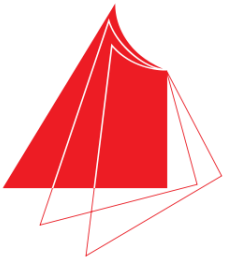


Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Hochschule Karlsruhe - Technik und Wirtschaft

Studienarbeit

Weiterentwicklung eines autonom fahrenden RC-Fahrzeugs

Erweiterung Objekterkennung & -verfolgung

Projektcode: 19WS-OL-CaroloCup

Projektkürzel:
MECM150
MECM250

Mechatronic Competition Team

Wintersemester 2019-2020

Daniel Autenrieth 59538

Prof. Dr.-Ing. Ferdinand Olawsky
Dipl.Ing.(FH) Bernhard Beck
Dipl.Inform. Jurgen Ewert

22. Mai 2020

MMT

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Aufgabenstellung Projektarbeit

Fakultät für Maschinenbau und Mechatronik

Projekttitel: Carolo-Cup

Studentischer Wettbewerb zum Thema „Autonomes Fahren“

Hintergrund zum Projektthema:

An der TU Braunschweig findet jedes Jahr ein studentischer Wettbewerb zum Thema **„autonomes Fahren“** statt. Bei diesem Wettbewerb müssen die Teams ihre selbstentwickelten autonomen Modellautos präsentieren und demonstrieren, dass das Modellauto vorgegebene Disziplinen beherrscht. Der Wettbewerb ist auf recht hohen Niveau, in der Jury sitzen Vertreter großer Industrieunternehmen. Die Fakultät MMT hat bereits viermal an diesem Wettbewerb teilgenommen.

Aufgabenstellung des Projektes:

In den letzten Semestern wurde ein Modellauto entworfen und gebaut, das im Februar 2019 erfolgreich am Carolo-Cup-Wettbewerb teilgenommen hat. Das Potential des Fahrzeugs ist noch nicht ausgereizt, die Leistungsfähigkeit kann noch deutlich gesteigert werden. Der Schwerpunkt liegt momentan in der **Weiterentwicklung der Software** (autonome Steuerung, Einparkroutine, **Bildverarbeitung**, Geschwindigkeitsregelung, Hinderniserkennung, ...). Im Bereich der Elektronik/Sensorik/Aktorik sind noch kleinere Baustellen zu bearbeiten.

Die Projektarbeit wird folgende Ziele bzw. Ergebnisse beinhalten:

- Inbetriebnahme des aktuellen Carolo-Cup-Fahrzeugs
- Einarbeiten in die eingesetzten Software-Bibliotheken (ROS, OpenCV, CUDA) und in die eingesetzte Hardware (Nvidia Jeston Board, Arduino)
- Testen und Bewerten der vorhandenen Software
- Weiterentwicklung der Software, insbesondere im Hinblick auf Hinderniserkennung und Hindernisumfahrung
- Entwicklung einer schnellen Bildverarbeitung auf dem Jetson TX2 Board für die autonome Steuerung

Ziel des Projekts im Wintersemester 2019/20 ist es, alle Wettbewerbsdisziplinen gemäß den aktuellen Wettbewerbsregeln zu beherrschen.

Bearbeitung im:	**Zielgruppe:**	**Projekt-Code:**
WS 201920	Bachelor oder Master	19ws_OL CaroloCup

Betreuer/Kontakt: Prof. Dr.-Ing. Ferdinand Olawsky; Tel.:1710; E-Mail: ferdinand.olawsky@hs-karlsruhe.de
Prof. Dr. Stefan Ritter (Fakultät EIT)

Projektbearbeiter	**Matr.Nr.**	**Studiengang/Semester**	**Unterschrift/Datum**
Alexander Haaf	71346	MECM / 2	A. Haaf / 04.11.19
Miroslav Andjelkovic	65877	MECM / 2	/ 04.11.19
Maurus MANATSCHAL	71652	MECM / 2	/ 04.11.19
David BLOCH	71674	MECM / 2	/ 06.11.19
Stefanie Wagner	62717	MECB / 5	S. Wagner 07.11.19
Daniel Autenrieth	59538	MECB / 6	07.11.19
Ydbert Yosua Trihardianto	58001	MECB / 6	07.11.19
Jan VAILLANT	73471	MECM / 1	07.11.19

Betreuer/Dozent: Prof. Dr.-Ing. Ferdinand Olawsky F. Olawsky

Eidesstattliche Erklärung

Eidesstattliche Erklärung zur Arbeit

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Karlsruhe, den 22.05.2020

Ort, Datum

Daniel Autenrieth

Inhaltsverzeichnis

Anmerkung

Diese Arbeit wurde im Rahmen des Carolo Cups verfasst und dient zur Dokumentation der geleisteten Arbeit sowie als Grundlage zum Verständnis der entwickelten Software. Aufgrund der Verlängerung der Arbeitszeit konnte diese Ausarbeitung nicht zeitgleich mit den Berichten der anderen Teammitglieder abgegeben werden und wird nun als Erweiterung des Gesamtberichtes nachgereicht. Alle hier gezeigten Darstellungen entstammen der beschriebenen Software oder der beiliegenden Power-Point-Präsentation. Aufgrund des begrenzten Umfangs dieser Dokumentation wurde nicht auf alle Details des Hauptprogrammes eingegangen und das Framework wurde nur in seinen Grundzügen beschrieben. Genauere Beschreibungen sowie Nutzungsanleitungen wurden den Programmen beigefügt.

Aufgabenstellung

Für die zukünftige Teilnahme an dem Carolo Cup sollen Objekte auf der Fahrbahn erkannt werden. Im Master-Cup des Carolo Cups müssen neben Start- und Stopplinien auch Hindernisse erkannt und aufgrund derer bestimmte Manöver ausgeführt werden. Zur Detektion stehen die Bilder der bereits vorhandenen Kamera sowie die Funktionen der Fahrbahnbegrenzungen zur Verfügung. Auf Basis der transformierten Bilder sollen dann Objekte erkannt und verfolgt werden. Dies soll auch bei fehlenden Parametern weiterhin möglich sein.

Evaluation verschiedener Methoden / Vorgehensweise

Nach der Einarbeitung in das bestehende System wurden die verschiedenen Möglichkeiten evaluiert. Ein selbstlernendes System wurde aufgrund der nicht vorhandenen Trainingsdaten und deren aufwendigen Erzeugung ausgeschlossen. Der am Anfang verfolgte Ansatz, Objekte aufgrund ihrer Form auf dem Gesamtbild, mithilfe der in OpenCV bereits vorhandenen Funktionen, zu ermitteln, wurde nach einigen Versuchen aufgrund von nicht ausreichenden Ergebnissen verworfen. Die nun ausgearbeitete Software basiert auf einem selbstentwickelten Algorithmus und orientiert sich vom Grundprinzip an bereits verfassten Arbeiten, in denen nur bestimmte Bereiche des Bildes ausgelesen werden[1].

Software

Grundprinzipien

Geringe Rechenzeit. Möchte man die entwickelte Software möglichst leistungsfähig konzipieren, bietet es sich an nur die Bereiche/Pixel der Eingabe zu betrachten die von Relevanz sind. Aufgrund der Funktionen der Fahrbahnbegrenzungen, welche in der vorrausgehenden Fahrzeugsoftware ermittelt werden, lässt sich der zu betrachtende Bereich bereits deutlich einschränken. Des Weiteren sind Objekte erst ab einer gewissen Position interessant, weshalb der oberste Teil des Bildes ebenfalls nicht betrachtet werden muss. Das Grundprinzip des entwickelten Programms besteht nun darin immer nur eine geringe Zahl von Pixeln zu betrachten. Dies beginnt damit nur Pixel auf einer, orthogonal zu den Begrenzungsfunktionen stehenden,

[1] Bachelorarbeit, Waldemar Heck, Hindernis- und Kreuzungserkennung auf autonomen Fahrzeugen durch Kamera und Infrarotsensorik, Hochschule der Angewandten Wissenschaften Hamburg, 27.05.2013

Linie auszuwerten. Wurde hierbei ein bestimmter Prozentsatz an weiß-definierten Pixeln erfasst, werden erst die anderen Funktionen der Software ausgeführt. So besteht der einfache Durchlauf der Software, sollte kein Objekt verfolgt werden und die Anzahl der weißen Pixel gering sein, lediglich aus einer Abfrage von etwa 100 Pixeln.
Neben der Rechenzeit ist auch die Komplexität des Programmes ein wichtiger Faktor. Zum einen erleichtert es dem ursprünglichen Programmierer den Überblick zu behalten und schnell ganzheitliche Änderungen vorzunehmen. Zum anderen verkürzt sich die Einarbeitungszeit des darauffolgenden Programmierers enorm. Aus diesem Grund wurde versucht die grundlegende Logik und damit die Algorithmen auf ein paar wenigen Funktionen aufzubauen.
Erweiterbarkeit und Zugänglichkeit der Ergebnisse. Der Aufbau des Programmes richtet sich ebenfalls nach diesen Prinzipien. Objekte können, durch die objektorientierte Programmierung, den einfachen Aufruf der Detektionsfunktionen sowie der klassenübergreifenden Trackingfunktion, mit geringem Aufwand hinzugefügt oder erweitert werden. Die Ergebnisse der Funktionen werden in den Objekten gespeichert und können dann von anderen Teilen des Programmes jederzeit und ohne zusätzliche Funktionsaufrufe weiterverarbeitet werden.

Framework

Zur Programmierung und Evaluation des entwickelten Programmes werden gewisse Daten benötigt: Das transformierte Bild der Fahrzeugkamera wird im späteren Betrieb auf dem Fahrzeug ebenso der Funktion/ dem Programm bereitgestellt wie die dazu passenden Parameter der Begrenzungsfunktionen. Da diese jedoch nur bei aktiver Fahrt auf dem Fahrzeugcontroller zur Verfügung stehen, musste zuerst ein Framework entwickelt werden, welches es dem Nutzer/Programmierer erlaubt die benötigten Daten zu erstellen und dem Programm zuzuführen.

Erzeugung der Daten

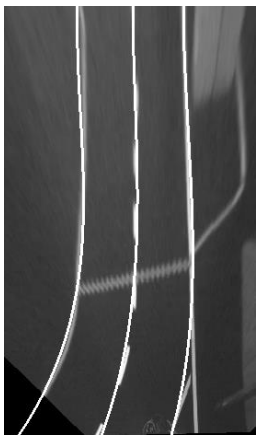
Die benötigten Bilder wurden aus gespeicherten Aufnahmen von verschiedenen Events entnommen. Hierzu wurden einfach in einem bestimmten Intervall mithilfe des VLC media players einzelne Frames der Aufnahmen extrahiert. Das hier ausgewählte Aufnahmeverhältnis war 17. Dies bedeutet das jedes 17te Bild der Aufnahme extern gespeichert wurde. Da die vorliegenden Bilder bereits transformiert wurden, ist eine weitere Bearbeitung vor Eingabe in die Software nicht mehr notwendig.
Zu den jeweiligen Bildern sind nun die Parameter der Begrenzungsfunktionen notwendig. Da das entwickelte Programm im Idealfall mit 3 Funktionen arbeitet (linke/ rechte Fahrbahnbegrenzung & Mittelstreifen), die momentan zu Verfügung stehende Fahrzeugsoftware jedoch nur 2 dieser Funktionen berechnet, war eine reine Entnahme der Parameter aus der Fahrzeugsoftware nicht möglich. Daher wurde ein zusätzliches Programm entwickelt, mit dessen Hilfe der Benutzer die Parameter der Funktionen von Hand für jedes einzelne Bild erzeugen kann. Dies begründet auch die geringere Anzahl der extrahierten Bilder. Da die Erzeugung von mehreren Tausenden Parametern dem Nutzen, den der Programmierer aus der besseren Datenlage zieht, nicht gerecht wird. Die erzeugten Parameter werden dann mit dem zusammengehörigen Bild in einer Textdatei gespeichert.

Einlesen der Daten

Um das entwickelte Programm später direkt in das Fahrzeug transferieren zu können, müssen die Daten in der gleichen Form der Gesamtfunktion „object_detection()“ übergeben werden. Aus diesem Grund wurde in dem Hauptprogramm ein Framework errichtet, welches die gespeicherten Daten einliest und diese dann, analog zu der Fahrzeugsoftware, der entwickelten Funktion übergibt.

Übergabeparameter

Die notwendigen Übergabeparameter bestehen zum einen, wie zuvor beschrieben, aus einem Bild und drei Funktionen 3ten Grades.



In Abbildung 1 wurden die Funktionen, auf dem dazugehörigen Graustufen-Bild, visualisiert. Da die Funktionen von Hand angepasst wurden, sind aufgrund der Komplexität dieser Aufgabe Abweichungen zu den tatsächlichen Ergebnissen der Fahrzeugsoftware zu erwarten. Liegen diese Abweichungen jedoch in nicht betrachteten Bereichen oder sind relativ gering, können diese vernachlässigt werden. Bei dem übergebenen Bild spielt es keine Rolle ob dieses, wie in Abb. 1 zu sehen, in Graustufen vorliegt oder bereits ein Binärer Treshold Filter angewendet wurde.

Abbildung 1 Parameter

Eingliederung in die Fahrzeugsoftware

Das entwickelte Programm soll in naher Zukunft als Funktion in die Fahrzeugsoftware übernommen werden. Aufgrund der Übergabeparameter der Funktion muss sich der Funktionsaufruf der entwickelten Funktion zeitlich hinter dem Einlesen und Transformieren des Kamerabildes befinden. Genauso wie hinter der Parametrierung der Fahrbahnbegrenzungsfunktionen. Da die Funktion jedoch wichtige Informationen über die Umgebung aktualisiert, sollte sie vor der Autopiloten-Funktion aufgerufen werden.

Aufbau der Objekte

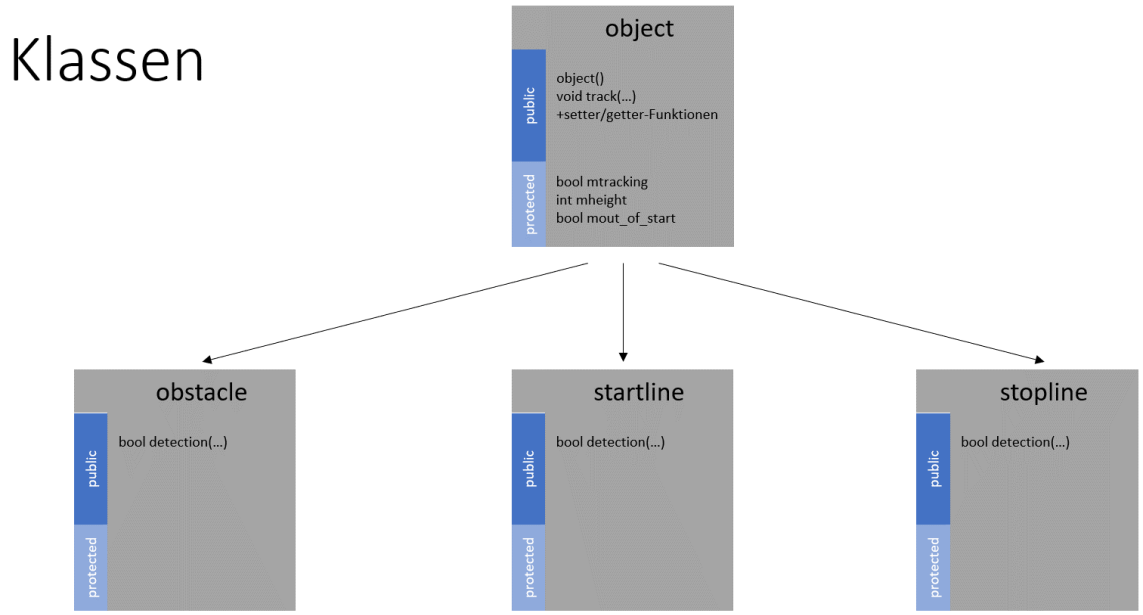


Abbildung 2 Klassen

In der entwickelten Software werden verschiedene Objekte instanziiert. In dem momentan verwendeten Softwarestand umfasst dies ein Startlinien-Objekt ein Stopplinien-Objekt und zwei Hindernis-Objekte (für

die rechte und linke Fahrspur). Alle diese verschiedenen Klassen erben von der Vaterklasse object. In dieser sind die Variablen definiert, in welchen die Ergebnisse der Software gespeichert werden. Des Weiteren existiert eine allgemeingültige Tracking-Funktion in der Vaterklasse, sowie die notwendigen Setter- und Getter-Funktionen für die Variablen. Jede Sohnklasse beinhaltet eine spezifische Detektionsfunktion, die evaluiert ob das Objekt erkannt wurde oder nicht. Ausgenommen davon ist allerdings die Startlinien-Detektionsfunktion da diese aufgrund des Ausschlussprinzips keine eigene Evaluation der Umgebung benötigt. Wird ein Objekt erkannt und kann weder einem Hindernis noch einer Startlinie zugeordnet werden, muss es sich dabei um eine Stopplinie handeln.

Vereinfachter Programmablauf

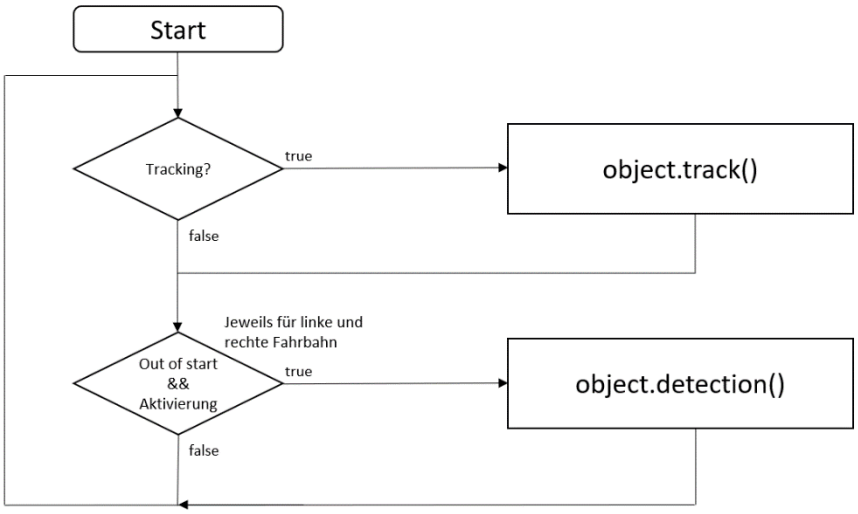


Abbildung 3 Programmablaufplan

Ist in einem der definierten Objekte die Tracking Variable auf wahr gesetzt, startet das Programm die Tracking-Funktion dieses Objektes. Sind die Variablen aktualisiert beziehungsweise die Funktion durchgelaufen, wird überprüft ob sich ein neues Objekt in dem relevanten Bereich befindet. Ist dies der Fall und es kann ausgeschlossen werden, dass es sich dabei nicht um ein bereits erkanntes Objekt handelt, wird die Detektions-Funktion ausgeführt. Diese prüft das unbekannte Objekt auf verschiedene Objekttypen und ordnet dieses einem zu.

Funktionen

Check-Pixel-Funktion/ Aktivierungsbedingung

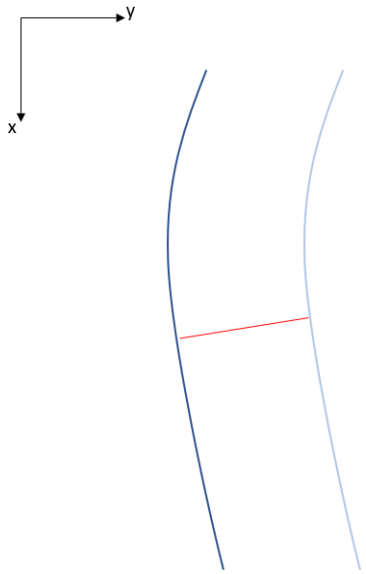


Abbildung 4 Check Pixel Funktion

Die Check_Pixel()-Funktion ist die Hauptfunktion auf der die anderen Detektions und Trackingsfunktionen aufgebaut sind. Die Funktion überprüft die Pixel welche sich auf einer, orthogonal zur Begrenzungsfunktion stehenden, Linie und zwischen den Fahrbahnbegrenzungen befinden. Ein Pixel wird als weiß definiert, wenn sein Wert über dem definierten Treshold liegt. Dieser Tresholdwert sollte, wenn möglich auf jedes Bild angepasst werden, um optimale Ergebnisse zu erlangen. Die Funktion gibt den prozentualen Anteil der als weiß-definierten Pixel zurück. Zum Beispiel ein Wert von 0.5 bedeutet, dass jeder zweite betrachtete Pixel nach Definition weiß ist. Aufgrund dieses Rückgabewertes können dann Entscheidungen getroffen werden. Überschreitet der Wert eine vorher festgelegte Grenze wird angenommen, dass sich auf der Linie ein Objekt befindet. Dies ist die Aktivierungsbedingung. Sobald diese erfüllt ist und sich kein bereits bekanntes Objekt auf der Linie befindet, werden die verschiedenen Detektionsfunktionen aufgerufen (siehe Vereinfachter Programmablauf).

Detektionsfunktion der Hindernis Klasse

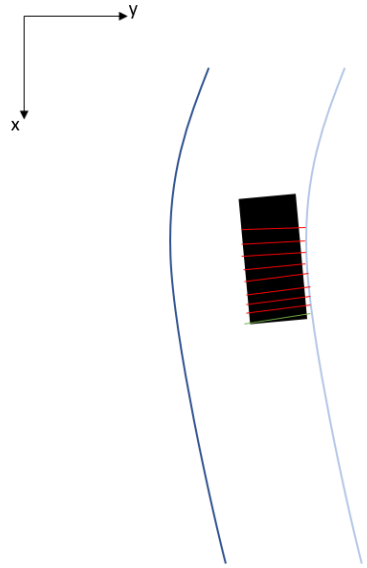
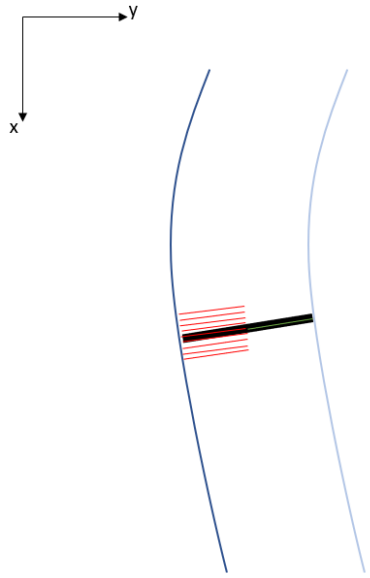


Abbildung 5 Hindernis Detektion

In der Detektionsfunktion der Hindernis Klasse wird der Bereich oberhalb der Aktivierungslinie betrachtet. Hier beispielhaft an der rechten Fahrbahnspur verdeutlicht. Die roten Linien stellen die überprüften Pixel dar. Werden bei mehreren Linien Weißanteile über dem definierten Wert erkannt, kann davon ausgegangen werden, dass sich das gefundene Objekt auch noch oberhalb der Aktivierungslinie fortsetzt. Aufgrund der Geometrie kann dann auf ein Hindernis geschlossen werden.

Detektionsfunktion der Startlinien Klasse



In der Detektionsfunktion der Starlinien Klasse wird der Bereich auf der linken Fahrbahnspur betrachtet. Da es sich bei dem betrachteten Bild nur selten um ein perfekt transformiertes Bild handelt, kann nicht davon ausgegangen werden, dass es sich bei der Startlinie um eine Gerade handelt. Aus diesem Grund wird nicht nur eine weitergeführte Linie auf der linken Seite betrachtet, sondern der gesamte Bereich. Wird in dem relevanten Bereich ein Weißanteil über dem definierten Wert erkannt, handelt es sich bei dem zu erkennenden Objekt um eine Startlinie. Da die Möglichkeit eines Hindernisses aufgrund der zuvor ausgeführten Hindernis-Detektionsfunktion nicht mehr gegeben ist kann es sich in diesem Beispiel auch nicht um ein überlappendes Hindernis handeln.

Abbildung 6 Startlinien Detektion

Detektionsfunktion der Stopplinie

Die Stopplinien-Klasse besitzt ebenfalls eine Detektionsfunktion, wie die anderen zuvor genannten Klassen, allerdings besitzt diese keine tatsächliche Logik zu Detektion eines Objektes. Aufgrund des Ausschlussverfahrens kann davon ausgegangen werden, dass es sich bei dem zu erkennenden Objekt um eine Stopplinie handelt, wenn ausgeschlossen wurde, dass es sich dabei um ein Hindernis oder eine Startlinie handeln könnte. Da in den Detektionsfunktionen auch die Variablen der Objekte verändert werden, ist die Existenz der Stopplinien-Detektionsfunktion trotzdem notwendig.

Tracking Funktion

Die Tracking-Funktion überprüft ob sich das erkannte Objekte relativ zu dem Fahrzeug bewegt hat. Hierzu sucht die Funktion die untere Kante des Objektes. Durch Vibrationen sowie eine reale relative Bewegung besteht die Möglichkeit, dass sich das Objekt auf dem Bild nach Oben bewegt. Aufgrund dessen beginnt der Tracking Algorithmus wenige Pixel über der letzten bekannten Position des Objektes nach der unteren Kante zu suchen. Es werden immer drei Linien (Check Pixel-Funktionsaufrufe) überprüft. Sind die beiden oberen Linien als weiß definiert, also ist der Weißanteil größer als der definierte Wert, und die untere Linie wird als Schwarz erkannt, kann davon ausgegangen werden, dass die Kante des Objektes gefunden wurde. Ist diese Bedingung nicht erfüllt arbeitet der Algorithmus sich Pixelreihe um Pixelreihe auf dem Bild nach unten. Wurde die Kante des zu trackenden Objektes, innerhalb eines gewissen Bereiches, nicht gefunden, wird die das Objekt in Zukunft nicht mehr verfolgt. Verschwindet ein Objekt aus dem relevanten Bereich kann davon ausgegangen werden, dass es sich dabei um eine Spiegelung handeln musste, da kein reales Objekt in der Lage wäre sich mit einer so unverhältnismäßig hohen Geschwindigkeit zu bewegen.

Globale Parameter/ Definitionen

In dem entwickelten Programm gibt es viele verschiedene Parameter, die die Funktionsweise und die Genauigkeit der Software beeinflussen. Die global gültigen sind in einer externen Header-Datei aufgeführt und können bei Bedarf angepasst werden. Auch die einzelnen Detektionsfunktionen besitzen verschiedene Definitionen. Diese können bei Fehlern oder Verbesserungen der Eingangsparameter angepasst werden. Verbessert sich zum Beispiel die Qualität des transformierten Kamerabildes, kann der relevante Bereich in der Startlinien Detektion verkleinert werden, was die Rechenzeit verkürzt.
Ein paar Parameter werden aufgrund ihrer enormen Wichtigkeit hier näher beleuchtet:

- START_HEIGHT
  Dieser Parameter bestimmt ab welcher Höhe ein Objekt erkannt und getrackt werden soll. Dieser kann nach Evaluation aller möglicher Situation auf einen festen Wert eingestellt werden. Hierbei ist das geltende Koordinatensystem zu beachten, welches von OpenCV übernommen wurde (siehe Abbildung 4 bis Abbildung 6).
- TRESHOLD
  Dieser Parameter definiert, ab welchem Wert ein Pixel als Weiß angenommen werden kann. Dieser Parameter sollte im Optimalfall auf jedes einzelne Bild angepasst werden. Denn auch hier sind Schwankungen durch den Lichteinfall möglich.
- TRIGGER_PERC
  Dieser Parameter definiert, ab welchem prozentualen Weißanteil ein Objekt erkannt werden soll. Wird dieser Parameter erhöht, verringert sich die Wahrscheinlichkeit eine kleine lokale Spieglung als Objekt zu erkennen. Jedoch erhöht sich die Wahrscheinlichkeit, ein schlecht beleuchtetes oder falsch parametriertes (Treshold) Objekt zu übersehen.

Bekannte Problematiken

Zum Zeitpunkt der Erstellung dieser Dokumentation war es leider noch nicht möglich alle Problematiken des Hauptprogrammes zu beseitigen. Die im Folgenden beschriebenen Probleme entsprechen demnach einer momentanen Aufnahme der Entwicklung.

- Die reibungsfreie Ausführung der Software soll auch ohne Verfügbarkeit aller Funktionsparameter gewährleistet sein. Deshalb ist eine gewisse Redundanz in der check_pixel()-Funktion notwendig. Sollte eine Fahrbahnbegrenzung und/oder die Mittellinie nicht gefunden werden, müssen dennoch dieselben Pixel überprüft werden, als ob alle Funktionsparameter breitgestellt worden wären. Da die Ausführung des mathematischen Problems in der Power Point Dokumentation bereits aufgezeigt ist, wird auf eine ausführliche Beschreibung hier verzichtet.

- Die zweite Problematik besteht in der möglichen Überlappung eines Hindernisses mit der anderen Fahrbahn und wird durch die Transformation des Kamerabildes verstärkt.

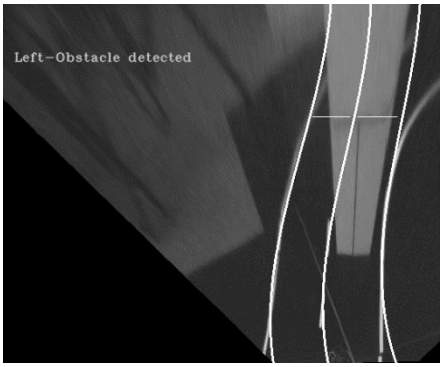


Abbildung 7 Überlappendes Hindernis

Wie in der Abbildung zu erkennen ist, ragt das Hindernis von der rechten auf die linke Fahrspur und aktiviert dort die Objekterkennung, welche diese Überlappung ebenfalls als Hindernis wahrnimmt. Da der zur Behebung des Problems notwendige Algorithmus auf demselben Grundsatz wie die check_pixel()-Funktion beruhen würde, ist es notwendig, zuerst das oben genannte mathematische Problem zu lösen, bevor eine Lösung dieser Problematik möglich ist.

Hinweise zur Implementierung in die Fahrzeugsoftware

Bei der Implementierung des Programmes in die Fahrzeugsoftware sowie die darauf aufbauende Logik müssen einige Punkte beachtet werden. Ein Hindernis, welches von außerhalb der Fahrbahn in diese fährt, kann je nach Richtung und Höhe unterschiedlich klassifiziert werden. Deshalb darf die Gesamtfunktion während eines Kreuzungsaufenthalts nicht aufgerufen werden. Da vorbeifahrende Objekte/ Fahrzeuge falsch klassifiziert und darauffolgend falsche Manöver ausgeführt werden könnten.
Aber auch die Detektionssoftware an sich ist nicht für alle Situationen ausgelegt. Wird zum Beispiel ein bestimmtes Objekt verfolgt und es wird ein neues Objekt derselben Klasse erkannt, wird dieses nicht verfolgt, da von einer Spiegelung ausgegangen wird. Bereits getrackten Objekten werden eine höhere Priorität und eine höhere Wahrscheinlichkeit, keine Spiegelung zu sein, eingeräumt. Da unter normalen Umständen nicht mehr als ein Objekt pro Klasse verfolgt werden muss, ist die Implementierung von mehreren Objekten derselben Klasse nicht notwendig. Sollte es doch zu der Notwendigkeit kommen, können durch kleine Änderungen neue Objekte der Software hinzugefügt werden.

Zusammenfassung

Die hier vorgestellte Software besteht aus 2 Programmen und lässt sich des Weiteren in zwei grobe Bausteine unterteilen: Das Framework, welches benötigt wurde, um die Daten für die Hauptsoftware zu erzeugen und dieser die Daten dann zu übergeben. Und die Hauptfunktion welche, sobald diese in die Fahrzeugsoftware implementiert wurde, in der Lage sein wird, verschiedenen Objekte in Echtzeit zu erkennen, sie zu klassifizieren und sie zu verfolgen. Da aufgrund der momentanen Situation und der zeitlichen Beschränkung die Implementierung sowie der Test der Software auf dem Fahrzeug nicht mehr möglich ist, müssen die oben genannten Parameter (siehe Globale Parameter/ Definitionen) zu einem späteren Zeitpunkt experimentell bestimmt werden.

Ausblick / Weitere Schritte

Die entwickelte Software sollte im Fahrzeug implementiert und die Parameter angepasst werden. Darauf aufbauend sollten Algorithmen zur Reaktion auf die verschiedenen Objekte und Situationen entwickelt werden. Hierbei wird auch die Kombination, mit der von Herr Trihardianto entwickelten Kreuzungserkennungssoftware eine wichtige Rolle spielen. Sollten die zuvor beschriebenen Problematiken (Bekannte Problematiken) noch nicht gelöst worden sein, ist auch die Behebung dieser ein wichtiger Schritt.