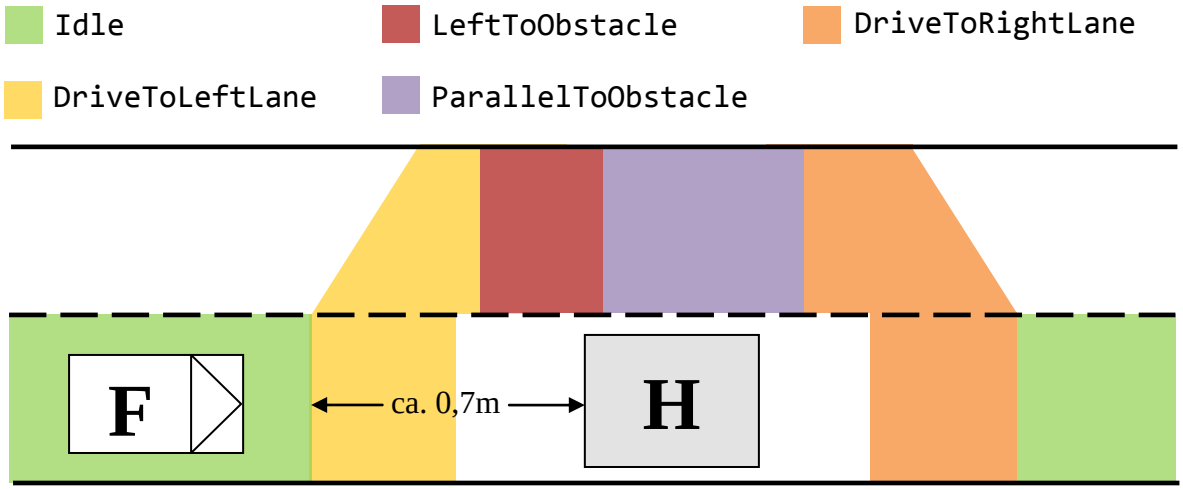


Abbildung 4.1.2: Grafische Darstellung des Ausweichvorgangs mit den dazugehörigen Subzuständen

4.2 Subautomat Kreuzungserkennung

Der Subautomat für die Kreuzungserkennung besteht aus vier Zuständen: `WaitOnCrossing`, `ProveRightBeforeLeft`, `WaitUntilCrossingClear`, `CrossingClear` und steuert den Analysevorgang des Fahrzeugs an einer Kreuzung. Der übergeordnete Hauptzustand ist `CrossingRecognized`. Er verwaltet die vier Subzustände und startet deren `execute()` Methoden (vgl. Abbildung 4.2.1).

Wird die Haltelinie einer Kreuzung erkannt befindet sich der Hauptautomat im Hauptzustand `CrossingRecognized` in diesem Zustand wird das Fahrzeug abgebremst bis es vor der Haltelinie der Kreuzung zum stilstand kommt, danach wird die `execute()` Methode des ersten Subzustandes `WaitOnCrossing` gestartet. In diesem Zustand wartet das Fahrzeug

mindestens zwei Sekunden an der Haltelinie [Carolo-Cup Regelwerk (2013)] (die Wartezeit kann als Parameter in `ParameterForOAandCR.h` übergeben werden). Ist der Timer abgelaufen findet ein Zustandswechsel in den nächsten Subzustand `ProveRightBeforeLeft` statt. In diesem Zustand wird geprüft ob die Straßenkreuzung frei ist, oder ob ein dynamisches Hindernis von rechts kommt. Ist die Kreuzung frei so wird in den Zustand `CrossingClear` gewechselt. Wird ein dynamisches Hindernis von rechts erkannt, so bleibt das Fahrzeug weiter an der Stopplinie stehen und gewährt Vorfahrt. Es findet ein Zustandswechsel in den Zustand `WaitUntilCrossingClear` statt. In diesem Zustand wartet das Fahrzeug so lange, bis das dynamische Hindernis die Kreuzung passiert hat. Ist die Kreuzung frei, so wird in den Zustand `CrossingClear` gewechselt. Hier wird der Subautomat beendet und es wird zurück in den Hauptzustand `Idle` gesprungen. In dem `Idle` Zustand übernimmt Polaris (vgl. Kapitel 7 Einleitung) wieder die Steuerung des Fahrzeugs und überquert die Kreuzung.

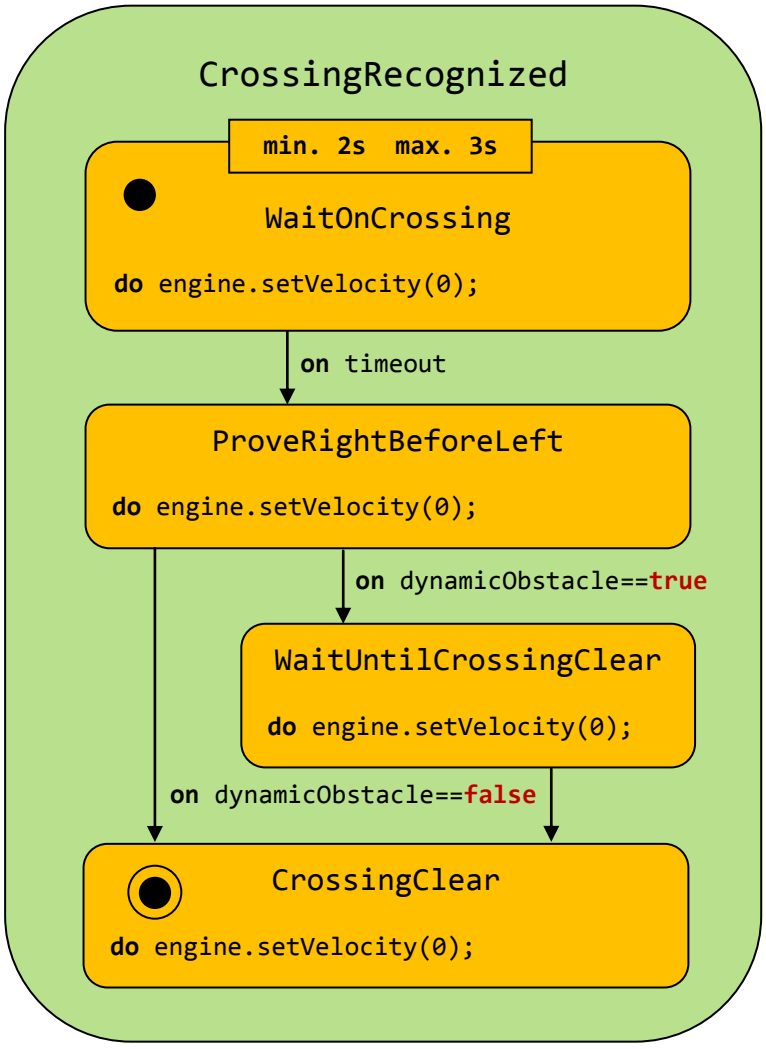


Abbildung 4.2.1: Subautomat Kreuzungserkennung

5 Hinderniserkennung und Ausweichvorgang

Eine der Disziplinen beim Carolo-Cup ist der Rundkurs mit Hindernissen. Beim Durchlauf befinden sich an mehreren Stellen des Rundkurses jeweils statische Hindernisse auf dem eigenen und gegenüberliegenden Fahrstreifen bzw. außerhalb der Fahrbahn. Alle Hindernisse bestehen aus weißen Kartons mit Abmessungen wie in Abbildung 5.0.1 angegeben. Darüber hinaus können die Kartons am Boden befestigt sein. Der minimale Abstand zwischen den Hindernissen beträgt 1000mm. Eine eindeutige Fahrstreifenzuordnung der statischen Hindernisse ist nicht immer gegeben [Carolo-Cup Regelwerk (2013)].

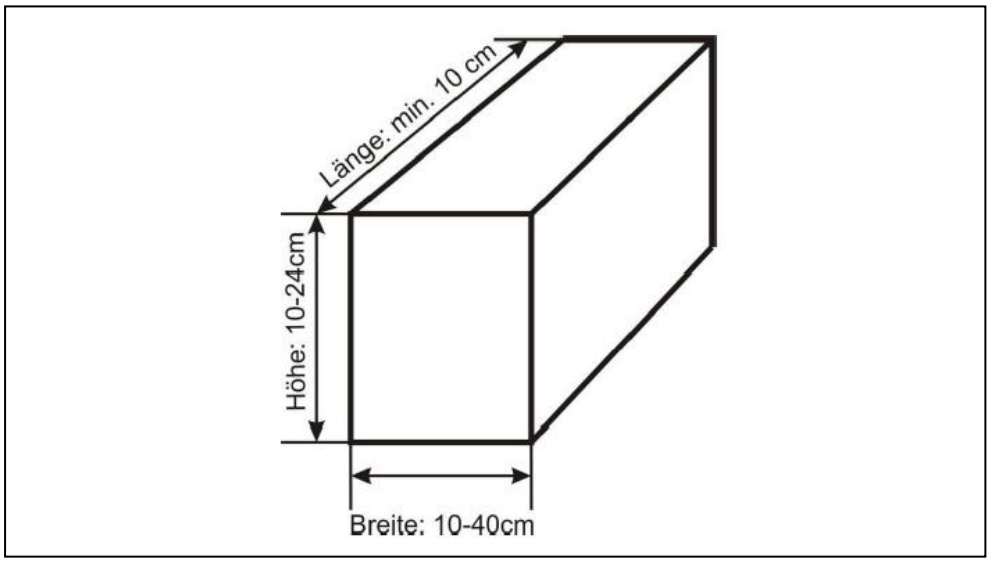


Abbildung 5.0.1: Hindernisabmessungen [Carolo-Cup Regelwerk (2013)]

Zunächst werden die Anforderungen an die Hinderniserkennung spezifiziert (vgl. Abschnitt 5.1). Anschließend wird das Erkennen von Hindernissen mit Hilfe der Kamera durch das Anlegen einer dynamischen ROI in das Kamerabild behandelt (vgl. Abschnitt 5.2). Der nächste Abschnitt beinhaltet das Erkennen von Hindernissen mit Hilfe der Infrarotsensoren (vgl. Abschnitt 5.3). Hat das Fahrzeug ein Hindernis erkannt wird der Ausweichvorgang eingeleitet, dieser ist im Abschnitt 5.4 beschrieben. Der Abschnitt 5.5 befasst sich mit dynamischen Hindernissen. Abschließend werden im Abschnitt 5.6 die Einstellparameter für die Hinderniserkennung beschrieben und erläutert.

5.1 Spezifikation der Anforderung an die Hinderniserkennung

Die Anforderungen an die Hinderniserkennung entstanden durch im Vorfeld durchgeführte Tests und Beobachtungen und orientieren sich an dem Carolo-Cup Regelwerk [Carolo-Cup Regelwerk (2013)]. Nachfolgend werden die Anforderungen aufgelistet und spezifiziert:

- **Zuverlässigkeit:** Die Hinderniserkennung soll auf den geraden Streckenabschnitten wie auch in den Kurven zuverlässig funktionieren. Es sollen alle Hindernisse mit den im Carolo-Cup Regelwerk aufgeführten Abmessungen (vgl. Abbildung 5.0.1) erkannt werden. Es müssen sowohl statische wie auch dynamische Hindernisse, die sich auf der Fahrbahn vor dem Fahrzeug befinden, bis zu einer Fahrzeuggeschwindigkeit von 1,2m/s (entspricht einem Integer-Wert von 17 für velocity vgl. Abschnitt 5.5) erkannt werden. Um ein Ausweichvorgang erfolgreich einzuleiten muss das Fahrzeug ein Hindernis in einer Entfernung von 0,6m zuverlässig erfassen. Der Algorithmus sollte stabil sein gegenüber folgenden Lichtschwankungen:
    - Lichtreflektionen auf der Fahrbahn die durch ungleichmäßige Ausleuchtung der Fahrbahn entstehen können.
    - Schatten die von Personen oder Gegenständen auf die Fahrbahn und / oder die Hindernisse geworfen werden.
- **Laufzeit:** Der Algorithmus der Hinderniserkennung sollte bei der Ausführung nicht länger als 40ms Rechenzeit benötigen, um in den FAUSTcore Systemtakt zu passen. Aufwendige double-Arithmetik sollte daher nach Möglichkeit vermieden werden.

5.2 Anlegen einer dynamischen ROI für die Hinderniserkennung

Um ein Hindernis zu erkennen wird in das Kamerabild des Fahrzeugs eine ROI mit einer bestimmten Anzahl an LOIs gelegt (vgl. Abschnitt 3.2 und Abschnitt 3.3). Über Parameter lässt sich die ROI für die Hinderniserkennung im Kamerabild positionieren (vgl. Abschnitt 5.6). Zu beachten ist, dass die ROI möglichst mittig auf der Fahrbahn positioniert ist. Wird die ROI näher am Fahrzeug positioniert, so wird ein Hindernis später erkannt. In diesem Fall ist das Fahrzeug näher am Hindernis dran bevor es dieses erkennt. Wird die ROI im Kamerabild weiter vom Fahrzeug weg positionier, so wird ein Hindernis früher erkannt. Da die ROI dazu dient Hindernisse zu erkennen wird sie nachfolgend als Hindernis-ROI bezeichnet.

Soll das Fahrzeug einem erkannten Hindernis ausweichen, darf die Erkennungsdistanz d zwischen Fahrzeug und Hindernis nicht kleiner als d_{min} ausfallen. Das ist die Distanz bei der das Fahrzeug noch rechtzeitig ausweichen kann, ohne mit dem Hindernis zu kollidieren. Bei einer Geschwindigkeit unter 1,2m/s beträgt diese minimale Distanz $d_{min} = 0{,}6m$. Damit ein Hindernis welches mit den vorne am Fahrzeug befestigten Infrarotsensoren erkannt wurde mit der Kamera verifiziert werden kann, muss die Distanz zwischen Fahrzeug und Hindernis bei beiden Hinderniserkennungsmethoden (Infrarotsensoren und Kamera) auf dieselbe Erkennungsdistanz d eingestellt sein (vgl. Abschnitt 5.3).

Um die Erkennungsdistanz einer Hindernis-ROI einzustellen wird als erstes das Fahrzeug auf einer geraden Strecke vor ein Hindernis gestellt. Die Entfernung zum Hindernis sollte die gewünschte Erkennungsdistanz *d* betragen. Die Hindernis-ROI wird nachfolgend so in das Kamerabild positioniert, das die Unterseite der Hindernis-ROI genau auf der Unterseite des Hindernisses liegt. Die Breite der Hindernis-ROI sollte in der Erkennungsdistanz *d* der Breite *b* des Fahrzeugs entsprechen. Die Höhe der Hindernis-ROI sollte in der Erkennungsdistanz *d* der Höhe *h* des kleinsten zulässigen Hindernisses entsprechen (vgl. Abbildung 5.2.1).

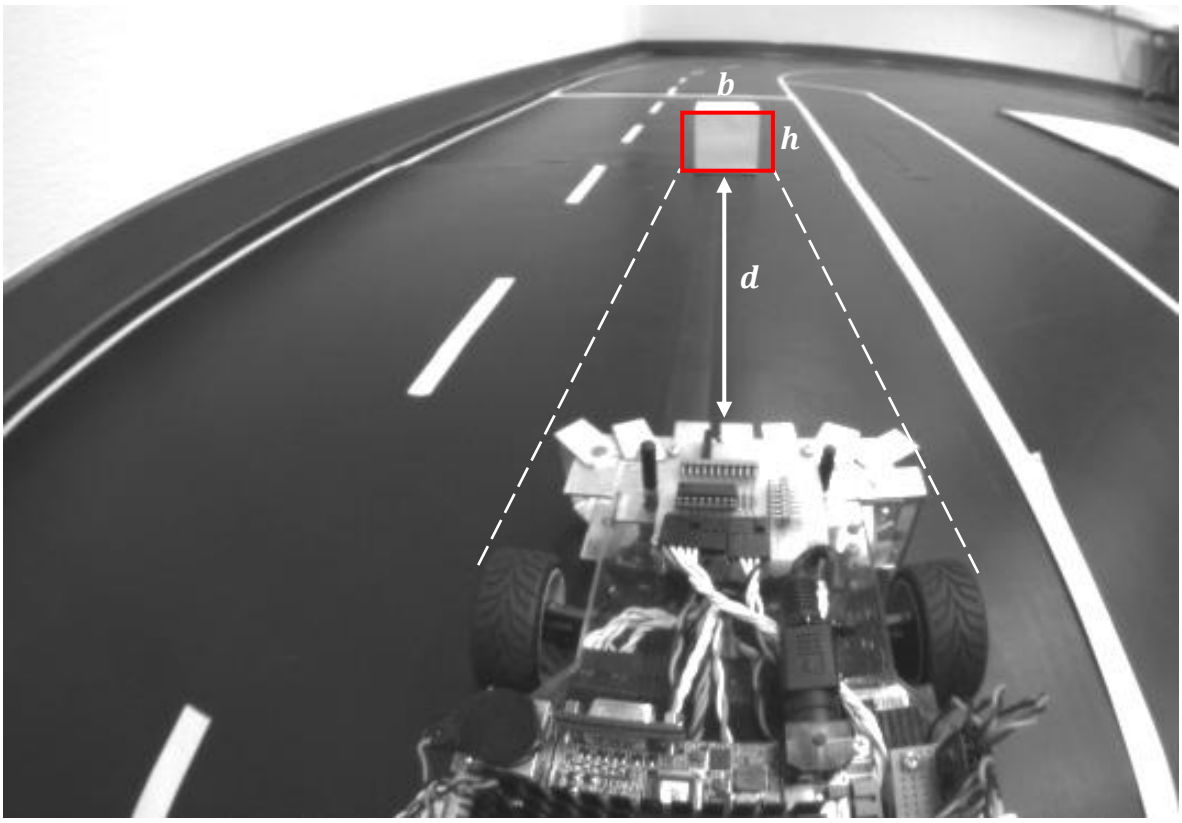


Abbildung 5.2.1: *Positionierung der Hindernis-ROI im Kamerabild der Fahrzeugkamera.*
d = Erkennungsdistanz zum Hindernis
b = Breite des Fahrzeugs in der Erkennungsdistanz d
h = Höhe des kleinsten Hindernisses in der Erkennungsdistanz d

Um zu überprüfen ob sich ein Hindernis vor dem Fahrzeug befindet muss der Rückgabewert der Methode `getPixelOverLimitInPercent()` der Hindernis-ROI überwacht werden. Dieser Wert ist abhängig von der Größe und Breite der Hindernis-ROI, von der Anzahl der LOIs und der Größe des vor dem Fahrzeug befindlichen Hindernisses. Wird die Hindernis-ROI so positioniert wie in Abbildung 5.2.2 und steigt der Rückgabewert über 35 befindet sich ein Hindernis in einer Entfernung von ca. 0,8m vor dem Fahrzeug.

Mit einer starren Hindernis-ROI wie sie oben beschrieben ist, können nur Hindernisse auf einer geraden Fahrbahn erkannt werden. Damit Hindernisse auch in einer Kurve zuverlässig erkannt werden können, muss sich die Hindernis-ROI dynamisch an die Fahrbahn anpassen.

Es muss dafür gesorgt werden, dass die Hindernis-ROI stets mittig auf der Fahrbahn positioniert ist, auch wenn sich das Fahrzeug in einer Kurve befindet.

Übergabeparameter der Methode `getPixelOverLimitInPercent()`:
`leftX = 360`
`leftY = 160`
`rightX = 400`
`rightY = 190`
} Koordinaten der Hindernis-ROI
`numberOfLines = 7` (Anzahl LOIs)
`limit = 165` (Schwellwert)

Entfernung zum Hindernis: ca. 0,9m

Rückgabewert = 30%

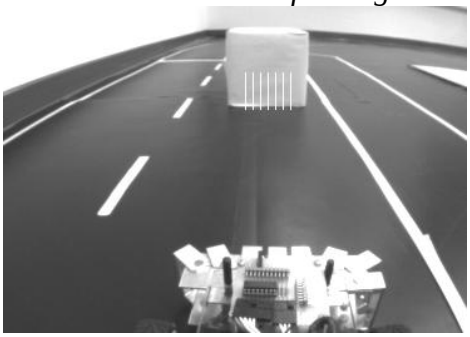
Rückgabewert = 19%

Entfernung zum Hindernis: ca. 0,8m

Rückgabewert = 47%

Rückgabewert = 35%

Entfernung zum Hindernis: ca. 0,7m



Rückgabewert = 89%

Rückgabewert = 46%

Abbildung 5.2.2: Rückgabewerte der Methode `getPixelOverLimitInPercent` *in Abhängigkeit von Entfernung und Größe der Hindernisse (rechts kleinstes zulässiges Hindernis [Carolo-Cup Regelwerk (2013)])*

Das kann erreicht werden, indem die Hindernis-ROI an den Lenkwinkel geknüpft wird. Die Berechnung des Lenkwinkels wurde entwickelt im Rahmen der Masterarbeit von Ivo Nikolov [Nikolov 2011 Kapitel 5]. Die Lenkwinkelberechnung befindet sich im Verzeichnis:

`FAUSTplugins/Tasks/SteeringControl/SteeringControl.cpp`

Über die Methode `getSteeringAngle()` die sich im Verzeichnis:

`FAUSTplugins/Data/SensorValues.h`

befindet, lässt sich der eingestellte Lenkwinkelwert abgreifen. Der Lenkwinkel wird als Integer-Wert zurückgegeben. Ein negativer Rückgabewert von -100 ist ein vollausschlag der Vorderräder nach links. Ein positiver Rückgabewert von 100 ist ein vollausschlag der Vorderräder nach rechts. Rückgabewerte der Methode `getSteeringAngle()` in abhängigkeit von der Fahrbahn sind in der Abbildung 5.2.3 aufgeführt. Das Mathematische Fahrspurmodell zur Abbildung von Fahrspurmarkierungen, sowie die Identifikation von Fahrzeuglage, Fahrzeugsteuerkurs und Fahrspurradius können in der Masterarbeit von Eike Jenning [Jenning 2008 Kapitel 2 und Kapitel 4] eingesehen werden.

SteeringAngle										
13	1	-5	25	75	83	78	73	71	64	7
11	1	-5	31	77	83	77	73	71	63	5
11	1	-5	35	79	83	77	73	71	60	1
11	1	-4	41	79	82	77	73	71	57	-1
10	0	-4	44	80	82	77	72	71	43	-4
10	0	-3	47	81	82	76	72	71	56	-4
9	-1	0	50	82	81	75	71	70	56	-11
9	-2	0	53	82	81	75	71	70	45	-14
9	-2	0	57	82	81	75	71	70	42	-22
7	-2	2	57	83	81	75	71	70	37	-25
6	-3	6	63	83	81	75	70	70	32	-28
5	-3	9	65	83	80	74	70	71	29	-31
4	-4	10	67	83	80	74	70	71	26	-36
4	-4	14	69	83	80	73	70	70	26	-39
3	-4	16	72	83	78	73	70	69	16	-39
2	-4	22	73	83	78	74	71	68	16	-42
1	-5	25	73	83	78	73	71	67	11	

Rechtskurve gerade Strecke Linkskurve

Abbildung 5.2.3: Rückgabewerte der Methode `getSteeringAngle` *in Abhängigkeit von der Fahrbahn (Abmessungen der Rechtskurve siehe Abbildung 5.2.4)*

Damit sich die Hindernis-ROI dynamisch an die Fahrspur anpasst, wird diese über zwei Faktoren an den Lenkwinkel geknüpft. Der eine Faktor ist der Offset für die X-Koordinate, der andere Faktor ist der Offset für die Y-Koordinate der Hindernis-ROI. Die beiden Faktoren für den Offset können mit Hilfe des Kamerabildes und des Lenkwinkels berechnet werden (vgl. Abbildung 5.2.4). Dazu wird neben die Hindernis-ROI aus Abbildung 5.2.2 mit deren Hilfe Hindernisse auf geraden Fahrbahnabschnitten erkannt werden (rot) eine weitere ROI so in das Bild gelegt, dass sie in der Rechtskurve mittig auf der Fahrbahn liegt (grün). Um den X-Offset für die Hindernis-ROI zu berechnen wird von der X-Koordinate der rechten unteren

Ecke der grünen ROI die X-Koordinaten der rechten unteren Ecke der roten Hindernis-ROI subtrahiert und durch den Lenkwinkel dividiert:

$$obstacleOffsetX = \frac{rightX_{green} - rightX_{red}}{SteeringAngle} = \frac{520 - 400}{44} \approx 2{,}7 \qquad (4.1)$$

Um den Y-Offset für die Hindernis-ROI zu berechnen wird von der Y-Koordinate der rechten unteren Ecke der grünen ROI die Y-Koordinate der rechten unteren Ecke der roten Hindernis-ROI subtrahiert und der Lenkwinkel durch das Ergebnis dividiert:

$$obstacleOffsetY = \frac{SteeringAngle}{rightY_{green} - rightY_{red}} = \frac{44}{210 - 190} = 2{,}2 \qquad (4.2)$$

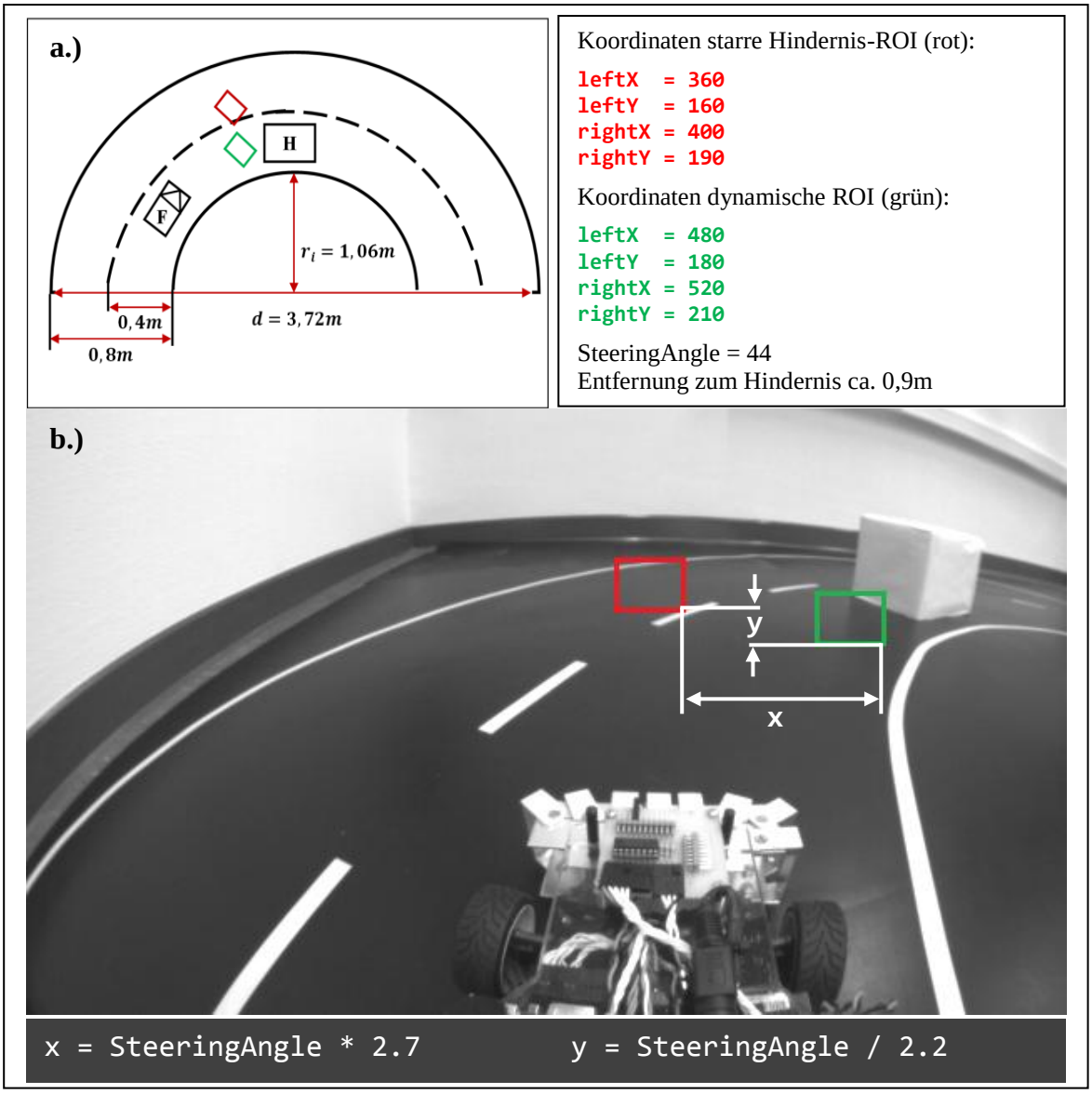


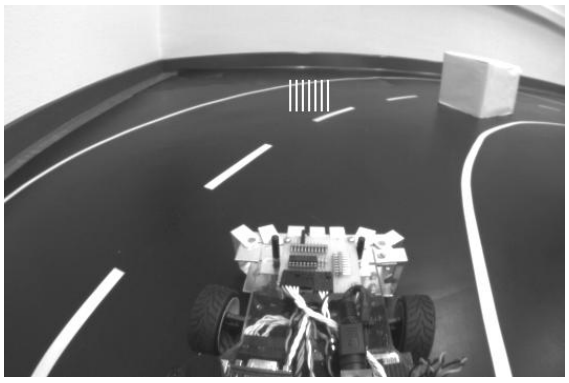
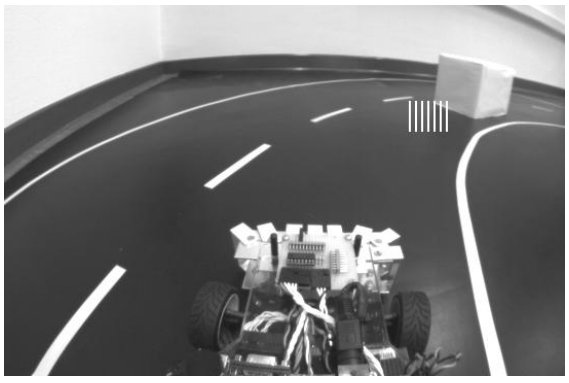
Abbildung 5.2.4: *Offsetberechnung für eine dynamische ROI (grün).*
a.) Schematische Darstellung. b.) Ausschnitt aus original Kamerabild.

Damit sich die Hindernis-ROI dynamisch an die Fahrbahn anpasst, müssen ihre Koordinaten wie folgt an den Lenkwinkel geknüpft werden:

```
if(SteeringAngle > 0){
      obstacleROI->setOffsetX(SteeringAngle * obstacleOffsetX);
      obstacleROI->setOffsetY(SteeringAngle / obstacleOffsetY);
}                                                                  } Rechtskurve
else{
      obstacleROI->setOffsetX(SteeringAngle * obstacleOffsetX);
      obstacleROI->setOffsetY(SteeringAngle /-obstacleOffsetY);
}                                                                  } Linkskurve
```

Bei der Implementierung des Offsets für die dynamische Hindernis-ROI muss zwischen Rechts -und Linkskurve unterschieden werden. Bei einer Linkskurve sind die werte von `SteeringAngle` negativ, deshalb muss der `obstacleOffsetY` mit (-1) multipliziert werden, damit sich die Hindernis-ROI nach unten verschiebt (vgl. Abbildung 5.2.5).

starre Hindernis-ROI	dynamische Hindernis-ROI
`SteeringAngle = 44;`	

	`obstacleOffsetX = 2.7`
	`obstacleOffsetY = 2.2`
	 

`SteeringAngle = -36;`	

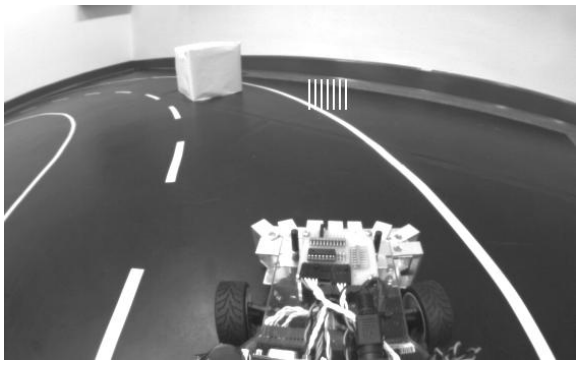
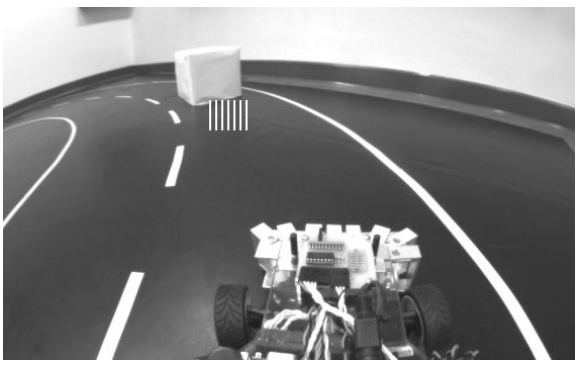
	`obstacleOffsetX =  2.7`
	`obstacleOffsetY = -2.2`
	 

Abbildung 5.2.5: Kurvenverhalten einer starren Hindernis-ROI im Vergleich zu einer vom Lenkwinkel abhängigen dynamischen Hindernis-ROI.

5.3 Hinderniserkennung mit den Infrarotsensoren

Um die Distanz zu einem Objekt messen zu können, wurden an der Vorderseite des Fahrzeugs sieben Infrarot-Entfernungsmesser der Sharp Electronics GmbH angebracht. Diese Sensoren sind an fünf Aluminiumträger befestigt. Die vorderen drei Infrarotsensoren sind fest am Fahrzeug montiert, ihr Winkel lässt sich nicht verstellen. Der Winkel der beiden linken und rechten Infrarotsensoren ist variabel und kann verstellt werden. Fünf der Sensoren sind für die Hinderniserkennung zuständig und haben einen Messbereich von 20cm bis 150cm. Die anderen beiden Infrarotsensoren werden für das Einparken benötigt und haben einen Messbereich von 10cm bis 80cm und 4cm bis 30cm (vgl. Abbildung 5.3.1).

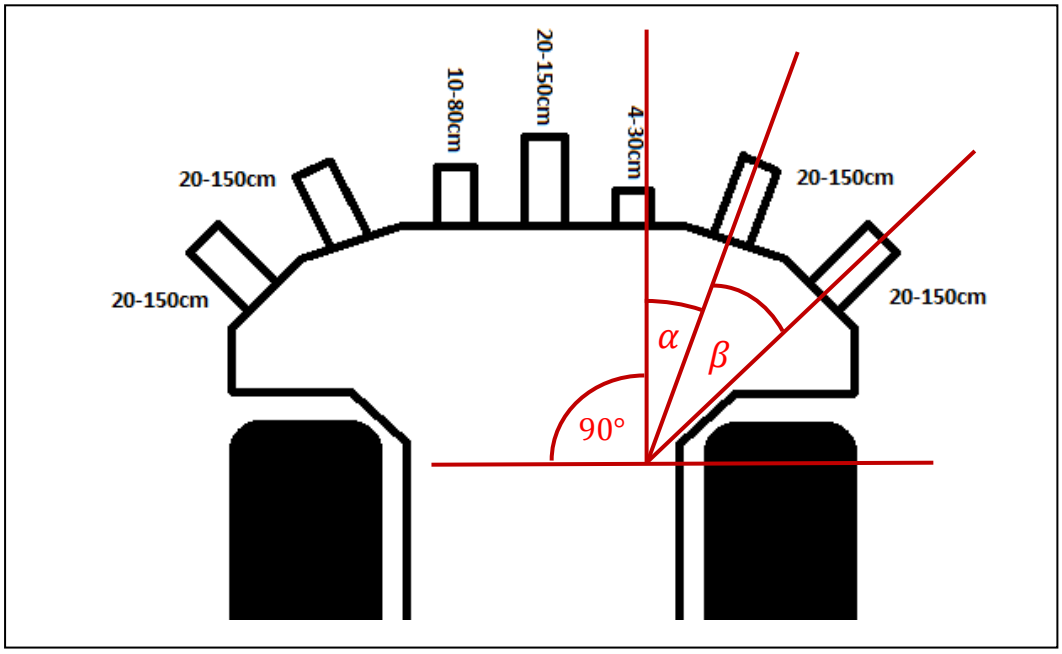


Abbildung 5.3.1: Anordnung der Infrarotsensoren an der Vorderseite des Fahrzeugs

Die für die Hinderniserkennung zuständigen Infrarotsensoren haben folgende Bezeichnung: `IrLongFrontLeftLeft; IrLongFrontLeft; IrLongFrontCenter; IrLongFrontRight; IrLongFrontRightRight` (v. l.) Die technischen Daten der Infrarotsensoren sind in der Tabelle 5.1 aufgelistet.

Messbereich:	20cm bis 150 cm
Betriebsspannung (VCC):	4,5V - 5,5V
Ausgangsspannung (Vout):	0,4V bis 2,6V (150cm - 20cm)
Stromaufnahme:	33mA (max. 50mA)
Bezeichnung:	GP2Y0A02YK
Hersteller:	Sharp

Tabelle 5.1: Technische Daten der Infrarotsensoren [Datenblatt GP2Y0A02YK]

Funktionsweise:

Der Sender strahlt einen durch die Linse stark gebündelten IR-Strahl aus, der vom Hindernis reflektiert wird. Die Detektion der Entfernung erfolgt mittels Triangulation (Je weiter entfernt das Hindernis, desto kleiner der Winkel zwischen emittierten und reflektierten Lichtstrahl). Anstelle einer üblichen Photodiode enthalten die SHARP-Sensoren ein PSD ("Position Sensitive Device", ist mit einer in die Länge gezogenen Photodiode zu vergleichen). Je nachdem, an welcher Stelle der empfangene Lichtstrahl auf die Photofläche trifft, teilt sich der Fotostrom entsprechend der Position des Punktes in zwei Teile (vgl. Abbildung 5.3.2). Aus dem Quotienten von Differenz und Summe der beiden Ströme erhält man ein von der Lichtleistung unabhängiges Positionssignal [TUM].

a.) *b.)*

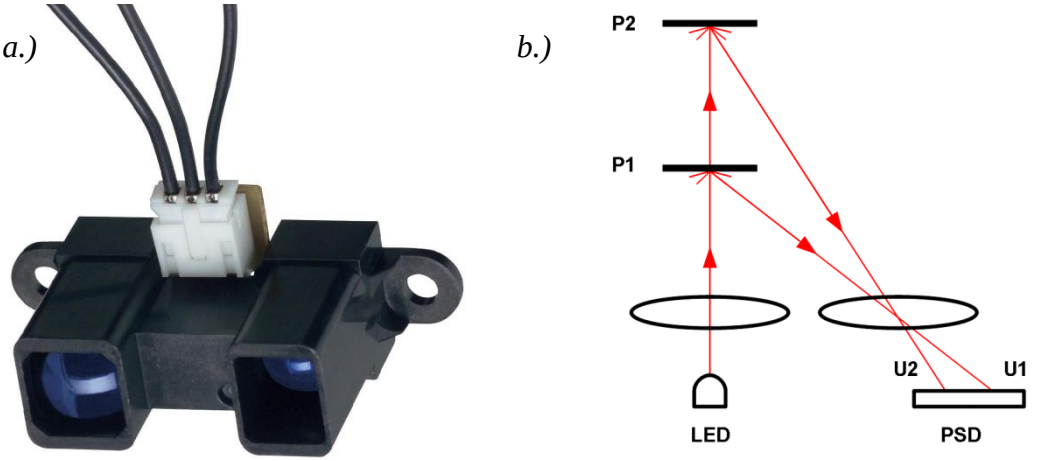


Abbildung 5.3.2: *a.) Distanz-Sensor von Sharp [*GP2Y0A02YK Abbildung*]*
b.) Funktionsweise des Distanz-Sensors [roboticlab]

Die Sharp Infrarotsensoren verfügen über drei Leitungen: Versorgungsleitung, Masse und Signalleitung. Für die Ansteuerung muss kein Taktsignal generiert werden. Die Elektronik im Sensor ermittelt die Entfernung und gibt diese über ein Analogsignal an den Analog-Digital-Konverter des am Fahrzeug montierten Mikrokontroller (MCB9B500) weiter. Dabei entspricht eine Spannung von etwa 2,6V einer Entfernung zum Hindernis von etwa 20 cm und eine Spannung von ca. 0,4V einer Entfernung von 150cm (vgl. Abbildung 5.3.3).

Um die Entfernung der Infrarotsensoren in Zentimeter auslesen zu können, wurden in der Datei `RS232RxFrame.cpp`, die sich im Verzeichnis:

`FAUSTplugins/Driver/RS232/RS232RxFrame.`cpp

befindet, zwei Lookup-Tabellen in Form von zwei Integer Arrays angelegt:

`static int cmLong[131]{...}` für die Werte in Zentimeter
`static int analogLong[131]{...}` für die Analogwerte der Sensoren

um aus den Analogwerten die Entfernungsangaben in Zentimeter zu bekommen, wird die Methode `AnalogToCM(int *m, int n, int x){...}` verwendet.

Der Zugriff auf die Sensorwerte erfolgt folgendermaßen:

```
SensorValuesPtr sValues = DataContainer<SensorValues>::instance().getData();

std::cout<<"FrontLeft:   " << sValues->getIrLongFrontLeft()<<std::endl;
std::cout<<"FrontCenter: " << sValues->getIrLongFrontCenter()<<std::endl;
std::cout<<"FrontRight:  " << sValues->getIrLongFrontRight()<<std::endl;
```

Der Rückgabewert ist ein Integer-Wert der die Entfernung zum Hindernis in Zentimeter angibt.

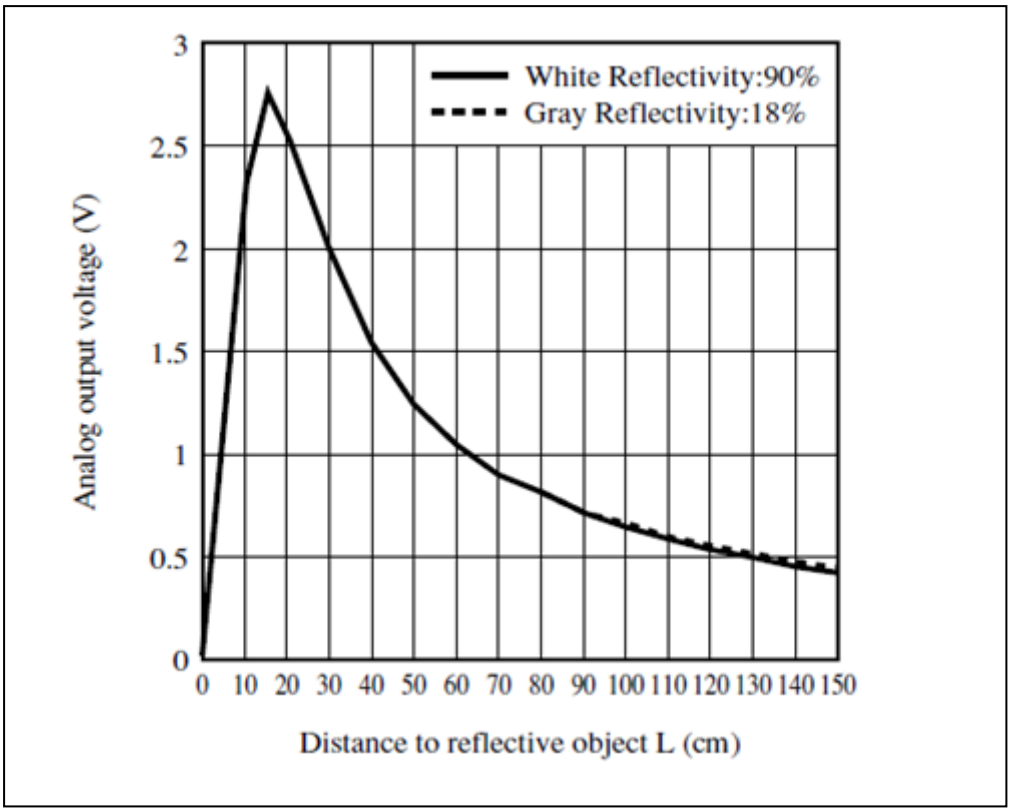


Abbildung 5.3.3: Spannungswerte des Distanz-Sensor von Sharp am Analogausgang in Abhängigkeit von der Entfernung [Datenblatt GP2Y0A02YK]

Hinderniserkennung mit den Infrarotsensoren auf einer geraden Fahrbahn:

Um ein Hindernis mit den Infrarotsensoren auf einer geraden Fahrbahn zu erkennen muss als erstes die Distanz festgelegt werden in der ein Hindernis als erkannt gilt. Diese Erkennungsdistanz d darf den maximalen Messbereich $SensorRange_{max}$ der am Fahrzeug befestigten Infrarotsensoren nicht überschreiten. Sie sollte auch nicht zu gering ausfallen damit das Fahrzeug noch rechtzeitig ausweichen kann, ohne mit dem Hindernis zu kollidieren. Wie

schon bei der Hinderniserkennung mit der Kamera, darf die Erkennungsdistanz nicht kleiner sein als die minimale Distanz d_{min} in der das Fahrzeug sicher einem Hindernis ausweichen kann ohne mit diesem zu kollidieren (Bei einer Geschwindigkeit unter 1,2m/s: $d_{min} = 0{,}6m$ vgl. Abschnitt 5.2). Um Ungenauigkeiten der Infrarotsensoren bei der Entfernungsmessung abzufangen wird zusätzlich ein Sicherheitspuffer d_{Puffer} benötigt. Der Sicherheitspuffer sollte je nach Genauigkeit der Sensoren zwischen 0,05m bis 0,1m betragen. Um ein Hindernis mit den Infrarotsensoren zu erkennen und ihm erfolgreich ausweichen zu können, muss die Erkennungsdistanz d auf ein Wert eingestellt sein zwischen $SensorRange_{max}$ und der Summe von d_{min} und d_{Puffer}:

$$SensorRange_{max} > d > d_{min} + d_{Puffer} \tag{5.1}$$

Um feststellen zu können ob sich auf einer geraden Fahrbahn vor dem Fahrzeug ein Hindernis befindet, müssen die drei vorderen Infrarotsensoren auf die Erkennungsdistanz d abgefragt werden. Ist der Rückgabewert aller drei vorderen Sensoren größer d so befindet sich kein Hindernis in der Erkennungsdistanz vor dem Fahrzeug. Sinkt der Wert eines der drei vorderen Sensoren unter d so befindet sich ein Hindernis vor dem Fahrzeug. Dieselbe Erkennungsdistanz d muss auch für die Hindernis-ROI eingestellt werden, damit ein Hindernis welches mit den Infrarotsensoren erkannt wurde mit der Kamera verifiziert werden kann.

Damit möglichst alle Hindernisse die sich vor dem Fahrzeug auf einer geraden Fahrbahn befinden erkannt werden, müssen die beiden Infrarotsensoren `IrLongFrontLeft` und `IrLongFrontRight` (die sich links und rechts von dem im Zentrum montierten Sensor befinden) in einem Winkel α abstrahlen der die gesamte Breite b des Fahrzeugs in der gewünschten Erkennungsdistanz d abdeckt (vgl. Abbildung 5.3.4).

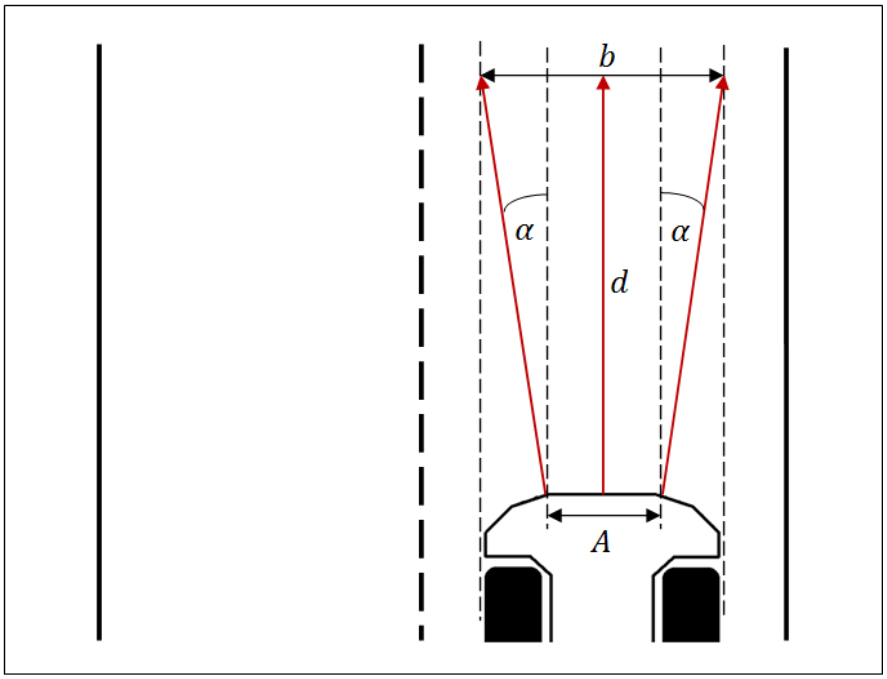


Abbildung 5.3.4: Winkeleinstellung der Infrarotsensoren: `IrLongFrontLeft` *und* `IrLongFrontRight`

Dieser Winkel α wird folgendermaßen berechnet:

$$\tan(\alpha) = \frac{\frac{b-A}{2}}{d} = \frac{b-A}{2d} \tag{5.2}$$

dabei ist b die Breite des Fahrzeugs, d die gewünschte Erkennungsdistanz in der ein Hindernis erkannt werden soll und A der Abstand des linken Infrarotsensors vom rechten Infrarotsensor.

Hinderniserkennung mit den Infrarotsensoren in einer Kurve:

Auf einer geraden Fahrbahn ist es nicht notwendig alle Infrarotsensoren abzufragen. Die äußersten Sensoren `IrLongFrontLeftLeft` und `IrLongFrontRightRight` werden auf einer geraden Fahrbahn nicht benötigt. Diese Infrarotsensoren sind für die Hinderniserkennung in Kurven zuständig. Ihr Winkel wird so gewählt, dass ihr Infrarotstrahl in die jeweilige Kurve strahlt. Das Fahrzeug kann mit ihrer Hilfe in die Kurve "hinein schauen" und feststellen ob sich dort ein Hindernis befindet (vgl. Abbildung 5.3.5). Um festzustellen ob sich das Fahrzeug in einer Kurve befindet wird der Lenkwinkel δ abgefragt. Ist der Lenkwinkel positiv und überschreitet er einen vorher festgelegten Wert ab dem eine Kurve als erkannt gilt z. B. 15, so befindet sich das Fahrzeug in einer Rechtskurve. In dieser werden nur die beiden rechten Infrarotsensoren `IrLongFrontRightRight` und `IrLongFrontRight` abgefragt. Die Werte der anderen Sensoren werden nicht benötigt, da sie in einem festen Winkel am Fahrzeug angebracht sind, strahlt ihr Infrarotstrahl über den Mittelstreifen auf die Gegenfahrbahn und liefert dadurch keine brauchbaren Informationen. Befindet sich das Fahrzeug in einer Linkskurve so werden nur die Infrarotsensoren `IrLongFrontLeftLeft` und `IrLongFrontLeft` abgefragt. Der Infrarotstrahl der anderen Sensoren strahlt über den Seitenstreifen der Fahrbahn und liefert deshalb keine brauchbaren Informationen.

Der Winkel β, der sich zwischen `IrLongFrontRight` und `IrLongFrontRightRight` befindet, wird mit Hilfe des Kosinussatzes und dem Winkel α aus der Gleichung 5.2 berechnet. Dazu wird ein Hindernis mittig in die Kurve mit dem kleinsten Innenradius gestellt und das Fahrzeug auf der Fahrbahn vor dem Hindernis so platziert, dass es die gewünschte Entfernung bei der ein Hindernis erkannt werden soll einnimmt (vgl. Abbildung 5.3.5).

$$\beta = 90° - \alpha - \gamma \tag{5.3}$$

$$\cos(\gamma) = \frac{r^2 - d^2 - R^2}{-2dR} \tag{5.4}$$

Die Bestimmung des Winkels β zwischen den beiden Sensoren auf der linken Seite des Fahrzeugs `IrLongFrontLeftLeft` und `IrLongFrontLeft` erfolgt nach demselben Verfahren.

Befindet sich das Fahrzeug in einer Rechtskurve, so werden nur die beiden rechten Sensoren abgefragt. Sinkt ihr Wert unter die für die Hinderniserkennung festgelegte Erkennungsdistanz d, so befindet sich ein Hindernis vor dem Fahrzeug. In einer Linkskurve werden nur die beiden linken Infrarotsensoren abgefragt. Sinkt ihr Wert unter d so befindet sich auch in

diesem Fall ein Hindernis vor dem Fahrzeug. Ist in beiden Fällen der Rückgabewert der Sensoren größer als d so befindet sich kein Hindernis vor dem Fahrzeug in der Erkennungsdistanz.

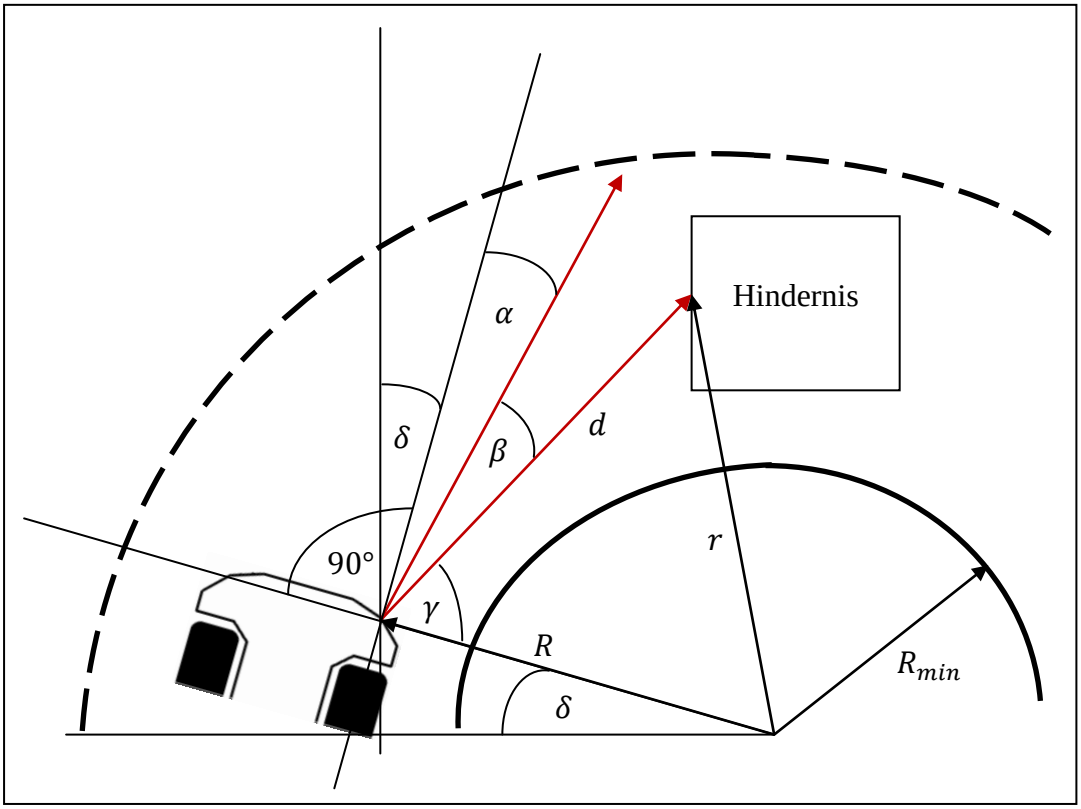


Abbildung 5.3.5: Einstellung des Winkels β des äußeren Infrarotsensors für die Hinderniserkennung in einer Kurve.

5.4 Ausweichvorgang

Befindet sich das Fahrzeug im eingeschalteten Zustand auf der Fahrbahn, so folgt es der rechten Fahrspur mit Hilfe von Polaris dem kamerabasierten Spurerkennungsalgorithmus (vgl. Kapitel 7 Einleitung). Dieser Zustand ist der `Idle`-Zustand der Hinderniserkennung. In diesem Zustand wird die Fahrbahn vor dem Fahrzeug nur mit den Infrarotsensoren nach Hindernissen "gescannt". Dazu werden die jeweiligen Infrarotsensoren, je nachdem wo sich das Fahrzeug befindet (in einer Kurve oder auf einer gerader Strecke vgl. Abschnitt 5.3), abgefragt. Wird durch eines der abgefragten Infrarotsensoren ein Hindernis erkannt, so wird dieser Befund mit Hilfe der Kamera und einer für die Hinderniserkennung in das Kamerabild gelegten dynamischen ROI (vgl. Abschnitt 5.2) verifiziert. Wird von der Kamera kein Hindernis erkannt, bleibt der Hauptautomat der Hinderniserkennung im `Idle`-Zustand. Wird jedoch auch von der Kamera ein Hindernis erkannt, so wird in den Zustand `ObstacleRecognized` gewechselt. In diesem Zustand befindet sich ein Subautomat der den Ausweichvorgang steuert (vgl. Abschnitt 4.1). Während des Ausweichvorgangs werden weder die Infrarotsensoren noch die Kamera abgefragt. Ist der Ausweichvorgang abgesch-

lossen wird mit Hilfe der Infrarotsensoren wieder nach Hindernissen vor dem Fahrzeug "gescannt". Die Hinderniserkennung mit der Kamera dient weiterhin nur als Verifizierung für die mit den Infrarotsensoren erkannten Hindernisse. Auf diese Weise verbrauch die Hinderniserkennung weniger Ressourcen.

5.5 Dynamische Hindernisse

Beim Carolo-Cup Wettbewerb befinden sich nicht nur statische Hindernisse auf der Fahrbahn, es ist auch mindestens ein dynamisches Hindernis unterwegs. Ein derartiges Hindernis ähnelt von seinem Äußeren den statischen Hindernissen („fahrender weißer Karton mit den entsprechenden Abmaßen“ vgl. Abbildung 5.0.1) und kann auf dem eigenen Fahrstreifen, auf dem Gegenfahrstreifen oder auch an Kreuzungen auftauchen. Es bewegt sich mit einer maximalen Geschwindigkeit von 0.6 m/s auf seinem Fahrstreifen. Situationen, bei denen durch das dynamische Hindernis die komplette Fahrbahn blockiert wird, sind ausgeschlossen [Carolo-Cup Regelwerk (2013)].

Das Erkennen dieser dynamischen Hindernisse erfolgt nach demselben Prinzip wie auch das Erkennen von statischen Hindernissen (vgl. Abschnitt 5.2 und Abschnitt 5.3). Bei Überholen eines dynamischen Hindernisses muss jedoch festgestellt werden wie schnell das Fahrzeug fahren muss um ein dynamisches Hindernis zu überholen, wie lange es dauert und welche Strecke wird bei unterschiedlichen Geschwindigkeiten zurückgelegt. Damit diese Fragen beantwortet werden können wurde ein Versuch durchgeführt.

Versuchsbeschreibung:

Um das Fahrzeug mit einer bestimmten Geschwindigkeit fahren zu lassen wird folgender Aufruf benötigt:

```
engine.setVelocity(12);
```

Die übergebene Zahl in diesem Fall die zwölf ist ein Integer-Wert. Ist der Wert positiv fährt das Fahrzeug vorwärts. Bei einem negativen Wert fährt das Fahrzeug rückwärts. 100 ist der maximale positive Wert und -100 ist der maximale negative Wert. Bei einem Integer-Wert von Null bleibt das Fahrzeug stehen. Je nach Ladung des Akkumulators bewegt sich das Fahrzeug erst ab einem Wert zwischen acht und zehn vorwärts.

Um eine genaue Aussage treffen zu können wie schnell das Fahrzeug sich fortbewegt, müssen wir wissen wie lange ($Zeit$: $t[s]$) das Fahrzeug für eine konstante $Strecke$ ($s[m]$) braucht. Aus den Messergebnissen wird die $Geschwindigkeit$ ($v[m/s]$) für die jeweiligen Integer-Werte berechnet.

Versuchsaufbau und Versuchsdurchführung:

Die Geschwindigkeitsmessung findet auf einem geraden Abschnitt der Teststrecke statt. Die Länge des Abschnitts beträgt 3,50m (vgl. Abbildung 5.5.1). Zum Versuchszeitpunkt sind beide Akkumulatoren des Fahrzeugs vollständig aufgeladen. Es werden die Zeiten für folgende für das Fahrzeug gebräuchliche Integer-Werte gemessen:

```
11, 12, 13, 15, 17, 20
```

Zu jedem Integer-Wert finden fünf Messungen statt und es werden fünf Zeiten (t_1 bis t_5) ermittelt (vgl. Tabelle 5.2). Anschließend wird aus den fünf Messwerten die jeweilige Durchschnittszeit ($\bar{t}$) berechnet. Aus der Strecke und der berechneten Durchschnittszeit wird für jeden Integer-Wert die dazugehörige Geschwindigkeit errechnet (vgl. Tabelle 5.3).

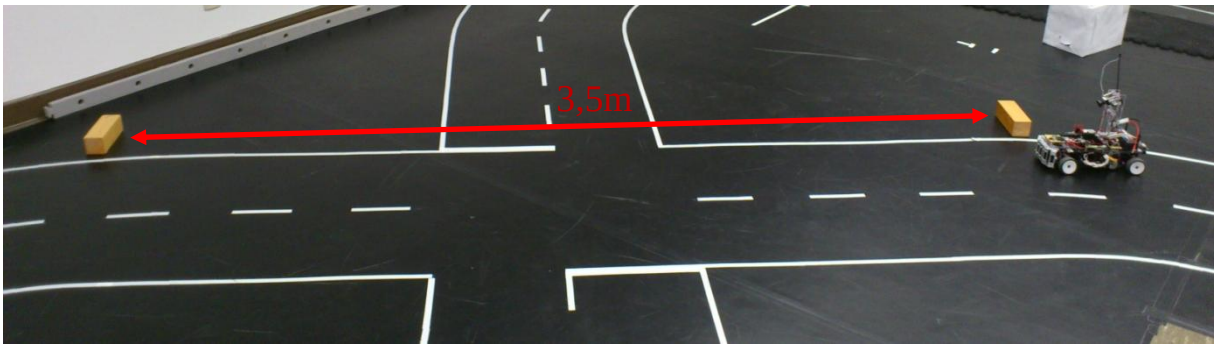


Abbildung 5.5.1: Versuchsaufbau zur Geschwindigkeitsmessung

Beobachtung:

velocity (int)	distance ($s[m]$)	time ($t[s]$)				
		t_1	t_2	t_3	t_4	t_5
11	3,50	4,483	4,555	4,861	4,882	4,875
12	3,50	4,109	4,181	4,085	4,140	3,859
13	3,50	3,468	3,598	3,347	3,762	3,595
15	3,50	3,268	3,015	3,106	2,928	3,232
17	3,50	2,961	2,778	2,901	2,946	3,041
20	3,50	2,385	2,451	2,445	2,512	2,572

Tabelle 5.2: Ergebnis der Zeitmessungen

Ergebnis:

Die Durchschnittszeit zum jeweiligen Integer-Wert wird berechnet durch:

$n = Anzahl\ Messungen$

$$\bar{t} = \frac{\sum_{i=1}^{n} t_i}{n} \tag{5.5}$$

Die Geschwindigkeit zum jeweiligen Integer-Wert wird berechnet durch:

$$v = \frac{s}{\bar{t}} \tag{5.6}$$

Die berechneten Geschwindigkeiten stehen in Tabelle 5.3

velocity(*int*)	time (*t[s]*)					av. time ($\bar{t}$ [s])	velocity ($v[\frac{m}{s}]$)
11	4,483	4,555	4,861	4,882	4,875	4,7312	0,74
12	4,109	4,181	4,085	4,140	3,859	4,0748	0,86
13	3,468	3,598	3,347	3,762	3,595	3,5540	0,98
15	3,268	3,015	3,106	2,928	3,232	3,1098	1,13
17	2,961	2,778	2,901	2,946	3,041	2,9254	1,20
20	2,385	2,451	2,445	2,512	2,572	2,4730	1,42

Tabelle 5.3: Ergebnis der Geschwindigkeitsmessung

Nachdem die Geschwindigkeiten zu den jeweiligen Integer-Werten bekannt sind, kann berechnet werden wie viel Zeit das Fahrzeug für den Überholvorgang eines dynamischen Hindernisses mit der Länge $l[m]$ benötigt und welche Strecke es dabei zurücklegt.

die Dauer des Überholvorganges wird berechnet durch:

$t \quad = Dauer\ des\ Überholvorgangs$
$v_F = Geschwindigkeit\ des\ Fahrzeugs$
$v_H = Geschwindigkeit\ des\ dynamischen\ Hindernisses$
$l \quad = Länge\ des\ dynamischen\ Hindernisses$

$$t = \frac{l}{v_F - v_H} \tag{5.7}$$

Für ein dynamisches Hindernis mit einer Länge von $l = 0{,}6m$ und einer Geschwindigkeit von $v_H = 0{,}6m/s$ lassen sich die Dauer des Überholvorganges $t[s]$ und die Länge der zurückgelegten Strecke $s[m]$ bei einer Fahrzeuggeschwindigkeit von $v_F = 0{,}74m/s$ folgendermaßen berechnen:

gegeben: Fahrzeuggeschwindigkeit $v_F = 0{,}74m/s$ (Integer-Wert elf); Geschwindigkeit dynamisches Hindernis $v_H = 0{,}6m/s$; Länge dynamisches Hindernis $l = 0{,}6m$
gesucht: Überholdauer $t[s]$; Überholstrecke $s[m]$

$$t = \frac{l}{v_F - v_H} = \frac{0{,}6m}{0{,}74m/s - 0{,}6m/s} = \frac{30}{7}s \approx 4{,}29s$$

$$s = v_F * t = 0{,}74m/s * 4{,}29s = 3{,}17m$$

Um ein dynamisches Hindernis mit einer Länge von $l = 0{,}6m$ und einer Geschwindigkeit von $v_H = 0{,}6m/s$ zu überholen benötigt der Saphir bei einer Geschwindigkeit von $v_F = 0{,}74m/s$ was einem Integer-Wert von elf entspricht, ca. $4{,}29s$ und legt dabei eine Strecke von $3{,}17m$ zurück. Weitere Werte für die Überholdauer und die Überholstrecke stehen in der Tabelle 5.4. Um einen schnelleren Überholvorgang zu gewährleisten und die Überholstrecke zu verkürzen,

gibt es zwei Möglichkeiten. Die erste Möglichkeit ist das Fahrzeug generell mit einer höheren Geschwindigkeit fahren zu lassen. Zum Beispiel mit 1,13m/s (Integer-Wert 15) dadurch würde das Fahrzeug eine Überholstrecke von 1,27m benötigen. Ein Nachteil dieser Methode ist, dass Polaris Schwierigkeiten hat beim Ausweichvorgang die Fahrspur einwandfrei zu wechseln und die neue Fahrspur zu halten. Wird die Fahrspur bei dieser Geschwindigkeit gewechselt, verliert sich Polaris und das Fahrzeug verlässt die Fahrbahn. Zu gegebenem Zeitpunkt ist das Fahrzeug in der Lage bis zu einer Geschwindigkeit von 1m/s die Fahrspur sicher zu wechseln. Das Verlassen der Fahrbahn kann verhindert werden, indem das Fahrzeug nachdem es ein Hindernis erkannt hat auf eine Geschwindigkeit von 0,86m/s (Integer-Wert 12) abgebremst wird und erst dann die Fahrspur wechselt. Nach dem Fahrspurwechsel wird das Fahrzeug auf die ursprüngliche Geschwindigkeit beschleunigt. Wurde das Hindernis erfolgreich überholt wird das Fahrzeug wieder auf eine Geschwindigkeit von 0,86m/s abgebremst bevor es erneut die Fahrspur wechselt. Die zweite Möglichkeit wäre das Fahrzeug mit einer konstanten Geschwindigkeit fahren zu lassen, bei der Polaris keine Probleme hat die Fahrspur zu wechseln zum Beispiel 0,98m/s (Integer-Wert 13). Bei einer Geschwindigkeit von 0,98m/s beträgt die Überholstrecke 1,55m. Um die Überholstrecke und somit auch die Überholzeit nochmals zu verkürzen, kann das Fahrzeug beim Überholen eines dynamischen Hindernisses beschleunigt werden. Die Erhöhung der Geschwindigkeit darf jedoch erst erfolgen, nachdem die Spur erfolgreich gewechselt wurde und Polaris sich auf die neue Spur eingestellt hat. Wurde das Hindernis erfolgreich überholt muss das Fahrzeug auf eine Geschwindigkeit von unter 1m/s abgebremst werden, bevor die Fahrspur gewechselt und somit der Überholvorgang abgeschlossen wird.

velocity (*int*)	Geschw. v_F [$\frac{m}{s}$]	Geschw. v_H [$\frac{m}{s}$]	Überholdauer t [s]	Überholstrecke s [m]
11	0,74	0,6	4,29	3,17
12	0,86	0,6	2,31	1,98
13	0,98	0,6	1,58	1,55
15	1,13	0,6	1,13	1,27
17	1,20	0,6	1,00	1,20
20	1,42	0,6	0,73	1,04

Tabelle 5.4: Überholdauer und Überholstrecke in Abhängigkeit vom Integer-Wert für die Überholung eines 0,6m langen dynamischen Hindernisses, der sich mit einer Geschwindigkeit von 0.6m/s bewegt

5.6 Einstellparameter für die Hinderniserkennung

In diesem Abschnitt werden die Einstellparameter für die Hinderniserkennung erläutert. Damit das Fahrzeug möglichst viele Hindernisse rechtzeitig erkennt und ihnen erfolgreich ausweicht, müssen folgende Parameter mit Sorgfalt ermittelt, berechnet und richtig eingestellt werden. Wird das Fahrzeug auf einer neuen Strecke betrieben, oder ändern sich die Lichtverhältnisse, müssen vor der Fahrt die Parameter für die Hinderniserkennung überprüft und eventuell neu eingestellt werden.

Die Parameter für die Hinderniserkennung können mit Hilfe der FAUST Web-Oberfläche (vgl. Abschnitt 2.3) eingestellt werden. Sie befinden sich unter dem Unterpunkt "OAandCR" (Obstacle Avoidance and Crossing Recognition). Mit einem Klick auf das Plus Zeichen öffnet sich eine Liste mit den in alphabetischer Reihenfolge sortierten Parametern für die Hindernis- und Kreuzungserkennung. In diesem Abschnitt wird nur auf die Einstellparameter der Hinderniserkennung eingegangen. Die Parameter für die Kreuzungserkennung befinden sich im Abschnitt 6.5. Die allgemeinen Parameter für die Hindernis -und Kreuzungserkennung werden im Abschnitt 2.3 erläutert.

Einstellparameter der Hinderniserkennung:

Mit `obstacleROILeftX` wird die linke X-Koordinate der Hindernis-ROI im Kamerabild eingestellt. Mit `obstacleROILeftY` wird die linke Y-Koordinate der Hindernis-ROI im Kamerabild eingestellt.

Mit `obstacleROIRightX` wird die rechte X-Koordinate der Hindernis-ROI im Kamerabild eingestellt. Mit `obstacleROIRightY` wird die rechte Y-Koordinate der Hindernis-ROI im Kamerabild eingestellt. Die vier Integer-Werte beschreiben die beiden Koordinaten die eine ROI für das Erkennen von Hindernissen im Kamerabild aufspannen (vgl. Abschnitt 5.2).

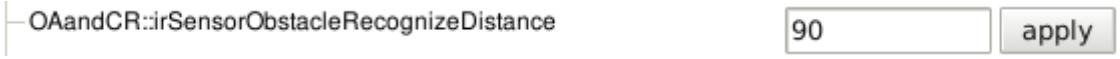
Mit `obstacleOffsetX` wird der Offset für die X-Koordinate der Hindernis-ROI gesetzt. Mit `obstacleOffsetY` wird der Offset für die Y-Koordinate der Hindernis-ROI gesetzt. Durch diese beiden Offsets wird die Hindernis-ROI an den Lenkwinkel gekoppelt und passt sich somit dynamisch der Fahrspur an (vgl. Abschnitt 5.2).

Mit `obstacleROINumberOfLines` wird die Anzahl der LOIs (Lines Of Interrest) angegeben, die in die Hindernis-ROI gelegt werden (vgl. Abschnitt 3.3 und Abschnitt 5.2).

Mit `percentLimitObstacle` wird der Prozentwert eingestellt zu dem eine Hindernis-ROI durch ein Hindernis bedeckt sein muss, damit ein Hindernis als erkannt gilt. Steigt der Rück-

gabewert der Methode `getPixelOverLimitInPercent()` der Hindernis-ROI über den Wert der in `percentLimitObstacle` eingegeben wurde so gilt ein Hindernis als erkannt (vgl. Abschnitt 5.2). Je niedriger der Wert desto kleiner ist die Fläche der Hindernis-ROI die bedeckt sein muss damit ein Hindernis als erkannt gilt. Dieser Wert muss immer mindestens um fünf Prozentpunkte höher sein als der maximale Rückgabewert der Methode `getPixel-OverLimitInPercent()` der Hindernis-ROI, während sich eine Haltelinie bzw. eine Start-linie vollständig innerhalb einer Hindernis-ROI befindet. Ist er das nicht so besteht die Gefahr, dass sowohl die Startlinie wie auch die Haltelinie an Kreuzungen als Hindernisse erkannt werden.

Mit `obstaclePOILimit` wird der Schwelwert für die Graustufe eines POI der sich auf einer LOI innerhalb einer Hindernis-ROI befindet eingestellt. Ist der Grauwert eines Pixels über dem Wert der in `obstaclePOILimit` übergeben wurde, handelt es sich bei dem Pixel um ein POI (Pixel Of Interest). Ist der Grauwert des Pixels unter dem übergebenen Wert so ist der untersuchte Pixel kein POI. Dieser Wert ist abhängig von der Ausleuchtung des Raumes in dem das Fahrzeug betrieben wird. Ändern sich die Lichtverhältnisse des Raumes muss dieser Wert angepasst werden um Hindernisse besser erkennen zu können. (vgl. Abschnitt 3.1 und Abschnitt 3.4).



Mit `irSensorObstacleRecognizeDistance` wird die Erkennungsdistanz d zwischen Fahr-zeug und Hindernis eingestellt, bei der ein Hindernis mit den Infrarotsensoren als er-kannt gilt. Sinkt der Rückgabewert eines der abgefragten IR-Sensoren unter diesen Wert, gilt das Hindernis als erkannt (vgl. Abschnitt 5.3).

Mit `slowForwardVelocity` wird die Geschwindigkeit eingestellt mit der das Fahrzeug die Fahrspur wechselt. Die Geschwindigkeit sollte so gewählt werden, dass Polaris keine Probleme hat die Fahrspur zu wechseln und sich auf die neue Fahrspur einzustellen. Ist der Wert zu hoch eingestellt, führt es zu Problemen beim Spurwechsel. Das Fahrzeug könnte ausbrechen und die Fahrbahn verlassen.

6 Kreuzungserkennung und Dynamisches Hindernis an Kreuzung

Bei dem Carolo-Cup 2013 in der Disziplin Rundkurs mit Hindernissen muss das Fahrzeug mehrere Kreuzungen passieren. Alle Kreuzungen sind rechtwinklig aufgebaut. Die Haltelinie an der Kreuzung ist 40mm breit. Das Fahrzeug muss vor der Haltelinie mindestens zwei Sekunden anhalten. Die Fahrzeugvorderkante muss sich dabei vor der Haltelinie befinden, darf aber nicht weiter als 15 cm von der Haltelinie entfernt sein. Anschließend kann die Kreuzung im Normalfall geradeaus passiert werden. Durch ein dynamisches Hindernis wird ein Verkehrsteilnehmer symbolisiert. Auf dieses Hindernis muss insbesondere an Kreuzungen in Abhängigkeit der Haltelinienanordnung (Vorfahrtsberechtigung) Rücksicht genommen werden. Liegt eine Vorfahrtssituation vor, muss vor der Haltelinie der Kreuzung gewartet werden, bis das dynamische Hindernis die Kreuzung vollständig passiert hat. In keinem Fall darf eine Kollision mit dem Hindernis auftreten [Carolo-Cup Regelwerk (2013)].

Zunächst werden die Anforderungen an die Kreuzungserkennung spezifiziert (vgl. Abschnitt 6.1). Anschließend wird das Erkennen der Haltelinie mit Hilfe der Kamera durch das Anlegen einer dynamischen ROI in das Kamerabild behandelt (vgl. Abschnitt 6.2). Der nächste Abschnitt beinhaltet die Differenzierung zwischen Haltelinie und Startlinie (vgl. Abschnitt 6.3). Das Erkennen dynamischer Hindernisse an Kreuzungen mittels Infrarotsensoren wird in Abschnitt 6.4 behandelt. Abschließend werden im Abschnitt 6.5 die Einstellparameter für die Kreuzungserkennung beschrieben und erläutert.

6.1 Spezifikation der Anforderung an die Kreuzungserkennung

Die Anforderungen an die Kreuzungserkennung entstanden durch im Vorfeld durchgeführte Tests und Beobachtungen und orientieren sich an dem Carolo-Cup Regelwerk [Carolo-Cup Regelwerk (2013)]. Nachfolgend werden die Anforderungen aufgelistet und spezifiziert:

- **Zuverlässigkeit:** Die Differenzierung zwischen einer Haltelinie an einer Kreuzung und der Startlinie sollte sowohl auf geraden Streckenabschnitten wie auch in Kurven zuverlässig funktionieren. Die Haltelinie sowie die Startlinie sollten bis zu einer Geschwindigkeit von 1,2m/s (dies entspricht einem Integer-Wert von 17 für velocity

vgl. Abschnitt 5.5) zuverlässig erkannt werden. Der Algorithmus für die Kreuzungserkennung sollte stabil sein gegenüber folgenden Lichtschwankungen:
    - Lichtreflektionen auf der Fahrbahn die durch ungleichmäßige Ausleuchtung der Fahrbahn entstehen können.
    - Schatten die von Personen oder Gegenständen auf die Fahrbahn und / oder die Haltelinie bzw. Startlinie geworfen werden.
- **Flexibilität:** Es sollte möglich sein die Entfernung in der die Haltelinie erkannt wird flexibel einzustellen, damit das Fahrzeug auch bei Geschwindigkeiten über 1m/s in der Lage ist vor der Haltelinie zum Stillstand zu kommen.
- **Laufzeit:** Der Algorithmus der Haltelinienerkennung sowie die Differenzierung zwischen Halte -und Startlinie sollte bei der Ausführung nicht länger als 40ms Rechenzeit benötigen, um in den FAUSTcore Systemtakt zu passen. Aufwendige double-Arithmetik sollte daher nach Möglichkeit vermieden werden.

6.2 Anlegen einer dynamischen ROI für die Haltelinie

Die Haltelinie an einer Kreuzung unterscheidet sich in Ihren Grauwerten im Kamerabild signifikant von den Grauwerten der Fahrbahn (vgl. Abschnitt 3.1). Um festzustellen ob sich eine Haltelinie in einer bestimmten Distanz vor dem Fahrzeug befindet, muss in das Kamerabild des Fahrzeugs für die rechte Fahrspur eine ROI mit einer bestimmten Anzahl an LOIs gelegt werden (vgl. Abschnitt 3.2 und Abschnitt 3.3). Über Parameter lässt sich die ROI für die Haltelinienerkennung im Kamerabild positionieren (vgl. Abschnitt 6.5). Zu beachten ist, dass die ROI möglichst mittig auf der rechten Fahrspur positioniert ist. Wird die ROI näher am Fahrzeug positioniert, so wird eine Haltelinie später erkannt. In diesem Fall ist das Fahrzeug näher an der Haltelinie dran bevor es diese erkennt. Wird die ROI im Kamerabild weiter vom Fahrzeug weg positionier, so wird eine Haltelinie früher erkannt. Da die ROI dazu dient die Haltelinie an einer Kreuzung zu erkennen wird sie nachfolgend als Haltelinie-ROI bezeichnet.

Um die Höhe der Haltelinien-ROI im Kamerabild zu bestimmen, muss als erstes die Strecke berechnet werden die das Fahrzeug zurücklegt bis ein neues Bild von der Fahrzeugkamera zu Verfügung steht. Diese Strecke $s[m]$ ist abhängig von der Fahrzeuggeschwindigkeit $v[m/s]$ und Anzahl der von der Fahrzeugkamera bereitgestellten Bilder Anz_{Bilder} pro Zeiteinheit $t[s]$ (vgl. Gleichung 6.1).

$$s = v * \frac{t}{Anz_{Bilder}} \tag{6.1}$$

Nachdem die Strecke s berechnet wurde, werden zwei Kamerabilder benötigt. Beim ersten Bild befindet sich das Fahrzeug in der gewünschten Erkennungsdistanz d zur Haltelinie und beim zweiten Bild befindet sich das Fahrzeug in der Distanz:

$$d' = d + s \tag{6.2}$$

von der Haltelinie entfernt. Die Höhe der Haltelinie-ROI wird anschließend berechnet indem die Anzahl der Pixel zwischen der Oberkante der Haltelinie aus dem zweiten Kamerabild und

der Unterkannte der Haltelinie aus dem ersten Kamerabild (damit wird gewehrleistet, das die Haltelinie mindestens einmal vollständig innerhalb der Haltelinie-ROI zu sehen ist) mit zwei multipliziert wird. Das Ergebnis ist die Höhe der Haltelinie-ROI in Pixeln innerhalb der, die Haltelinie in zwei auf einander folgenden Bildern zu sehen ist.

Beispiel zur Bestimmung der Höhe einer Haltelinien-ROI:

Das Fahrzeug bewegt sich mit einer Geschwindigkeit von 2m/s. Die Fahrzeugkamera liefert 20 Bilder pro Sekunde. Gesucht ist die Höhe h der Haltelinie-ROI.

gegeben: Fahrzeuggeschwindigkeit $v = 2m/s$;
Zeit $t = 1s$;
Anzahl Bilder in der Zeit t: $Anz_{Bilder} = 20$;
gewünschte Erkennungsdistanz $d = 0{,}5m$
gesucht: Höhe h der Haltelinie-ROI in Pixel

$$s = v * \frac{t}{Anz_{Bilder}} = 2m/s * \frac{1s}{20} = 0{,}1m$$

$$d' = d + s = 0{,}5m + 0{,}1m = 0{,}6m$$

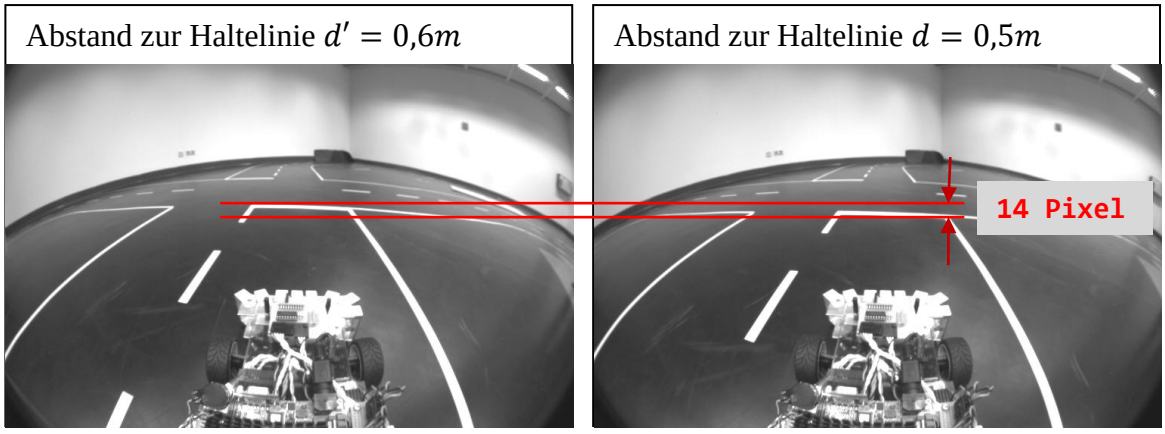


Abbildung 6.2.1: Bestimmung der Anzahl der Pixel zwischen der Oberkante der Haltelinie in der Distanz d' und der Unterkannte der Haltelinie in der Erkennungsdistanz d.

Die Höhe der Haltelinien-ROI beträgt:

$$h = 14\ Pixeln * 2 = 28\ Pixeln$$

Diese ermittelte Höhe h in Pixeln kann jetzt für alle Fahrzeuggeschwindigkeiten unter 2m/s und Bildraten über 20 Bilder pro Sekunde verwendet werden. Das würde jedoch Performance

kosten. Denn jeder zusätzliche Pixel auf einer LOI innerhalb der Haltelinie-ROI der mit dem Schwellwert verglichen werden muss (vgl. Kapitel 3) kostet Laufzeit. Um diese Laufzeit zu minimieren muss für jede neue Geschwindigkeit und Bilderrate die Höhe der Haltelinie-ROI neu berechnet werden.

Die Breite der Haltelinien-ROI sollte in der Erkennungsdistanz d mindestens die Hälfte der Fahrbahn bedecken. Dadurch kann eine Haltelinie auch dann erkannt werden, wenn Teile von ihr verschmutzt sind. Sie sollte jedoch nicht die gesamte Breite der Fahrbahn einnehmen, denn das Fahrzeug befindet sich beim Fahren nicht immer direkt in der Mitte der Fahrbahn. Somit würde der Mittel -und der Seitenstreifen von der Haltelinie-ROI erfasst werden, und den Rückgabewert der Methode `getPixelOverLimitInPercent()` der Haltelinie-ROI verfälschen. Damit das nicht passiert müssen Puffer von ca. ein Viertel der Fahrbahn rechts und links der Haltelinien-ROI in der Erkennungsdistanz d eingebaut werden. Um die Haltelinien-ROI im Kamerabild zu positionieren wird das Fahrzeug in der gewünschten Erkennungsdistanz d vor die Haltelinie gestellt. Die Haltelinien-ROI wird anschließend so in das Kamerabild positioniert, das die Haltelinie vollständig (in voller Höhe) in der oberen Hälfte der Haltelinie-ROI liegt (vgl. Abbildung 6.2.2).

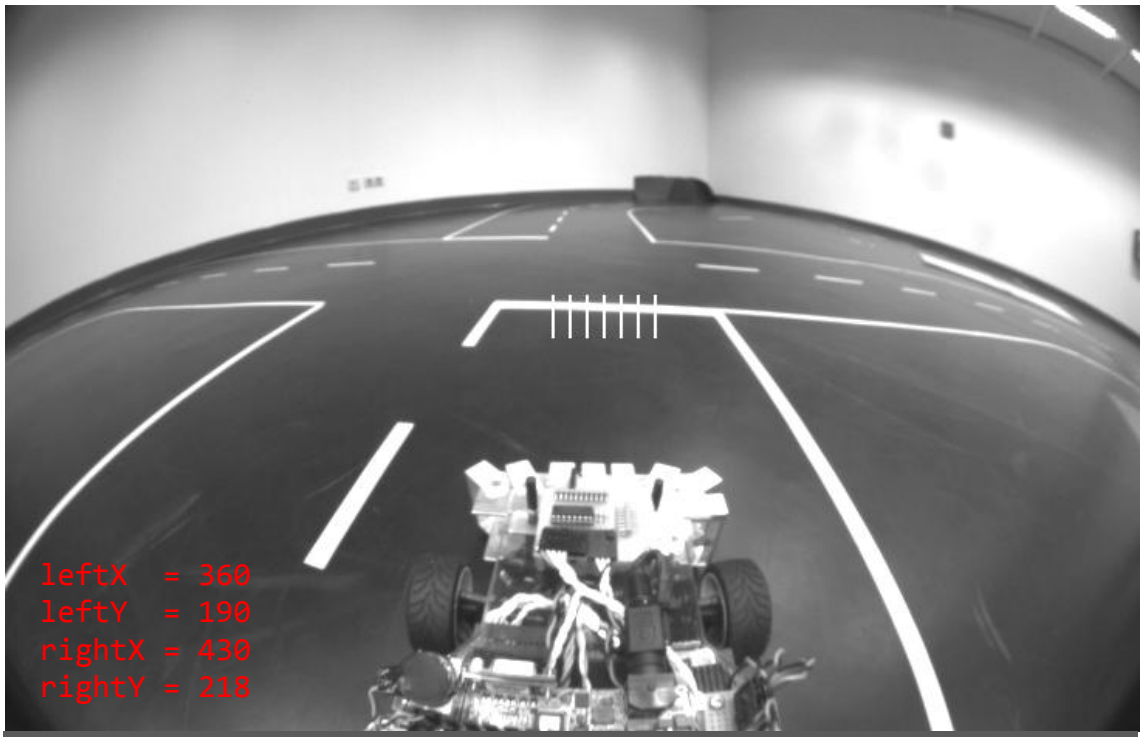


Abbildung 6.2.2: Positionierung der Haltelinie-ROI im Kamerabild der Fahrzeugkamera in der Erkennungsdistanz d = 0,5m.

Um zu überprüfen ob sich eine Haltelinie vor dem Fahrzeug befindet muss der Rückgabewert der Methode `getPixelOverLimitInPercent()` der Haltelinie-ROI überwacht werden. Steigt er über einen vorher festgelegten Wert, so gilt die Haltelinie als erkannt. Damit eine

Haltelinie erkannt wird sollte der Wert nicht höher sein als der maximale Rückgabewert der Methode. In Abbildung 6.2.2 ist der maximale Rückgabewert 21%, das ist der Wert bei dem sich die Haltelinie vollständig innerhalb der Haltelinie-ROI befindet. Um eine Haltelinie zu erkennen kann der Rückgabewert der Methode `getPixelOverLimitInPercent()` der Haltelinie-ROI auf einen Wert höher 15% abgefragt werden. Steigt der Rückgabewert über 15% befindet sich eine Haltelinie vor dem Fahrzeug und der Subautomat für die Kreuzungserkennung wird gestartet (vgl. Abschnitt 4.2).

Damit sich die Haltelinie-ROI dynamisch an die Fahrspur anpasst, wird diese genauso wie die Hindernis-ROI über zwei Faktoren an den Lenkwinkel geknüpft. Der eine Faktor ist der Offset für die X-Koordinate, der andere Faktor ist der Offset für die Y-Koordinate der Haltelinie-ROI. Die beiden Faktoren für den Offset werden nach demselben Prinzip berechnet wie die beiden Offset Faktoren für die Hindernis-ROI mit Hilfe des Kamerabildes und des Lenkwinkels (vgl. Abschnitt 5.2). Zu beachten ist das die Haltelinie-ROI im Kamerabild stets unter der Hindernis-ROI positioniert werden muss, damit ein vor dem Fahrzeug befindliches Hindernis über den Rückgabewert der Methode `getPixelOverLimitInPercent()` der Haltelinie-ROI nicht irrtümlich als Haltelinie bzw. Startlinie erkannt wird (vgl. Abbildung 6.2.3).

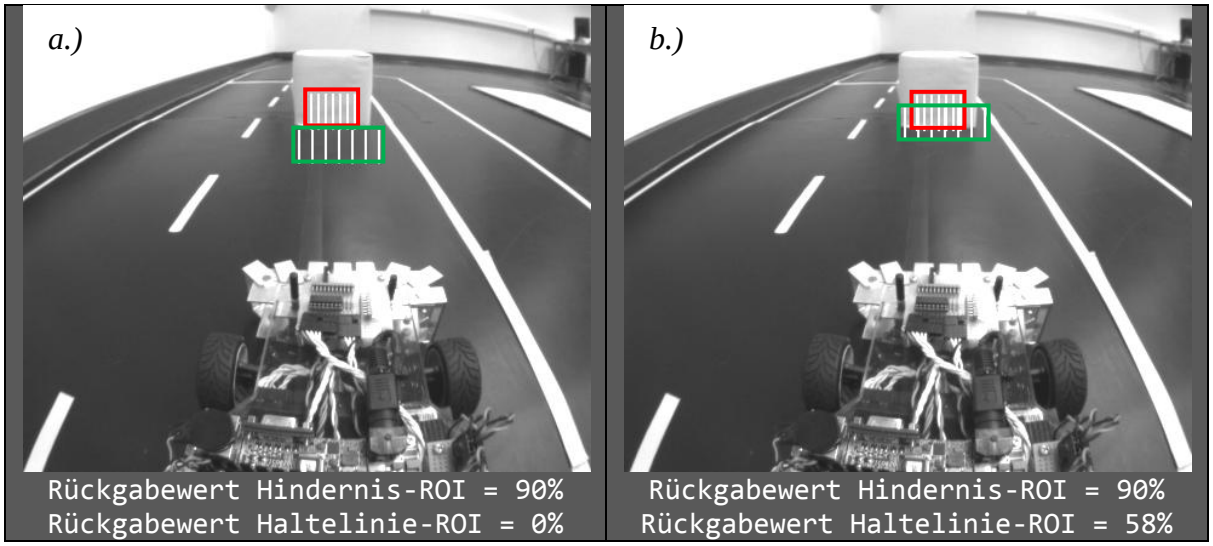


Abbildung 6.2.3: *a.) Richtige Positionierung der Haltelinie-ROI (grün) im Kamerabild*
b.) Falsche Positionierung der Haltelinie-ROI (grün) im Kamerabild

Damit die dynamische Haltelinie-ROI nicht nur auf gerader Fahrbahn, sondern auch in den Kurven unterhalb der Hindernis-ROI positioniert bleibt, muss der Y-Offset der Haltelinie-ROI (`lanesOffsetY`) gleich oder größer dem Y-Offset (`obstacleOffsetY`) der Hindernis-ROI sein.

Beispielparameter für eine Haltelinien-ROI:

Die folgenden Beispielparameter für eine Haltelinie-ROI können zusammen mit denen im Abschnitt 5.2 für die Hindernis-ROI berechneten Parametern verwendet werden. Diese Beispielparameter sind für eine Erkennungsdistanz von $d = 0{,}5m$ ausgelegt. Die Haltelinie

einer Kreuzung wird somit in einer Distanz von 0,5m vor dem Fahrzeug erkannt. Damit in den Kurven die Haltelinie-ROI genau unter der Hindernis-ROI positioniert bleibt wird für beide ROIs derselbe Offset für die Y-Koordinate verwendet. Der Offset für die X-Koordinate der Haltelinie-ROI fällt kleiner aus, weil diese breiter und näher am Fahrzeug positioniert ist.

```
leftX  = 360
leftY  = 190
rightX = 430
rightY = 218
lanesOffsetX = 2.0
lanesOffsetY = 2.2
rightLaneROINumberOfLines = 7 (Anzahl LOIs für die Haltelinie-ROI)
```

6.3 Differenzierung zwischen Halte -und Startlinie

Bei dem Carolo-Cup beginnt jedes der teilnehmenden Fahrzeuge den Rundkurs an einer Startlinie. Diese Startlinie ist, genauso wie die Haltelinie an einer Kreuzung, weiß und ihre Breite beträgt 40mm. Sie unterscheidet sich in ihren Grauwerten, wie auch die Haltelinie, signifikant von der Fahrbahn. Einer der Unterschiede zur Haltelinie ist das die Startlinie über beide Fahrspuren verläuft (vgl. Abbildung 6.3.1). Damit das Fahrzeug nach einer Runde die Startlinie nicht fälschlicherweise für eine Haltelinie hält und an dieser stoppt, muss es in der Lage sein die beiden zu unterscheiden.

Abbildung 6.3.1: Eine Startlinie verläuft im Gegensatz zu einer Haltelinie über beide Fahrspuren.

Um die Startlinie von der Haltelinie zu unterscheiden, wird nachdem das Fahrzeug eine Linie auf der rechten Fahrspur erkannt hat (vgl. Abschnitt 6.2) sogleich eine weitere ROI mit einer bestimmten Anzahl an LOIs für die linke Fahrspur in das Kamerabild gelegt. Diese zweite ROI wird nachfolgend als Startlinien-ROI bezeichnet (vgl. Abbildung 6.3.5). Mit der Startlinie-ROI wird überprüft ob es sich bei der erkannten Linie um eine Startlinie oder eine Haltelinie handelt. Ist der Rückgabewert der Methode `getPixelOverLimitInPercent()` der Startlinien-ROI über einem vorher festgelegten Prozentsatz, so handelt es sich um eine

Startlinie und das Fahrzeug kann weiterfahren. Ist der Rückgabewert unter dem vorher festgelegten Prozentsatz handelt es sich um eine Haltelinie und somit um eine Kreuzung an der das Fahrzeug laut Carolo-Cup Regelwerk [Carolo-Cup Regelwerk (2013)] für mindestens zwei Sekunden anhalten muss (vgl. Abbildung 6.3.2).

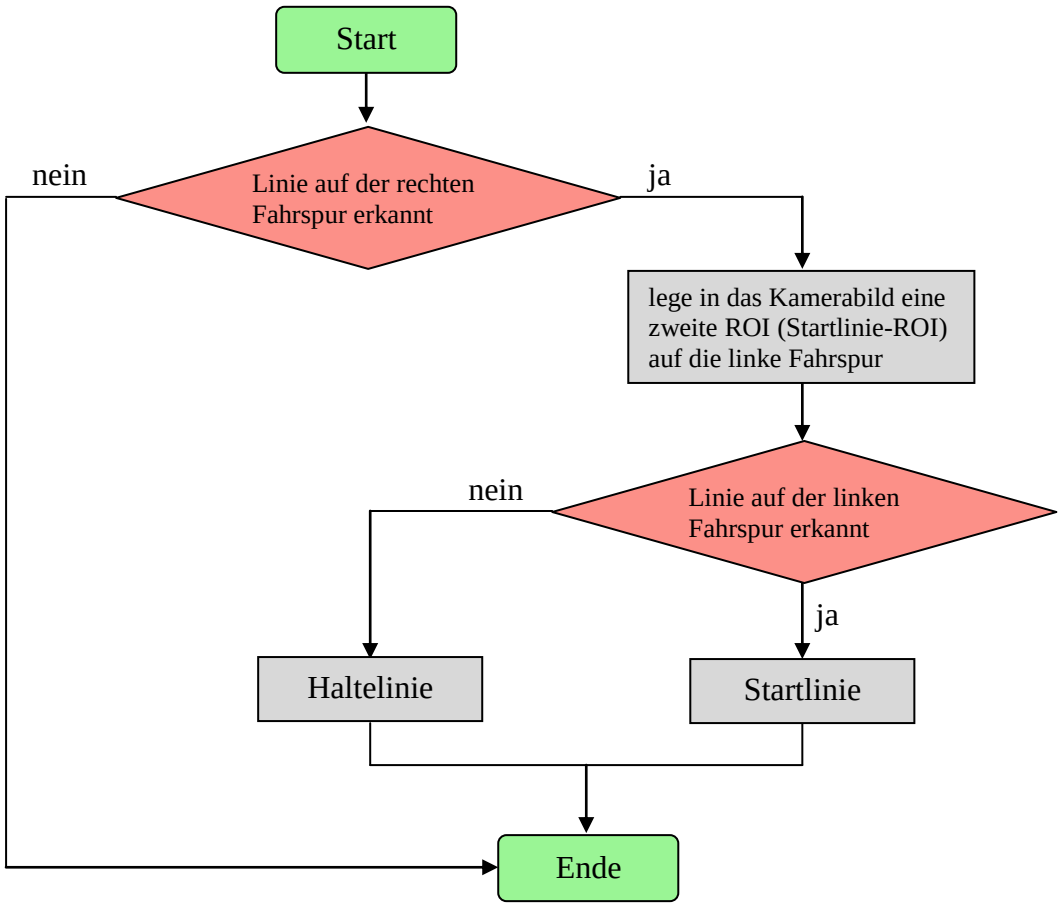


Abbildung 6.3.2: Flussdiagramm zur Unterscheidung zwischen Startlinie und Haltelinie

Positionierung der Startlinie-ROI im Kamerabild:

Die Startlinien-ROI wird über Parameter im Kamerabild positioniert (vgl. Abschnitt 6.5). Um Ressourcen zu sparen wird der Rückgabewert der Methode `getPixelOverLimitInPercent`() der Startlinien-ROI nur abgefragt wenn mithilfe der Haltelinie-ROI eine Linie auf der rechten Fahrspur in der Erkennungsdistanz d erkannt wurde. Die Höhe der Startlinie-ROI sollte der Höhe der Haltelinie-ROI entsprechen (vgl. Abschnitt 6.2). Die Breite der Startlinie-ROI sollte in der Erkennungsdistanz d die Hälfte der linken Fahrbahn bedecken. Die Positionierung der Startlinie-ROI erfolgt nach demselben Prinzip wie die Positionierung der Haltelinie-ROI (vgl. Abschnitt 6.2) Damit sich die Startlinie-ROI dynamisch an die Fahrspur anpasst, wird diese über dieselben zwei Faktoren wie die Haltelinie-ROI (`lanesOffsetX` und `lanesOffsetY`) an den Lenkwinkel geknüpft. (vgl. Abschnitt 6.2).

Startlinienerkennung innerhalb einer Kurve:

Über die Methode `getSlopeOfRecognizedLine()` der Haltelinie-ROI (vgl. Abschnitt 3.5) kann die Steigung der auf der rechten Fahrspur erkannten Linie berechnet werden um damit die Startlinie-ROI im Kamerabild über den Y-Offset in Abhängigkeit von der Steigung zu positionieren. Die berechnete Steigung kann dazu benutzt werden um eine Startlinie in einer Kurve zu erkennen (vgl. Abbildung 6.3.3). Mit der Gleichung 6.4 kann der zusätzliche Offset für die Y-Koordinate der Startlinie-ROI berechnet werden. Dabei ist c der Abstand zwischen der Startlinie-ROI und der Haltelinie-ROI in Pixeln und x der zusätzliche Y-Offset in Abhängigkeit von der Steigung für die Startlinie-ROI in Pixeln.

$m = Steigung$

$$tan(\alpha) = \frac{x}{c} = m \tag{6.3}$$

$$m = \tan(\alpha) \quad für\ \alpha \neq 90°$$

$$x = m * c \tag{6.4}$$

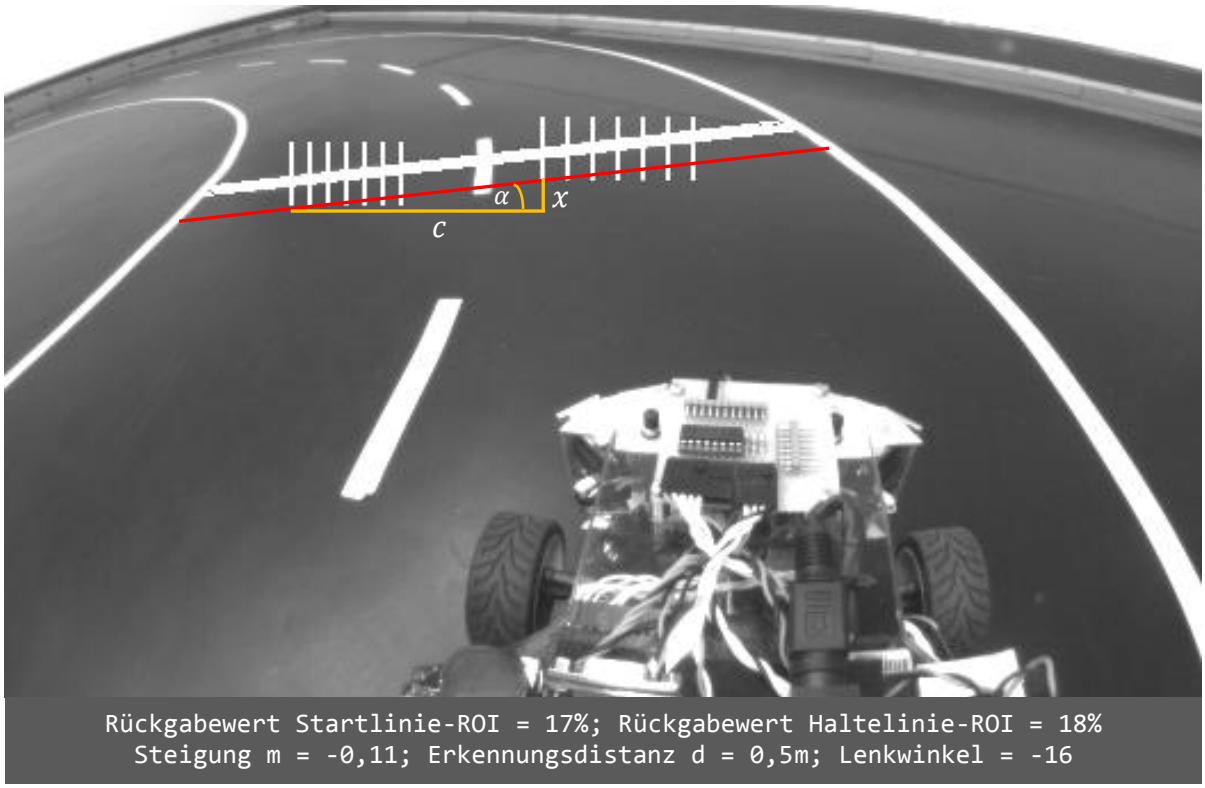


Abbildung 6.3.3: Berechnung des Y-Offsets für die Startlinie-ROI mit Hilfe der Steigung

Die Steigung der durch die Haltelinie-ROI erkannten Linie ist negativ (vgl. Abbildung 6.3.3). Das liegt daran das der Bilduhrsprung links oben liegt und das Koordinatensystem des Bildes

von dort aus mit den Koordinaten (y=0, x=0) beginnt (vgl. Einleitung von Kapitel 3). Befindet sich eine Startlinie in einer Rechtskurve ist ihre Steigung positiv (vgl. Abbildung 6.3.4). Der Y-Offset (`offsetY`) für die Startlinie-ROI wird anschließend folgendermaßen berechnet:

$$offsetY = \begin{cases} \dfrac{Lenkwinkel}{lanesOffsetY} - m * c, & \text{falls } Lenkwinkel \geq 0\ (Rechtskurve) \\ \dfrac{Lenkwinkel}{-lanesOffsetY} - m * c, & \text{falls } Lenkwinkel < 0\ (Linkskurve) \end{cases}$$

Beim Y-Offset (`offsetY`) muss drauf geachtet werden das bei einer Linkskurve der Lenkwinkel negativ ist. Deshalb muss der Wert von `lanesOffsetY` mit (-1) multipliziert werden, damit sich die Startlinie-ROI nach unten verschiebt. Die negative Steigung der auf der Rechten Fahrspur durch die Haltelinie-ROI erkannten Linie trägt dazu bei das das Produkt $m * c$ negativ wird und die Startlinie-ROI sich in Abhängigkeit der Steigung im Kamerabild weiter nach unten verschiebt (vgl. Abbildung 6.3.3).

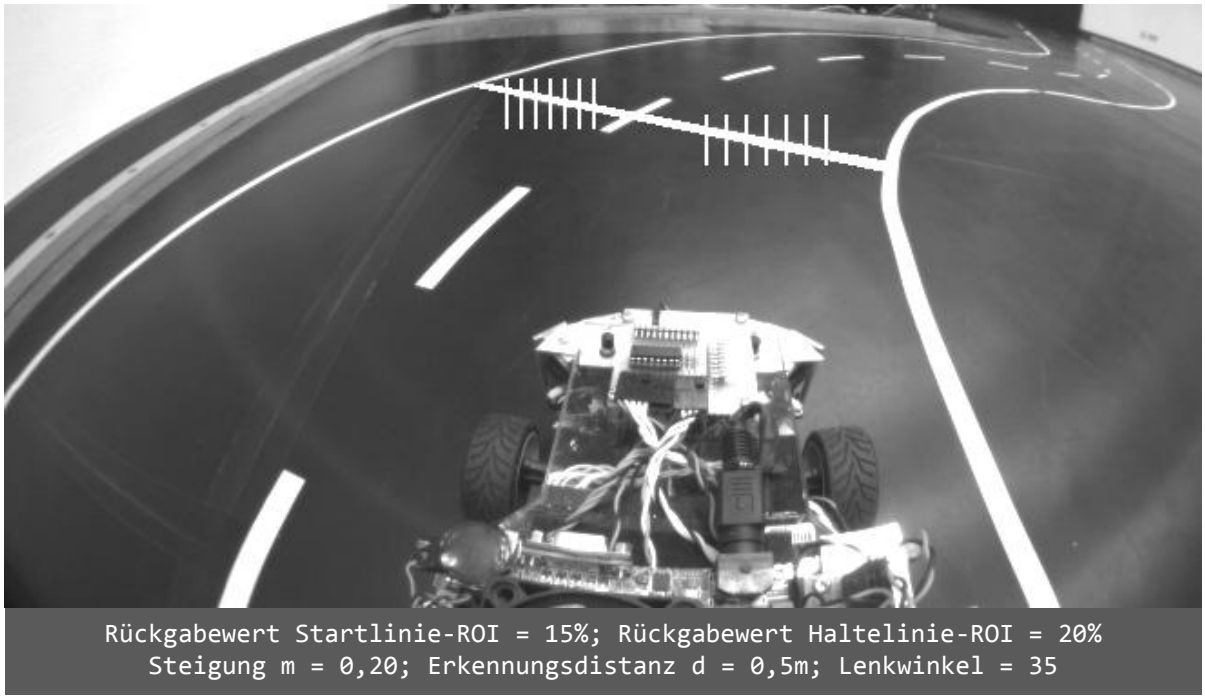


Abbildung 6.3.4: Steigung und Rückgabewerte der Startlinie-ROI und der Haltelinie-ROI in einer Rechtskurve

Damit der linke Teil einer Startlinie auf der linken Fahrspur erkannt wird sollte der vorher festgelegte Prozentsatz, ab dem eine Linie auf der linken Fahrspur als erkannt gilt, nicht höher sein als der maximale Rückgabewert der Methode `getPixelOverLimitInPercent()` der

Startlinien-ROI. In Abbildung 6.3.5 ist der maximale Rückgabewert 17%, das ist der Wert bei dem sich der linke Teil der Startlinie auf der linken Fahrspur vollständig innerhalb der Startlinie-ROI befindet. Um eine Startlinie zu erkennen kann der Rückgabewert der Methode `getPixelOverLimitInPercent()` der Startlinie-ROI auf einen Wert höher 10% abgefragt werden.

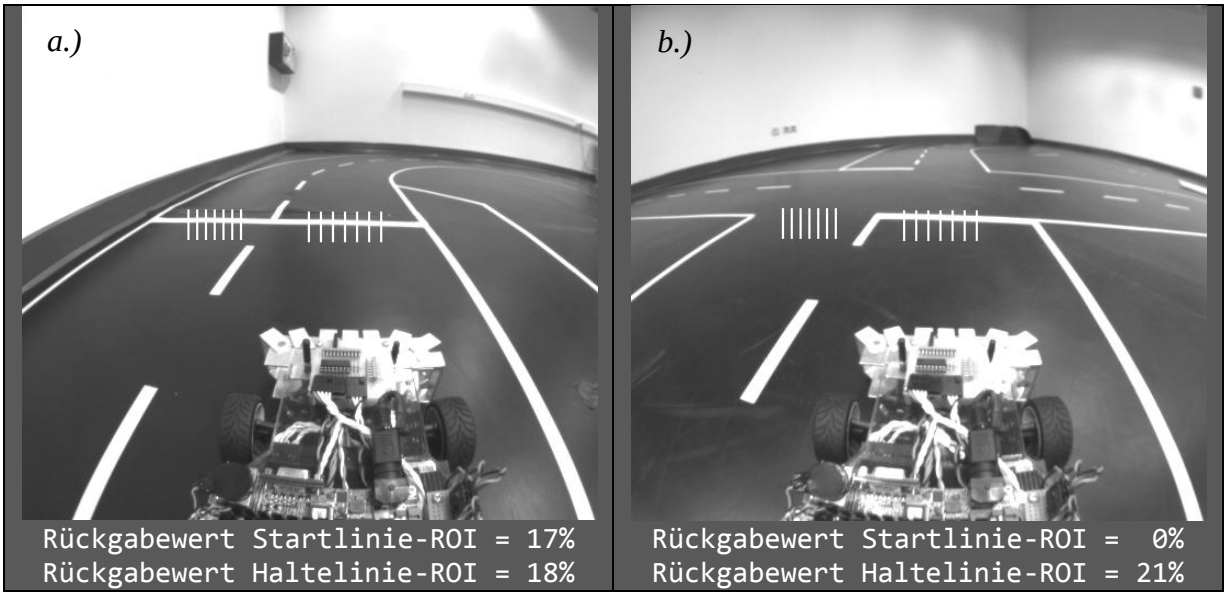


Abbildung 6.3.5: Rückgabewert der Methode `getPixelOverLimitInPercent()` *der Startlinien-ROI und der Haltelinie ROI in der Erkennungsdistanz d = 0,5m bei einer erkannten Startlinie (a.) und einer Kreuzung (b.)*

6.4 Erkennung dynamischer Hindernisse an Kreuzung mittels Infrarotsensoren

Wie schon in Abschnitt 5.5 beschrieben befinden sich beim Carolo-Cup dynamische Hindernisse auf der Fahrbahn. Auf diese dynamischen Hindernisse muss insbesondere an Kreuzungen in Abhängigkeit der Haltelinienanordnung (Vorfahrtsberechtigung) Rücksicht genommen werden (vgl. Abbildung 6.4.1). Liegt eine Vorfahrtssituation vor, muss vor der Haltelinie der Kreuzung gewartet werden, bis das dynamische Hindernis die Kreuzung vollständig passiert hat. In keinem Fall darf eine Kollision mit dem Hindernis auftreten [Carolo-Cup Regelwerk (2013)].

Steht das Fahrzeug an der Haltelinie einer Kreuzung, so muss es Vorfahrt gewähren. Ob sich ein dynamisches Hindernis auf der Kreuzung befindet wird anhand der an der Vorderseite des Fahrzeugs angebrachten Infrarotsensoren ermittelt. Es werden dafür dieselben fünf Infrarotsensoren verwendet die schon für die Hinderniserkennung auf der Fahrbahn zuständig sind (vgl. Abschnitt 5.3). Hat das Fahrzeug die Haltelinie einer Kreuzung erkannt, bremst es und bleibt vor der Haltelinie mindestens zwei Sekunden lang stehen. Nach Ablauf der Zeit werden alle fünf Infrarotsensoren abgefragt, ob sich in der für jeden Infrarotsensor separat eingestellten Erkennungsdistanz ein Hindernis befindet. Befindet sich kein Hindernis in der Erken-

nungsdistanz ist die Kreuzung frei und kann passiert werden. Schlägt eines der Infrarotsensoren aus, so befindet sich ein dynamisches Hindernis auf der Kreuzung. Daraufhin wartet das Fahrzeug so lange an der Haltelinie bis keines der vorderen Infrarotsensoren ausschlägt. Ist es der Fall kann die Kreuzung überquert werden.

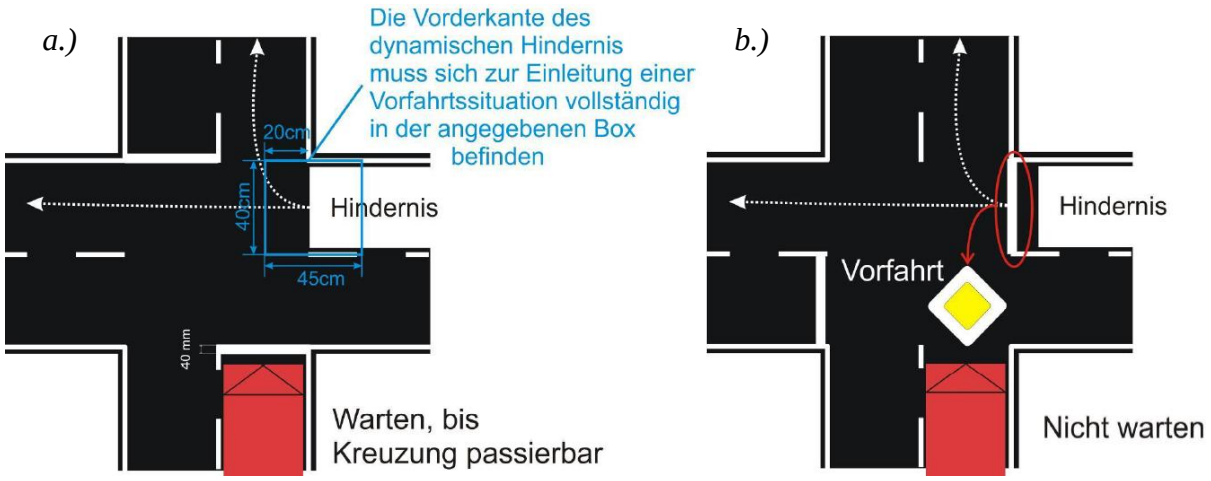


Abbildung 6.4.1: *a.) Fahrzeug muss dem dynamischen Hindernis Vorfahrt gewähren b.) Fahrzeug hat Vorfahrt [Carolo-Cup Regelwerk (2013)].*

Um die Kreuzung mit den Infrarotsensoren bestmöglich abzudecken muss die Erkennungsdistanz für jeden Infrarotsensor in Abhängigkeit von den Abmessungen der Kreuzung berechnet und eingestellt werden. Dabei darf die Erkennungsdistanz nicht über die Kreuzung hinausgehen, damit Gegenstände die in der Nähe der Kreuzung außerhalb der Fahrbahn positioniert sind nicht fälschlicherweise als dynamische Hindernisse erkannt werden (vgl. Abbildung 6.4.2). Der vordere Infrarotsensor `IrLongFrontCenter` wird dabei so eingestellt, dass er die gesamte Fahrbahnbreite plus die Haltelinienbreite und den Abstand des Fahrzeugs zur Haltelinie abdeckt. Der Winkel β zwischen den Infrarotsensoren `IrLongFrontLeftLeft` und `IrLongFrontLeft` sowie `IrLongFrontRight` und `IrLongFrontRightRight` wird nach der Gleichung 5.3 berechnet. der Winkel α wird nach der Gleichung 5.2 berechnet (vgl. Abschnitt 5.3). Die Erkennungsdistanz der fünf vorderen Infrarotsensoren wird folgendermaßen berechnet (vgl. Abbildung 6.4.2):

$IRll$ $= Erkennungsdistanz\ von$ `IrLongFrontLeftLeft`
IRl $= Erkennungsdistanz\ von$ `IrLongFrontLeft`
IRc $= Erkennungsdistanz\ von$ `IrLongFrontCenter`
IRr $= Erkennungsdistanz\ von$ `IrLongFrontRight`
$IRrr = Erkennungsdistanz\ von$ `IrLongFrontRightRight`
$b = Fahrbahnbreite$
$b_{Haltelinie} = Breite\ der\ Haltelinie$
$a_{Haltelinie} = Abstand\ des\ Fahrzeugs\ zur\ Haltelinie$

$$IRc = b + b_{Haltelinie} + a_{Haltelinie} \tag{6.5}$$

$$IRll = IRrr = \frac{IRc}{\cos(\alpha + \beta)} \tag{6.6}$$

$$IRl = IRr = \frac{IRc}{\cos(\alpha)} \tag{6.7}$$

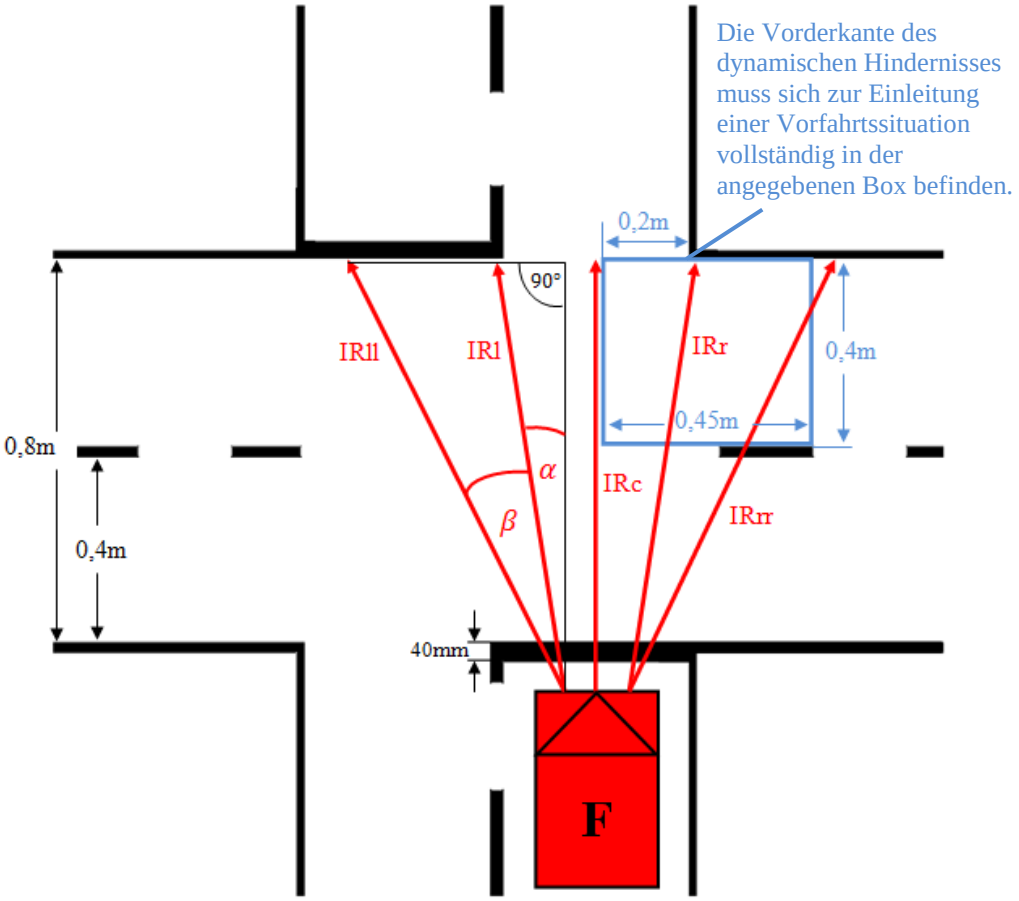


Abbildung 6.4.2: Erkennungsdistanzen IRll, IRl, IRc, IRr und IRrr der vorderen Infrarotsensoren für das dynamische Hindernis an einer Kreuzung

Fehlt die Haltelinie an einer Kreuzung, handelt es sich um eine Vorfahrtssituation des Fahrzeugs gegenüber dem dynamischen Hindernis (vgl. Abbildung 6.4.1). Somit wird auch keine Haltelinie vom Fahrzeug mit Hilfe der Haltelinie-ROI erkannt (vgl. Abschnitt 6.2). In diesem Fall überquert das Fahrzeug die Kreuzung als erstes. Das dynamische Hindernis muss warten bis das Fahrzeug die Kreuzung passiert hat.

6.5 Einstellparameter für die Kreuzungserkennung

In diesem Abschnitt werden die Einstellparameter für die Kreuzung -und Startlinienerkennung erläutert. Damit das Fahrzeug die Haltelinie an einer Kreuzung oder die Startlinie rechtzeitig erkennt und zwischen den beiden erfolgreich unterscheiden kann, müssen folgende Parameter mit Sorgfalt ermittelt, berechnet und richtig eingestellt werden. Wird das Fahrzeug auf einer neuen Strecke betrieben, oder ändern sich die Lichtverhältnisse, müssen vor der Fahrt die Parameter für die Kreuzung -und Startlinienerkennung überprüft und eventuell neu eingestellt werden.

Die Parameter für die Kreuzung -und Startlinienerkennung können mit Hilfe der FAUST Web-Oberfläche (vgl. Abschnitt 2.3) eingestellt werden. Sie befinden sich unter dem Unterpunkt "OAandCR" (Obstacle Avoidance and Crossing Recognition). Mit einem Klick auf das Plus Zeichen öffnet sich eine Liste mit den in alphabetischer Reihenfolge sortierten Parametern für die Hindernis- und Kreuzungserkennung. In diesem Abschnitt wird nur auf die Einstellparameter der Kreuzung -und Startlinienerkennung eingegangen. Die Parameter für die Hinderniserkennung befinden sich im Abschnitt 5.6. Die allgemeinen Parameter für die Hindernis -und Kreuzungserkennung werden im Abschnitt 2.3 erläutert.

Einstellparameter der Kreuzung -und Startlinienerkennung:

Mit `leftLaneROILeftX` wird die linke X-Koordinate der Startlinie-ROI im Kamerabild eingestellt. Mit `leftLaneROILeftY` wird die linke Y-Koordinate der Startlinie-ROI im Kamerabild eingestellt.

Mit `leftLaneROIRightX` wird die rechte X-Koordinate der Startlinie-ROI im Kamerabild eingestellt. Mit `leftLaneROIRightY` wird die rechte Y-Koordinate der Startlinie-ROI im Kamerabild eingestellt. Die vier Integer-Werte beschreiben die beiden Koordinaten die eine ROI für das Erkennen der Startlinie im Kamerabild aufspannen (vgl. Abschnitt 6.3).

Mit `leftLaneROINumberOfLines` wird die Anzahl der LOIs (Lines Of Interrest) angegeben, die in die Startlinie-ROI gelegt werden (vgl. Abschnitt 3.3 und Abschnitt 6.3).

Mit `percentLimitLeftLane` wird der Prozentwert eingestellt zu dem eine Startlinie-ROI durch den linken Teil einer Startlinie bedeckt sein muss, damit eine Startlinie als erkannt gilt. Steigt der Rückgabewert der Methode `getPixelOverLimitInPercent()` der Startlinie-ROI über den Wert der in `percentLimitLeftLane` eingegeben wurde so gilt eine Startlinie als erkannt. (vgl. Abschnitt 6.3).

Mit `rightLaneROILeftX` wird die linke X-Koordinate der Haltelinie-ROI im Kamerabild eingestellt. Mit `rightLaneROILeftY` wird die linke Y-Koordinate der Haltelinie-ROI im Kamerabild eingestellt.

Mit `rightLaneROIRightX` wird die rechte X-Koordinate der Haltelinie-ROI im Kamerabild eingestellt. Mit `rightLaneROIRightY` wird die rechte Y-Koordinate der Haltelinie-ROI im Kamerabild eingestellt. Die vier Integer-Werte beschreiben die beiden Koordinaten die eine ROI für das Erkennen der Haltelinie im Kamerabild aufspannen (vgl. Abschnitt 6.2).

Mit `rightLaneROINumberOfLines` wird die Anzahl der LOIs (Lines Of Interrest) angegeben, die in die Haltelinie-ROI gelegt werden (vgl. Abschnitt 3.3 und Abschnitt 6.2).

Mit `percentLimitRightLane` wird der Prozentwert eingestellt zu dem eine Haltelinie-ROI durch die Haltelinie bedeckt sein muss, damit eine Haltelinie als erkannt gilt. Steigt der Rückgabewert der Methode `getPixelOverLimitInPercent()` der Haltelinie-ROI über den Wert der in `percentLimitRightLane` eingegeben wurde so gilt eine Haltelinie als erkannt. (vgl. Abschnitt 6.2).

Mit `lanesOffsetX` wird der Offset für die X-Koordinate der Startlinie-ROI und der Haltelinie-ROI gesetzt. Mit `lanesOffsetY` wird der Offset für die Y-Koordinate der Startlinie-ROI und der Haltelinie-ROI gesetzt. Durch diese beiden Offsets werden die Startlinie-ROI sowie die Haltelinie-ROI an den Lenkwinkel gekoppelt und passen sich somit dynamisch der Fahrspur an (vgl. Abschnitt 6.2 und Abschnitt 6.3).

Mit `lanesPOILimit` wird der Schwelwert für die Graustufe eines POI eingestellt, der sich auf einer LOI innerhalb einer Startlinie-ROI oder einer Haltelinie-ROI befindet. Ist der Grauwert eines Pixels über dem Wert der in `lanesPOILimit` übergeben wurde, handelt es sich bei dem Pixel um ein POI (Pixel Of Interest). Ist der Grauwert des Pixels unter dem übergebenen Wert so ist der untersuchte Pixel kein POI. Dieser Wert ist abhängig von der Ausleuchtung des Raumes in dem das Fahrzeug betrieben wird. Ändern sich die Lichtverhältnisse des Raumes muss dieser Wert angepasst werden um die Startlinie und die Haltelinie besser erkennen zu können. (vgl. Abschnitt 3.1 und Abschnitt 3.4).

Mit `IRc` wird die Erkennungsdistanz des Infrarotsensors `IrLongFrontCenter` für ein dynamisches Hindernis an einer Vorfahrtskreuzung eingestellt.
Mit `IRl` wird die Erkennungsdistanz des Infrarotsensors `IrLongFrontLeft` für ein dynamisches Hindernis an einer Vorfahrtskreuzung eingestellt.
Mit `IRll` wird die Erkennungsdistanz des Infrarotsensors `IrLongFrontLeftLeft` für ein dynamisches Hindernis an einer Vorfahrtskreuzung eingestellt.
Mit `IRr` wird die Erkennungsdistanz des Infrarotsensors `IrLongFrontRight` für ein dynamisches Hindernis an einer Vorfahrtskreuzung eingestellt.
Mit `IRrr` wird die Erkennungsdistanz des Infrarotsensors `IrLongFrontRightRight` für ein dynamisches Hindernis an einer Vorfahrtskreuzung eingestellt (vgl. Abschnitt 6.4).

Mit `waitingTimeOnCrossInSeconds` Wird die Zeit in Sekunden eingestellt, die das Fahrzeug an einer erkannten Haltelinie halten soll.

7 Test und Evaluation

Zur Evaluation der in Kapitel drei bis sechs entwickelten Algorithmen wird das im Rahmen des FAUST-Projektes (Fahrerassistenz- und Autonome Systeme) der Hochschule für Angewandte Wissenschaften Hamburg entwickelte autonome Modellfahrzeug "Saphir" verwendet (vgl. Abbildung 7.0.1). Das Fahrzeug basiert auf einem Tamiya TT-01 [FAUST-Webseite]. Als Karosserie wird ein Modell des Audi RS8 im Maßstab 1:10 mit den folgenden Abmessungen (L x B x H) 440 x 200 x 120mm verwendet. Die Hauptrecheneinheit des Saphirs besteht aus einem PICO820. Als Festplatte dient eine 8GB Compact-Flash-Karte. Die Leistungsdaten des PICO820 sind in der Tabelle 7.1 dargestellt. Die eingesetzte Kamera zeichnet Bilder in einem breiten Format auf und ist mit einem Weitwinkelobjektiv ausgestattet (vgl. Tabelle 7.2). Das Weitwinkelobjektiv ermöglicht eine Erkennung der Fahrspurmarkierungen in engen Kurven [Matthaei 2007]. Am Fahrzeug sind elf Infrarotsensoren montiert, die für die Hindernis-erkennung (vgl. Abschnitt 5.3) und den Einparkvorgang verwendet werden. Des Weiteren besitzt der Saphir ein Hall Sensor für die Distanzmessung. Dabei sind 21 Neodym-Magnete auf der Vorderachse montiert, so dass eine Auflösung von einem Zentimeter erreicht wird [FAUST-Webseite]. Die Steuerung der Aktorik und die Auswertung der Sensorik übernimmt der Mikrokontroller MCB9B500 [Guide MCB9B500].

Abbildung 7.0.1: Autonomes FAUST-Modelfahrzeug "Saphir"

PICO820	
Prozessor	Intel® Atom™ Z510 1.1GHz
Arbeitsspeicher	2GB DDR2 533 MHz
Festplatte	8GB Compact-Flash-Karte
Betriebssystem	Debian 6.0 mit 2.6 Echtzeit-Kernel

Tabelle 7.1: Leistungsdaten des PICO820 [Datenblatt PICO820]

IDS uEye UI-1226LE	
Bildauflösung	752 x 480 Pixel
Maximale Bildrate	87 fps
Belichtungszeit	80μs - 5,5s
Bildsensor	1/3" CMOS monochrom
Datenschnittstelle	USB 2.0
Abmessungen	B = 36mm, H = 36mm, T = 20mm
Gewicht	12g
Objektiv	Weitwinkelobjektiv: f = 2,27mm, 1/3"

Tabelle 7.2: Leistungsdaten Fahrzeugkamera UI-1226LE [Datenblatt UI-1226LE]

Als Teststrecke für das autonome Modelfahrzeug Saphir dient ein Rundkurs. Der Rundkurs besteht aus zwei Geraden, mehreren Kurven und einer Kreuzung an der die Vorfahrtsregel beachtet werden müssen (vgl. Abbildung 7.0.2). Die Breite der Fahrbahn beträgt 0,8m. Eine Fahrspur ist 0,4m Breit. Die Breite der Startlinie und der Haltelinie beträgt 0,04m. Somit sind die Abmessungen des Rundkurses identisch mit denen im Carolo-Cup Regelwerk 2013 [Carolo-Cup Regelwerk (2013)].

Auf dem Saphir befindet sich die Fahrspurerkennung Polaris, die im Rahmen der Masterarbeit von E. Jenning [Jenning 2008 Kapitel 3] entwickelt wurde. Polaris hält das Fahrzeug mittig auf der Fahrspur, indem es für eine der beiden Fahrspuren zwei Polynome zur Abbildung der Fahrspurmarkierungen ermittelt. In der Regel wird die rechte Fahrspur überwacht, so dass die Mittelstreifenmarkierung und die rechte Außenmarkierung abgebildet werden. Über einen Parameter der Kontrolldatenstruktur `PolarisControlData` kann Polaris dazu aktiviert werden, die linke oder rechte Fahrspur zu überwachen. Bei allen nachfolgenden Systemtests wird auf dem Saphir die Polaris Version: `PolarisV2` verwendet.

Bei den nachfolgenden Tests handelt es sich um Systemtests bei denen das gesamte System gegenüber den Anforderungen (vgl. Abschnitt 5.1 sowie Abschnitt 6.1) auf einer Teststrecke (vgl. Abbildung 7.0.2) getestet wird. Zunächst werden die durchgeführten Tests für die Hinderniserkennung und den Ausweichvorgang sowie die erzielten Ergebnisse vorgestellt (vgl. Abschnitt 7.1). Im Abschnitt 7.2 werden Tests für die Erkennung von Haltelinien an Kreuzungen, der Startlinie und der Beachtung von Vorfahrt an einer Kreuzung zusammen mit den erreichten Ergebnissen vorgestellt.

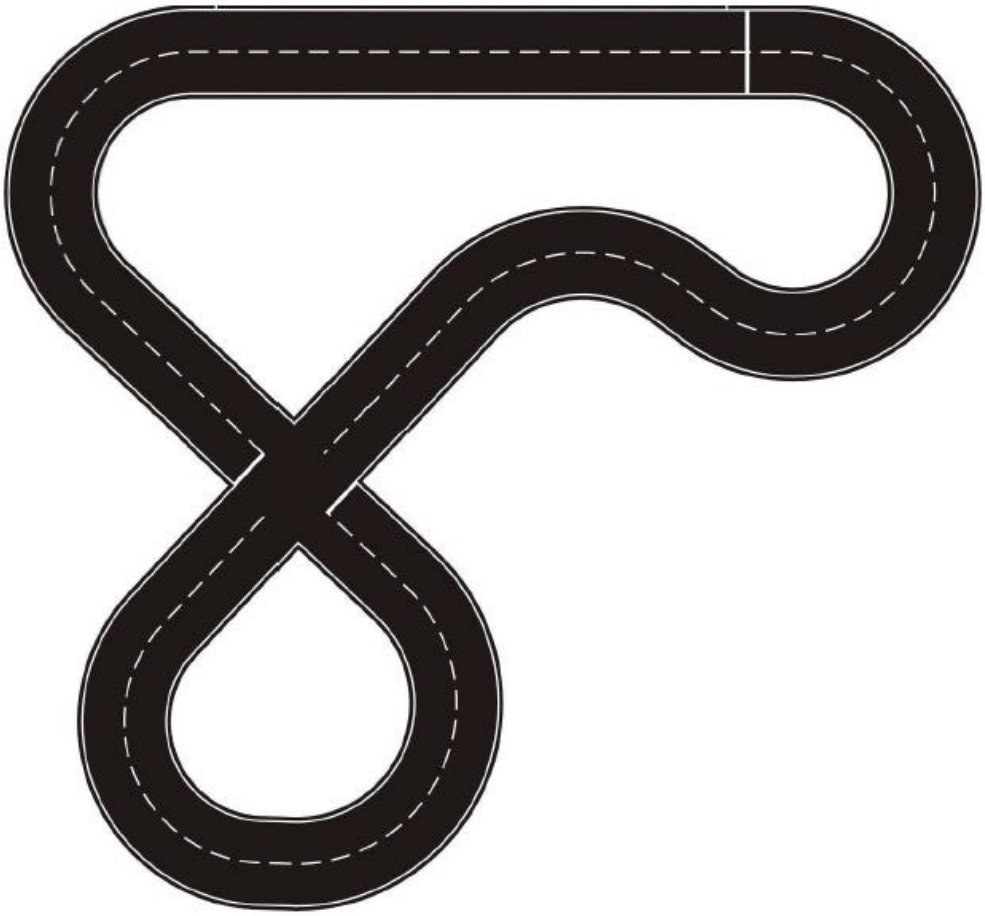


Abbildung 7.0.2: Teststrecke als Rundkurs für das Modelfahrzeug "Saphir"

7.1 Hinderniserkennung und Ausweichvorgang

Bei dem ersten Systemtest wird die Hinderniserkennung mit den Infrarotsensoren getestet. Dabei werden nur die Infrarotsensoren für die Hinderniserkennung genutzt. Ziel ist es festzustellen wie genau die Infrarotsensoren sind und wie schnell auf Hindernisse reagiert werden kann. Dazu befinden sich fünf Hindernisse auf der Teststrecke H1 bis H5.

Versuchsaufbau und Versuchsdurchführung:

Der Versuch wird mit dem Fahrzeug Saphir ohne Karosserie durchgeführt. Für die Fahrspurerkennung wird die Polaris Version `PolarisV2` verwendet. Die Position der Hindernisse auf der Teststrecke kann der Abbildung 7.1.1 a.) entnommen werden. Die Abmessungen der Hindernisse stehen in der Tabelle 7.3. Die Anordnung der vorderen Infrarotsensoren (Winkel α und β) ist in der Abbildung 7.1.1 b.) dargestellt (vgl. Abschnitt 5.3). Die Erkennungsdistanz

für die vorderen Infrarotsensoren `irSensorObstacleRecognizeDistance` ist auf 0,9m eingestellt (vgl. Abschnitt5.3 und Abschnitt 5.6).

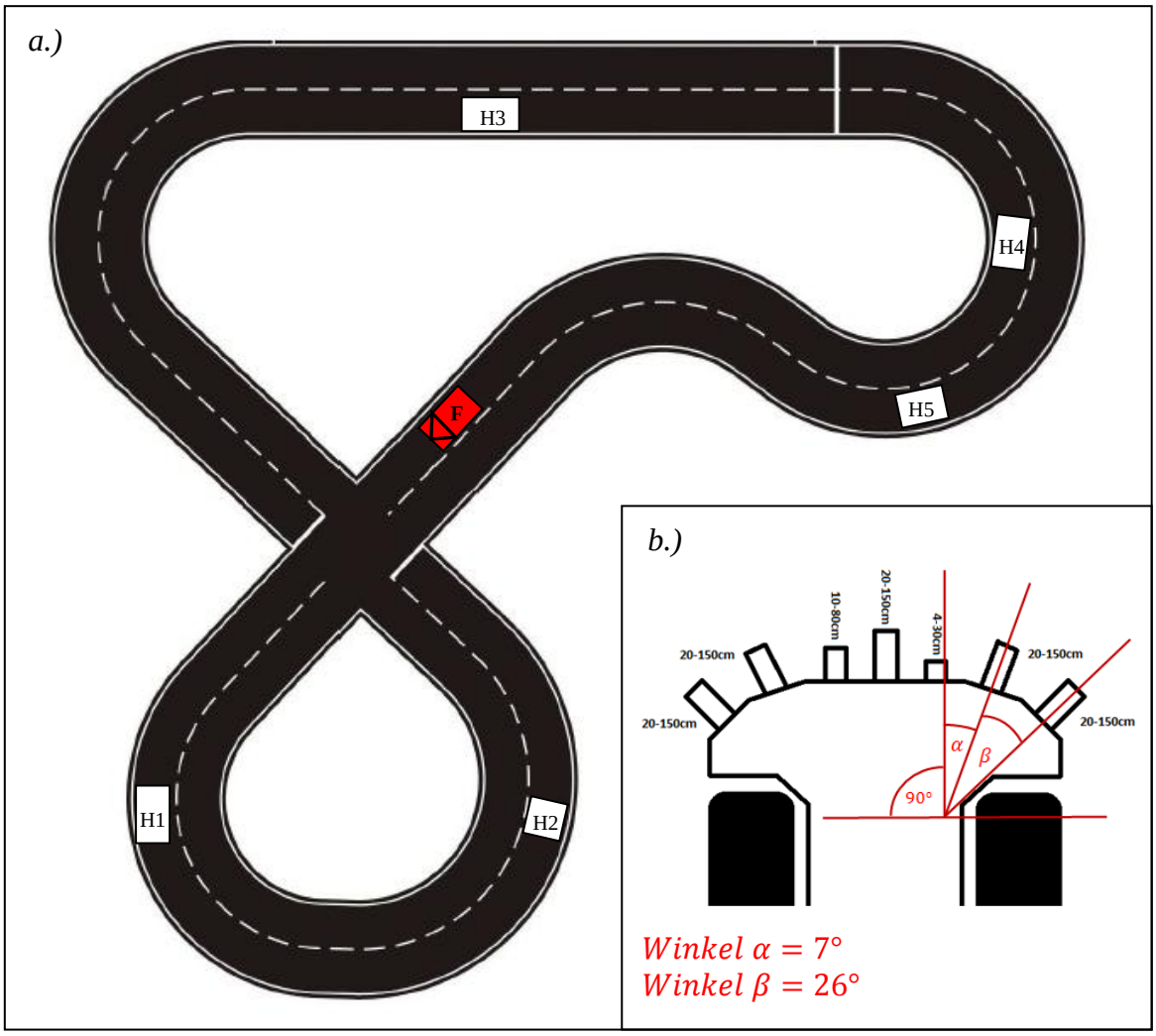


Abbildung 7.1.1: *a.) Standorte der Hindernisse (weiß) H1 bis H5 auf der Teststrecke. Startposition des Fahrzeugs Saphir (rot).*
b.) Winkeleinstellung (α und β) der vorderen Infrarotsensoren.

Hindernis	Länge[m]	Breite[m]	Höhe[m]
H1	0,48	0,28	0,22
H2	0,10	0,10	0,10
H3	0,18	0,28	0,24
H4	0,32	0,22	0,25
H5	0,40	0,23	0,12

Tabelle 7.3: Abmessungen der Hindernisse H1 - H5

Das Fahrzeug absolviert den Rundkurs drei Mal mit einer Geschwindigkeit von ca. 1m/s (entspricht einem Integer-Wert von 13 vgl. Abschnitt 5.5). Beobachtet wird ob die Hindernisse H1 bis H4 vom Fahrzeug erkannt und sämtliche Ausweichvorgänge kollisionsfrei ausgeführt werden. Das Hindernis H5 steht am Ende einer Rechtskurve auf der Gegenfahrbahn und sollte von den Infrarotsensoren ignoriert werden.

Beobachtung:

Runde	Hindernis	Erkannt		Kollision	
		ja	nein	ja	nein
1	H1	x			x
	H2	x			x
	H3	x			x
	H4	x			x
	H5		x		x
2	H1	x			x
	H2	x		x	
	H3	x			x
	H4	x			x
	H5	x		x	
3	H1	x			x
	H2	x		x	
	H3	x			x
	H4	x			x
	H5		x		x

Tabelle 7.4: Hinderniserkennung mit den Infrarotsensoren

In der ersten Runde wurden die Hindernisse H1 bis H4, die sich auf derselben Fahrbahn wie das Fahrzeug befanden (vgl. Abbildung 7.1.1), vom Fahrzeug erkannt. Der Ausweichvorgang verlief bei allen vier Hindernissen (H1 bis H4) ohne Kollision. Das Hindernis H5, welches auf der Gegenfahrbahn positioniert war, wurde von den Infrarotsensoren nicht erkannt.

In der zweiten Runde wurde das kleinste Hindernis H2 in der Linkskurve zu spät durch die Infrarotsensoren erkannt und das Fahrzeug kollidierte beim Ausweichvorgang mit der linken Ecke des Hindernisses. In der Rechtskurve wurde das Hindernis H5, welches auf der Gegenfahrbahn stand, von den Infrarotsensoren als Hindernis auf der eigenen Fahrbahn interpretiert und das Fahrzeug kollidierte mit dem Hindernis indem es den Ausweichvorgand einleitete. Die Hindernisse H1, H3 und H4 wurden vom Fahrzeug erkannt und der Ausweichvorgang wurde ohne Kollisionen vollzogen.

In der dritten Runde wurde das kleinste Hindernis H2 in der Linkskurve erneut zu spät durch die Infrarotsensoren erkannt und das Fahrzeug kollidierte daraufhin beim Ausweichvorgang mit dem Hindernis. Die Hindernisse H1, H3 und H4 wurden vom Fahrzeug erkannt und der

Ausweichvorgang wurde ohne Kollisionen vollzogen. Das Hindernis H5 wurde von den Infrarotsensoren nicht erkannt (vgl. Tabelle 7.4).

Weiterhin ist beim Test aufgefallen, dass obwohl die Erkennungsdistanz der Infrarotsensoren auf 0,9m eingestellt war das Fahrzeug erst ca. 0,2m bis 0,1m vor den Hindernissen die Fahrbahn wechselte. Dabei schwankten die Werte der vorderen analogen Infrarotsensoren während sich das Fahrzeug bewegte um bis zu 0,1m. Beim Hindernis H4 in der Rechtskurve hat das Fahrzeug nach ca. 0,8m hinter dem Hindernis wieder auf die rechte Fahrspur zurückgewechselt. Bei den Hindernissen H1 bis H3 wechselte das Fahrzeug nach ca. 0,2m bis 0,4m hinter dem Hindernis wieder auf die rechte Fahrspur.

Ergebnis:

Infrarotsensoren sind für die Hinderniserkennung auf autonomen Fahrzeugen nur bedingt geeignet. Die an der Vorderseite montierten analogen Infrarotsensoren GP2Y0A02YK (vgl. Abschnitt 5.3) sind langsam und unpräzise für die Hinderniserkennung während der Fahrt. Obwohl die Erkennungsdistanz der vorderen Infrarotsensoren auf 0,9m eingestellt war, wechselte das Fahrzeug erst ca. 0,2m vor dem Hindernis die Fahrspur. Durch die Schwankungen in der Erkennungsdistanz von bis zu 0,1m werden Hindernisse die auf der Gegenfahrbahn in Kurven positioniert sind durch die Infrarotsensoren erkannt und als Hindernis auf der eigenen Fahrspur interpretiert. Durch den daraufhin eingeleiteten Ausweichvorgang kommt es zu Kollisionen mit diesen Hindernissen.

Die Anzahl der vorderen Infrarotsensoren ist mit zwei Sensoren (vgl. Abschnitt 5.3) zu gering um eine gute Abdeckung in den Kurven zu gewährleisten (vgl. Abbildung 7.1.2). Das Hindernis H2 mit den kleinsten nach dem Carolo-Cup Regelwerk zulässigen Abmessungen wurde bei der zweiten und dritten Runde vom Fahrzeug zu spät erkannt und es kam zu einer Kollision beim Ausweichvorgang. Die zwei größeren ebenfalls in den Kurven positionierten Hindernisse H1 und H4 wurden in allen drei Runden zuverlässig und rechtzeitig erkannt.

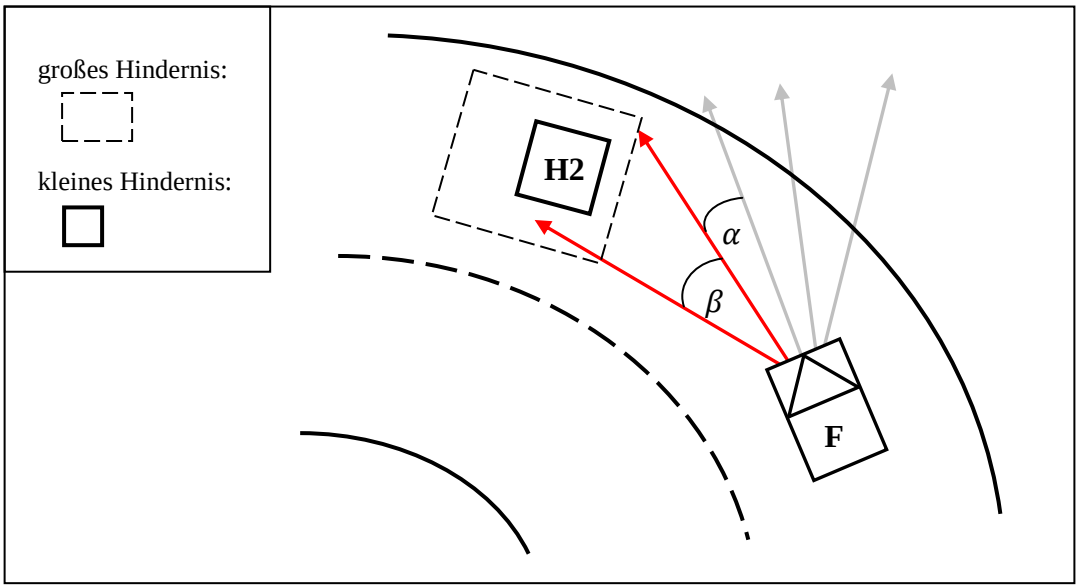


Abbildung 7.1.2: Abdeckung einer Kurve durch die Infrarotsensoren

Bei dem zweiten Systemtest wird die Hinderniserkennung mit der Fahrzeugkamera und einer in das Kamerabild gelegten Hindernis-ROI getestet. Dabei wird nur die Fahrzeugkamera für die Hinderniserkennung genutzt (vgl. Abschnitt 5.2). Ziel ist es festzustellen wie präzise Hindernisse erkannt werden und wie schnell auf diese reagiert werden kann. Dazu befinden sich fünf Hindernisse auf der Teststrecke H1 bis H5.

Versuchsaufbau und Versuchsdurchführung:

Der Versuch wird mit dem Fahrzeug Saphir ohne Karosserie durchgeführt. Für die Fahrspurerkennung wird die Polaris Version `PolarisV2` verwendet. Die Position der Hindernisse auf der Teststrecke ist dieselbe wie bei der Hinderniserkennung mit Infrarotsensoren und kann der Abbildung 7.1.1 a.) entnommen werden. Die Abmessungen der Hindernisse stehen in der Tabelle 7.3. Die Erkennungsdistanz zwischen Fahrzeug und Hindernis beträgt ca. 0,9m.

Koordinaten und Einstellungen der Hindernis-ROI:

```
leftX  = 360
leftY  = 140
rightX = 400
rightY = 170

numberOfLines = 5 (Anzahl LOIs)
limit = 160 (Schwellwert)

obstacleOffsetX = 2.7
obstacleOffsetY = 2.2
```

Das Fahrzeug absolviert den Rundkurs drei Mal mit einer Geschwindigkeit von ca. 1m/s das entspricht einem Integer-Wert von 13 vgl. Abschnitt 5.5. Beobachtet wird ob die Hindernisse H1 bis H4 vom Fahrzeug erkannt und sämtliche Ausweichvorgänge kollisionsfrei ausgeführt werden. Das Hindernis H5 steht am Ende einer Rechtskurve auf der Gegenfahrbahn und sollte vom Fahrzeug ignoriert werden.

Beobachtung:

In allen drei Runden wurden die Hindernisse H1 bis H4 zuverlässig vom Fahrzeug erkannt und sämtliche Ausweichvorgänge kollisionsfrei ausgeführt. Auch das kleinste Hindernis H2 welches in einer Linkskurve positioniert war, wurde alle drei Runden zuverlässig von der im Kamerabild positionierten Hindernis-ROI erkannt. Die Hindernis-ROI befand sich während der gesamten Fahrt stets mittig auf der rechten Fahrspur. Dadurch wurde das Hindernis H5 welches in einer Rechtskurve auf der Gegenfahrbahn positioniert war erfolgreich ignoriert. Wurde ein Hindernis erkannt, wechselte das Fahrzeug ca. 0,4m bis 0,3m vor den Hindernissen die Fahrbahn. Beim Hindernis H4 in der Rechtskurve hat das Fahrzeug nach ca. 0,8m hinter dem Hindernis wieder auf die rechte Fahrspur zurückgewechselt. Bei den Hindernissen H1 bis H3 wechselte das Fahrzeug nach ca. 0,2m bis 0,4m hinter dem Hindernis wieder auf die rechte Fahrspur (vgl. Tabelle 7.5).

Runde	Hindernis	Erkannt		Kollision	
		ja	nein	ja	nein
1	H1	x			x
	H2	x			x
	H3	x			x
	H4	x			x
	H5		x		x
2	H1	x			x
	H2	x			x
	H3	x			x
	H4	x			x
	H5		x		x
3	H1	x			x
	H2	x			x
	H3	x			x
	H4	x			x
	H5		x		x

Tabelle 7.5: Hinderniserkennung mit der Fahrzeugkamera

Ergebnis:

Mit der Fahrzeugkamera wurden die auf der rechten Fahrspur positionierten Hindernisse H1 bis H4 zuverlässig erkannt. Das Fahrzeug hat schnell auf die Hindernisse reagiert und ihnen rechtzeitig ausgewichen. Bei keinem Ausweichvorgang kam es zur Kollision. Somit ist die Methode mit der in das Kamerabild gelegten dynamischen Hindernis-ROI (vgl. Abschnitt 5.2) gut für die Hinderniserkennung geeignet.

7.2 Kreuzungserkennung

Es werden zwei Systemtests zur Kreuzungserkennung durchgeführt. Im ersten Test wird die Erkennung der Haltelinie, sowie die Differenzierung zwischen Halte -und Startlinie mit Hilfe der in das Kamerabild positionierten Haltelinie-ROI und Startlinie-ROI getestet (vgl. Abschnitt 6.2 und Abschnitt 6.3). Der zweite Systemtest beschäftigt sich mit der Erkennung eines dynamischen Hindernisses mittels Infrarotsensoren bei einer Vorfahrtssituation an einer Kreuzung (vgl. Abschnitt 6.4).

Versuchsaufbau und Versuchsdurchführung:

Der Versuch wird mit dem Fahrzeug Saphir ohne Karosserie durchgeführt. Für die Fahrspurerkennung wird die Polaris Version `PolarisV2` verwendet. Als Teststrecke wird ein Rundkurs ohne Hindernisse verwendet (vgl. Abbildung 7.0.2). Das Fahrzeug absolviert jeweils drei Runden mit folgenden Geschwindigkeiten: 0,98m/s, 1,13m/s und 1,42m/s. Beobachtet wird ob das Fahrzeug die Haltelinie an einer Kreuzung, sowie die Startlinie zuverlässig erkennt.

Bei einer Haltelinie soll das Fahrzeug für mindestens zwei Sekunden anhalten. Nach Ablauf der Zeit soll das Fahrzeug die Kreuzung passieren und seine Fahrt fortsetzen. Bei einer Startlinie soll das Fahrzeug weiterfahren ohne anzuhalten. Es soll eine Ausgabe im Terminal erscheinen ob es sich bei der erkannten Linie um eine Haltelinie oder eine Startlinie handelt. Die Erkennungsdistanz für die Haltelinie sowie die Startlinie wird durch die Koordinaten der Halte -und Startlinie-ROI im Kamerabild auf ca. 0,5m eingestellt (vgl. Abschnitt 6.2).

Koordinaten und Einstellungen der Startlinie-ROI:

```
leftX  = 230
leftY  = 190
rightX = 280
rightY = 118

numberOfLines = 5 (Anzahl LOIs)
limit = 155 (Schwellwert)

obstacleOffsetX = 2.0
obstacleOffsetY = 2.2
```

Koordinaten und Einstellungen der Haltelinie-ROI:

```
leftX  = 340
leftY  = 190
rightX = 410
rightY = 218

numberOfLines = 5 (Anzahl LOIs)
limit = 155 (Schwellwert)

obstacleOffsetX = 2.0
obstacleOffsetY = 2.2
```

Beobachtung:

Die Haltelinie an der Kreuzung wurde bei jeder getesteten Geschwindigkeit erkannt. Das Fahrzeug blieb immer an der erkannten Haltelinie stehen und setzte nach einer Wartezeit von zwei Sekunden seine Fahrt fort. Die Startlinie wurde bei einer Geschwindigkeit von 0,98m/s und 1,13m/s in jeder Runde erkannt (Ausgabe im Terminal) und ohne anzuhalten vom Fahrzeug passiert. Bei einer Geschwindigkeit von 1,42m/s wurde die Startlinie in der zweiten und dritten Runde erkannt. In der ersten Runde wurde die Startlinie nicht erkannt (keine Ausgabe im Terminal) wurde vom Fahrzeug dennoch ohne zu halten passiert. (vgl. Tabelle 7.6).

Geschwindigkeit $v_F[m/s]$	Runde	Haltelinie erkannt		Startlinie erkannt	
		ja	nein	ja	Nein
0,98	1	x		x	
	2	x		x	
	3	x		x	
1,13	1	x		x	
	2	x		x	
	3	x		x	
1,42	1	x			x
	2	x		x	
	3	x		x	

Tabelle 7.6: Haltelinie -und Startlinienerkennung mit der Fahrzeugkamera

Ergebnis:

Mit den im Versuchsaufbau vorgenommenen Einstellungen für die Halte -und Startlinie-ROI werden die Startlinie und die Haltelinie an Kreuzungen bis zu einer Fahrzeuggeschwindigkeit von 1,13m/s zuverlässig erkannt. Somit ist das im Abschnitt 6.2 und Abschnitt 6.3 beschriebene Verfahren zur Erkennung der Haltelinie und der Differenzierung zwischen der Haltelinie und Startlinie geeignet.

Der zweite Systemtest beschäftigt sich mit der Erkennung eines dynamischen Hindernisses mittels Infrarotsensoren bei einer Vorfahrtssituation an einer Kreuzung (vgl. Abschnitt 6.4).

Versuchsaufbau und Versuchsdurchführung:

Der Versuch wird mit dem Fahrzeug Saphir ohne Karosserie durchgeführt. Für die Fahrspurerkennung wird die Polaris Version `PolarisV2` verwendet. Als Teststrecke dient der in Abbildung 7.0.2. abgebildeter Rundkurs. Für das dynamische Hindernis wird ein weißen Karton mit folgenden Abmessungen: Länge = 0,32m, Breite = 0,22m und Höhe = 0,25m benutzt. Dieses dynamische Hindernis wird im ersten Durchlauf mit der Vorderkante an die linke Seite der blauen Box positioniert (vgl. Abbildung 7.2.1 *a.)*). Im zweiten Durchlauf befindet sich nur die Vorderkante des dynamischen Hindernisses innerhalb der blauen Box (vgl. Abbildung 7.2.1 *b.)*). In beiden Durchläufen muss das Fahrzeug an der Haltelinie der Kreuzung halten und das dynamische Hindernis mit Hilfe der vorderen Infrarotsensoren erkennen (vgl. Ab-schnitt 6.4). Hat das Fahrzeug das Hindernis erkannt bleibt es so lange stehen bis das dyna-mische Hindernis die Kreuzung vollständig passiert. Ist die Kreuzung frei muss das Fahrzeug diese überqueren und seine Fahrt fortsetzen.

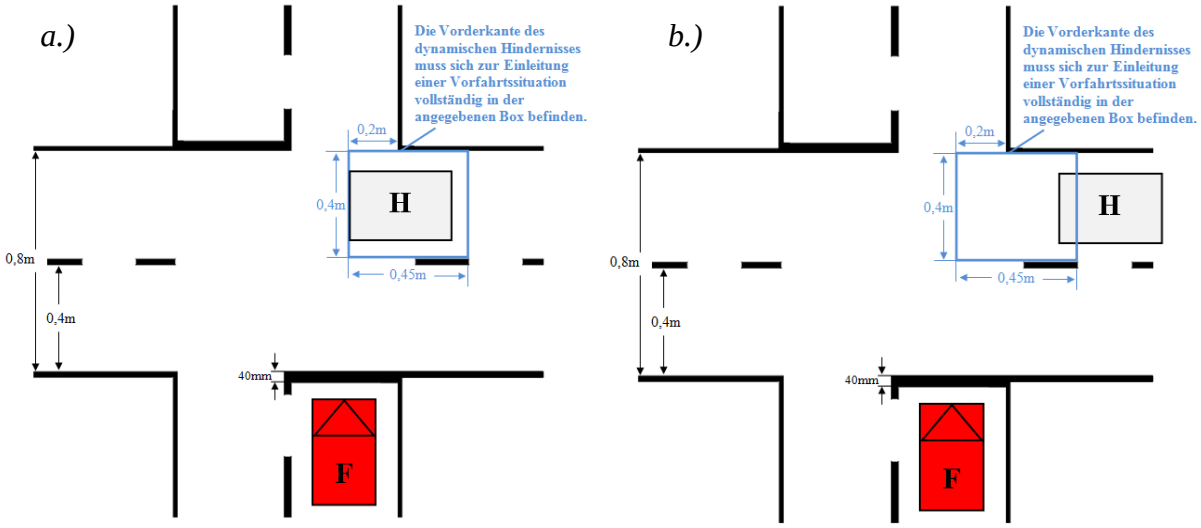


Abbildung 7.2.1: *a.) Position des dynamischen Hindernisses beim ersten Durchlauf*
b.) Position des dynamischen Hindernisses beim zweiten Durchlauf

Es wurden folgende Einstellungen für die Erkennungsdistanzen der jeweiligen vorderen Infrarotsensoren benutzt: (vgl. Abschnitt 6.4)

$IRll = 107cm$ ($Erkennungsdistanz\ von$ `IrLongFrontLeftLeft`)
$IRl = 92cm$ ($Erkennungsdistanz\ von$ `IrLongFrontLeft`)
$IRc = 90cm$ ($Erkennungsdistanz\ von$ `IrLongFrontCenter`)
$IRr = 92cm$ ($Erkennungsdistanz\ von$ `IrLongFrontRight`)
$IRrr = 107cm$ ($Erkennungsdistanz\ von$ `IrLongFrontRightRight`)

Einstellung der Winkeln zwischen Infrarotsensoren: $\beta = 26°$ und $\alpha = 7°$ (vgl. Abschnitt 5.3).

Beobachtung:

In beiden Durchläufen wurde das dynamische Hindernis vom Fahrzeug erkannt. Das Fahrzeug hielt an der Haltelinie und setzte seine Fahrt erst fort, nachdem das dynamische Hindernis die Kreuzung vollständig passiert hat.

Ergebnis:

Die Vorfahrtssituation wurde vom Fahrzeug in beiden Durchläufen richtig gedeutet und das dynamische Hindernis wurde in beiden Fällen mit den Infrarotsensoren erkannt. Somit ist das im Abschnitt 6.4 beschriebene Verfahren zur Erkennung von Vorfahrtssituationen und dynamischen Hindernissen an Kreuzungen geeignet.

8 Fazit

In dieser Arbeit wurden Methoden entwickelt zur Hindernis- und Kreuzungserkennung auf dem autonomen Modelfahrzeug "Saphir". Die Ziele dabei waren das Erkennen und das berührungslose überholen von dynamischen und statischen Hindernissen auf einem Rundkurs, sowie die Erkennung von Kreuzungen und derer Vorfahrtssituation. Als Sensoren dienten dabei eine monochrome 0,36 Megapixel Kamera von IDS und Infrarot-Entfernungsmesser der Sharp Electronics GmbH.

Die Anforderungen an die Hinderniserkennung (vgl. Abschnitt 5.1) sowie die Anforderungen an die Kreuzungserkennung (vgl. Abschnitt 6.1) sind alle erfüllt (vgl. Kapitel 7). Das Erkennen der Hindernisse, der Startlinie und der Haltelinie mit der Fahrzeugkamera und einem zu den Lichtverhältnissen im Raum geeignetem Schwellwert (vgl. Kapitel 3, 5 und 6) funktioniert (vgl. Kapitel 7). Tests haben gezeigt, das Hindernisse und Haltelinie selbst bei Lichtreflektionen auf der Fahrbahn (die durch ungleichmäßige Ausleuchtung der Fahrbahn entstehen) oder Schatten (die von Personen oder Gegenständen auf die Hindernisse, die Startlinie oder die Haltelinie geworfen werden) zuverlässig erkannt werden.

Die Sharp Infrarotsensoren der GP2-Serie sind zur Hinderniserkennung auf einem autonomen Fahrzeug weniger gut geeignet (vgl. Kapitel 7). Einige der Sensoren sind bei Tests öfters ausgefallen oder mussten neu kalibriert werden. Die Entfernungsangaben während sich das Fahrzeug auf ein Hindernis zubewegt sind ungenau. Es treten Messfehler von bis zu 0,14m auf. Die Sensoren sind in einem festen Winkel am Fahrzeug montiert, sind somit nicht in der Lage sich einer Kurve dynamisch anzupassen. Dies führt zu einer schlechten Abdeckung in den Kurven. Dadurch werden besonders kleine Hindernisse in Kurven zu spät oder gar nicht erkannt. Große Hindernisse die in einer Kurve auf der Gegenfahrbahn Nahe am Mittelstreifen stehen werden von den Infrarotsensoren als Hindernisse auf der eigenen Fahrbahn interpretiert, wodurch es bei einem eingeleiteten Ausweichmanöver zur Kollision mit dem Hindernis kommt (vgl. Kapitel 7).

Die Erkennung dynamischer Hindernisse mit Hilfe der vorderen Infrarotsensoren (GP2-Serie von Sharp) an einer Kreuzung (vgl. Abschnitt 6.4) hat bei allen Tests zuverlässig funktioniert (vgl. Abschnitt 7.2). Aufgrund der erhöhten Ausfallwahrscheinlichkeit der Infrarotsensoren empfiehlt es sich zur Unterstützung und als Ausfallsicherheit eine weitere Prüfung mit der Fahrzeugkamera zu unternehmen. Dies kann realisiert werden, indem eine oder mehrere ROIs (vgl. Abschnitt 3.2) in das Kamerabild gelegt und auf ein dynamisches Hindernis an der Kreuzung abgefragt werden.

Der Fahrspurwechsel besonders in den Kurven muss weiter verbessert werden. Das Fahrzeug sollte nachdem es ein Hindernis überholt hat schneller auf die eigene Fahrspur wechseln und dabei weniger Strecke zurücklegen. Im Moment legt das Fahrzeug bis zu 0,8m an Strecke zurück, bis es in einer Kurve nach einem Überholvorgang wieder auf die eigene Fahrbahn wechselt (vgl. Abschnitt 7.1).

9 Abbildungsverzeichnis

10 Tabellenverzeichnis

11 Literaturverzeichnis

[Caroline 2007] CarOLO-Team der: *autonomen Fahr-zeug "Caroline"*, Technischen Universität Braunschweig, August 2007, URL: https://www.tu-braunschweig.de/presse/medien/presseinformationen?year=2007&pinr=137 (2013-02-21)

[Carolo-Cup Regelwerk (2013)] *„Carolo-Cup" Regelwerk 2013* von Juni 2012 URL: http://www.carolo-cup.de/fileadmin/user_upload/2013/regelwerk/Regelwerk.pdf (2012-10-24)

[DARPA] *The DARPA Urban Challenge 2007* URL: http://archive.darpa.mil/grandchallenge/index.asp (2013-02-21)

[Datenblatt GP2Y0A02YK] SHARP Long Distance Measuring Sensor GP2Y0A02YK URL: http://www.produktinfo.conrad.com/datenblaetter/175000-199999/185364-da-01-en-Distanzsensor_GP2Y0A02YK.pdf (2013-02-21)

[Datenblatt GP2Y0D340K] SHARP Distance Measuring Sensor Unit Digital output (400mm) type URL: http://sharp-world.com/products/device/lineup/data/pdf/datasheet/gp2y0d340k_e.pdf (2013-02-21)

[Datenblatt PICO820] The PICO820 is a Pico-ITX board with support for Intel® ATOM™ processor Z500 Series URL: http://www.axiomtek.com/Download/Download/eBOX530-820-FL/PICO820%20User%27s%20Manual.pdf (2013-01-15)

[Datenblatt UI-1226LE] IDS Imaging Development Systems GmbH *Datenblatt UI-1226LE* URL: http://www.ids-imaging.de/frontend/products.php?cam_id=12&sc=2 (2013-01-15)

[FAUST-Webseite] Webseite des FAUST-Projektes (FAUST Fahrerassistenz- und Autonome Systeme) der Hochschule für Angewandte Wissenschaften Hamburg URL: http://faust.informatik.haw-hamburg.de/index.php?section=fahrzeuge&sub=1 (2013-03-21)

[GP2Y0A02YK Abbildung] SHARP Long Distance Measuring Sensor GP2Y0A02YK URL: http://www.conrad.de/ce/de/product/185364/Distanz-Sensor-GP-2-Y0A-02-YK-Sharp-GP-2-Y0A-02-YK-Messbereiche-Erfassungsbereich-20-150-cm-5-VDC (2013-02-21)

[Guide MCB9B500] KEIL Tools by ARM *Mikrokontroller MCB9B500 User's Guide* URL: http://www.keil.com/support/man/docs/mcb9b500/mcb9b500_to_cpu.htm (2013-01-15)

[Hackenkamp 2001] Prof. Dr. Hackenkamp: *Histogramm und Grauwerttransformationen* April 2001 URL: http://www.fbmn.fh-darmstadt.de/home/heckenkamp/bv1/vorlesung/histo/histo_5b.html (2013-01-24)

[Heinrich & Biehl] Stefan Heinrich & Markus Biehl: *ChameleonVision Kapitel 4 Eindimensionale Segmentierung* URL: http://kik.informatik.fh-dortmund.de/abschlussarbeiten/chameleon/ (2013-01-25)

[Jenning 2008] Eike Jenning: *Systemidentifikation eines autonomen Fahrzeugs mit einer robusten, kamerabasierten Fahrspurerkennung in Echtzeit*. Hochschule für Angewandte Wissenschaften Hamburg, Dezember 2008

[JENOPTIK] JENOPTIK: *Region Of Interest* URL: http://www.jenoptik.com/02B80D5086542B17C125784E0033792B (2013-02-03)

[Jiang 2007] Prof. Xiaoyi Jiang: *Bildverarbeitung Script Kapitel 11 Binärisierung und Verarbeitung von Binärbildern* Januar 2007 URL: http://cvpr.uni-muenster.de/teaching/ws06/bildverarbeitungWS06/script/BV11-2.pdf (2013-01-30)

[Matthaei 2007] Richard Matthaei: *Entwurf einer kamerabasierten Fahrspurerkennung für ein autonom fahrendes Modellauto*. Studienarbeit, Institut für Regelungstechnik, Technische Universität Braunschweig. 2007

[Meisel RV02 2012] Prof. Dr. -Ing. Andreas Meisel: *Robot Vision Script RV02* März 2012, Seite 1 - 11

[Meisel RV03 2012] Prof. Dr. -Ing. Andreas Meisel: *Robot Vision Script RV03* März 2012, Seite 23 ff.

[Niblack 1986] W. Niblack: *An Introduction to Digital Image Processing*, Prentice-Hall, 1986.

[Nikolov 2011] Ivo Nikolov: *Maschinelles Lernen zur Optimierung einer autonomen Fahrspurführung*. Hochschule für Angewandte Wissenschaften Hamburg, November 2011

[OpenCV Wiki] *Wiki for OpenCV (Free Open Source Computer Vision)* URL: http://opencv.willowgarage.com/wiki/FullOpenCVWiki (2012-11-05)

[Otsu 1979] N. Otsu: *A Threshold Selection Method from Gray-Level Histograms, IEEE Trans. on Systems, Man, and Cybernetics*, Bd. 9, 1979, S. 62–66.

[PEARL-News] PEARL-News: *Implementation von hierarchischen Zustandsautomaten in Echtzeitsystemen*, Ausgabe 24, November 2006

[RIES 2005] Sascha Ries *Hinderniserkennung* Seminar "Mobile Systeme" Institut für Informatik, Universität Koblenz-Landau Januar 2005, Seite 3

[Risse] Prof. Dr. Thomas Risse: hs-bremen.de: *Segmentierung* URL: http://www.weblearn.hs-bremen.de/risse/AWI/SEGMENT/SEGMENTI.HTM (2013-01-28)

[roboticlab] Robotic & Mechatronics *Infrarot-Entfernungsmesser* URL: http://home.roboticlab.eu/de/examples/sensor/ir_distance (2013-03-08)

[Robotik 2008] Dr. Johannes Steinmüller: *Robotik*, TU Chemnitz, 2008/2009 URL: http://www-user.tu-chemnitz.de/~stj/lehre/ROBO.pdf (2012-11-20)

[TUM] Technische Universität München *IR-Abstandssensoren von SHARP* URL: http://www.daedalus.ei.tum.de/index.php/de/dokumentation/material/sensoren/ir-abstandssensor (2013-03-05)

[Vacek 2009] Stefan Vacek: *Videogestützte Umfelderfassung zur Interpretation von Verkehrssituationen für kognitive Automobile*, Karlsruhe (Universitätsverlag Karlsruhe), 2009

[vhs] vhs-seminar.de: Farbtiefe URL: http://www.vhs-seminar.de/farbtiefe.html (2013-01-18)

[Wienke 2008] Johannes Wienke: *Bildverarbeitung mit OpenCV*, Universität Bielefeld, Mai 2008, URL: http://www.semipol.de/wp-content/uploads/sites/4/2008/07/tutorial.pdf (2012-11-12)

[Wirtschaftsspiegel 2006] Dr. Sören Kammel, *Universität entwickelt kognitive Automobile* Universität Karlsruhe (TH), Institut für Mess- und Regelungstechnik Karlsruher Wirtschafts-spiegel 2006/2007 URL: http://www1.karlsruhe.de/Wirtschaft/img/standort/profile/down233.pdf (2013-02-21)

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, den _____ _____