Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Ilona Blanck

Ausweichmanöver von kognitiven autonomen Fahrzeugen an statischen Hindernissen

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Ilona Blanck

Ausweichmanöver von kognitiven autonomen Fahrzeugen an statischen Hindernissen

Masterarbeit eingereicht im Rahmen der Masterprüfung
im Studiengang 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 16. Januar 2012

Ilona Blanck

Thema der Masterarbeit

Ausweichmanöver von kognitiven autonomen Fahrzeugen an statischen Hindernissen

Stichworte

FAUST, Bildverarbeitung, autonomes Fahrzeug, Carolo-Cup, Schwellwertverfahren, Newton-Verfahren, Bayes-Theorem, Kartierung, Zustandsautomat, Ausweichverfahren

Kurzzusammenfassung

In dieser Arbeit wird ein Verfahren für das autonome Ausweichen von statischen Hindernissen für die Fahrzeugplattform Onyx innerhalb des Projektes FAUST der HAW Hamburg entwickelt. Dieses soll im Carolo-Cup 2010 der Technischen Universität Braunschweig zum Einsatz kommen. Zur Entwicklung des Ausweichverfahrens fehlten einige Grundvoraussetzungen, so dass diese im Rahmen der Arbeit entwickelt werden mussten. Hierbei handelt es sich um das Erkennen der Hindernisse im Kamerabild, deren Bestimmung im Raum und Zuordnung zur Fahrbahn, sowie einer Darstellung der Fahrzeugumgebung in Form einer Karte. Für den Ausweichvorgang wurde ein Automat entwickelt, der durch eine Situationsanalyse anhand der Karte entscheidet, in welchen Zustand er gelangt. Es wird überprüft ob ein Ausweichvorgang eingeleitet werden muss. Ist dies der Fall, wird ein Fahrspurwechsel auf die andere Fahrspur und nach dem Überholen des Hindernisses die Rückkehr auf die ursprüngliche Fahrspur angeregt. Für den Fahrspurwechsel selbst wird die bestehende Fahrspurführung, die das Fahrzeug in der Fahrspur halten soll, benutzt.

Ilona Blanck

Title of the paper

Avoidance of static obstacles performed by a cognitive autonomous vehicle

Keywords

FAUST, image processing, autonomous vehicle, Carolo-Cup, thresholding, Newton's method, Bayes' theorem, mapping, state machine, obstacle avoidance

Abstract

A procedure for autonomous avoidance of static obstacles for the vehicular platform Onyx employed by project FAUST by the HAW Hamburg has been developed in the course of this master's thesis. This procedure will be used during the Carolo-Cup 2010 conducted by the Technical University of Braunschweig. Some basic requirements had not been met for developing the obstacle avoidance procedure. Specifically the recognition of obstacles via camera, their assignment to a location and lane, as well as a depiction of the proximity of the vehicle by means of a map. A state machine, which analyses the situation based on information provided by the map, has been developed. Based on the collected data the machine selects the correct state to go into. It is checked if an avoidance procedure has to be initiated. If this is the case, a lane change to the other lane and after passing the obstacle a return to the original lane will be performed. The existing lane guidance system is used for the lane changing procedure itself.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einführung

Während einer Autofahrt muss der Autofahrer in der realen Welt diversen Hindernissen ausweichen. Dabei kommt es immer wieder zu Kollisionen mit anderen Verkehrsteilnehmern. Solche Unfälle geschehen, weil der Fahrer nicht die gesamte Umgebung wahrnehmen kann, er bei der Entscheidungsfindung zögert, oder den Ausweichvorgang nicht korrekt durchführt. In der Autoindustrie gewinnen Fahrassistenzsysteme immer mehr an Bedeutung. In einer idealen Zukunft ist der Mensch nicht mehr auf die eigene Fahrtüchtigkeit angewiesen, denn hier übernimmt ein hochentwickeltes autonomes Fahrsystem seine Aufgaben.

Ausweichen bedeutet allgemein: „aus der Bahn eines anderen gehen [und Platz machen]“ ([3]). Wird Fahrzeugen im Straßenverkehr durch ein Hindernis auf der genutzten Fahrspur die Fahrt erschwert, muss unter Einhaltung der Straßenverkehrsordnung ausgewichen werden. In vielen Fällen des alltäglichen Lebens muss Hindernissen ausgewichen werden, die weit im Voraus zu erkennen sind, so dass ein Verlassen der Fahrbahn vermieden werden kann. Eine solche Situation ist beispielsweise ein in zweiter Reihe parkender Kleintransporter. Für den anstehenden Ausweichvorgang muss die Umgebung beobachtet und die Entfernung zu den anderen Verkehrsteilnehmern ermittelt und ausgewertet werden. Der Ausweichvorgang muss im richtigen Moment eingeleitet werden. Der Fahrspurwechsel sollte dabei möglichst sanft ablaufen, um Insassen nicht in eine unkomfortable Beschleunigung zu bringen und zu verletzen. Letztlich soll das Fahrzeug wieder auf die zuvor genutzte Fahrbahn wechseln, um keine Behinderung des Gegenverkehrs darzustellen.

Für eine Ausweichtechnik ist die Unterscheidung verschiedener Verkehrsteilnehmer wie Fahrradfahrer, Autos, LKWs oder Bussen zunächst nicht relevant. Ebenso müssen erhöhte Schwierigkeiten der Umgebungserkennung durch verschmutzte oder nicht vorhandene Fahrbahnmarkierungen in Bezug auf die Ausweichtechnik nicht berücksichtigt werden. Daher kann die reale Welt auf ein Modell reduziert werden.

Der Carolo-Cup ist ein von der TU Braunschweig veranstalteter Wettbewerb, in dem alltägliche Situationen im Straßenverkehr simuliert werden. Die Umgebung im Carolo-Cup ist klar strukturiert und reduziert. So haben zur Vermeidung der Komplexität beim Ausweichen die Hindernisse alle dieselbe definierte Form (Kapitel 1.3.2), so dass für das im Rahmen dieser Arbeit zu verwirklichende Ausweichverfahren auf ein bereits definiertes Modell der realen Welt zurückgegriffen werden kann.

1.1 Zielsetzung und Aufbau der Arbeit

Unter Einhaltung der im Carolo-Cup 2010 vorgegebenen Richtlinien soll ein Algorithmus für das Ausweichen von Hindernissen der aus dem FAUSTProjekt der HAW Hamburg zur Verfügung stehenden, autonomen Fahrzeugplattform Onyx realisiert werden.

Zu diesem Zweck wird im ersten Teil der Arbeit eine Übersicht über das Gesamtprojekt FAUST, den Rahmenbedingungen des Wettbewerbs Carolo-Cup 2010, sowie der fahrzeuginternen Sicht auf die Welt gegeben.

Ein elementarer Baustein für den Ausweichvorgang ist das Erkennen von Hindernissen vor dem Fahrzeug. Dieser Bereich wird über eine Kamera erfasst. Das Erkennen der Hindernisse wird mit Hilfe eines Bildverarbeitungsalgorithmus realisiert, welcher in **Kapitel 2** beschrieben wird.

Um festzustellen, ob ein Hindernis, dem ausgewichen werden muss, im Weg ist, muss die Umgebung des Fahrzeugs bekannt sein. Zur Modellierung der Umgebung wird eine Karte erstellt. Der Detailgrad der Karte und deren Erstellung wird in **Kapitel 3** beschrieben.

Für das Ausweichverfahren wird ein Automat genutzt, der anhand der Karte eine Situationsanalyse der Umgebung durchführt und dementsprechend den Ausweichvorgang plant. Das für den Carolo-Cup 2010 genutzte Ausweichverfahren wird in **Kapitel 4** beschrieben.

In **Kapitel 5** werden einige Testszenarios und entstandene Probleme, sowie der Einsatz im Carolo-Cup beschrieben.

Die Inhalte dieser Arbeit und der Stand des Verfahrens werden in **Kapitel 6** auf ihre Nachhaltigkeit hin diskutiert.

Diese Arbeit schließt mit einem Fazit in **Kapitel 7**.

1.2 Einordnung in das Gesamtprojekt

Im Projekt FAUST (Fahrerassistenz- und Autonome Systeme) der HAW Hamburg ([6]) ist eine Softwarearchitektur entwickelt worden, die verschiedenen Projekten mit unterschiedlichen Plattformen ein Grundgerüst für das autonome Fahren zur Verfügung stellt. Die Softwarearchitektur besteht aus dem FAUSTcore und den FAUSTplugins. Der FAUSTcore stellt dem Anwender dabei eine vollständige Ausführungsumgebung seiner definierten Module zur Verfügung. Zudem ermöglicht der FAUSTcore die bequeme Steuerung der Anwendung über eine Web-Oberfläche. Über die Web-Oberfläche können den Modulen zu jedem Programmstart Parameter übergeben werden. Die Parameter können manuell vorgegeben werden. Die FAUSTplugins beschreiben die einzelnen Module wie z.B. die Spurerkennung, die Lenkungssteuerung, das Erkennen und Ausweichen von Hindernissen etc. Diese werden gemäß ihrer Einstellungen vom FAUSTcore-Scheduler gestartet, wobei die Reihenfolge durch eine Priorität der einzelnen Module vorgegeben wird. Ähnlich wie bei der Subsumption-Architektur können höhere Ebenen Signale auf niedrigeren Ebenen überschreiben (Abb. 1-1). Die Module der FAUSTplugins sind fahrzeugspezifisch, da sie von den genutzten Sensoren und Aktoren abhängig sind.

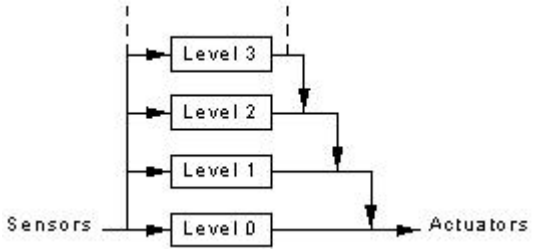

Abb. 1-1: Subsumption-Architektur

Eines der Unterprojekte von FAUST ist die Teilnahme am Carolo-Cup. In Tabelle 1-1 sind die einzelnen Module der FAUSTplugins dargestellt. Der unterste Eintrag hat die niedrigste und der oberste Eintrag hat die höchste Priorität. Die Module mit niedriger Priorität werden zuerst ausgeführt. So können sie den folgenden Modulen Information über einen Datencontainer zur Verfügung stellen. Höher priorisierte Module können Signalwerte, wie z.B. Geschwindigkeit oder Lenkwinkel, überschreiben.

Name	**Kurzbeschreibung**
breakLightAssistant	Ein Bremslichtassistent, der bei Reduktion der Geschwindigkeit die Bremslichter anschaltet und bei einer Beschleunigung des Fahrzeugs die Bremslichter ausschaltet. Durch dieses Modul braucht sich kein weiteres Modul um die Aktivierung und Deaktivierung der Bremslichter kümmern.
BreakOnCrossing	Regelt das Verhalten an Stopp-Kreuzungen, inklusive dem Erkennen von Kreuzungen. Das Kreuzungserkennen könnte auch als eigenständiges Modul hinter oder vor dem Hinderniserkennen platziert werden. Die Daten könnten somit anderen Modulen, wie

	einer Kartenerstellung der gesamten Fahrstrecke, zur Verfügung gestellt werden.
obstacleAvoidance	Regelt das Ausweichen an statischen Hindernissen (Kapitel 4).
obstacleMap	Erstellt eine Karte der direkten Umgebung des Fahrzeugs (Kapitel 3).
laserLineObstacleDetection	Ein Bildverarbeitungsalgorithmus, der Hindernisse erkennt (Kapitel 2).
steeringControl	Dient der Fahrspurführung auf einer Strecke ohne Hindenisse (Kapitel 1.5.2). Sie kann für den Fahrspurwechsel während des Ausweichvorgangs (Kapitel 4) ausgenutzt werden.
POLARIS	Ein Bildverarbeitungsalgorithmus, der die Straße erkennt (Kapitel 1.5.1). Hierbei wird eine Fahrspur überwacht, nicht die gesamte Fahrbahn. Das Modul stellt zwei Polynome der beiden Fahrspurmarkierungen, sowie die Information über die überwachte Fahrbahnseite zur Verfügung.

Tabelle 1-1: Auflistung der Module in den FAUST-Plugins

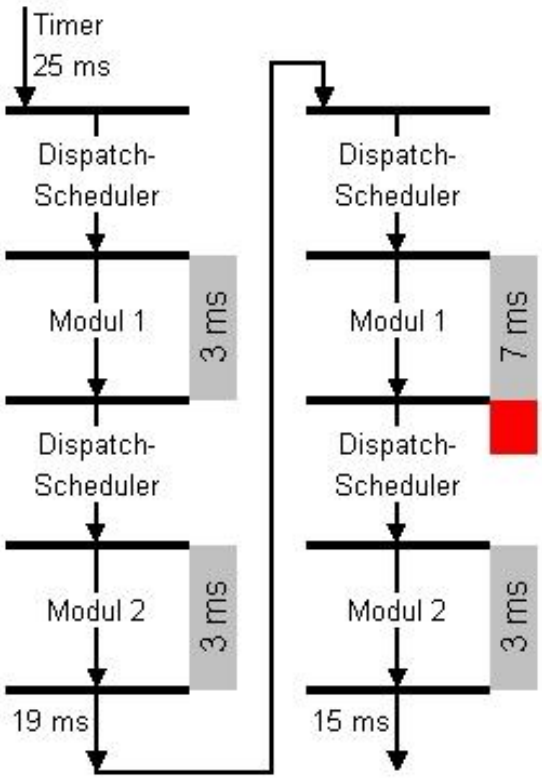

Abb. 1-2: Scheduling und Ausführungszeit

Dem FAUSTcore wird ein Systemtakt fest vorgegeben, in dem alle Module durchlaufen werden müssen. Dieser ist auf 25ms festgelegt. Durch einen Timer, der den Systemtakt bestimmt, wird sichergestellt, dass nach Ablauf der 25ms das Scheduling der einzelnen Module neu gestartet wird. Beim Scheduler sind alle auszuführenden Module angemeldet. Die Priorität der Module bestimmt die Reihenfolge, in der sie abgearbeitet werden. Der Dispatch-Scheduler veranlasst die Ausführung des Moduls, welches in der Abarbeitungsliste als nächstes steht. Das Modul arbeitet seine Methoden ab und wird nach deren Abarbeitung beendet. Daraufhin wird das nächste in der Abarbeitungsliste stehende Modul gestartet. Nach Ablauf des eingestellten Systemtaktes wird dieser Kreislauf erneut gestartet und der Dispatch-Scheduler fängt erneut an seine Liste der Module von oben abzuarbeiten. Das Scheduling wird durch die Ausführungszeit der einzelnen Module bestimmt. Die einzelnen Module können unterschiedliche Ausführungszeiten haben (Abb. 1-2). Beispielsweise muss für den Ausweichvorgang erst das Ausweichen berechnet werden, sobald ein Hindernis vorhanden ist, dem ausgewichen werden muss. Es kann nicht garantiert werden, dass jedes Modul genau alle 25ms zur Ausführung kommt.

1.3 Umgebungsbeschreibung

Für einen Ausweichvorgang muss die Umgebung des Fahrzeugs wahrgenommen werden, um auf sie zu reagieren. Dabei handelt es sich um ein reduziertes Modell der Welt. Vorgegeben wird die Umgebung durch das Szenario Carolo-Cup. Laut Regelwerk ([20]) des Carolo-Cups handelt es sich um einen Rundkurs mit langen Geraden, Kurven und vorfahrtsberechtigten Kreuzungen. Des Weiteren befindet sich eine Parkbucht zum parallelen Einparken an einer Stelle des Rundkurses. Ein möglicher Aufbau des Rundkurses ist im Anhang A dargestellt. Auf und neben der Fahrbahn können diverse Hindernisse aufgebaut sein. Die Fahrbahn wird in Kapitel 1.3.1 und die Hindernisse in Kapitel 1.3.2 genauer beschrieben.

Neben den Details der Umgebung wird auch dargestellt, wie die Umgebung mit der Kamera aufgenommen wird.

1.3.1 Fahrbahn

Die Fahrbahn befindet sich auf einer Ebene und besteht aus einem schwarzen Untergrund mit weißen Markierungen. Die Markierungen sind zwei durchgezogene Außen- und eine gestrichelte Mittelspur, eine Startlinie, sowie Stopplinien an den Kreuzungen. Die Fahrbahnmarkierungen können Lücken bis zu einem Meter aufweisen.

Abb. 1-3 und Abb. 1-4 zeigen die im Regelwerk vorgegebenen Abmessungen der Fahrbahnmarkierungen sowie der Fahrbahnbreite.

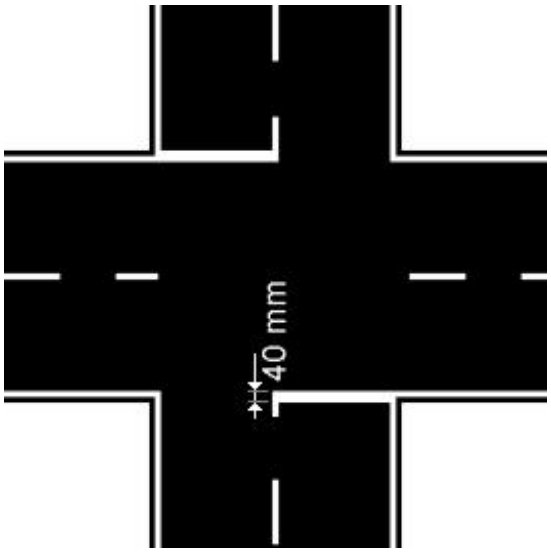

Abb. 1-3: Grafische Darstellung einer Kreuzung mit Stopplinie ([20])

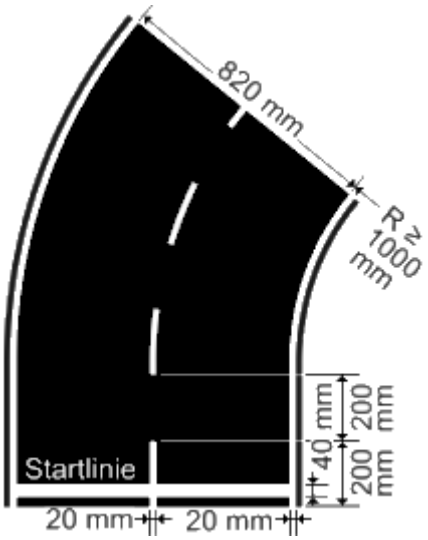

Abb. 1-4: Grafische Darstellung der Startlinie und eines Kurvenabschnitts mit kleinstmöglichem Radius ([20])

Die Parkbucht befindet sich rechts neben der Fahrbahn und wird links von der rechten Außenmarkierung der Fahrbahn abgegrenzt sowie rechts von einer weiteren durchgezogenen weißen Linie, die ebenfalls 20mm breit ist (Abb. 1-5). Da die Parkbucht außerhalb der Fahrbahn liegt, ist sie für den Ausweichvorgang nicht von Bedeutung.

Abb. 1-5 und Abb. 1-6 zeigen zwei Kamerabilder von der Fahrbahn auf der Teststrecke der HAW Hamburg im Jahr 2009. Bei dem Bodenbelag handelt es sich um das gleiche Material wie es auch im Carolo-Cup verwendet wird. Auf beiden Bildern wurde ein Hindernis auf der Fahrbahn platziert. Auf den Hindernissen projiziert der Linienlaser eine Linie parallel zum

Boden. Abb. 1-5 zeigt einen geraden Streckenabschnitt mit einer Haltebucht auf der rechten Seite. Auf der Abb. 1-6 befindet sich das Fahrzeug in einer Rechtskurve.

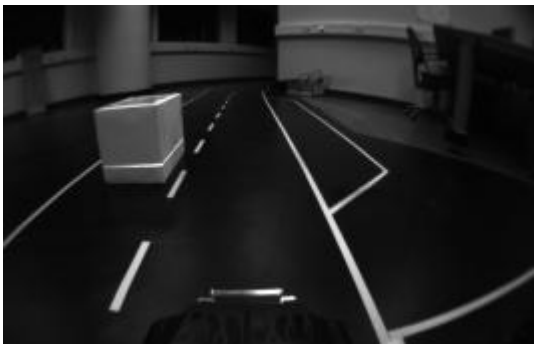

Abb. 1-5: Kamerabild eines geraden Streckenabschnitts mit einer Parkbucht rechts und einem Hindernis auf der linken Fahrspur.

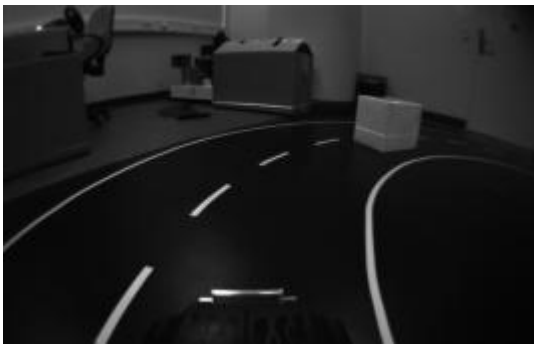

Abb. 1-6: Kamerabild einer Rechtskurve mit Hindernis auf der rechten Fahrspur

Abb. 1-7 zeigt die Rundstrecke am Wettkampftag des Carolo-Cup 2009. Es sind zwei Lücken in der Fahrbahnmarkierung zu erkennen. Einmal direkt hinter einer langgezogenen Kurve (vorne im Bild) sowie in der Mitte einer S-Kurve (hinten rechts im Bild). In beiden Fällen erschwert zusätzlich ein Hindernis die Weiterfahrt.

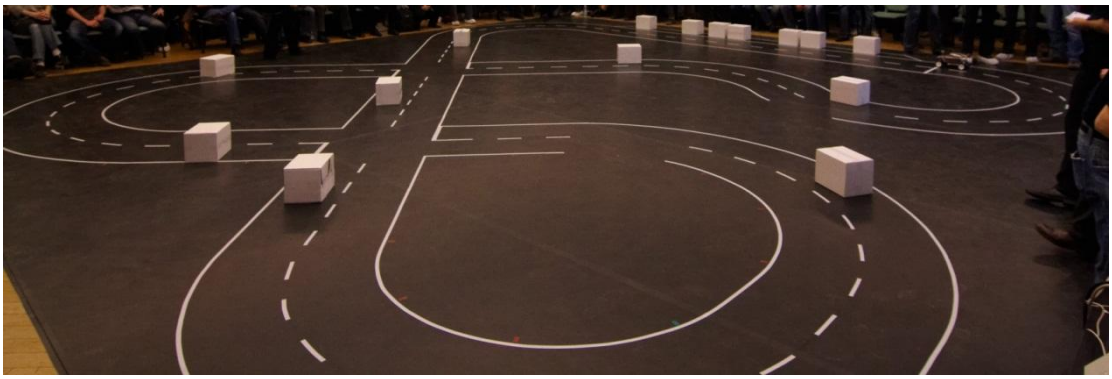

Abb. 1-7: Foto des Rundkurses am Wettkampftag mit Hindernissen und Lücken in der Fahrbahnmarkierung

1.3.2 Hindernis

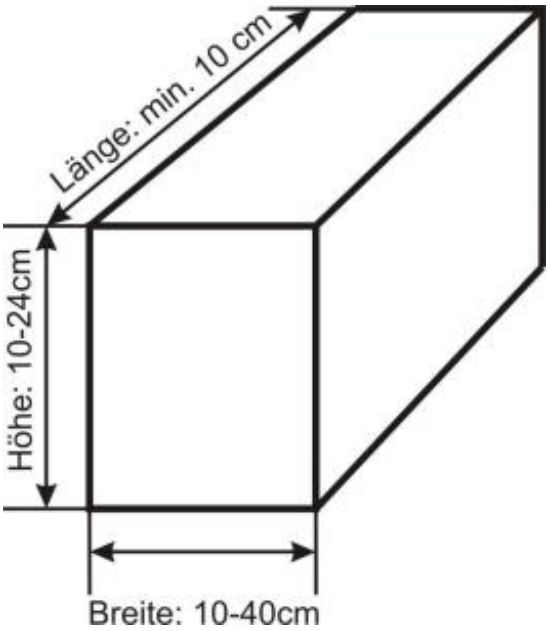

Abb. 1-8: Dimensionen der Hindernisse ([20])

Auf und neben der gesamten Fahrbahn können sich Hindernisse befinden. Die Hindernisse werden durch weiße Pappkartons dargestellt (Abb. 1-7). Jedoch ist nicht auszuschließen, dass sich auch Menschen neben der Fahrbahn befinden. Die Dimensionen der Hindernisse ist in Abb. 1-8 angegeben. Das kleinste Hindernis kann eine Größe von 10x10x10cm besitzen. Die Breite und Höhe der Hindernisse ist auf maximal 40x24cm begrenzt. Eine maximale Länge der Hindernisse ist nicht angegeben.

Die Hinderniskartons im Wettbewerb 2009 hatten ungefähr die Maße: 23x24x32cm. Durch Zusammenstellen mehrerer Kartons konnte die Länge jedoch erhöht werden. Die genutzten Hindernisse standen am Testtag bereits zur Verfügung, so dass die Breite und Höhe der Hindernisse während des Wettkampfes bekannt sind.

Im Folgenden sind einige Hindernispositionen schematisch dargestellt. Ein schwarzer Pfeil markiert die Fahrspur und Fahrtrichtung des Fahrzeugs. Das Hindernis ist als grauer Kasten abgebildet.

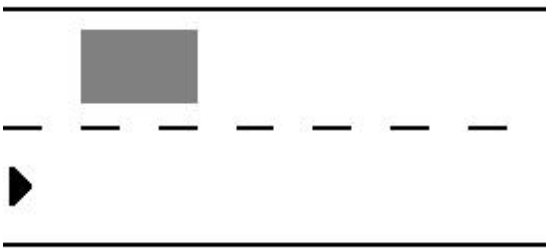

Abb. 1-9: Ein Hindernis befindet sich auf der linken Fahrspur

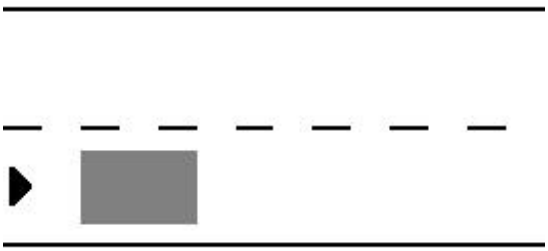

Abb. 1-10: Ein Hindernis befindet sich auf der rechten Fahrspur

Abb. 1-11: Zwei Hindernisse befinden sich hintereinander mit einem Mindestabstand von einem Meter auf der linken Fahrspur

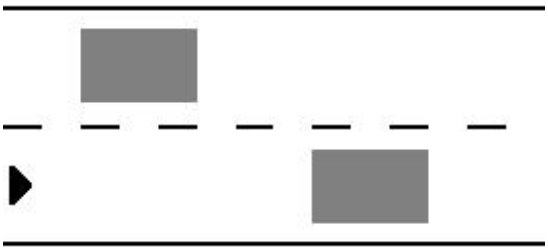

Abb. 1-12: Es befinden sich zwei Hindernisse mit einem Mindestabstand von einem Meter versetzt auf der linken und rechten Fahrspur

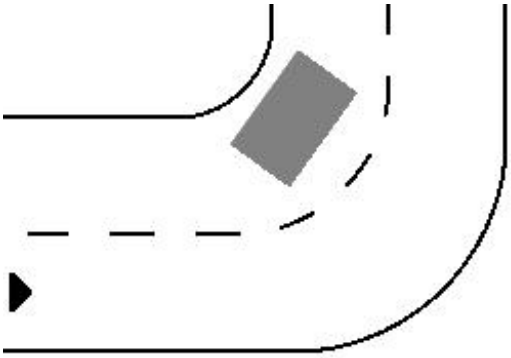

Abb. 1-13: Ein Hindernis befindet sich innerhalb einer Linkskurve auf der linken Fahrspur

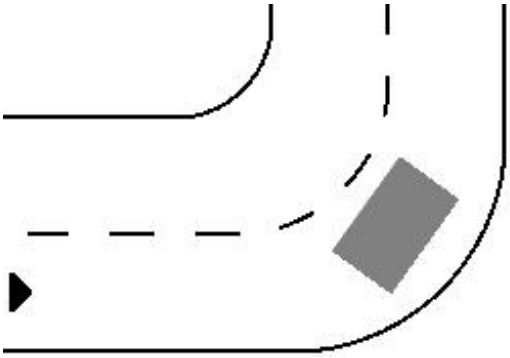

Abb. 1-14: Ein Hindernis befindet sich innerhalb einer Linkskurve auf der rechten Fahrspur

Die Fahrbahn und die Hindernisse werden über eine Kamera aufgenommen und dann mittels Bildverarbeitung erkannt (Kapitel 2). Daher folgt eine Analyse ausgewählter Kamerabilder. Zunächst werden alle relevanten Linien im Bild erläutert. Dann folgen Hindernisse auf einem geraden Streckenabschnitt und in einer Kurve.

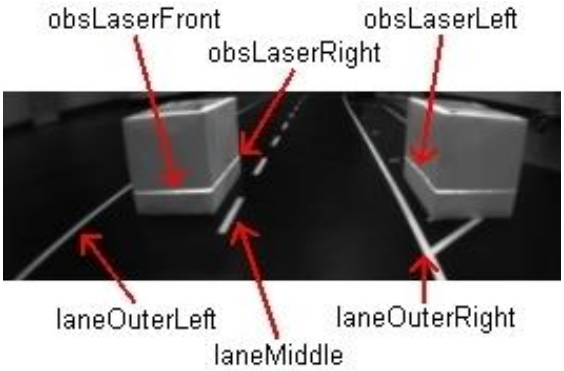

Abb. 1-15: Linienbezeichnungen im Kamerabild

In Abb. 1-15 sind die im Kamerabild relevanten Linien beschriftet, so dass in den folgenden Beschreibungen der unterschiedlichen Fälle eine eindeutige Zuweisung der Linien gegeben ist. Die Fahrbahnmarkierungen im Kamerabild werden als laneOuterLeft für die linke Außenmarkierung, laneMiddle für die Mittelstreifenmarkierung und laneOuterRight für die rechte Außenmarkierung bezeichnet. Die Laserlinien im Kamerabild werden in Fahrtrichtung als obsLaserFront für die Laserlinie auf der Vorderseite der Hindernisse, obsLaserRight für die rechte Seite des Hindernisses und obsLaserLeft für die linke Seite des Hindernisses bezeichnet. Es

stehen 50cm vor dem Fahrzeug zwei Hindernisse auf einem geraden Streckenabschnitt. Die Seitenlinien obsLaserRight und obsLaserLeft sind deutlich zu erkennen. Des Weiteren ist eine Abschwächung des Laserstrahls auf dem rechts neben der Fahrbahn stehenden Hindernis zu erkennen. ObsLaserFront wird nach rechts außen hin deutlich dunkler.

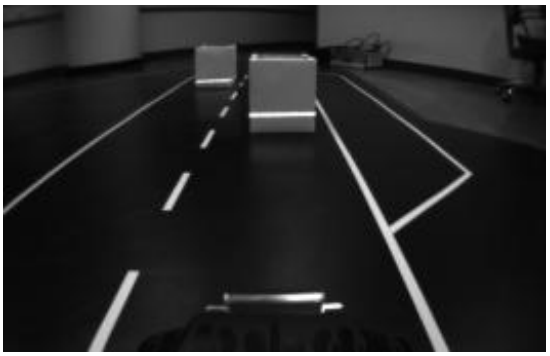

Abb. 1-16: Zwei Hindernisse in 1m und 2m Entfernung

Die in Abb. 1-16 dargestellte Szene ist eine nicht regelgerechte Stellung der Hindernisse, da lediglich deren Fronten einen Abstand von einem Meter aufweisen. Das Bild wurde zu Darstellungszwecken aufgenommen. Steht das Hindernis auf einem geraden Streckenabschnitt der eigenen Fahrspur, so ist nur obsLaserFront zu sehen. Außerdem ist im Gegensatz zu einem linken, nah stehenden Hindernis aus Abb. 1-15 an einem Hindernis, das weit hinten auf der Gegenfahrspur steht (Abb. 1-16), nur obsLaserFront zu sehen.

In den folgenden zwei Abbildungen befindet sich das Fahrzeug vor, bzw. in einer Rechtskurve. Steht das Hindernis weit entfernt im Kurvenanfang wie in Abb. 1-17, so sind obsLaserFront und obsLaserRight zu sehen. Beide Linien sind relativ deutlich zu erkennen und bilden zusammen nahezu eine Gerade.

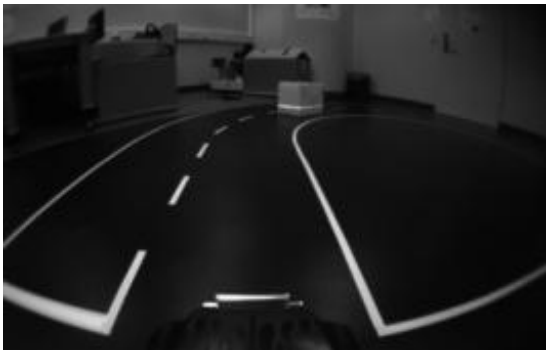

Abb. 1-17: Das Hindernis befindet sich 2m vom Fahrzeug entfernt

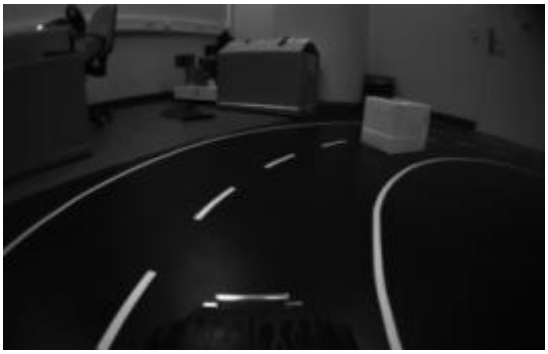

Abb. 1-18: Das Hindernis befindet sich 1m vom Fahrzeug entfernt

In Abb. 1-18 befindet sich das Fahrzeug im Kurvenanfang und das Hindernis steht nur etwa einen Meter entfernt. Die obsLaserFront Linie ist nur schwach zu erkennen. Die obsLaserRight ist nur noch vage zu ahnen und für die Bildverarbeitung nicht mehr erkennbar, da der Kontrast zum Hindernis zu gering ist. Des Weitern ist mit steigender Entfernung ein Abschwächung der Außenbogenmarkierung in Kurven (hier die laneOuterLeft) zu erkennen.

1.4 Fahrzeugbeschreibung

Abb. 1-19: Fahrzeug FAUST-Onyx

Als autonomes Fahrzeug dient das im Rahmen des FAUST-Projekts entwickelte Modellfahrzeug Onyx (Abb. 1-19). Als Basis der Fahrzeugplattform dient der allradbetriebene Pickup-Truck Ford F-350 im Modellmaßstab 1:10 von Tamiya. Die Entscheidungskriterien für dieses Modell lagen im robusten Aufbau, der hohen Zuladungsfähigkeit, sowie der Abmessungen, die anhand des CaroloCup-Regelwerkes vorgegeben waren. Weiterhin waren der Gesamtaufbau und die Möglichkeit zur Unterbringung neuer Komponenten ausschlaggebend. Zur Verbesserung der Fahrpräzision und Stabilität wurden diverse Teile, wie Radlager, Reifen und Motor ausgetauscht. Mit allen Umbaumaßnahmen hat das Fahrzeug eine Länge von 47cm (carLength) und eine Breite von 22cm (carWidth).

In der schwarzen Box, die die Ladefläche des Trucks ersetzt, ist eine IO-Box eingebaut, um die Kommunikation zwischen den Sensoren und der Recheneinheit zu regeln. Zusätzlich dient sie als Halterung für die Recheneinheit, einem Asus Aspire One (Anhang C.1). Auf ihr finden die Bildverarbeitung, sowie die Berechnung aller Fahrmanöver statt.

Für den autonomen Einsatz werden unterschiedliche Sensoren genutzt, die Informationen zu Umfeld und Fahrzeug liefern. Das Sensoren - Konzept und die eingesetzten Sensoren werden in folgenden Kapiteln vorgestellt.

1.4.1 Sensoren - Konzept

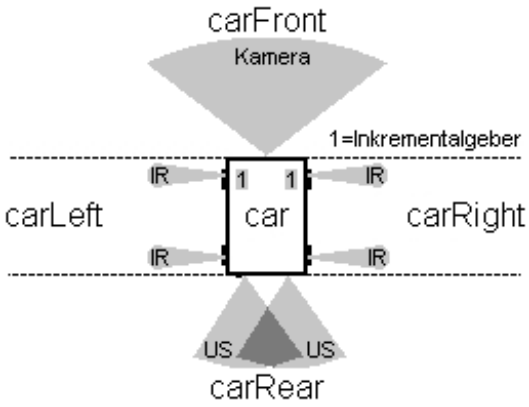

Abb. 1-20: Beobachtungsbereich für Hindernisse um das Fahrzeug

Damit das Fahrzeug seine Umgebung wahrnehmen kann, ist es mit verschiedenen Sensoren ausgestattet. Der Bereich um das Fahrzeug herum wurde in vier verschiedene Sektoren unterteilt, in denen die Aufgabe der eingesetzten Sensoren unterschiedlich ist. In Abb. 1-20 sind die vier Bereiche carFront für den Bereich vor dem Fahrzeug, sowie carLeft und carRight für den linken und rechten Seitenbereich, und carRear für den Bereich hinter dem Fahrzeug dargestellt. Die grauen Bereiche zeigen die Abdeckung der Fahrzeugumgebung durch die im Folgenden beschriebenen Sensoren auf. Eingetragen sind die Sensoren, die für den Ausweichvorgang genutzt werden. Die weißen Bereiche dazwischen sind Bereiche, die nicht von den Sensoren erfasst werden, die sog. *Blinden Bereiche*.

Im Bereich carFront sind sowohl Fahrbahn als auch Hindernisse zu erkennen. Da sich das Fahrzeug vorwärts auf einem Rundkurs bewegt, werden in diesem Bereich die Hindernisse zuerst erkannt. Die Erfassung des Bereichs vor dem Fahrzeug von mindestens 140° ist wünschenswert, um Hindernisse auch in Kurven, oder beim Spurwechsel möglichst frühzeitig zu erkennen.

In den Bereichen carLeft und carRight werden Hindernisse erst erkannt, wenn sich das Fahrzeug bereits daneben befindet. Während der Geradeausfahrt befinden sich Hindernisse die im Bereich carRight erkannt werden, rechts neben der Fahrbahn und sind nur für ein paralleles Einparken interessant. Hindernisse, die während der Geradeausfahrt im Bereich carLeft erkannt werden, befinden sich auf der linken Fahrspur oder links neben der Fahrbahn. Die hier erkannten Hindernisse beeinträchtigen die normale Fahrt nicht. Bei einem Ausweichvorgang sind beide Bereiche jedoch von hoher Relevanz. Soll einem Hindernis ausgewichen werden, das sich auf der rechten Fahrbahn befindet, muss der Bereich carLeft insoweit frei sein, dass ein Fahrspurwechsel erfolgen kann, ohne dass ein links stehendes Hindernis berührt wird. Fährt das Fahrzeug während des Ausweichmanövers auf der linken Fahrspur, wird im Bereich carRight erkannt, wann das Hindernis, dem ausgewichen werden soll, passiert wurde. Das Hindernis muss so weit entfernt sein, dass es genügend Platz für einen erneuten Fahrspurwechsel auf die rechte Fahrspur gibt, ohne dass das Hindernis berührt wird.

Der Bereich carRear ist für das Ausweichen insofern von Bedeutung, als dass bei einer eventuellen Rückwärtsfahrt die Hindernisse erkannt werden müssen, die hinter dem Fahrzeug stehen,

um diese nicht zu berühren. Es kann zu einer Rückwärtsfahrt kommen, wenn Hindernisse zu spät erkannt wurden, oder der Fahrspurwechsel nicht planmäßig funktioniert hat.

1.4.2 Eingesetzte Sensoren

In der Robotik und Automatisierung werden Objekte auf verschiedene Weisen erkannt. Es werden jedoch immer Sensoren zur Hilfe genommen, um die Umgebung zu erkennen. Sensoren können taktil ([13]) oder nicht taktil sein. Je nach Geschwindigkeit ist das Erkennen eines Hindernisses ab 70cm Entfernung notwendig. Ein so langer Fühler wäre jedoch nicht praktikabel. Für das Szenario Carolo-Cup bieten sich berührungslose Sensoren an. Zu den berührungslosen Sensoren für die Entfernungsmessung, bzw. Objekterkennung, zählen: Infrarotsensoren, Ultraschallsensoren und Bildsensoren (Kameras). Im Folgenden wird die Auswahl der genutzten Sensoren beschrieben. Im Anhang C sind die eingesetzten Sensoren zusammengefasst.

Infrarotsensoren

Zum Einsatz kommen vier GP2D12 IR-Sensoren von Sharp ([18]) mit einer Reichweite von 10-80cm und einem Erfassungswinkel von 4,5°.

Es sind jeweils zwei IR-Sensoren auf der linken und rechten Seite des Fahrzeugs montiert, um Hindernisse neben dem Fahrzeug zu erkennen. Die Sensoren werden senkrecht zum Fahrzeug ausgerichtet. Durch den kleinen Erfassungswinkel ist eine Positionsbestimmung der Hindernisse im Bezug zum Fahrzeug exakt genug. Für eine ausreichende Abdeckung der Seiten reichen zwei Sensoren aus. Ein Sensor wird an der Front angebracht (IRFrontPos = 3cm), so dass ein Hindernis erkannt wird, sobald das Fahrzeug neben diesem ist. Ein zweiter Sensor wird an das Heck angebracht (IRRearPos = 42cm), so dass erkannt werden kann, wenn das Fahrzeug ganz neben dem Hindernis ist und wann es das Hindernis vollständig passiert hat.

Die IR-Sensoren werden aus zwei Gründen nicht im Bereich carFront eingesetzt. Die Signallaufzeiten von IR-Sensoren, die im Modellbau eingesetzt werden, liegen, wie im genutzten Modell, bei ca. 40ms. Eine Aktualisierung der Signalwerte alle 40ms ist zu langsam, um genügend präzise Entfernungsangaben zu liefern. Des Weiteren besitzen IR-Sensoren einen durchschnittlichen Erfassungsbereich von 6°. Der Erfassungsbereich von den genutzten IR-Sensoren ist sogar geringer. Für eine ausreichend deckende Belegung des carFront-Bereichs müssten zu viele Sensoren eingesetzt werden, was einem Ressourcen sparendem Konzept entgegensteht.

Ultraschallsensoren

Mit Hilfe von Ultraschallsensoren (US-Sensoren), kann einem Objekt eine zentimetergenaue Entfernung zum Sensor zugewiesen werden. Aufgrund des kegelförmigen Messbereichs ist die exakte Bestimmung der Position im Erfassungswinkel jedoch nicht möglich. Zur Verbesserung der Positionsbestimmung im Erfassungswinkel könnten mehrere Sensoren mit überlappenden Erfassungswinkeln eingesetzt werden. Durch das Einsetzen von US-Sensoren treten uner-

wünschte Reflektionen auf, die zu Messfehlern führen. Je mehr US-Sensoren also eingesetzt werden, umso mehr Messfehler treten auf.

Zum Einsatz kommen vier SRF10 von Devantech ([17]) mit einem Erfassungswinkel von ca. 70°. Für das parallele Einparken werden zwei US-Sensoren jeweils vorne und hinten genutzt. Die vorderen US-Sensoren können nicht primär zur Erkennung von Hindernissen genutzt werden, da für eine Zuordnung zu den Fahrspuren eine genaue Positionsbestimmung notwendig ist. Sie könnten jedoch zur Validierung der ermittelten Entfernung genutzt werden. Die hinteren US-Sensoren können jedoch für den Bereich carRear gut genutzt werden. Bei einem Rückwärtsfahren sind in der Regel die Positionen der Hindernisse auf der Fahrbahn bekannt, so dass lediglich deren Entfernung relevant ist.

Bildsensor – Kamera

Zum Einsatz kommt eine UI-1226LE-M-GL von iDS ([7]) mit einem Weitwinkelobjektiv, so dass ein Erfassungswinkel von 140° erreicht wird. Die Kamera ist so montiert, dass sie den Bereich vor dem Fahrzeug aufnimmt. Mit Hilfe von Bildverarbeitungsmethoden, lassen sich aus dem Kamerabild Objekte extrahieren. Durch verschiedene Verfahren und Zusatzinformationen kann aus dem Kamerabild sogar auf eine zentimetergenaue Positionierung der erkannten Objekte in der realen Welt geschlossen werden.

Zur Unterstützung der Bildverarbeitung ist ein Linienlaser an der Fahrzeugfront installiert, der eine zum Boden parallele Linie ausstrahlt. Die Laserlinie ist nur sichtbar, wenn sie auf ein Hindernis projiziert wird (Abb. 1-5). Sind keine Hindernisse im Erfassungsbereich der Kamera, ist keine Laserlinie im Bild vorhanden. Bei dem Linienlaser handelt es sich um einen LH650-16-3 von PicoTronic ([16]).

Anstelle der Kombination von Kamera und Linienlaser könnte für die reale Welt ein Laserentfernungsmesser zum Einsatz kommen. Für den Einsatz bei 1:10-Modellen sind diese jedoch nicht geeignet, da sie zu groß sind.

Inkrementalgeber

Damit die Entfernungsangaben der Hindernisse in den Blinden Bereichen aktualisiert werden können, kommen zwei Inkrementalgeber (TCUT1300X01 von Vishay ([21])) im linken und rechten Vorderrad des Fahrzeugs zum Einsatz. Da in Kurven die inneren und äußeren Räder unterschiedlich viele Radumdrehungen zurücklegen, müssen für eine korrekte Ermittlung der gefahrenen Distanz die Werte aus dem rechten und linken Rad gemittelt werden.

Die berechnete Distanz wird für diese Arbeit als eine geradeaus gefahrene Strecke interpretiert. Da das Fahrzeug selbst bei einer geraden Strecke nicht in einer Geraden, sondern in einer leichten Schlängelbewegung fährt, sind die Entfernungsbestimmungen fehlerhaft. Da jedoch die Gebiete, in denen kein Sensor die Entfernungsangaben korrigieren kann, nicht groß sind (Abb. 1-20), kann dieser Fehler vernachlässigt werden.

1.5 Vorstellung genutzter Module

Für das Ausweichverfahren werden zwei bestehende Module genutzt. Die Fahrspurerkennung POLARIS (Kapitel 1.5.1) liefert die Fahrbahndaten und weitere Mechanismen, die für die Positionierung der Hindernisse zur Fahrbahn notwendig sind. Die Fahrspurführung (Kapitel 1.5.2) wird während des Ausweichens für die Fahrzeugführung ausgenutzt.

1.5.1 Fahrspurerkennung – POLARIS

Die Fahrspurerkennung POLARIS, die im Rahmen der Masterarbeit von E. Jenning ([8] Kapitel 3) entwickelt wurde, ermittelt für eine der beiden Fahrspuren zwei Polynome zur Abbildung der Fahrspurmarkierungen. 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.

Die Fahrbahnmarkierungen werden im Kamerabild, das den Bereich vor dem Fahrzeug abbildet, gesucht. Bedingt durch die Krümmung der Kameralinse treten bei handelsüblichen Kameras Linsenverzeichnungen auf, wodurch gerade Strecken als Kurve dargestellt werden (Abb. 1-21). Damit möglichst viel von der Umgebung aufgenommen werden kann, wird eine Kamera mit Weitwinkelobjektiv (Kapitel 1.4.1) eingesetzt. Gerade bei Weitwinkelkameras kommt es zu einer deutlichen Linsenverzerrung. Mit Hilfe einer Linsenverzeichnungskorrektur kann dies behoben werden (Abb. 1-22).

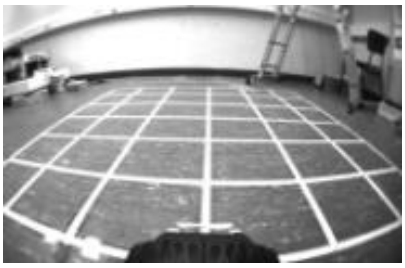

Abb. 1-21: Linsenverzeichnetes Quellbild

Abb. 1-22: Resultat der Linsenverzeichnungskorrektur. Der rote Rahmen entspricht der gleichen Größe des Bildausschnitts wie in Abb. 1-21

Durch die Kenntnis der internen Kameraparameter, kann das verzerrte Bild entzerrt werden, so dass gerade Linien wieder gerade abgebildet sind. Als interne Parameter bezeichnet man die Bildkoordinaten des Bildhauptpunktes, die Brennweite und die Parameter der Linsenverzeichnung. Die Parameter ändern sich je nach Fokus- oder Zoomeinstellung. Die genutzte Kamera

besitzt jedoch kein Zoom und wird so befestigt, dass der Fokus nicht unabsichtlich verändert wird. Somit reicht eine einmalige Bestimmung der Parameter aus. Dies geschieht mit Hilfe der photogrammetrischen Vermessungssoftware Photomodeler ([5]) und wurde von E. Jenning im Rahmen seiner Masterarbeit durchgeführt. Da keine erneute Kalibrierung der internen Parameter vorgenommen werden muss, wird das Verfahrend zur Bestimmung der internen Kameraparameter hier nicht weiter beschrieben.

In Abb. 1-22 sind schwarze Linien im Bild zu erkennen. Dieses Linienmuster entsteht durch das Verfahren der Linsenverzeichnungskorrektur, wobei aus dem Quellbild das Zielbild erstellt wird. Für die schwarzen Bildstellen im Zielbild existiert kein Quellbildpunkt der die Zielbildkoordinate abbilden kann. Die nicht definierten Pixel sind aufgrund der Voreinstellung schwarz gewählt, so dass sie nicht als Fahrbahnmarkierung oder Laserlinie fehlinterpretiert werden.

Da die Linsenverzeichnungskorrektur für das gesamte Bild zu zeitaufwändig ist, erfolgen die Bildverarbeitungsalgorithmen auf den verzeichneten Kamerabildern. Nur ausgewählte Pixel werden korrigiert. Die Suche geschieht über einige wenige horizontale Betrachtungsbereiche, die anhand der Fahrspurdaten aus dem vorherigen Zyklus platziert werden. Jeder Betrachtungsbereich extrahiert maximal einen Punkt, der die Lage der Fahrspurmarkierung im Bild beschreibt. Das Kamerabild im Betrachtungsbereich wird durch eine Faltung mit dem Sobel-Operator nach Kanten gefiltert. Einer Kante im Bild entsteht, wenn sich zwei benachbarte Pixel in ihrem Wert stark unterscheiden. Die weißen Fahrbahnmarkierungen auf dem schwarzen Fahrbahnuntergrund bilden einen starken Unterschied der Pixelwerte. Im gefilterten Betrachtungsbereich wird ausgehend vom Mittelpunkt simultan in beide Richtungen zu den Rändern hin durch die Pixel iteriert und das erste helle Pixel, welches eine Kante markiert, als Fahrspur selektiert.

Die Bildkoordinaten der extrahierten Punkte werden anschließend durch die Linsenverzeichnungskorrektur in verzeichnungsfreie Bildkoordinaten umgewandelt. Die Bildkoordinaten werden zur weiteren Verwendung in Fahrzeugkoordinaten (Kapitel 1.6.1) umgewandelt. Dies geschieht mit Hilfe einer projektiven Transformation. Dabei werden die Bildebene und die Fahrbahnebene aufeinander abgebildet. Für die projektive Transformation werden die kartesischen Koordinaten des Fahrzeugkoordinatensystems ($\vec{x} = (x_k \quad y_k)^T$) und der Bildkoordinaten ($\overrightarrow{x^*} = (x_k^* \quad y_k^*)^T$) in homogene Koordinaten umgewandelt ($\overrightarrow{x_h} = (x_h \quad y_h \quad 1)^T$ und $\overrightarrow{x_h^*} = (x_h^* \quad y_h^* \quad 1)^T$). Mit Hilfe einer Transformationsmatrix B werden nun auf den Bildkoordinaten die Fahrzeugkoordinaten berechnet. Für die Berechnung gilt die Gleichung 3.25 aus [8]: $\overrightarrow{x_h} = B\overrightarrow{x_h^*}$. Aus dieser Gleichung lassen sich $x_k = \frac{x_h}{1}$ und $y_k = \frac{y_h}{1}$ berechnen. Die Transformationsmatrix B wird durch eine einmalige Kalibrierung berechnet. Da auch eine Transformation von Fahrzeugkoordinaten in Bildkoordinaten notwendig ist, muss für diese Richtung die Transformationsmatrix durch eine einmalige Kalibrierung berechnet werden. Die Berechnung der beiden Transformationsmatrizen erfolgt anhand eines linearen Ausgleichssystems. Für das lineare Ausgleichssystem werden korrespondierende Punkte im linsenverzeichnungsfreien Kame-

rabild und genau bekannte Punkten im Fahrzeugkoordinatensystem ermittelt (in Kapitel 1.6.3 ist das Kalibrierverfahren für die Laserlinie angepasst).

Sobald die Punkte im Fahrzeugkoordinatensystem vorliegen, werden die linken und rechten Fahrspurpolynome durch eine Ausgleichsrechnung approximiert. Diese Polynome werden in der Datenstruktur PolarisLane abgespeichert und anderen Modulen zur Verfügung gestellt.

Die Fahrspurerkennung ist ein elementarer Bestandteil für die Kartenerstellung (Kapitel 3). Daher wird im Folgenden auf die Stabilität des Verfahrens eingegangen. Während des gesamten Verarbeitungsablaufs zur Polynombestimmung treten an verschiedenen Stellen Messfehler auf.

Bereits bei der Kalibrierung der internen Kameraparameter für die Linsenverzeichnungskorrektur ([8] Kapitel 5.2.1) treten Messfehler aufgrund eines nicht optimal abgestimmten Kalibriermusters für ein Weitwinkelobjektiv auf. Das Muster bedeckt nicht die gesamte Breite des Kamerabildes, so dass gerade die stark verzeichneten Bildecken nicht korrekt aufgenommen werden. Hieraus ergibt sich, dass die Kameraparameter nicht korrekt bestimmt werden. Aus dem hieraus resultierenden Fehler ergibt sich eine Differenz von 1-2cm nah dem Fahrzeug.

Für die Kalibrierung der Parameter der Transformationsmatrix für die projektive Transformation ist es wichtig, dass die Kalibrieranordnung exakt ist. Ein nicht korrekt platziertes Fahrzeug vor dem Kalibriermuster hat zur Auswirkung, dass die bekannten Entfernungen (in Fahrzeugkoordinaten) nicht korrekt sind. Zudem sind die korrespondierenden Bildpunkte nicht immer genau einem Pixel zuzuweisen. (Mehr zu den Messfehlern der Kalibrierung der Transformationsmatrix steht in Kapitel 1.6.3)

Während der Fahrt kann es durch Beschleunigung und Bremsen des Fahrzeugs zu einer Neigung kommen, so dass die Ebenen des Kamerabildes und der Fahrbahn ein anderes Verhältnis zueinander haben. Bei einer maximalen Neigung nach vorne, bei der immer noch alle vier Reifen die Fahrbahn berühren, kommt es zu einer Differenz von 10 Pixeln, was eine Differenz von bis zu 30cm bedeutet.

Abb. 1-23: Vergrößerte Bildausschnitte der Fahrbahnmarkierung. Oben: Gerade. Mitte: unscharfer Kurvenabschnitt in ca. 30cm Entfernung. Unten: äußere Kurvenmarkierung in ca. 1,5m Entfernung

Zur Identifikation der Fahrspurmarkierung werden horizontale Betrachtungsbereiche eingesetzt, in denen Kanten gesucht werden. Bei der Analyse des Kamerabild (Abb. 1-23), wird deutlich, dass die Fahrspurmarkierungen in Kurven in Fahrzeugnähe unscharf abgebildet sind. Ein unscharfes Bild hat zur Folge, dass die Kanten im Bild nicht deutlich ausgeprägt sind. Zudem sind die Betrachtungsbereiche auch in Kurven horizontal, welches zu einer Verstärkung des Unschärfeeffekts führt (Abb. 1-23 unten).

Da zur Identifikation der Fahrspurmarkierung lediglich eine Kante gesucht wird und nicht beide Kanten der Fahrspurmarkierung, kann es vorkommen, dass der Betrachtungsbereich so ungünstig gesetzt ist, dass ein nah an der Fahrbahnmarkierung stehendes Hindernis als Fahrspur fehlinterpretiert wird. Da die Farbe der Hindernisse weiß ist, entstehen an deren Rändern ebenfalls Kanten.

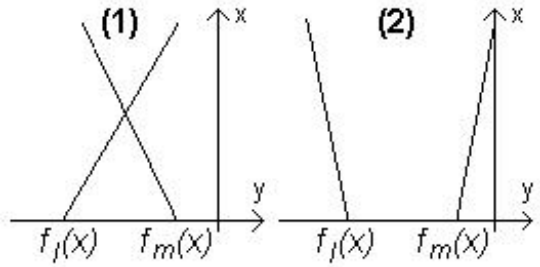

Abb. 1-24: Schematische Darstellung von fehlerhaft berechneten Fahrspurpolynomen eines geraden Streckenabschnitts

Die Polynome sind approximiert und bilden nicht exakt die Fahrbahnkurve ab. In Abb. 1-24 sind die Auswirkungen auf einem geraden, konstant breiten Streckenabschnitt schematisch dargestellt. Es wird die linke Fahrspur überwacht, während sich das Fahrzeug auf der rechten befindet. Die linke Außenmarkierung wird durch das Polynom $f_l(x)$ approximiert und die Mittelstreifenmarkierung durch das Polynom $f_m(x)$. Diese Auswirkungen gelten im Wesentlichen auch für Kurvenabschnitte. In der Teilgrafik 1 sind die Steigungen entgegengesetzt, so dass sich die beiden Polynome schneiden. Die Fahrspurbreite, der Abstand zwischen den beiden Polynomen $f_l(x)$ und $f_m(x)$, verringert sich bis zum Schnittpunkt. Nach dem Schnittpunkt verbreitert sich die Fahrspur stetig. In der Teilgrafik 2 sind die Steigungen entgegengesetzt, so dass sich der Abstand der beiden Polynome stetig vergrößert. Wird aus den gegebenen Polynomen ein Mittelpunkt der Fahrspur berechnet, gilt in beiden Fällen, dass der Mittelpunkt nicht in der tatsächlichen Mitte der Fahrspur liegt, da die Steigungen in der Regel unterschiedliche Werte haben. Eine Berechnung der dritten unbekannten Fahrbahnmarkierung führt, je nach dem welches Polynom zur Berechnung herangezogen wurde, zu komplett unterschiedlichen Resultaten.

Der Kommunikation zwischen POLARIS und den anderen Modulen stehen zwei Datenstrukturen zur Verfügung, die PolarisLane und die PolarisControlData. In der PolarisLane stellt POLARIS Informationen über die Fahrspuren bereit. Die PolarisControlData dient dazu, POLARIS zu zwingen die linke oder rechte Fahrspur zu beobachten, sowie ein Reset durchzuführen. Die beiden Klassen werden im Anhang B beschrieben.

1.5.2 Fahrspurführung

Auf den hindernisfreien Fahrbahnabschnitten fährt das Fahrzeug mit der Fahrspurführung, die im Rahmen der Bacherlorarbeit von I. Nikolov ([15]) entwickelt wurde. Die Fahrspurführung basiert auf dem „Pure Pursiut“-Verfahren und hat die Aufgabe, das Fahrzeug in der Fahrspur zu halten. Dieses Verfahren läuft stabil (vgl. [15] Abschnitt 4.3.5). Bei einem Ausweichen muss das Fahrzeug die Fahrspur wechseln. Ein Fahrspurwechsel steht jedoch der Aufgabe der Fahrspurführung, des halten der Fahrspur, entgegen. Durch einen Trick, auf den im Folgenden ein-

gegangen wird, ist der Fahrspurwechsel trotzdem möglich, ohne eine weitere Fahrzeugführung zu entwickeln.

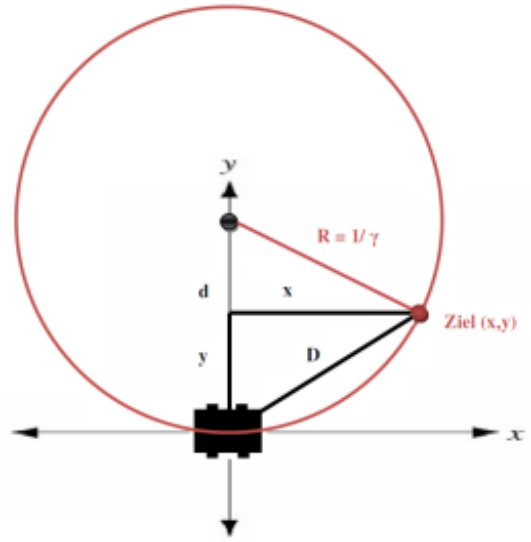

Abb. 1-25: Pure Pursuit ([15] Abbildung 9)

Für die Fahrzeugführung wird ausgehend von der momentanen Fahrzeugposition und einem vor dem Fahrzeug liegender Zielpunkt ein Kreisbogen berechnet, der als Grundlage für den einzustellenden Lenkwinkel dient (Abb. 1-25).

Die Fahrspur wird durch die beiden Fahrspurpolynome von POLARIS (Kapitel 1.5.1) angegeben. Der Zielpunkt wird so gewählt, dass er sich in der Mitte der Fahrspur, also in der Mitte der beiden Fahrspurpolynome, befindet.

Die Distanz des Zielpunktes (in Abb. 1-25 mit D gekennzeichnet) wird so gewählt, dass die Fahrt möglichst stabil ist. Dies bedeutet, dass auf geraden Strecken eine größere Distanz zum Fahrzeug gewählt wird. In Kurven ist die Distanz geringer, so dass der Zielpunkt noch innerhalb der Fahrspur liegt.

Ein Fahrspurwechsel auf die linke Fahrspur wird dadurch erzwungen, dass die beiden Fahrpurpolynome von POLARIS anstelle der rechten Fahrspur die linke Fahrspur darstellen. Der Zielpunkt für die Fahrzeugführung wird wie gewohnt zwischen den beiden Polynomen festgelegt. Durch das Verschieben der beiden Polynome ist die Fahrzeugposition in Bezug zu diesen verändert. Der IST-Wert der Fahrspurführung wird somit sprungartig verändert, so dass ein starker Lenkwinkel eingeschlagen wird und das Fahrzeug die Fahrspur wechselt.

Da im Regler jedoch Sicherheitsmechanismen greifen, wenn die Polynome fehlerhafte Daten liefern (erkennbar etwa durch einen plötzlichen Sprung), muss dem Regler mitgeteilt werden, dass die linke Fahrspur befahren werden soll. Diese Information muss für die gesamte Fahrt auf der linken Fahrspur übermittelt werden. Neben dem Abschalten des Sicherheitsmechanismus, wird die Entfernung des Zielpunktes verringert, so dass ein enges Ausweichen möglich ist. Für den Wechsel zurück auf die rechte Fahrspur wird POLARIS angewiesen die rechte Fahrspur zu überwachen.

Die Fahrzeugführung stellt bei der Kommunikation eine SteeringControlData-Datenstruktur zur Verfügung, die über den DataContainer der Fahrspurführung bereitgestellt werden kann. Sie enthält Informationen über die Distanz zum Zielpunkt und über die Orientierung innerhalb der linken Fahrspur.

1.6 Internes Weltbild

Die im Kapitel 1.4.2 vorgestellten Sensoren liefern sowohl Entfernungsdaten in Zentimetern (US-Sensor und IR-Sensor), als auch Positionsdaten in Pixeln (Kamera und deren Bildverarbeitung). Für einen Ausweichvorgang müssen alle Sensordaten in ein einheitliches Modell überführt und normiert werden. Aus diesem Grunde sind hier die verschiedenen Koordinatensysteme und deren Transformation zueinander beschrieben.

1.6.1 Koordinatensysteme

Ein Koordinatensystem dient der Bezeichnung der genauen Position von Punkten in einem geometrischen Raum. In einer Ebene wird zunächst ein rechtwinkliges Koordinatensystem festgelegt. Die horizontale Achse wird als x-Achse, und die vertikale als y-Achse bezeichnet. Jeder Punkt kann eindeutig mit Hilfe seiner x- und y-Koordinaten, also über seine Abstände in Richtung der beiden Achsen, beschrieben werden. Die Abstände werden dabei in der mit dem Koordinatensystem festgelegten Maßeinheit gemessen. Der Nullpunkt, bei dem alle Koordinaten den Wert Null annehmen, nennt man den Koordinatenursprung. Die Reihenfolge der beiden Koordinaten ist von wesentlicher Bedeutung: (2, 7) bezeichnet einen anderen Punkt als (7, 2) (nach [11]).

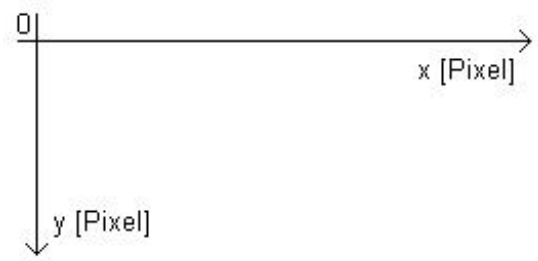

Abb. 1-26: Bildkoordinatensystem

Über die Bildverarbeitungsmethoden, die aus den Kameradaten Informationen über die Hindernisse im Bereich carFront (Abb. 1-20) ermitteln, werden die Informationen in Pixeln angegeben. Diese sind in einem Bildkoordinatensystem angeordnet. Das Bildkoordinatensystem (Abb. 1-26) hat seinen Ursprung in der linken oberen Ecke. Die y-Achse gibt die Anzahl der Bildreihen an und entspricht der Sichtweite des Fahrzeugs in Fahrtrichtung. Für die ebene Fahrbahn ohne Hindernisse gilt: je niedriger der y-Wert eines Bildpunktes ist, desto weiter weg liegt der abgebildete Punkt in der Umgebung. Die x-Achse gibt die Anzahl der Bildspalten an und entspricht der horizontalen Sichtweite des Fahrzeugs. Die Mitte der x-Achse entspricht der Fahrzeuglängsachse. Besitzt ein Bildpunkt einen kleinen x-Wert, liegt der abgebildete Weltpunkt links vom Fahrzeug. Besitzt der Bildpunkt einen hohen x-Wert, liegt der abgebildete Weltpunkt rechts vom Fahrzeug. Die beiden Achsen haben die Maßeinheit Pixel.

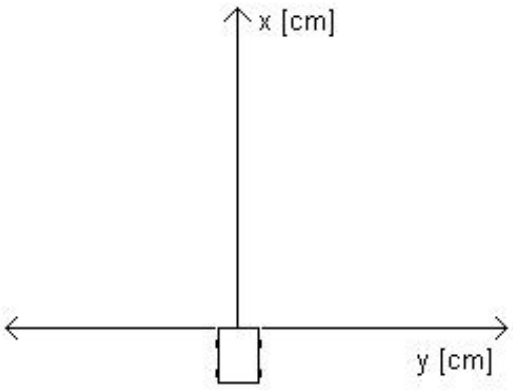

Abb. 1-27: Fahrzeugkoordinatensystem bei dem die x-Achse die Fahrtrichtung in cm darstellt.

Die Fahrbahninformation, mit denen ermittelt wird, auf welcher Fahrspur sich ein Hindernis befindet, liegt in Fahrzeugkoordinaten vor ([8], Abschnitt 2.2). Als Ursprung für das Fahrzeugkoordinatensystem (Abb. 1-27) wurde die Mitte der vorderen Stoßstange definiert. Das Fahrzeugkoordinatensystem ist starr mit dem Fahrzeug verbunden und wird gemeinsam mit ihm bewegt. Es wird die aktuelle Sicht des Fahrzeugs abgebildet. Die x-Achse gibt die Sichtweite des Fahrzeugs wieder und entspricht der Fahrzeuglängsachse. Zu beachten ist hierbei, dass die x-Achse die senkrechte und die y-Achse die waagerechte Achse ist. Dies hat den Hintergrund, dass eine Kurve aus Sicht des Fahrzeugs vertikal verläuft und die mathematische Abbildung der Kurve durch eine Funktion der Form $f(x) = y$ dargestellt werden kann. Die Maßeinheit dieses Koordinatensystems ist Zentimeter.

Eine zuverlässige Entfernungsbestimmung ist bei derzeitigem Stand nur bis zu einer Entfernung von 150cm möglich. Diese Entfernung wird als WDMax definiert. Sie ergibt sich aus der Stabilität von POLARIS (Kapitel 1.5.1).

Die US- und IR-Sensoren liefern ihre Daten in Zentimetern und können anhand ihrer festen Position am Fahrzeug direkt in das Fahrzeugkoordinatensystem übertragen werden.

1.6.2 Transformation Bild-/ Fahrzeugkoordinaten

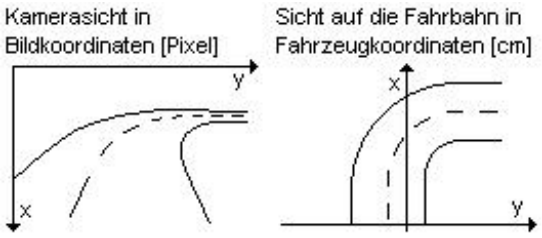

Abb. 1-28: Darstellung einer Kurve in Bild- und Fahrzeugkoordinaten

Um die verschiedenen Sensordatenwerte vergleichen zu können, müssen die beiden Koordinatensysteme in Übereinkunft gebracht werden. Dies wird durch eine projektive Transformation erreicht. Durch die projektive Transformation wird aus der Kamerasicht eine Sicht auf die Fahrbahn (Abb. 1-28), wobei aus dem Bildkoordinatensystem mit der Maßeinheit Pixel ein Fahrzeugkoordinatensystem mit der Maßeinheit Zentimeter wird. Jedem Bildpunkt kann eine genaue Entfernungsangabe in Fahrtrichtung und Positionierung quer dazu zugeordnet werden. Andersherum gilt dies auch: Jedem Punkt auf der Fahrbahn (im Sichtbereich der Kamera) kann ein Bildpunkt zugeordnet werden.

Die für POLARIS (Kapitel 1.5.1) entwickelte projektive Transformation kann nicht direkt übernommen werden. Sie wird aber für die Positionsbestimmung der Hindernisse adaptiert. Bei der projektiven Transformation können Ebenen aufeinander abgebildet werden. Die Umgebung in der sich das Fahrzeug bewegt, ist unsere dreidimensionale Welt. Da die Fahrbahn sich auf einer Ebene befindet, und die Fahrbahnmarkierungen direkt auf dem Boden angebracht sind, kann zur Vereinfachung die Welt auf eine 2D- Umgebung reduziert werden. Hierfür werden die Informationen zur Höhe entfernt. Bei der projektiven Transformation handelt es sich somit um eine bijektive Abbildung $\mathcal{W}$ von einer 2D- Weltsicht, der Weltebene W, in eine 2D- Bildsicht, der Bildebene I:

$\mathcal{W}: W \leftrightarrow I$, wobei $W \subset \mathbb{R}^2$ und $I \subset \mathbb{R}^2$

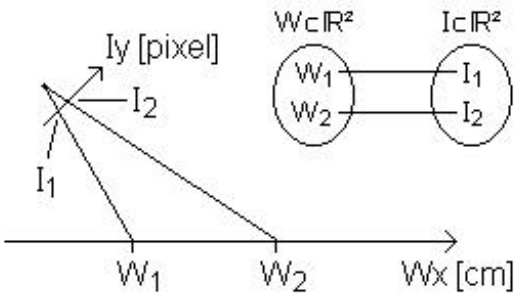

Abb. 1-29: Bijektive Abbildung von Welt- in Bildkoordinaten

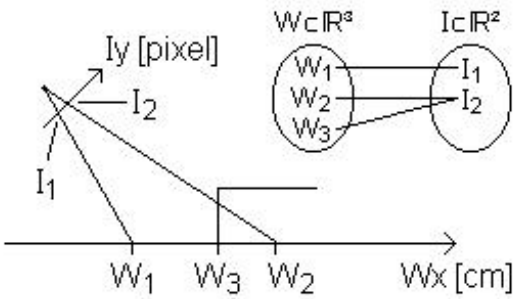

Abb. 1-30: Surjektive Abbildung von Welt- in Bildkoordinaten

Wie Abb. 1-29 verdeutlicht, gilt: $I_1 = \mathcal{W}(W_1)$ und $I_2 = \mathcal{W}(W_2)$. Aufgrund der Bijektivität, kann aus der Bildebene zurück auf die Weltebene geschlossen werden. Da die Fahrbahnmarkierungen, und für unser Verfahren die Hindernisse, im Bild gesucht werden, ist der Rückschluss vom Bild in die Welt nötig. Die Richtung Welt zu Bild wird benötigt, um die Suchbereiche im Bild einzuschränken.

Durch die Hindernisse wird die 2D-Weltebene um eine dritte Dimension, der Höhe der Hindernisse, erweitert (Abb. 1-30). Dadurch wird mehreren Weltpunkten ein Bildpunkt zugeordnet, somit ist die neue Abbildung $\mathcal{W}'$ nicht mehr bijektiv.

$\mathcal{W}': W \rightarrow I$, wobei $W \subset \mathbb{R}^3$ und $I \subset \mathbb{R}^2$

Aus Abb. 1-30 wird deutlich dass $I_2 = \mathcal{W}'(W_2) = \mathcal{W}'(W_3)$ gilt. Somit kann von I2 nicht mehr nur auf einen Weltpunkt geschlossen werden. Da die Positionierungszuordnung der Hindernisse jedoch aus den Bilddaten erfolgen soll, muss hier eine Lösung gefunden werden.

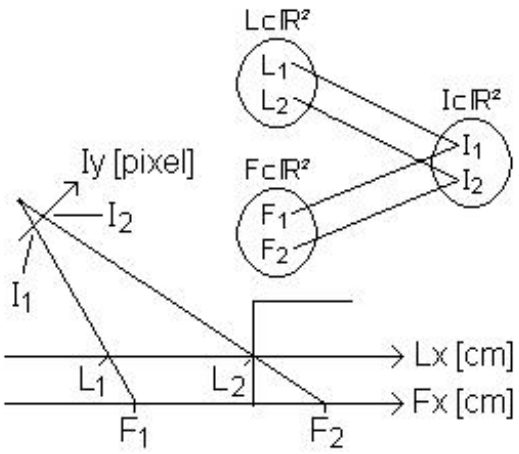

Abb. 1-31: Einführung von zwei Ebenen

Um das Problem zu lösen, wird ein Linienlaser als Hilfsmittel hinzugenommen. Der Laser spannt eine weitere Ebene in der Welt auf, die parallel zur Fahrbahnebene liegt (Abb. 1-31).

Fahrbahnebene:
$F \subset W$, mit $F \subset \mathbb{R}^2$ und $W \subset \mathbb{R}^2$

Laserebene:
$L \subset W$, mit $L \subset \mathbb{R}^2$ und $W \subset \mathbb{R}^2$

So können zwei bijektive Abbildungen beschrieben werden:

$\mathcal{F}: F \leftrightarrow I$
$\mathcal{L}: L \leftrightarrow I$

Da nicht die gesamte Fahrbahnebene von Interesse ist, sondern nur die Fahrbahnmarkierungen M, gilt: $M \subset F$. Da der Laser parallel zum Boden angebracht wird, existiert zwar eine Ebene der Laserlinie, jedoch ist diese nur sichtbar, wenn ein Hindernis existiert. Für die Hindernisse O gilt folglich: $O \subset L$. Für das Bild ergibt sich folgende Zuordnung:

$$I \supset \mathcal{F}(M) \cup \mathcal{L}(O) = I'$$

Zur Unterscheidung, ob es sich bei einem Punkt aus I' um einen Punkt aus $\mathcal{F}(M)$ oder $\mathcal{L}(O)$ handelt, wird ein Abgleich der Pixelwerte durchgeführt (Kapitel 2).

1.6.3 Kalibrierung der Laserlinie

Für die projektive Transformation sind einige bekannte Wertepaare von Welt- und Bildpunkten notwendig, um für jeden Bildpunkt einen Weltpunkt (und umgekehrt) zu errechnen. Da der Laser jedoch eine höher liegende Ebene als die Fahrbahnebene aufspannt, muss diese neue Ebene kalibriert werden. Solange die genaue Höhe und exakte Ausrichtung der Kamera und des Lasers nicht verstellt werden, reicht eine einmalige Kalibrierung aus. Das Verfahren zur Kalibrierung von E. Jenning ([8] Abschnitt 3.3) kann prinzipiell übernommen werden, muss aber für die hinzukommende Ebene angepasst werden.

Für die Kalibrierung der Fahrbahnebene ist ein Muster angelegt worden, bei dem einige markante Stellen genau ausgemessen wurden (Abb. 1-32). Die markanten Stellen sind die Kreuzungen der weißen Linien. Sie sind von rechts oben bis links unten durchnummeriert. Das Fahrzeug wird so vor dem Muster positioniert, dass die mittlere Rasterlinie (Nummer 4) eine Verlängerung der Mittelachse des Fahrzeugs darstellt und die Stoßstange die erste Markie-

rungslinie berührt. Von diesem Punkt an sind markante Stellen auf einen Zentimeter genau bekannt.

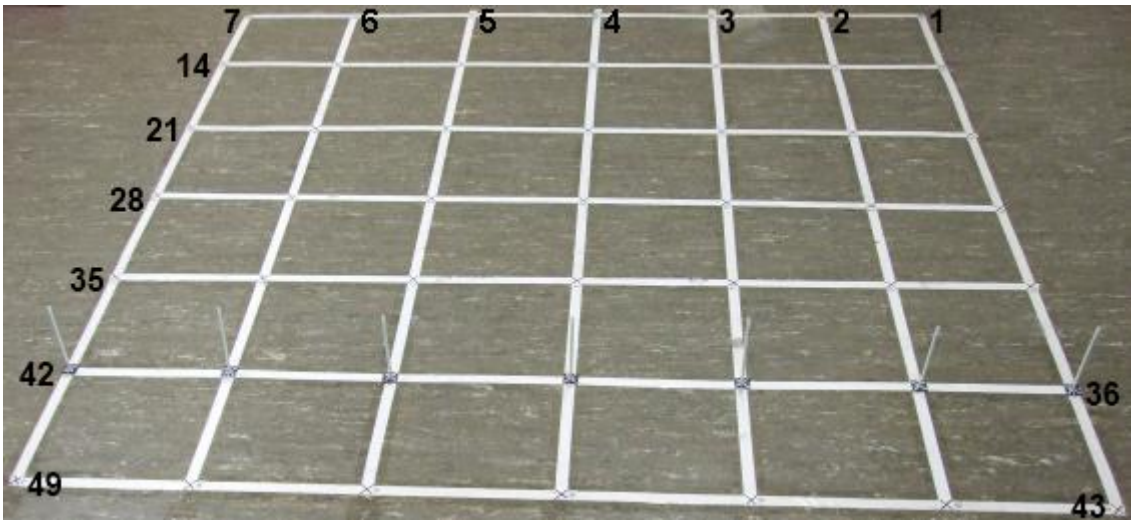

Abb. 1-32: Kalibriermuster

Für die Kalibrierung des Fußbodens ist lediglich ein Foto aus der genau vorgeschrieben Position notwendig. Damit die Ebene der Laserlinie fotografiert werden kann, wurden Stelzen angefertigt (in der Abb. 1-32 in der Reihe 36 - 42). Würde man alle Kreuzungen belegen, würden sich die Stelzen gegenseitig verdecken. Daher muss für jede Kreuzung ein Foto angefertigt werden. Die Stelzen werden so auf die Kreuzungen platziert, dass ihr vorderer Mittelpunkt auf der Mitte der Kreuzung liegt. Aus diesen Fotos muss nun per Hand der Pixelpunkt herausgeschrieben werden, an dem die Laserlinie die Stelze trifft. In einer Tabelle muss per Hand der Pixelwert dem entsprechenden Kreuzungswert zugewiesen werden.

Bei der Kalibrierung kann es leicht zu Messfehlern kommen, wenn das Fahrzeug nicht korrekt vor das Kalibriermuster gestellt, sondern z.B. 10cm vor das Muster. Hierbei würden die berechneten Entfernungsangaben nicht mehr stimmen. Auch eine Schrägstellung des Fahrzeugs vor dem Kalibriermuster führt zu Fehlern, da dann vor allem die hinteren markanten Stellen (Nummer 1-7, Abb. 1-32) deutlich versetzt sind. Das Fahrzeug an exakt dieselbe Stelle und derselben Ausrichtung zu positionieren, wie es für die Bestimmung der markanten Stellen positioniert war, ist nahezu unmöglich. Daher liefern die Berechnungen immer fehlerhafte Ergebnisse, so dass die Entfernungsbestimmungen im relevanten Bildbereich nur zentimetergenau sind. Damit der Vergleich zwischen den Hindernissen und den Fahrspuren möglichst genau ist, sollten beide Ebenen zusammen kalibriert werden, ohne dass das Fahrzeug hierbei verstellt wird. Im schlimmsten Fall wird das Fahrzeug schräg gesetzt, so dass sich die Fehler aufaddieren.

Beim Ablesen der markanten Stellen im Kamerabild kann es zu weiteren Messfehlern kommen. Die Kreuzungsmitte ist nicht immer einem Bildpixel zuzuordnen, sondern nur in einem Bereich von 2x2 Pixeln abgebildet. Je nach Wahl des Pixels gehen hierbei einige Millimeter verloren. Zudem wird die Laserlinie im vorderen Mittelbereich breit auf die Stelzen projiziert und verläuft somit über mehrere Pixelzeilen.

2 Hindernisse erkennen durch Bildverarbeitung

Ein elementarer Baustein für den Ausweichvorgang ist das Erkennen von Hindernissen vor dem Fahrzeug, was zum Entwicklungsstand des Ausweichverfahrens nicht vorhanden war. Dieses Manko zu beheben stellt die Kernanforderung dieser Arbeit dar. Der Bereich vor dem Fahrzeug wird über eine Kamera mit Weitwinkelobjektiv erfasst (Kapitel 1.4.2). Für das Erkennen der Hindernisse wurde im Rahmen dieser Arbeit ein Bildverarbeitungsalgorithmus realisiert. Das Verfahren zum Erkennen der Hindernisse kann grob in sechs Abschnitte unterteilt werden:

- Reduktion des zu untersuchenden Bildbereichs in zwei Stufen.
- Identifikation von Hindernispunkten innerhalb einer COI
- Gruppieren der gefundenen Hindernispunkte
- Ermittlung der Randpunkte im Bild
- Umrechnen der Hindernisrandpunkte in Fahrzeugkoordinaten
- Bereitstellung der Hindernisinformationen zur weiteren Verarbeitung

Für eine Veranschaulichung der einzelnen Verfahrensschritte werden in diesem Kapitel schematische Darstellungen des Kamerabildes verwendet.

Reduktion des zu untersuchenden Bildbereichs in zwei Stufen

Trotz leistungsstarker Hardware können die angewandten Bildverarbeitungsalgorithmen nicht beliebig komplex sein, da der vom FAUSTcore fest vorgegebene Systemtakt die Berechnungszeiten der einzelnen Module beschränkt (Kapitel 1.2). Durch Reduktion der Bildinformationen kann die Rechenzeit bereits vor der Wahl des Bildverarbeitungsalgorithmus verringert werden. Die Reduktion des zu untersuchenden Bildbereichs erfolgt in zwei Stufen.

Reduktion des zu untersuchenden Bildbereichs: Stufe 1

Abb. 2-1 stellt schematisch das Kamerabild in Bildkoordinaten (Kapitel 1.6.1), sowie die erste Stufe der Reduktion des zu untersuchenden Bildbereichs dar. Das Kamerabild hat eine Auflösung von IXMax*IYMax Pixeln (die aktuelle Auflösung beträgt 752x480 Pixel). Zur Reduktion der Bildinformationen, und um den Rechenaufwand möglichst gering zu halten, werden uninteressante Bereiche aus dem Kamerabild entfernt. So bleibt ein interessanter Bereich über, die sogenannten ROI (*Region of Interest*). Im unteren Bildteil ist die Motorhaube mit Stoßstange zu sehen, so dass aus diesem Bildteil keine Hindernisinformationen ermittelt werden kön-

nen. Im oberen Bildteil befindet sich der Horizont (hellgrauer Bereich), in dem keine Fahrbahninformationen mehr vorliegen. Es wird zusätzlich ein Abschnitt der Fahrbahn entfernt, da eine zuverlässige Entfernungsbestimmung nur bis zu einer Entfernung von WDMax möglich ist (Kapitel 1.6.1).

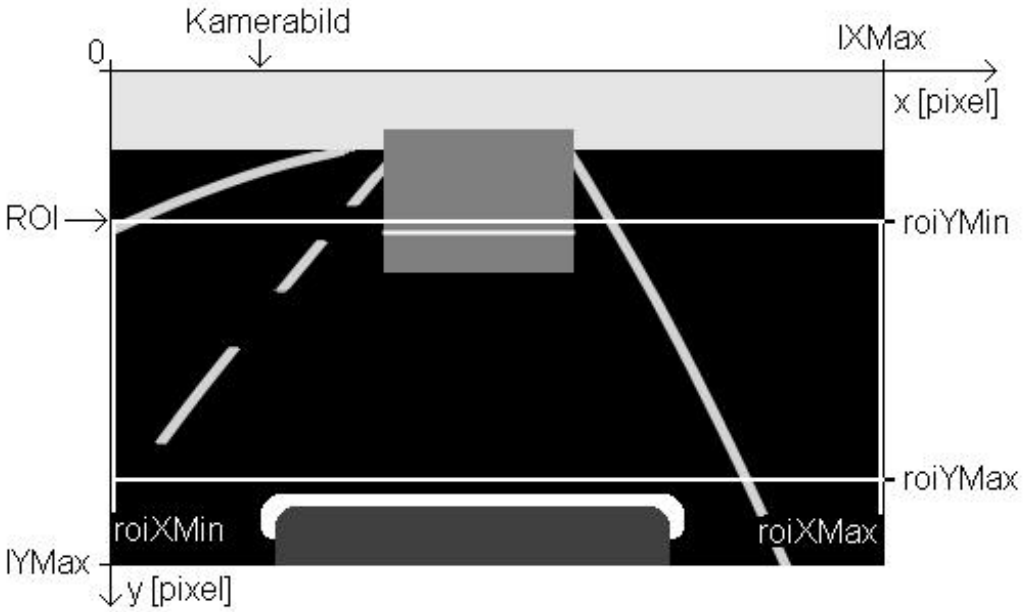

Abb. 2-1: Schematische Darstellung des Kamerabildes mit Koordinatensystem und Grenzen für die ROI

Die untere Grenze der ROI wird durch den Bildzeilenwert roiYMax vorgegeben. Dieser wird so gewählt, dass er knapp oberhalb der Stoßstange liegt. So wird der untere Bildteil, in dem die Motorhaube mit Stoßstange zu sehen ist, aus dem ROI ausgeschlossen. RoiYMax beschreibt den Mindestabstand, bei dem ein Hindernis erkannt werden kann.

Die obere Grenze der ROI wird durch den Bildzeilenwert roiYMin vorgegeben. Dieser wird durch die Angabe von WDMax über die Projektive Transformation der Laserebene (Kapitel 1.6.2) dynamisch zum Programmstart berechnet:

$$roiyMin = \mathcal{L}(WDMax)$$

Die Betrachtungsgrenzen für die x-Achse des Bildes werden durch roiXMin und roiXMax beschrieben. Bislang wird die gesamte Bildbreite untersucht, wodurch gilt:

roiXMin = 0 und roiXMax = IXMax.

Eine Einschränkung der Bildbreite ist möglich, wenn die Fahrbahninformationen hinzugezogen werden, da der Ausweichvorgang nur auf der Fahrbahn stattfinden soll. Jedoch erschweren Hindernisse die Sicht auf die Fahrbahnmarkierungen. Daher sind diese so stark verdeckt, dass

nicht zu jedem Bild korrekte Fahrbahninformationen vorliegen. Anhand dieser könnte die Einschränkung der Bildbreite. berechnet werden.

Der Betrachtungsbereich wird von 752x480 Pixel auf circa 752x260 Pixel reduziert (je nach dem, in welcher Bildzeile roiYMin liegt).

Reduktion des zu untersuchenden Bildbereichs: Stufe 2

Eine Möglichkeit, die Bilddaten weiter zu reduzieren, ist es, anstelle einer großen ROI mehrere Zeilen- oder Spalten-ROIs zu betrachten. Dies ist möglich, da die Größe der zu suchenden Objekte, die eines Pixels übersteigt.

Die Laserlinien sind in der Regel horizontale Linien im Kamerabild und bilden eine horizontale Kante. Folglich wird der Bildbereich innerhalb der ROI in mehrere Spalten-ROIs unterteilt. Die einzelne zu untersuchende Spalte wird im folgenden COI (*Column of Interest*) genannt. Abb. 2-2 zeigt die zweite Stufe zur Reduktion der zu untersuchenden Bildbereiche.

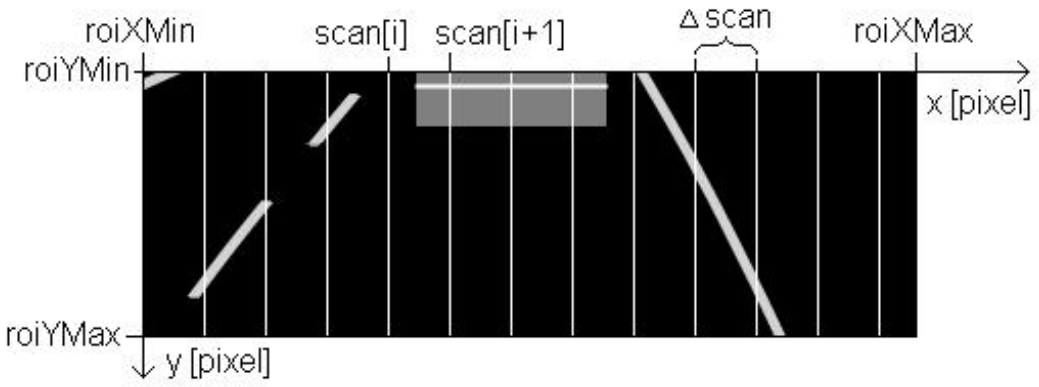

Abb. 2-2: Schematische Darstellung der ROI mit Koordinatensystem, Grenzbezeichnungen und Scanlinien

Die im Verfahren genutzten COIs haben eine Breite von einem Pixel. Das Koordinatensystem und die Einheiten bleiben gleich. Die Anzahl der COIs hängt von deren Abstand (Δscan) ab. Je weniger COIs erzeugt werden, also je größer Δscan ist, desto weniger Rechenaufwand wird benötigt. Δscan muss so gewählt werden, dass genügend COIs auf der Laserlinie des entferntesten Hindernisses auftreffen. Dies impliziert, dass die Breite des schmalsten Hindernisses (IObsWidthMin) an der entferntesten gültigen Stelle innerhalb der ROI (roiYMin) bekannt ist. Die Berechnung von Δscan findet direkt nach der Berechnung von roiYMin statt. Die Mindestanzahl an COIs, die die Laserlinie eines Hindernisses treffen müssen, wird über den Parameter, scanHits, vorgegeben. Für die Beschreibung des Verfahrens wird dieser mit drei belegt.

Für die Berechnung von IObsWidthMin werden Informationen aus dem Regelwerk des Carolo-Cups ([20]) herangezogen. Die Mindestbreite der Hindernisse beträgt laut Regelwerk 10cm. Da die Angabe aus der Welt stammt, wird eine Variable WObsWidthMin definiert. WObsWidthMin kann über einen Parameter zum Programmstart vorgegeben werden, da sich gezeigt hat,

dass im Wettbewerb die Hindernisse mindestens 20cm breit waren. Je breiter die Hindernisse sind, desto größer wird Δscan. Mit den Informationen WDMax und WObsWidthMin kann IObsWidthMin berechnet werden. Es wird angenommen, dass die linke Kante des Hindernisses auf dem Punkt *LPL = (WDMax, 0)*, und die rechte Kante auf dem Punkt *LPR = (WDMax, WObsWidthMin)* der Laserebene liegt. Wodurch für IObsWidthMin und Δscan gilt:

$$IObsWidthMin = \mathcal{L}(LPR) - \mathcal{L}(LPL)$$

$$\Delta scan = \frac{IObsWidthMin}{scanHits}$$

Zur Beschreibung der Berechnung der COIs wurde davon ausgegangen, dass die Laserlinie zu jedem Zeitpunkt als horizontale Linie im Kamerabild abgebildet wird. Bei einem Spurwechsel und in Kurven ist dies nicht der Fall. Während des Spurwechsels steht das Fahrzeug schräg auf der Fahrspur, so dass die Laserlinie schräger im Bild abgebildet ist. Wird ein Spurwechsel vollzogen, wurde das Hindernis bereits erkannt und darauf reagiert, so dass für dessen Weiterverfolgung andere Sensoren zum Einsatz kommen können. Zudem ist das Hindernis während des Fahrspurwechsels sehr nah am Fahrzeug, wodurch es größer abgebildet wird als entfernte Hindernisse. Da zur Berechnung von Δscan vom kleinstmöglichen, entferntesten Hindernis ausgegangen wird, ist das nahe Hindernis trotz der Schrägstellung groß genug, so dass genügend COIs die Laserlinie treffen.

Ist ein Hindernis in einer Kurve positioniert, steht es aus Sicht des Fahrzeugs schräg, so dass ab einer bestimmten Entfernung zwei Seiten zu sehen sind. Je näher sich das Fahrzeug auf das Hindernis zubewegt, desto gerader steht das Hindernis zum Fahrzeug. So ist immer weniger von der Hindernisseite zu erkennen. Die Laserlinien auf dem Hindernis sind hier keine horizontalen Linien mehr. Dennoch sind sie nah genug am Fahrzeug, so dass genügend COIs die Linie treffen.

Identifikation von Hindernispunkten innerhalb einer COI

Innerhalb einer COI wird die Laserlinie über ein Schwellwertverfahren ermittelt. Der Schwellwert, LBrightMin, gibt den Wert des dunkelsten Pixels, des zur Seite hin abschwächenden Laserstrahls, an, welches eindeutig der Laserlinie zugewiesen werden kann. Da dieser Wert von der Umgebungshelligkeit und den Einstellungen der Kamera abhängt, wird er über einen Parameter zum Programmstart vorgegeben. Dieser Wert muss empirisch aus Testbildern der Umgebung ermittelt werden.

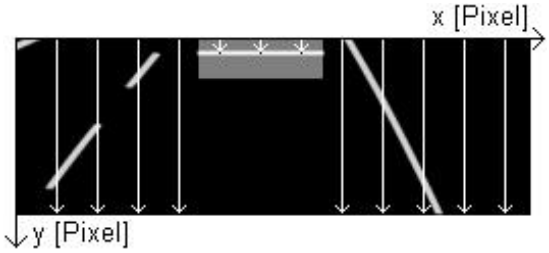

Abb. 2-3: Schematische Darstellung der Identifikation von Hindernispunkten innerhalb einer COI

Die COIs werden von links nach rechts einzeln mit dem Schwellwert verglichen. Ist der Wert eines Pixels größer oder gleich LBrightMin, so wird die Koordinate des Pixels als Laserlinie interpretiert und bildet damit ein Hindernis ab. Die Pixelkoordinaten werden in einem Vektor, scan, abgespeichert. Diese COI muss nicht weiter untersucht werden, da hintereinander stehende Hindernisse sich verdecken und die Laserlinie nur auf das vordere projiziert wird. Ist die Grenze der COI erreicht, wurde kein Pixel als Laserlinie erkannt. In diesem Fall wird eine Dummy-Koordinate (-1; -1) in scan abgespeichert. Am Ende des Verarbeitungsschrittes ist für jede COI ein Eintrag in scan enthalten. Die Größe des Vektors berechnet sich aus: roiXMax / Δscan. Das Verfahren ist in Abb. 2-3 schematisch dargestellt. Die COIs und deren Suchrichtung werden durch die weißen Pfeile angegeben. Die Pfeilspitze markiert das Ende der Suche innerhalb der COI.

Gruppierung der gefundenen Hindernispunkte

Die im Vektor scan abgespeicherten Hindernispunkte können zu verschiedenen Hindernissen gehören. Im nächsten Verarbeitungsschritt werden zusammengehörige Hindernispunkte ermittelt und auf zwei reduziert. Für jedes in scan vorhandene Hindernis wird ein rechter und linker Hindernispunkt ermittelt. Diese Punkte sind jedoch noch nicht mit Sicherheit die Randpunkte der Hindernisse.

Abb. 2-4: Schematische Darstellung der Gruppierung der gefundenen Hindernispunkte. Das Kamerabild ist zu Darstellungszwecken ausgegraut.

Um festzustellen, ob zwei Einträge zum selben Hindernis gehören, werden benachbarte Vektoreinträge untersucht. Dabei beginnt die Suche nach Pixelpunkten derselben Laserlinie mit einem Eintrag, der kein Dummy-Pixel ist und endet spätestens bei erneutem Auftreten eines Dummy-Pixels. Der Parameter, scanHits, der vom Anwender vor Programmstart festgelegt werden kann, gibt an, wie viele COIs mindestens auf eine Laserlinie treffen müssen, damit diese als ein Hindernis identifiziert wird. Wenn mindestens scanHits (hier mit drei belegt) benachbarte Vektoreinträge eine Linie bilden, wird der erste Punkt als linker Randpunkt (IOL) abgespeichert und der letzte Punkt als rechter Randpunkt (IOR) (Abb. 2-4).

Liegen in scan[i] und scan[i+1] gütige Koordinaten vor, wird der Mittelpunkt M der beiden Koordinaten berechnet. Ist der Pixelwert an der Bildkoordinate M größer oder gleich LBrightMin, so bilden die Punkte scan[i] und scan[i+1] eine Gerade. Der nächste Vektoreintrag gehört

ebenfalls zum selben Hindernis, falls der Pixelwert an der Koordinate des neuen M ebenfalls größer oder gleich LBrightMin ist. M wird aus scan[i] und scan[i+2] berechnet. Dies geschieht so lange, bis entweder der folgende Eintrag (scan[i+n]) eine Dummy-Koordinate ist, oder der Pixelwert des neuen M nicht mehr über LBrightMin liegt. Ist $(n-1) \geq scanHits$, so gilt:

IOL = scan[i]

IOR = scan[i+n-1]

Für gültige Randpunktepaare der Hindernisse wird eine Hindernisstruktur LaserObstacle erzeugt. Im LaserObstacle sind eigentlich die Fahrzeugkoordinaten der Randpunkte abgespeichert. Da diese jedoch noch nicht bekannt sind, werden zunächst die Bildkoordinaten übertragen. Die neu erzeugten Hindernisstrukturen werden in einem LaserObstaclesVektor, obstacles, abgespeichert.

Ermittlung der Randpunkte im Bild

Die mit Hilfe der COI ermittelten Randpunkte der Hindernisse sind nicht zwangsläufig die tatsächlichen Randpunkte der Hindernisse (Abb. 2-4).

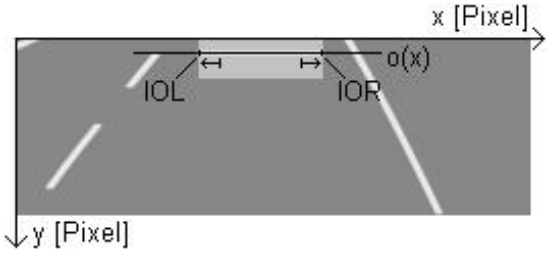

Abb. 2-5: Schematische Darstellung der Ermittlung der Randpunkte im Bild. Das Kamerabild ist zu Darstellungszwecken ausgegraut.

Für die Ermittlung der korrekten Randpunkte wird für jedes Hindernis in obstacles aus den vorläufigen Randpunkten eine Gerade mit Hilfe der Zwei-Punkt-Form berechnet. Die Gerade hat die Form:

$$o(x) = mx + n, \text{ wobei } m = \frac{IOR.y - IOL.y}{IOR.x - IOL.x}$$

und $n = IOL.y - m * IOL.x$

Für die Ermittlung des linken Randpunktes wird *o(x)* ausgehend von x = IOL.x – 1 in Richtung roiXMin überprüft. Solange der Pixelwert an der Bildkoordinate (x, *o(x)*) über dem Schwellwert LBrightMin liegt, wird IOL neu belegt. Andernfalls ist das linke Ende des Hindernisses gefunden (Abb. 2-5).

Für die Ermittlung des rechten Randpunktes wird *o(x)* ausgehend von x = IOR.x + 1 in Richtung roiXMax überprüft. Nachdem alle Einträge in obstacles überprüft wurden, sind diese nun die tatsächlichen Randpunkte der Hindernisse im Bild. Die eigentliche Bildverarbeitung ist mit diesem Schritt abgeschlossen. Anschließend folgt die Umwandlung der Bild- in Fahrzeugkoordinaten und die Bereitstellung der Hindernisinformationen an die restlichen Module.

Umrechnung der Hindernisrandpunkte in Fahrzeugkoordinaten

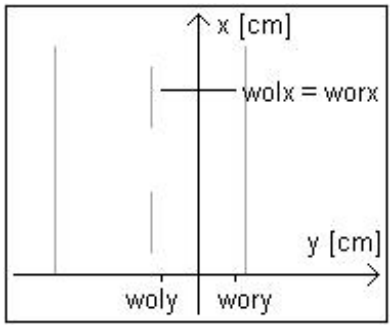

Abb. 2-6: Schematische Darstellung der Umrechnung der Hindernisrandpunkte in Fahrzeugkoordinaten. Die Fahrbahn ist zu Darstellungszwecken hellgrau hinzugefügt.

Die Umrechnung der Bildkoordinaten in Fahrzeugkoordinaten geschieht in zwei Schritten. Jedes Bildkoordinatenpaar aus obstacles muss zunächst in verzeichnungsfreie Bildkoordinaten umgerechnet werden.

Anschließend werden die verzeichnungsfreien Bildkoordinaten durch die projektive Transformation in Fahrzeugkoordinaten umgewandelt (Kapitel 1.6.2) und dann korrekt in obstacles als WOL=(wolx; woly) und WOR=(worx; wory) abgespeichert.

In Abb. 2-6 ist die Hindernisfront als gerade schwarze Linie eingezeichnet. Die Endpunkte der Linie sind die Koordinaten WOL und WOR. Die Fahrbahnmarkierungen sind als graue Linien abgebildet.

Bereitstellen der Hindernisinformationen zur weiteren Verarbeitung

Der LaserObstaclesVector obstacles enthält zu jeder gefundenen Laserlinie auf einem Hindernis einen Eintrag mit einem LaserObstacle. Das LaserObstacle beinhaltet Informationen über den linken und rechten Randpunkt des Hindernisses in Fahrzeugkoordinaten, sowie den Abstand zwischen den beiden Randpunkten in Zentimetern. Der LaserObstacleVector wird den restlichen Modulen über den Datencontainer zur Verfügung gestellt.

3 Kartenerstellung

Eine Karte dient zur Modellierung der Welt. Mit Hilfe eines Weltmodells können optimale Routen geplant werden. Um ein Weltmodell zu erhalten, werden Informationen über die Annahme der Umgebung abgespeichert. Diese Informationen sind nicht fehlerfrei, da sich die Umgebung verändern kann. Man unterschiedet zwei Arten von Karten: lokale Karten und globale Karten. In lokalen Karten wird der aktuelle Sichtbereich dargestellt. Die einzelnen, aktuellen Sensordaten werden in einer lokalen Karte zusammengefasst. Die globale Karte entspricht der Weltmodellierung. Für sie werden die einzelnen lokalen Karten zu einer Weltkarte zusammengefasst.

Für das Scenario Carolo-Cup wäre Wissen über den gesamten Rundkurs durch eine globale Karte von Vorteil. Da die statischen Hindernisse nicht verstellt werden, ist der Rundkurs für jedes Durchfahren identisch. Mit Hilfe einer globalen Karte könnte nach einmaligem Befahren des Rundkurses ein optimaler Weg mit optimalen Geschwindigkeiten berechnet werden, jedoch ist das exakte Erstellen solch einer Karte nicht trivial, da das Fahrzeug regelmäßig Sensordaten erhält, die fehlerbehaftet sind. Auch führt es anhand von Steuerdaten regelmäßig Bewegungen durch, die nicht exakt der Vorgabe entsprechen. Aufgrund der Sensor- und Steuerdaten wird die Position des Fahrzeugs auf seiner Umgebungskarte geschätzt, welche ebenfalls ungenau ist. Das Problem, des simultanen Aufbaus einer Karte und der Bestimmung der eigenen Position, wird als SLAM (simultaneous localization and mapping, [19] Kapitel 37.2) bezeichnet. Durch kleine Fehler bei der Selbstlokalisation werden die Abstandsmessungen an falscher Position und mit Orientierungsfehlern in die Karte eingetragen, so dass der Roboter bei der Kartierung tatsächlich eine andere Strecke befährt, als er annimmt. Durch das einmalige Auftreten der Startlinie könnte der Rundkurs nach dem ersten Durchlauf korrigiert werden. Aus diesen Gründen wird im Folgenden keine genaue globale Karte erstellt, da dies den Rahmen dieser Arbeit sprengen würde.

Für diese Arbeit reicht eine typische lokale Karte nicht aus, da diese nur eine Momentaufnahme der Umgebung ist. Durch die blinden Bereiche um das Fahrzeug herum, würden vorhandene Hindernisse nicht erkannt werden. Zudem können durch Fehlinterpretationen der Sensoren und in der Bildverarbeitung Hindernisse und die Fahrbahn gar nicht oder an falschen Positionen erkannt werden. Zum Einsatz kommt eine globale Karte mit begrenzter Reichweite. Die Reichweite (mapRange) wird durch die Stabilität der Fahrspurerkennung auf zwei Meter begrenzt. Die Entfernung reicht für das Erkennen aller relevanten Hindernisse aus, so dass diesen ausgewichen werden kann. Da die Reichweite der Karte begrenzt ist, ist das SLAM-Problem nicht relevant.

Um den Rechen- und Speicherbedarf so gering wie möglich zu halten, wird auf eine reduzierte Beschreibung der Umgebung zurückgegriffen. Da sich das Fahrzeug lediglich auf der Fahrbahn befindet, wird die Karte auf die Informationen der linken und rechten Fahrspur minimiert. Die Fahrspuren werden als Spalten abgebildet, auf denen das Fahrzeug und die Hindernisse zeilenweise eingetragen werden. Im folgenden Kapitel wird zunächst die Zuordnung der Hindernisse zur Fahrbahn beschrieben. Auf die genutzte Karte und deren Aufbau im Kapitel 3.2 eingegangen.

3.1 Hinderniszuordnung

Die Zuordnung der Hindernisse zur Fahrbahn ist nötig, da in Abhängigkeit von deren Position (Kapitel 1.3.2) entschieden wird, ob ein Ausweichvorgang gestartet werden muss. Um zu überprüfen, ob Hindernisse auf einer der beiden Fahrspuren liegen, muss die Position der Hindernisse im Verhältnis zu den Fahrspurpolynomen berechnet werden.

Als Fahrbahninformationen stehen die zwei Fahrspurpolynome leftEdge und rightEdge zur Verfügung (Kapitel 1.5.1). Dabei bildet das Polynom leftEdge entweder die linke Außenmarkierung oder die Mittelstreifenmarkierung ab, das Polynom rightEdge entweder die Mittelstreifenmarkierung oder die rechte Außenmarkierung. Welche Fahrspurmarkierung abgebildet wird, hängt von der Fahrspur ab, die durch POLARIS beobachtet wird. Durch PolarisLaneSide ist bekannt, welche der beiden Fahrspuren beobachtet wird. Durch diese Kenntnis können leftEdge und rightEdge jeweils einem der folgenden drei Polynome für die Fahrspurmarkierungen zugewiesen werden. Für die linke, äußere Fahrbahnmarkierung wird das Polynom $f_l(x)$, für die rechte, äußere Fahrbahnmarkierung wird das Polynom $f_r(x)$ und für die Markierung des Mittelstreifens wird das Polynom $f_m(x)$ definiert. Für die Zuweisung gilt:

PolarisLaneSide = = leftLane	PolarisLaneSide = = rightLane
$f_l(x) = leftEdge$ $f_m(x) = rightEdge$ $f_r(x) = null$	$f_l(x) = null$ $f_m(x) = leftEdge$ $f_r(x) = rightEdge$

Als Hindernisinformationen (Kapitel 2) stehen die beiden Endpunkte der Laserlinie von der Vorderseite des Hindernisses in Fahrzeugkoordinaten zur Verfügung. Laut Regelwerk ([20]) überlagern die Hindernisse nicht die Fahrbahnmarkierungen. Um Rechenzeit zu sparen, können die beiden Endpunkte der Hindernisse zu einem Mittelpunkt zusammengefasst werden. Befindet sich dieser auf einer Fahrspur, dann befindet sich auch das gesamte Hindernis auf dieser. Im Folgenden wird der Mittelpunkt der Hindernisfront als $H=(x_h, y_h)$ bezeichnet.

Für die Positionsbestimmung werden zunächst die Mittelpunkte der Fahrspuren berechnet. Der Mittelpunkt der linken Fahrspur wird als M_L, und der Mittelpunkt der rechten Fahrspur als M_R bezeichnet.

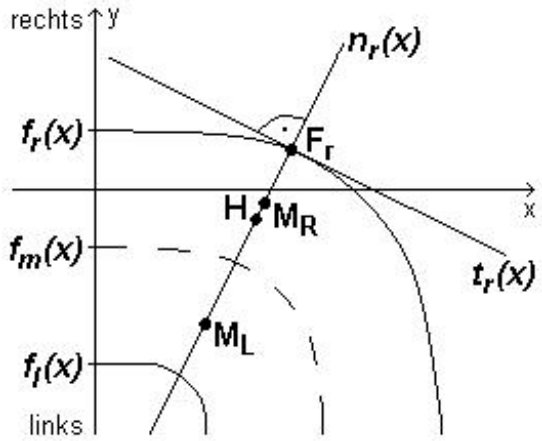

Abb. 3-1: Berechnungsgrundlage von F_r

Abb. 3-1 veranschaulicht die Idee des Verfahrens zur Bestimmung der Fahrspurmittelpunkte. Der zu untersuchende Hindernispunkt H befindet sich auf der rechten Fahrspur innerhalb einer Linkskurve. Die x-Richtung ist die Fahrtrichtung des Fahrzeugs. Positive y-Werte bilden die rechte Seite von der Fahrzeugmitte ab, negative y-Werte die linke Seite. Bei den Fahrspurpolynomen $f_r(x)$ und $f_m(x)$ handelt es sich um die vorgegebenen Polynome zweiten Grades (Kapitel 1.6.1). Zur Veranschaulichung wurde $f_m(x)$ gestrichelt dargestellt. Zudem ist das bei der Fahrt auf der rechten Fahrspur nicht bekannte dritte Fahrspurpolynom $f_l(x)$ eingetragen.

Die Mittelpunkte werden für jedes der gegebenen Polynome ($f_r(x)$, $f_m(x)$ oder $f_l(x)$) einzeln berechnet. Zur Bestimmung der Fahrspurmittelpunkte anhand des Polynoms $f_r(x)$ wird eine Normale $n_r(x)$ zu $f_r(x)$ am unbekannten Polynompunkt F_r gebildet (Abb. 3-1). Wobei $n_r(x)$ zusätzlich durch H verläuft, so dass anhand von $n_r(x)$ und F_r auf die Mittelpunkte der beiden Fahrspuren geschlossen werden kann. Da die Fahrspuren und die Mittelstreifenmarkierung eine definierte Breite besitzen (Abb. 1-4), können M_L und M_R aus F_r berechnet werden.

Da die Berechnung von $n(x)$ für jedes Polynom identisch ist, wird die Berechnung am Beispiel von $f_r(x)$ beschrieben. $n_r(x)$ wird durch zwei Punkte definiert: dem Hindernispunkt H und dem Punkt $F_r=(x_r, y_r)$, so dass $n_r(x) \perp f_r(x)$ (Abb. 3-1) ist. Mit anderen Worten: $n_r(x)$ muss senkrecht zur Tangente $t_r(x)$ des Polynoms $f_r(x)$ am Punkt F_r sein.

Für die Berechnung von $F_r=(x_r, y_r)$ ist gegeben:

$H = (x_h, y_h)$ der Hindernispunkt und

$f_r(x) = ax^2 + bx + c$ das Polynom zweiten Grades der äußeren Fahrbahnmarkierung.

Zur Berechnung von $F_r = (x_r, y_r)$, werden zwei Geradengleichungen benötigt:

$t_r(x)$ in der Form $t_r(x) = m_t x + n_t$

$$t_r(x) = p_r'(x_r)(x - x_r) + p_r(x) = (2ax_r + b)x - ax_r^2 + c \quad (3.1)$$

$n_r(x)$ in der Form $n_r(x) = m_n x + n_n$

$$n_r(x) = \frac{y_r - y_h}{x_r - x_h} x - \frac{y_r - y_h}{x_r - x_h} x_h + y_h \quad (3.2)$$

In beiden Formeln sind noch die Unbekannten x_r und y_r vorhanden. Durch die Eigenschaft: $t_r(x) \perp n_r(x)$, wodurch $m_t m_n = -1$ gilt, kann y_r errechnet werden:

$$-1 = (2ax_r + b)\frac{y_r - y_h}{x_r - x_h}$$
$$y_r = \frac{x_r - x_h}{-(2ax_r + b)} + y_h \tag{3.3}$$

Durch Einsetzen von y_r ergibt sich für $n_r(x)$:

$$n_r(x) = \frac{\left(\frac{x_r - x_h}{-(2ax_r + b)} + y_h\right) - y_h}{x_r - x_h}(x - x_h) + y_h$$
$$= \frac{x_h - x}{2ax_r + b} + y_h \tag{3.4}$$

Durch Einsetzen von y_r in $f_r(x)$ ergibt sich:

$$\frac{x_r - x_h}{-(2ax_r + b)} + y_h = ax_r^2 + bx_r + c$$
$$0 = x_r^3 + \frac{3b}{2a}x_r^2 + \frac{b^2 + 2ac - 2ay_h + 1}{2a^2}x_r + \frac{bc - by_h - x_h}{2a^2} \tag{3.5}$$

Um die kubische Gleichung (3.5) zu lösen, wird das Näherungsverfahren von Newton ([12]) genutzt. Mit $f(x)$ = (3.5) gilt:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad \text{, mit } f'(x_n) \neq 0 \tag{3.6}$$

Als Startwert x_0 wird x_h vom Hindernis genommen:

$$x_0 = x_h \tag{3.7}$$

Das Verfahren wird beendet sobald ε erreicht wird, oder mehr als 10 Iterationsschritte[1] benötigt werden. Für ε gilt:

[1] Die Begrenzung auf zehn Iterationsschritte wurde festgelegt um die Rechenzeit zu verkürzen. Tests haben ergeben, dass in Kurven bei einer Entfernung von H zum Fahrzeug bis zu 150cm bis zu fünf Iterationsschritte benötigt werden. Da nicht alle möglichen Polynom- und Hindernisposition-Kombinationen getestet wurden, wird die Anzahl der Iterationsschritte verdoppelt, da auch bei zehn Iterationsschritten die Rechenzeit noch im vorgegeben Rahmen liegt. Werden mehr Iterationsschritte benötigt, ist es möglich, dass die Newton-Folge nicht konvergent ist. Derzeit wird nicht überprüft, ob die Newton-Folge konvergiert. Liefert die Berechnung kein gültiges Ergebnis, wird dieses Polynom nicht weiter berücksichtigt. Das Ergebnis ist nicht gültig, falls $f'(x_n) \neq 0$ oder mehr als die maximal erlaubten Iterationsschritte benötigt werden.

$$\|x_{n+1} - x_n\| < \varepsilon = 0{,}001 \tag{3.8}$$

Durch Einsetzen von x_r in $f_r(x)$ wird der Polynompunkt $F_r = (x_r, f_r(x_r))$ vervollständigt und die Normale $n_r(x)$ (Gleichung (3.2)) kann berechnet werden.

Ausgehend von F_r kann über die definierte Fahrbahnumgebung auf die beiden Mittelpunkte der Fahrspuren (M_L und M_R) geschlossen werden. Die Berechnungsschritte für beide Mittelpunkte sind identisch, daher gilt für M_L und M_R als Mittelpunkt allgemein $M = (x_m, y_m)$. Die Entfernung von F_r zu M wird als d definiert.

$$d = \overline{F_r M} = \sqrt{(y_r - y_m)^2 + (x_r - x_m)^2} \tag{3.9}$$

In Gleichung (3.9) sind noch zwei Unbekannte x_m und y_m enthalten. Dadurch, dass F_r und M auf derselben Geraden $n_r(x)$ liegen, ist die Steigung zwischen den beiden Punkten bekannt (nämlich m_n). Somit kann das Steigungsdreieck so umgeformt werden, dass y_r ersetzt werden kann (Anhang D.1) und lediglich die Unbekannte x_m übrig bleibt:

$$\begin{aligned} d &= \sqrt{(m_n(x_r - x_m) + y_m - y_m)^2 + (x_r - x_m)^2} \\ 0 &= x_r^2 - 2x_m x_r + x_r^2 - \frac{d^2}{m_n^2 + 1} \\ x_m &= x_r \pm \sqrt{\frac{d^2}{m_n^2 + 1}} \end{aligned} \tag{3.10}$$

Ob die Wurzel von x_r addiert oder subtrahiert wird hängt von der jeweiligen Fahrbahnkrümmung ab, die sich in der Steigung m_n von $n_r(x)$ wiederspiegelt. Die Zuordnung der Addition und Subtraktion der Wurzel für $f_m(x)$ und $f_l(x)$ sind im Anhang D.2 angegeben. Für die Berechnungen anhand $f_r(x)$ gilt:

$m_n \geq 0$ Linkskurve $\quad x_m = x_r - \sqrt{\frac{d^2}{m_n^2 + 1}}$

$m_n < 0$ Rechtskurve $\quad x_m = x_r + \sqrt{\frac{d^2}{m_n^2 + 1}}$

Für die Berechnung von M_R ist d=20, also die halbe Fahrspurbreite. Für die Berechnung von M_L ist d=62, die gesamte rechte Fahrspurbreite plus die Breite der Fahrbahnmarkierungen plus die halbe linke Fahrspurbreite.

Durch Einsetzen von x_m in $n(x)$ wird der y_m ermittelt:

$$y_m = n(x_m) \tag{3.11}$$

Um Mess- und Berechnungsfehlern bei der Erzeugung der Fahrspurpolynome und der Hindernisinformationen entgegenzuwirken, wird die ermittelte Position der Hindernisse mit den Wahrscheinlichkeitswerten P_L und P_R erweitert. P_L gibt einen Wahrscheinlichkeitswert zwischen null und eins an. Hierbei bedeutet null, dass sich das Hindernis nicht auf der Fahrspur befindet und eins, dass es sich auf der Fahrspur befindet. Äquivalent gilt P_R für die rechte Fahrspur. Da keiner der verfügbaren Fahrspurenpolynome ein 100% richtiges Ergebnis erzeugt, werden zur Ermittlung P_L und P_R beide Fahrspurpolynome gleichzeitig berücksichtigt. Für $f_l(x)$ werden P_{lL} und P_{lR}, für $f_m(x)$ werden P_{mL} und P_{mR} und für $f_r(x)$ werden P_{rL} und P_{rR} gebildet und später zu P_L und P_R gemittelt.

Für eine prozentuale Wahrscheinlichkeitsangabe dazu, auf welcher Fahrspur sich H befindet, wird das Bayessche Entscheidungstheorem ([4]) benutzt. Hierfür werden drei Klassen identifiziert:

Ω_1: Das Hindernis befindet sich auf der linken Fahrspur

Ω_2: Das Hindernis befindet sich auf der rechten Fahrspur

Ω_3: Das Hindernis befindet sich neben der Fahrbahn

Wir betrachten die Position des Hindernisses als Zufallsvariable ω, die drei Zustände haben kann:

ω_1: Das aktuelle Hindernis befindet sich auf der linken Fahrspur

ω_2: Das aktuelle Hindernis befindet sich auf der rechten Fahrspur

ω_3: Das aktuelle Hindernis befindet sich links oder rechts neben der Fahrbahn.

Es gilt die bedingte Wahrscheinlichkeitsdichte:

$$p(x|\omega_i) \tag{3.12}$$

Wobei x die Entfernung vom aktuellen H (dem Hindernismittelpunkt) zur Fahrspurmitte ist. Gleichung (3.12) gibt die Wahrscheinlichkeit von x an, unter der Vorbedingung, dass die Umwelt im Zustand ω_i ist. So können folgende klassenbedingten Dichten definiert werden:

$p(x|\omega_1)$: klassenbedingte Entfernung des aktuellen H zur M_L

$p(x|\omega_2)$: klassenbedingte Entfernung des aktuellen H zur M_R

$p(x|\omega_3)$: klassenbedingte Entfernung des aktuellen H zu M_L und M_R.

Aufgrund der Positionierungsmöglichkeiten der Hindernisse auf der Fahrspur nehmen wir an, dass die klassenbedingte Wahrscheinlichkeitsdichte $p(x|\omega_i)$ normalverteilt $N(\mu, \sigma^2)$ ist, wobei $\sigma^2 = 9$ ist. Die neun Zentimeter der Varianz ergeben sich aus einem Messfehler für POLARIS und einer Toleranz für die Hindernisse, die nicht mittig auf die Fahrbahn gesetzt wurden. μ entspricht der Entfernung von $f_m(x)$ zu M_L bzw. M_R (jeweils 21cm) und von $f_m(x)$ zu dem Bereich neben der Fahrbahn. Für den Bereich neben der Fahrbahn wird davon ausgegangen, dass

die Hindernisse ähnlich weit Entfernt zur Fahrbahnmarkierung stehen wie auf den Fahrspuren. Ausgehend von der Mittelstreifenmarkierung entspricht dies einer Entfernung von 63cm. Somit gilt für die klassenbedingten Wahrscheinlichkeiten:

$$p(x|\omega_1) = N(-21, \sigma^2)$$
$$p(x|\omega_2) = N(+21, \sigma^2)$$
$$p(x|\omega_3) = N(-63, \sigma^2) + N(+63, \sigma^2) \quad (3.13)$$

$$N(\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right) \quad (3.14)$$

Für jeden Zustand der Zufallsvariablen ω, seien die a priori Wahrscheinlichkeiten $P(\omega_i)$:

$P(\omega_1) = 0{,}25$ für die linke Fahrspur, Klasse1

$P(\omega_2) = 0{,}25$ für die rechte Fahrspur, Klasse 2

$P(\omega_3) = 0{,}5$ für den Bereich neben der Fahrbahn, Klasse 3

Gesucht ist $P(\omega_i|x)$, die Wahrscheinlichkeit für den Zustand ω_i, wenn das Merkmal x beobachtet wurde.

$$P(\omega_j|x) = \frac{p(x|\omega_j) * P(\omega_j)}{p(x)} \quad (3.15)$$

$$p(x) = \sum_{j=1}^{3} p(x|\omega_j) * P(\omega_j) \quad (3.16)$$

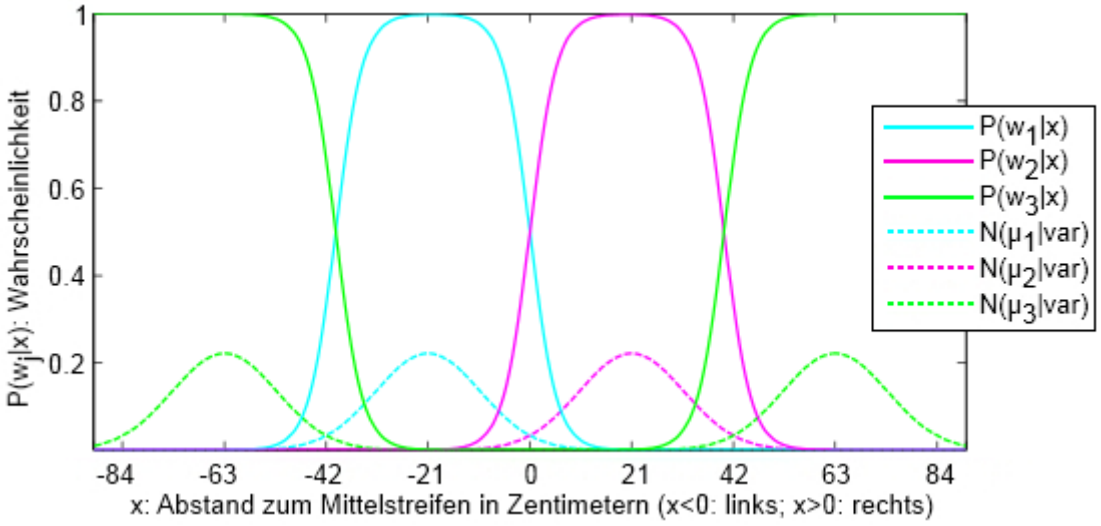

Abb. 3-2: Kurven der Wahrscheinlichkeitsberechnung nach Bayes

Nun kann durch Einsetzen des x-Wertes für das aktuelle Hindernis in die Gleichung (3.15) der Wahrscheinlichkeitswert direkt ermittelt werden. In Abb. 3-2 sind die Wahrscheinlichkeiten für die einzelnen Zustände aus Gleichung (3.15) als durchgezogene Linien grafisch dargestellt. Die gestrichelten Linien sind die entsprechenden Normalverteilungen aus Gleichung (3.14).

Für jedes Fahrspurpolynom werden die Wahrscheinlichkeiten $P(\omega_1|x)$ und $P(\omega_2|x)$ berechnet und addiert, wobei k=2 für die Anzahl der Polynome steht. Sollten in Zukunft für alle drei Fahrspurmarkierungen Polynome zur Verfügung gestellt werden, müssen lediglich für das dritte Polynom die beiden Wahrscheinlichkeiten $P(\omega_1|x)$ und $P(\omega_2|x)$ berechnet, und k auf drei festgelegt werden. Für P_L und P_R gilt:

$$P_L = \frac{\sum_{i=1}^{k} P_i(\omega_1|x)}{k} \tag{3.17}$$

$$P_R = \frac{\sum_{i=1}^{k} P_i(\omega_2|x)}{k} \tag{3.18}$$

Der in diesem Kapitel vorgestellte Algorithmus ist gegenüber dem Ansatz, der im Carolo-Cup 2010 verwendet wurde, um die Wahrscheinlichkeitsberechnung erweitert worden. Beim alten Ansatz wurde lediglich überprüft, ob H zwischen F_r und F_m, bzw. F_l und F_m liegt und so der rechten, bzw. linken Fahrspur zugewiesen. Die Erweiterung der Wahrscheinlichkeit ist insbesondere daher eine Verbesserung, da die fehlerhaften Darstellungen der Fahrspuren durch POLARIS (Kapitel 1.5.1) berücksichtigt werden. Dieses Verfahren wurde bereits in Matlab implementiert und kann nun in die Kartenerstellung integriert werden.

3.2 Kartenerstellung

Die direkte Umgebung des Fahrzeugs wird reduziert und beide Fahrspuren in einer begrenzten Reichweite (mapRange) dargestellt. Um die Zugriffe auf die Karte zu erleichtern, ist das Fahrzeug an eine feste Position der Karte gesetzt. Der Aufruf zum Überprüfen, ob sich ein Hindernis vor dem Fahrzeug befindet, ist immer gleich und nicht abhängig von der gerade befahrenen Fahrspur. Das hat zur Folge, dass die Spalten der Karte nicht immer dieselbe Fahrspur repräsentieren und die Einträge bei einem Fahrspurwechsel verschoben werden müssen.

	left	mid	right
front2nd	$a_{1,1}$ leftFront2nd	$a_{1,2}$ midFront2nd	$a_{1,3}$ rightFront2nd
front	$a_{2,1}$ leftFront	$a_{2,2}$ midFront	$a_{2,3}$ rightFront
car	$a_{3,1}$ left	$a_{3,2}$ carPos	$a_{3,3}$ right
rear	$a_{4,1}$ leftRear	$a_{4,2}$ midRear	$a_{4,3}$ rightRear

Abb. 3-3: Karte für den Ausweichvorgang

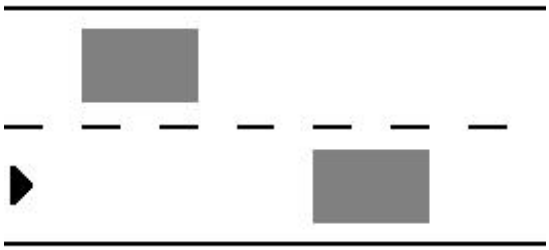

Abb. 3-4: Es befinden sich zwei Hindernisse versetzt auf der linken und rechten Fahrspur

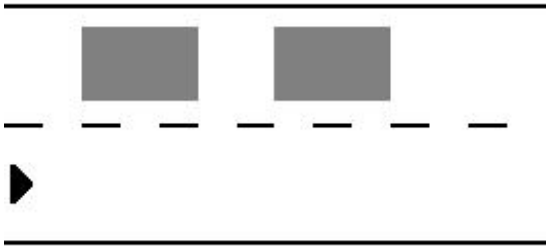

Abb. 3-5: Zwei Hindernisse befinden sich hintereinander auf der linken Fahrspur

Um das Fahrzeug herum wird eine 4x3 Matrix aufgebaut (Abb. 3-3). Die Entfernungsangaben der Hindernisse sind in Fahrzeugkoordinaten (Kapitel 1.6.1) angegeben. Das Fahrzeug befindet sich an Position $a_{3,2}$ (in Abb. 3-3 mit *carPos* gekennzeichnet). Die Spalte *mid* ($a_{i,2}$) stellt die gerade befahrene Fahrspur dar, also entweder die linke oder die rechte Fahrspur. Die Spalte *left* ($a_{i,1}$) entspricht entweder dem Bereich links neben der Fahrbahn, oder der linken Fahrspur. Die Spalte *right* ($a_{i,3}$) entspricht entweder der rechten Fahrspur, oder dem Bereich rechts neben der Fahrbahn.

Eine weitere Abstraktion der Umgebung stellen die Zeilen der Karte dar. In der Zeile *front* ($a_{2,k}$) sind die nächsten Hindernisse auf der jeweiligen Fahrspur abgebildet, wobei diese nicht dieselbe Entfernung zum Fahrzeug haben müssen. Beispielsweise kann in *leftFront* ein Hindernis mit einer Entfernung von 20cm und in *midFront* ein Hindernis mit einer Entfernung von 150cm eingetragen sein. Dies würde der Hindernissituation aus Abb. 3-4 entsprechen. Werden zwei Hindernisse auf derselben Fahrspur erkannt, wird das entferntere Hindernis in der Zeile *front2nd* ($a_{1,k}$) abgespeichert. Bei der Hindernissituation, in Abb. 3-5, wird das erste Hindernis in *leftFront* abgespeichert, da es das nächste Hindernis ist. Das zweite Hindernis wird an Position *leftFront2nd* abgespeichert. Die Zeile *car* ($a_{3,k}$) repräsentiert Hindernisse, die sich neben dem Fahrzeug befinden. Dies ist der Fall, solange die Vorderseite des Hindernisses eine negative Entfernung und die Rückseite des Hindernisses einen Entfernungswert größer als carRearPos = -47cm hat. Die 47cm entsprechen der Fahrzeuglänge (carLength). In der Zeile *rear* ($a_{4,k}$) sind Hindernisse abgebildet, die hinter dem Fahrzeug stehen. Ein Hindernis befindet sich hinter dem Fahrzeug, sobald die Hindernisfront eine kleinere Entfernung als carRearPos hat.

Fährt das Fahrzeug an einem Hindernis vorbei, wird dieses in zwei Zelleneinträgen der Spalten *left* und *right* vermerkt. Dasselbe Hindernis ist in den Zeilen *front* und *car* eingetragen, solange die Vorderseite des Hindernisses einen negativen Entfernungswert hat und die Hinterseite des Hindernisses nicht definiert ist (sie also noch nicht erkannt wurde), oder einen positive Entfernungswert hat. In den Zeilen *car* und *rear* kann dasselbe Hindernis eingetragen sein, falls die Vorderkante des Hindernisses kleiner carRearPos ist und die Hinterkante zwischen carFrontPos und carRearPos ist. Hindernisse mit einer Länge, die größer als 47cm ist, können sogar in *front*, *car* und *rear* gleichzeitig eingetragen sein.

Bei einem Fahrspurwechsel des Fahrzeugs werden die Inhalte der Karte verschoben. Das Fahrzeug fährt standardmäßig auf der rechten Fahrspur. Die Spalte *left* der Karte entspricht der linken Fahrspur. Die Spalte *mid* entspricht der rechten Fahrspur. Die Spalte *right* ist nicht relevant, da in dieser der Bereich rechts neben der Fahrbahn abgebildet wird. Bei einem Fahrspurwechsel auf die linke Fahrspur werden die Zelleneinträge von $a_{i,k}$ nach $a_{i,k+1}$ verschoben. Die Zelleneinträge der Spalte *left* werden gelöscht, da über den Bereich links neben der Fahrbahn keine Hindernisinformationen vorliegen. Bei einem Fahrspurwechsel von der linken auf die rechte Fahrspur wird die Karte in die andere Richtung verschoben. Die Spalten werden von $a_{i,k}$ nach $a_{i,k-1}$ kopiert und die Inhalte der Spalte *right* gelöscht. Da die Karte um das Fahrzeug herum aufgebaut ist, muss vor der Aktualisierung der Karte anhand der Sensordaten überprüft werden, ob der Fahrspurwechsel eingeleitet wurde. Ist dies der Fall, wird die Karte zunächst verschoben. Anschließend wird die Karte aktualisiert.

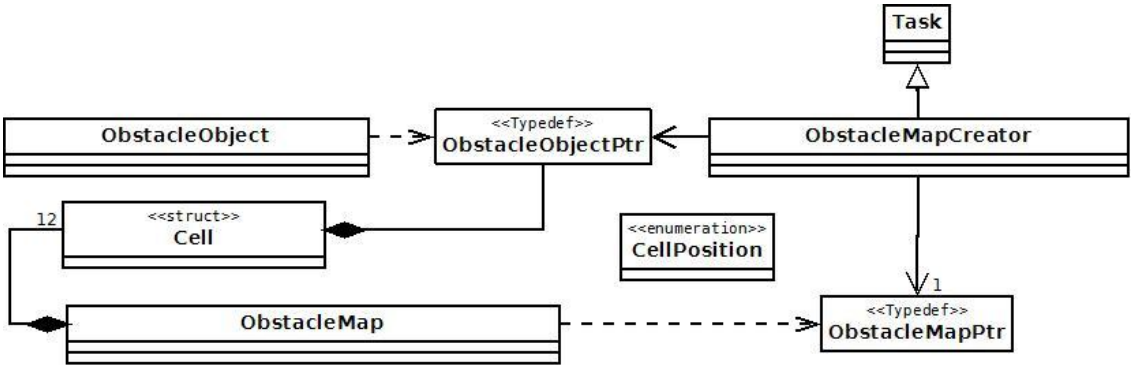

Abb. 3-6: reduziertes Klassendiagramm der ObstacleMap

Als Übersicht für den Aufbau der Karte ist in Abb. 3-6 die UML der Karte abgebildet. Eine ausführlichere Beschreibung der einzelnen Klassen ist im Anhang E gegeben. Die Karte besteht aus drei Klassen:

- ObstacleObject: Enthält die Daten eines Hindernisses.
- ObstacleMap: Karte bestehend aus 12 Zellen, die jeweils einen Zeiger zur Hindernisstruktur haben.
- ObstacleCreator: Erstellt und aktualisiert die Karte.

Für die Aktualisierung der Karte werden alle in Kapitel 1.4.1 beschriebenen Sensoren benutzt, wobei die Sensordaten der Reihe nach überprüft werden. Zunächst werden die neuen Sensordaten temporär abgespeichert.

Aktualisierung der Karte durch die Hindernisdaten aus der Bildverarbeitung

Die Bildverarbeitung zur Hinderniserkennung (Kapitel 2) stellt einen LaserObstacleVector zur Verfügung. Dabei handelt es sich um einen Vektor, in dem alle gefundenen Laserlinien einzeln als Zeiger auf LaserObstacle-Objekte abgespeichert sind. Die LaserObstacles enthalten die beiden Endpunkte der Laserlinie in Fahrzeugkoordinaten.

Bevor der LaserObstacleVector zur Aktualisierung der Karte genutzt werden kann, muss er sortiert werden. Für die Kartenerstellung ist nur die Frontseite der Hindernisse relevant, im LaserObstacleVector können jedoch zusätzlich noch Hindernisseiten enthalten sein. Sind zwei Seiten des Hindernisses erkannt worden, befinden sich dessen Linien hintereinander im LaserObstacleVector. Es wird diejenige Laserlinie als gültig befunden, die näher am Fahrzeug ist. Die andere wird nicht berücksichtigt.

Für die gültigen Laserlinien erfolgt die Positionszuordnung. Befindet sich das aktuelle Hindernis nicht auf der Fahrbahn, wird es nicht weiter berücksichtigt, und das nächste Hindernis wird überprüft. Befindet es sich auf einer Fahrspur, wird die entsprechende Kartenspalte aktualisiert. Befährt das Fahrzeug die rechte Fahrspur und das Hindernis befindet sich auf der linken Fahrspur, wird die Spalte *left* der Karte aktualisiert. Befindet sich das Hindernis jedoch auf der rechten Fahrspur, wird die Spalte *mid* der Karte aktualisiert.

Es folgt eine Auflistung der Kartenaktualisierung. Die Auflistung ist für die Spalten *left* und *right* identisch. Der erste Schritt kann bei der Spalte *mid* weggelassen werden, da sich in der Zeile *car* das Fahrzeug befindet und dort keine Hindernisse abgespeichert werden. curDist gibt die Entfernung des zur Aktualisierung herangezogenen Hindernisses an.

```
distToFront = |curDist – front.distanceFront|
distToFront2nd = |curDist – front2nd.distanceFront|
if ( front besetzt && car besetzt &&
     (car.obstacleID == front.obstacleID)) {
   front2ndÜberprüfen = true
} else if (front unbesetzt) {
   Neues Hindernis erstellen und in front einfügen
} else if (front besetzt) {
   if (das Fahrzeug fährt vorwärts) {
      if (curDist < front.distanceFront) {
         front.distanceFront = curDist
      } else if (distToFront < minDistanceOfObstacles) {
```

```
         nichts tun (dasselbe Hindernis),
         front.distanceFront wird als wahr angenommen
      } else if (distToFront > minDistanceOfObstacles) {
         front2ndÜberprüfen = true
      }
   } else if (das Fahrzeug fährt rückwärts) {
      if ( (front.distanceFront < curDist)  &&
           (distToFront < minDistanceOfObstacles) {
         front.distanceFront = curDist
      } else if (distToFront > minDistanceOfObstacles) {
         front2ndÜberprüfen = true
      } else if (curDist < front.distanceRear) {
         front.distanceFront = curDist
      } else if (curDist > front.distanceRear) {
         front2ndÜberprüfen = true
      }
   }
}
if (front2ndÜberprüfen == true) {
   if (front2nd unbesetzt) {
      Neues Hindernis erstellen und in front2nd einfügen
   } else if (front2nd besetzt) {
      if (das Fahrzeug fährt vorwärts) {
         if (curDist < front2nd.distanceFront) {
            front2nd.distanceFront = curDist
         } else {
            Nichts tun (3.Hindernis)
         }
      } else if (das Fahrzeug fährt rückwärts &&
                 (curDist > front2nd.distanceFront) &&
                 (distToFront2nd < minDistanceOfObstacles) ) {
         front2nd.distanceFront = curDist
      }
   }
}
```

Aktualisierung der Karte durch die Infrarotsensoren

Die Infrarotsensoren decken die Seitenbereiche des Fahrzeugs ab. Ein Hindernis auf der Fahrbahn wird erkannt, falls der IR-Sensor Werte im Intervall [0; detectRangeIR] liefert. Sind die Werte größer als detectRangeIR, so befinden sich die detektierten Objekte außerhalb der Fahrbahn und sind für die Aktualisierung der Karte nicht relevant. detectRangeIR wird über einen Parameterwert zum Programmstart vorgegeben. Bei der Annahme, dass die Hindernisse mindestens 20cm breit sind, wird detectRangeIR mit 40cm belegt. So ist gewährleistet, dass auch Hindernisse mit einer ungünstigen Stellung erkannt werden. Befindet sich ein 20cm breites Hindernis auf der linken Fahrspur an der linken Außenmarkierung und das Fahrzeug fährt fehlerhafterweise nicht in der Mitte der rechten Fahrspur sondern rechts auf dem äußersten Fahrspurenrand, liegen zwischen dem Hindernis und dem Fahrzeug genau 40cm.

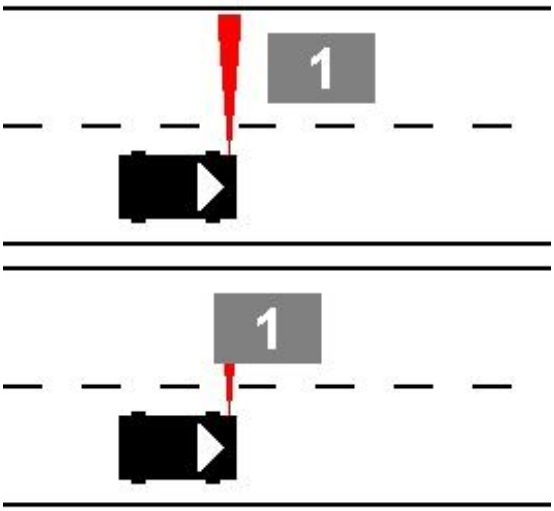

Abb. 3-7: Schematische Darstellung des Eintritts eines links stehendem Hindernis in den Bereich neben dem Fahrzeug (Zelle $a_{3,1}$ aus Abb. 3-3).

Die relevanten Ereignisse sind die Signalflanken. Sie entstehen, wenn die IR-Werte entweder in den gültigen Bereich hinein oder treten aus ihm heraus gelangen. Bei einer Vorwärtsfahrt bedeutet ein Flankenwechsel der vorderen IR-Sensoren in den gültigen Bereich, dass die Fahrzeugfront eine Hindernisfront passiert hat (Abb. 3-7). Das Hindernis aus *leftFront* (bzw. *rightFront*) wird zusätzlich in die Zelle *left* (bzw. *right*) eingetragen und distanceFront wird aktualisiert. Bis das Fahrzeug am Hindernis vorbei gefahren ist, kann anhand der IR-Sensordaten keine Aussage darüber getroffen werden, wie weit das Fahrzeug an dem Hindernis vorbeigefahren ist. Es kann lediglich verifiziert werden, dass das Hindernis noch existiert. Fährt das Fahrzeug mit seiner Front am Hindernis vorbei entsteht beim vorderen IR-Sensor ein Flankenwechsel aus dem gültigen Wertebereich heraus. Erst jetzt kann eine Aussage über die Position der Hinterkante des Hindernisses getroffen werden. DistanceRear wird aktualisiert und das Hindernis wird aus der Zeile front entfernt.

Wird beim hinteren IR-Sensor ein Flankenwechsel in den gültigen Bereich erkannt, wird die distanceFront des Hindernisses der Zeile *car* aktualisiert. Fährt das Fahrzeug mit seinem Heck am Hindernis vorbei, entsteht beim hinteren IR-Sensor ein Flankenwechsel aus dem gültigen Wertebereich heraus. Die distanceRear des Hindernisses aus der Zeile *car* wird aktualisiert.

Bei einer Rückwärtsfahrt werden analog zur vorhergehenden Beschreibung zuerst die rückwärtigen Sensoren zur Erkennung von Hindernissen verwendet.

Es folgt eine genaue Auflistung der Aktualisierung der Karte über IR-Sensoren, wobei zwischen den vorderen und hinteren IR-Sensoren unterschieden wird. Die IR-Sensoren sind fest am Fahrzeug montiert und die Position im Bezug zur Fahrzeugfront wird als IRFrontPos, für die vorderen IR-Sensoren, und IRRearPos, für die hinteren IR-Sensoren, bezeichnet. Um festzustellen, ob ein Flankenwechsel, also der erstmalige Ein- oder Austritt eines Hindernisses in die Zeile *car*, erfolgt ist, wird ein Merker validRangeFront und validRangeRear definiert.

```
// Aktualisierungsentscheidungen für die vorderen IR-Sensordaten
if (IR-Sensordaten im Wertebereich) {
   // Erstmaliges Erkennen des Hindernisses
   if (validRangeFront == false) {
      if (das Fahrzeug fährt vorwärts && front belegt) {
         front.distanceFront = IRFrontPos
         validRangeFront = true
      } else if (das Fahrzeug fährt rückwärts && car belegt) {
         car.distanceRear = IRFrontPos
         validRangeFront = true
      }
   } else {
      // Nichts tun. Das benachbarte, bekannte Hindernis wird
      // weiterhin erkannt, es ist jedoch keine Aktualisierung der
      // distanceFront anhand der IR-Angaben möglich.
   }
} else if (IR-Sensordaten außerhalb des Wertebereichs) {
   // Erstmaliges nicht Erkennen des Hindernisses
   if (validRangeFront == true) {
      if (Das Fahrzeug fährt vorwärts && car belegt) {
         car.distanceRear = IRFrontPos
      } else if (Das Fahrzeug fährt rückwätrs && front belegt){
         front.distanceFront = IRFrontPos
      }
      validRangeFront = false
   }
}

//  Aktualisierungsentscheidungen für die hinteren IR-Sensordaten
if (IR-Sensordaten im Wertebereich) {
   // Erstmaliges Erkennen des Hindernisses
   if (validRangeRear == false) {
```

```
      if (das Fahrzeug fährt vorwärts && car belegt) {
         car.distanceRear = IRRearPos
         validRangeRear = true
      } else (Das Fahrzeug fährt rückwärts && rear belegt) {
         rear.distanceRear = IRRearPos
         validRangeRear = true
      }
   } else {
      // Nichts tun. Das benachbarte, bekannte Hindernis wird
      // weiterhin erkannt, es ist jedoch keine Aktualisierung der
      // distanceRear anhand der IR-Angaben möglich.
   }
} else if (IR-Sensordaten außerhalb des Wertebereichs) {
   //Erstmaliges nicht Erkennen des Hindernisses
   if (validRangeRear == true) {
      if ( das Fahrzeug fährt vorwärts && rear belegt &&
           (rear.distanceRear > (carRearPos – calcDistMeasErr)) ) {
         rear.distanceRear = IRRearPos
      } else (das Fahrzeug fährt rückwärts && car belegt) {
         car.distanceRead = IRRearPos
      }
      validRangeRear = false
   }
}
```

Aktualisierung der Karte durch die Ultraschallsensoren

Die Positionen der Hindernisse hinter dem Fahrzeug werden bereits vor dem Fahrzeug ermittelt. Lediglich beim Starten sind die Positionen von möglichen Hindernissen hinter dem Fahrzeug nicht bekannt. Diese können jedoch vernachlässigt werden, da das Fahrzeug zu diesem Zeitpunkt steht, bzw. langsam an fährt, so dass eine Rückwärtsfahrt, die bis hinter die Startposition dauert, nicht vorkommt. Die US-Sensordaten werden lediglich zur Aktualisierung vorhandener Hindernisse genutzt. Die Daten werden zur Aktualisierung herangezogen, wenn sie im gültigen Wertbereich [-mapRange; minUSRange] liegen. Die minUSRange wird durch den Sensor und seine Befestigungsposition am Fahrzeug vorgegeben.

```
lrRear = leftRear oder rightRear, je nachdem welche der beiden Spalten
die Nebenfahrspur ist
maxDistPerCyc = maximal möglich gefahrene Entfernung
if (midRear belegt && lrRear belegt) {
   if (USRearLeftDist gültig && USRearRightDist gültig) {
```

```
    // Option I
    if ( (USRearLeftDist == USRearRightDist) &&
         (|USRearLeftDist + midRear.distanceRear| < maxDistPerCyc) {
       midRear.distanceRear = USRearLeftDist
    }
    // Option II
    else if ( (USRearLeftDist == USRearRightDist) &&
              ( |USRearLeftDist + lrRear.distanceRear|
                < maxDistPerCyc) ) {
       lrRear.distanceRear = USRearLeftDist
    }
    // Option III
    else if ( |USRearLeftDist + midRear.distanceRear|
              < maxDistPerCyc) {
       midRear.distanceRear = USRearLeftDist
    }
    // Option IV
    else if ( |USRearRightDist + midRear.distanceRear|
              < maxDistPerCyc) {
       midRear.distanceRear = USRearRightDist
    }
    // Option V
    else if ( |USRearLeftDist + lrRear.distanceRear|
              < maxDistPerCyc) {
       lrRear.distanceRear = USRearLeftDist
    }
    // Option VI
    else if ( |USRearRightDist + lrRear.distanceRear|
              < maxDistPerCyc) {
       lrRear.distanceRear = USRearRightDist
    }
 } else if (USRearLeftDist gültig) {
    // Option VII
    if ( |USRearLeftDist + midRear.distanceRear| < maxDistPerCyc) {
       midRear.distanceRear = USRearLeftDist
    }
    // Option VIII
```

```
      else if ( |USRearLeftDist + lrRear.distanceRear|
                < maxDistPerCyc) {
         lrRear.distanceRear = USRearLeftDist
      }
   } else if (USRearRightDist gültig) {
      // Option IX
      if ( |USRearRightDist + midRear.distanceRear| < maxDistPerCyc) {
         midRear.distanceRear = USRearRightDist
      }
      // Option X
      else if ( |USRearRightDist + lrRear.distanceRear|
                < maxDistPerCyc) {
         lrRear.distanceRear = USRearRightDist
      }
   }
} else if (midRear belegt && lrRear frei) {
   if (USRearLeftDist gültig && USRearRightDist gültig) {
      // siehe Option I, III und IV
   } else if (USRearLeftDist gültig) {
      // siehe Option VII
   } else if (USRearRightDist gültig) {
      // siehe Option IX
   }
} else if (lrRear belegt  && midRear frei){
   if (USRearLeftDist gültig && USRearRightDist gültig) {
      // siehe Option II, V und VI
   } else if (USRearLeftDist gültig) {
      // siehe Option VIII
   } else if (USRearRightDist gültig) {
      // siehe Option X
   }
}
```

Aktualisierung der Karte ohne objekterkennende Sensordaten

Abschließend wird für die Aktualisierung der Hinderniswerte in den blinden Bereichen der Karte eine Distanz über die gefahrenen Zentimeter des vergangenen Zyklus cmPerCyc berechnet. Dies geschieht mit Hilfe der Inkrementalgeber. Hierfür wird eine Differenz aus den aktuellen Inkrementalgeberdaten und den Inkrementalgeberdaten aus dem vergangenen Zyklus be-

rechnet. Dies geschieht sowohl für den linken als auch für den rechten Inkrementalgeber. Anschließend werden die beiden Werte gemittelt, um in Kurven die unterschiedlich viele Radumdrehungen der inneren und äußeren Räder auszugleichen.

Festschreiben der Karte und Überprüfung der Zellenwechsel

Sobald alle Sensoren überprüft worden sind, werden die temporären Daten fest in die Karte übernommen. Das Festschreiben der Sensorwerte geschieht über eine update-Methode, die in jedem ObstacleObject vorhanden ist. Hierfür werden die Zellen der Karte der Reihe nach überprüft. Da ein ObstacleObject in mehreren Zellen vorhanden sein kann, werden die IDs der Hindernisse in einem Vektor, updatedIDs, gespeichert, bei denen die update-Methode bereits aufgerufen wurde. Zur Aktualisierung der Hinderniswerte in den blinden Bereichen wird cmPerCyc der update-Methode übergeben. Es folgt der Pseudocode für die update-Methode:

```
for cellPosition c = leftFront2nd to rightRear{
   if (c belegt) {
      if (c.ID NOT IN updatedIDs) {
         c.update(cmPerCyc)
         updatedIDs.push_back(c.ID)
      }
   }
}
```

Nach dem Festschreiben der Sensorwerte muss überprüft werden, ob die Hindernisse noch im gültigen Sichtbereich der Karte sind, und ob Zellenübergänge umgesetzt werden müssen. Objekte können gelöscht werden, falls

- beide Distanzangaben ungültig sind, oder
- die FrontDistance > mapRange ist, oder
- die RearDistance < der negativen mapRange ist.

Die Zellenübergänge werden bei einer Vorwärtsfahrt in folgender Reihenfolge überprüft:

Ein Zellenwechsel von *car* nach *rear* erfolgt, falls gilt:
`car.distanceFront < carRearPos`

Der Inhalt aus Zelle *car* wird gelöscht, falls:
`car.distanceRear < carRearPos`

Ein Zellenwechsel von *front* nach *car* erfolgt, falls gilt:
`front.distanceFront < carFrontPos`

Der Inhalt aus Zelle *front* wird gelöscht, falls:
`front.distanceRear < carFrontPos`

Ein Zellenwechsel von *front2nd* nach *front* erfolgt, falls gilt:
`front frei`
Hierbei wird der Inhalt der Zelle *front2nd* gelöscht.

Für die Rückwärtsfahrt müssen die Zellenübergänge in derselben Reihenfolge mit invertierten Paaren überprüft werden:

Ein Zellenwechsel von *rear* nach *car* erfolgt, falls gilt:
`rear.distanceRear > carRearPos`

Der Inhalt aus Zelle *rear* wird gelöscht, falls:
`rear.distanceFront > carRearPos`

Ein Zellenwechsel von *car* nach *front* erfolgt, falls gilt:
`car.distanceRear > carRearPos`
Bevor der Zellenwechsel vollzogen werden kann, muss zunächst überprüft werden, ob in *front* ein anderes Hindernis vorhanden ist. Ist dies der Fall, erfolgt ein Zellenwechsel von *front* nach *front2nd*.

Der Inhalt aus Zelle *car* wird gelöscht, falls:
`rear.distanceFront > carRearPos`

4 Ausweichverfahren

Im Rahmen dieser Arbeit wurde für den Ausweichvorgang zunächst eine Grundversion eines Automaten entwickelt, welcher sich später leicht modular erweitern lässt. Der Automat wird im Folgenden beschrieben und hat das Ziel, statischen Hindernissen ordnungsgemäß auszuweichen.

Für eine Situationsanalyse der Umgebung, durch die entschieden wird, ob ein Ausweichvorgang eingeleitet werden muss, wird die in Kapitel 3 beschriebene Karte, obstacleMap, genutzt. Solange sich kein Hindernis auf der rechten Fahrspur befindet, hat das Ausweichverfahren eine beobachtende Rolle und fährt durch die in Kapitel 1.5.2 beschriebene Fahrspurführung. Sobald ein Hindernis auf der rechten Fahrspur erkannt wird, das so nah ist, dass diesem ausgewichen werden muss, verlässt der Automat die beobachtende Rolle und greift in die Fahrspurführung ein. Je nach Geschwindigkeit des Fahrzeugs variiert die Länge der Strecke, die für einen Fahrspurwechsel nötig ist. Je schneller das Fahrzeug fährt, desto mehr Strecke erfordert der Fahrspurwechsel. Der Startpunkt des Fahrspurwechsels wird so gewählt, dass zum einen kein weiteres Hindernis berührt wird, und zum anderen die Fahrt auf der linken Fahrspur möglichst kurz bleibt. Auch ein auf der linken Fahrspur stehendes Hindernis beeinflusst die dynamische Berechnung des Startpunktes, denn je näher die Hindernisse zueinander stehen, desto weniger Platz ist für den Fahrspurwechsel vorhanden. Da die Einstellung der Geschwindigkeit und des Lenkwinkeleinschlags abhängig von der Akkuladung ist (Kapitel 5) und das Ende eines Hindernisses erst bekannt ist, nachdem es passiert wurde (Kapitel 2), werden der Startpunkt des Fahrspurwechsels und die Geschwindigkeit des Fahrzeugs fest vorgegeben, solange der Automat sich nicht in beobachtender Rolle befindet. Um den Fahrspurwechsel innerhalb des Ausweichens durchzuführen, wird die bereits vorhandene Fahrspurführung genutzt. Wird ein Hindernis zu spät erkannt, so dass bei der gegebenen Geschwindigkeit kein korrekter Ausweichvorgang mehr stattfinden kann, muss das Fahrzeug zunächst bremsen und anschließend im Rückwärtsgang auf die entsprechende Entfernung zurückfahren. Sobald das Fahrzeug neben dem Hindernis auf der linken Fahrspur fährt, wird stetig überprüft, ob genügend Platz zum Wiedereinscheren vor dem Hindernis auf der rechten Fahrspur ist. Ist der Ausweichvorgang beendet, wird die Weiterfahrt fortgesetzt.

Da der Startpunkt des Fahrspurwechsels und die Geschwindigkeit des Fahrzeugs fest vorgegeben werden, können für das gesamte Ausweichverfahren folgende Parameter definiert werden:

- maxForwardSpeed: Die Geschwindigkeit der Vorwärtsfahrt während der in die Fahrspurführung eingreifenden Phase.
- maxBackwardSpeed: Die Geschwindigkeit der Rückwärtsfahrt.

- maxAvoidDistance: Die maximal benötige Ausweichdistanz, um bei gegebener Geschwindigkeit zwischen zwei Hindernissen mit einem Meter Entfernung die Fahrspur zu wechseln, ohne diese zu berühren.
- minAvoidDistance: Die minimal benötige Ausweichdistanz, um bei gegebener Geschwindigkeit zwischen zwei Hindernissen mit einem Meter Entfernung die Fahrspur zu wechseln, ohne diese zu berühren.
- stopDistance: Die minimal benötige Distanz für ein Anhalten vor dem Hindernis, so dass dieses nicht berührt wird.

In Abhängigkeit davon, in welcher Situation sich das Fahrzeug befindet, werden unterschiedliche Fahrentscheidungen getroffen. Für den Ausweichvorgang werden fünf unterschiedliche Bereiche definiert (Abb. 4-1), die den fünf Fahrsituationen, „kein Hindernis in Ausweichweite“ (A), „ein Hindernis ist vor dem Fahrzeug in Ausweichweite“ (B), „das Hindernis ist zu nah“ (C), „das Fahrzeug fährt neben einem Hindernis auf der linken Fahrspur“ (D) und „das Fahrzeug hat das Hindernis passiert“ (E), entsprechen.

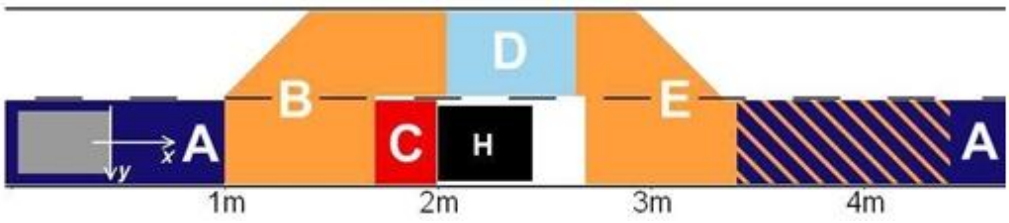

Abb. 4-1: Grafische Darstellung der Zustände des Ausweichverfahrens mit aktivem Wechsel der Fahrspur

Für diese fünf Fahrsituationen lassen sich direkt die Zustände für einen Zustandsautomaten ableiten (Abb. 4-2). Innerhalb der Zustände haben Hindernisse eine unterschiedliche Bedeutung. Ist ein Hindernis auf der linken Fahrspur, während sich das Fahrzeug in Zustand A befindet, hat das linke Hindernis keine Bedeutung. Befindet sich hingegen ein Hindernis auf der linken Fahrspur im Bereich B, während sich das Fahrzeug in Zustand B befindet, beeinträchtigt dieses den Fahrspurwechsel. Im Folgenden werden die einzelnen Zustände und Zustandsübergänge beschrieben.

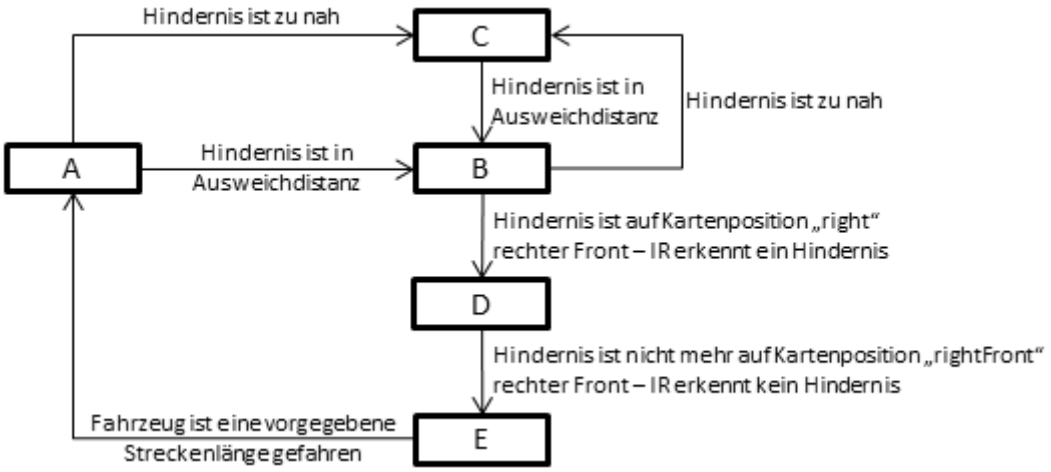

Abb. 4-2: Zustandsautomat für den Ausweichvorgang

Zustand A

Solange kein relevantes Hindernis vorhanden ist, befindet sich der Automat im Zustand A. In diesem Zustand hat das Ausweichverfahren lediglich eine beobachtende Rolle. Je nach Situation kann das Fahrzeug in die Folgezustände B und C gelangen. Hindernisse auf der linken Fahrspur sind in diesem Zustand nicht relevant, da sie nicht im Fahrbereich des Fahrzeugs liegen. Hingegen haben Hindernisse auf der rechten Fahrspur eine hohe Relevanz. Je nachdem, wie nah die Hindernisse dem Fahrzeug sind, erfolgt die Entscheidung, in welchen Zustand gewechselt wird. Da die maxForwardSpeed vom Zustand C verändert werden kann, wird diese auf den über den Parameter vorgegebenen Wert zurückgesetzt.

Im Regelfall erfolgt ein Zustandswechsel in den Zustand B. Taucht ein Hindernis plötzlich auf, oder wurde spät erkannt, könnte die stopDistance unterschritten und in den Notfallzustand C gewechselt werden. Ein Hindernis befindet sich auf der befahrenen, rechten Fahrspur, falls die Zelle *midFront* der obstacleMap belegt ist. Ein Zustandswechsel in den Zustand B erfolgt, falls gilt:

$$\texttt{stopDistance} < \textit{midFront}.\texttt{distanceFront} \leq \texttt{maxAvoidDistance}$$

Ein Zustandswechsel in den Zustand C erfolgt, falls gilt:

$$\textit{midFront}.\texttt{distanceFront} \leq \texttt{stopDistance}$$

Zustand B

Der Zustand B ist erreicht, falls ein Hindernis auf der rechten Fahrspur in Ausweichdistanz zum Fahrzeug ist. Die Geschwindigkeit wird auf maxForwardSpeed reduziert. Der linke Blinker des Fahrzeugs wird aktiv, um den Fahrspurwechsel anzuzeigen. Das Fahrzeug befindet sich solange im Zustand B, bis der Fahrspurwechsel zum Ausweichen vollzogen wurde oder misslingt.

Bis der Ausweichvorgang eingeleitet ist, wird stetig überprüft, ob sich das Fahrzeug mittlerweile nicht zu dicht am Hindernis befindet. Wird die minAvoidDistance unterschritten, ist nicht mehr genügend Platz für den Fahrspurwechsel, ohne dass das Hindernis berührt wird. Der Automat wechselt in den Zustand C.

Der Ausweichvorgang wird eingeleitet, sobald das links stehende Hindernis soweit überholt wurde, dass es bei einem Fahrspurwechsel nicht mehr berührt wird. Ist die Hinterkante des links stehenden Hindernisses bekannt, befindet sie sich mindestens 3cm hinter der Fahrzeugfront (Kapitel 3.2) und der Fahrspurwechsel kann eingeleitet werden. Da die Fahrspurführung aufgrund des Systemaufbaus (Kapitel 1.2) erst im folgenden Zyklus den Fahrspurwechsel einleiten kann, ist das Fahrzeug weit genug am linken Hindernis vorbeigefahren.

Ein relevantes Hindernis befindet sich auf der linken Fahrspur, falls in der obstacleMap die Zelle *leftFront* belegt und die Vorderseite des Hindernisses näher als die Vorderkante des auszuweichenden Hindernisses ist. Wurde das linke Hindernis ausreichend weit überholt, befindet sich das Fahrzeug nicht mehr vor dem Hindernis, so dass die Zelle *leftFront* wieder frei ist, bzw. ein weiter hinten stehendes Hindernis eingetragen ist. Der Ausweichvorgang wird eingeleitet, falls gilt:

```
leftFront frei ||
leftFront.distenceFront > midFront.distanceFront
```
(4.1)

Zum Einleiten des Ausweichvorgangs wird ein aktiver Wechsel der Fahrspur angeregt (Kapitel 1.5.2). Hierfür wird jeweils eine entsprechende Datenstruktur der Fahrspurführung und POLARIS im Datencontainer abgespeichert. Für die Fahrspurführung wird eine SteeringControlData–Datenstruktur gespeichert, die die Angaben enthält, dass die linke Fahrspur befahren werden soll und dass der Zielpunkt für die Fahrzeugführung näher am Fahrzeug liegen muss. Für POLARIS wird eine PolarisControlData–Datenstruktur abgespeichert, die die Angabe enthält, dass die linke Fahrspur überwacht werden soll.

Das Hindernis befindet sich rechts neben dem Fahrzeug, wenn die Zelle *right* belegt ist. An dieser Stelle ist der Fahrspurwechsel vollzogen. Die linken Blinker werden ausgeschaltet. Es erfolgt der Zustandswechsel in den Folgezustand D.

Zustand C

Der Zustand C ist ein Notfallzustand. Wurde ein Hindernis zu spät erkannt oder war das Fahrzeug zu schnell, um ein erfolgreiches Ausweichmanöver vorzunehmen, wird in diesen Notfallzustand gewechselt. In diesem Zustand wird die Fahrspurführung komplett überschrieben.

Zunächst wird die maxForwardSpeed reduziert, damit das Fahrzeug beim nächsten Versuch die Fahrspur zu wechseln eine kürzere Strecke benötigt. Das Fahrzeug wird durch langsames Rückwärtsfahren in die notwendige Ausweichdistanz gebracht. Hierfür wird die Geschwindigkeit auf maxBackwardSpeed gesetzt. Fährt das Fahrzeug vorwärts, bremst die Fahrzeugmechanik zunächst ab, bevor es rückwärts fährt. Der Zustandswechsel in den Zustand B erfolgt, wenn gilt:

```
midFront.distanceFront >= minAvoidDistance
```

Es wird minAvoidDistance und nicht maxAvoidDistance gewählt, da das Fahrzeug nach Verlassen des Zustands noch einen Zyklus die volle Geschwindigkeit maxBackwardSpeed rückwärts fährt, bis der Zustand B eintrifft und wieder eine Geschwindigkeitsangabe für eine Vorwärtsfahrt gegeben ist. Auch beim Wechsel von der Rückwärts- in die Vorwärtsfahrt bremst die Fahrzeugmechanik ab. So ist gewährleistet, dass ein ausreichender Abstand zum Hindernis gegeben ist.

Zustand D

Hat der Fahrspurwechsel auf die linke Fahrspur erfolgreich stattgefunden, befindet sich das Hindernis neben dem Fahrzeug und der Zustand D ist erreicht. Das Fahrzeug fährt auf der linken Fahrspur, parallel zum Hindernis. Hierbei übernimmt die Fahrspurführung die Fahrt innerhalb der Fahrspur. Für diese wird eine SteeringControlData–Datenstruktur in den Datencontainer abgespeichert, die die Angaben enthält, dass die linke Fahrspur befahren werden soll und dass der Zielpunkt für die Fahrzeugführung näher am Fahrzeug liegen muss.

Während der Parallelfahrt wird beobachtet, ob sich Hindernisse vor dem Fahrzeug befinden. Hierbei sind im Wesentlichen zwei Situationen zu berücksichtigen:

- Es werden keine Hindernisse vor dem Fahrzeug erkannt, bzw. sind sie weit genug entfernt, so dass sie den Fahrspurwechsel zurück auf die rechte Fahrspur nicht beeinträchtigen.
- Ein weiteres Hindernis befindet sich hinter dem ersten Hindernis auf der rechten Fahrspur (Abb. 4-3). Das zweite Hindernis steht so nah am ersten, dass das Fahrzeug keine zwei Fahrspurwechsel (auf die rechte Fahrspur zurück und dann wieder auf die linke, um dem zweiten Hindernis auszuweichen) durchführen kann. Die minimal nötige Entfernung zwischen den beiden Hindernissen entspricht der maxAvoidDistance plus der stopDistance. Ist dieser Mindestabstand nicht gegeben, wird die Fahrt auf der linken Fahrspur verlängert, bis das zweite Hindernis passiert wurde.

Wenn eine dynamische Anpassung der Geschwindigkeit und des Startpunktes für einen Fahrspurwechsel möglich wären, müsste zusätzlich berücksichtigt werden, dass sich ein weiteres Hindernis auf der linken Fahrspur (Abb. 4-4), befindet, so dass die Geschwindigkeit des Fahrzeugs angepasst werden muss, damit das zweite Hindernis nicht beim Zurückfahren auf die rechte Fahrspur berührt wird. Reicht eine Anpassung der Geschwindigkeit nicht aus, müsste ein Zustand C* eingefügt werden, der dem Zustand C ähnelt.

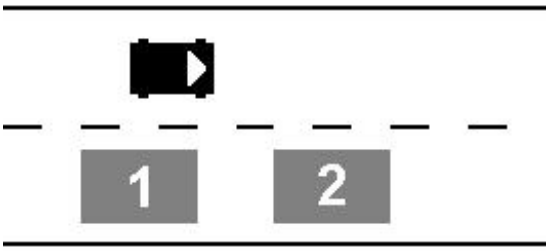

Abb. 4-3: Schematische Darstellung einer Parallelfahrt mit zwei hintereinander stehenden Hindernissen.

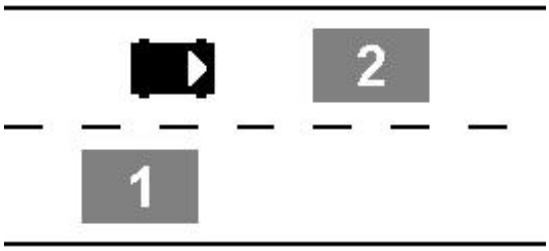

Abb. 4-4: Schematische Darstellung einer Parallelfahrt mit einem zweiten Hindernis auf der linken Fahrspur.

Der Zustand D ist frühestens beendet, falls das Ende des auszuweichenden Hindernisses bekannt ist. Sie ist bekannt, wenn gilt:

```
right belegt && (right.distanceFront < carFrontPos)
```
(4.2)

Um die zweite Hindernissituation (Abb. 4-3) zu berücksichtigen, muss für die Zelle *rightFront* zusätzlich folgende Bedingung erfüllt sein:

```
rightFront frei ||
(rightFront.distanceFront > 2*minAvoidDistance)
```
(4.3)

Nun ist gewährleistet, dass ein Wiedereinscheren möglich ist und der Zustandswechsel in den Zustand E erfolgt.

Zustand E

Ist der Bereich vor dem Fahrzeug frei, so dass ein Fahrspurwechsel zurück auf die rechte Fahrspur erfolgen kann, ist Zustand E erreicht. Um den Fahrspurwechsel anzuzeigen, wird zunächst der rechte Blinker aktiviert.

Zum Einleiten des Fahrspurwechsels wird wiederum ein aktiver Wechsel der Fahrspur angeregt. Hierfür wird für die Fahrspurführung und POLARIS jeweils eine entsprechende Datenstruktur in den Datencontainer gespeichert. Für die Fahrspurführung wird eine SteeringControlData–Datenstruktur gespeichert, die die Angaben enthält, dass nun die rechte Fahrspur befahren werden soll und dass der Zielpunkt für die Fahrzeugführung näher am Fahrzeug liegen muss. Für POLARIS wird eine PolarisControlData–Datenstruktur gespeichert, die die Angabe enthält, dass die rechte Fahrspur überwacht werden soll.

Ein Fahrspurwechsel in den Anfangszustand A erfolgt unter Verwendung eines Timers, da keine Sensordaten zur Verfügung stehen, durch die ermittelt werden kann, ob das Fahrzeug wieder auf der rechten Fahrspur fährt. Die Timerzeit wurde empirisch aus dem Fahrspurwechsel auf die linke Fahrspur (Zustand B) ermittelt. Da ein Zustandswechsel erfolgen kann, bevor das Fahrzeug wieder exakt auf der rechten Fahrspur fährt, um gegebenenfalls genügend Platz

für einen weiteren Ausweichvorgang zu haben, wird der Timer auf 4/5 der benötigten Zeit eingestellt.

POLARIS bietet zwar die Methode getLateralOffset an, welche die Position des Fahrzeugs innerhalb der von POLARIS berechneten Fahrspur angibt, jedoch ist die Methode zu ungenau. Das liegt daran, dass die Polynome die Fahrspur im Bereich des Fahrzeugs nur ungenau approximieren, da dort keine Fahrspurmarkierungen durch die Bildverarbeitung erkannt werden. Zusätzlich reicht es nicht aus, zu wissen, ob die Fahrzeugfront bereits innerhalb der Fahrspur ist. Denn dieses gibt keine Aussage darüber, in welchem exakten Bereich sich das Fahrzeug auf der Fahrspur befindet.

5 Erfahrungsbericht

Die erste Version des Automaten zum Ausweichen der Hindernisse kam bereits im Carolo-Cup 2009 zum Einsatz. Der Automat benutzte noch keine Karte, sondern wertete die Sensorsignale direkt aus. In Ermangelung einer funktionierenden Fahrspurführung konnte diese Version nicht vor dem Wettkampf komplett getestet werden. Pünktlich zum Wettkampf funktionierte die Fahrspurführung, so dass der erste richtige Test während des Wettkampfes stattfinden konnte. Die erste Hindernisschikane bestand aus einem links und einem recht stehenden Hindernis mit einer Entfernung von einem Meter. Während des Fahrspurwechsels wechselte das Fahrzeug die Fahrspur nicht korrekt und kam dem rechten Hindernis zu nah, so dass die Rückwärtsfahrt einsetzte. Das Fahrzeug erkannte das sich nun hinter ihm befindliche linke Hindernis nicht richtig, wodurch es dieses in eine ausreichende Ausweichdistanz zurück schob. Das Berühren des Hindernisses führte zu Strafmetern. Beim zweiten Ausweichversuch kam das Fahrzeug bis neben das rechte Hindernis, dann aber hatten die Akkus nicht mehr genügend Ladung, so dass sich das Fahrzeug nicht mehr von der Stelle bewegte. Unser Team war zwar eins der wenigen, die diese Disziplin fuhren, leider aber das einzige mit einer negativen Fahrdistanz.

Daraus ergab sich folgendes Ergebnis: Die gesamte Umgebung des Fahrzeugs muss beobachtet werden, unabhängig davon, welches Ausweichverfahren genutzt wird. Um im knappen Zeitrahmen eine geeignete Karte zu entwickeln, wurde diese extrem auf das Aufgabengebiet zugeschnitten.

Während der Untersuchungen, in denen ein Fahrzeug zum Einsatz kam, fiel auf, dass die Geschwindigkeit und der Lenkwinkeleinschlag stark von der Akkuladung abhängig sind. Bei gleichen Werten für die Aktoren von Motor und Lenkung, entstehen unterschiedliche Resultate. Ist beispielsweise die Akkuladung niedrig, fährt das Fahrzeug bei gleicher Geschwindigkeitsangabe langsamer und die Lenkausschläge fallen geringer aus, so dass der Fahrspurwechsel eine weitere Strecke in Anspruch nimmt als bei einer vollen Akkuladung. An einem Tempomat wird gearbeitet, dieser war zum Implementationsstand noch nicht vorhanden. Aufgrund dieser Tatsache kann der Startpunkt zum Ausweichen nicht anhand der Geschwindigkeit berechnet werden, sondern wird fest vorgegeben. Die Entfernung zum Hindernis ab der der Ausweichvorgang starten soll (avoideDistance) wurde so gewählt, dass sowohl bei einem schwachen, als auch bei einem vollen Akku der Fahrspurwechsel ohne Kollisionen mit dem Hindernis abläuft.

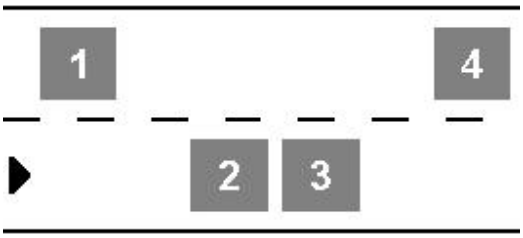

Abb. 5-1: schematische Darstellung des Testszenarios

Für die Kartenerstellung und das Ausweichverfahren wurden Hindernissituationen in geraden Streckenabschnitten, in einer Kurve und an einer Kreuzung getestet. Im geraden Streckenabschnitt wurden verschiedene Schikanen mit zwei Hindernissen getestet. In Abb. 5-1 ist eine Schikane dargestellt, die alle zu testenden Hindernissituationen beinhaltet. Stehen die Hindernisse 1 und 2 weit genug entfernt, wird die Vorbeifahrt an einem links stehenden Hindernis getestet. Stehen die Hindernisse 2 und 3 zusammen, bilden sie ein langes Hindernis. Stehen sie bis zu einer Entfernung von 2*minAvoidDistance (Kapitel 4) auseinander, ist kein Platz vorhanden, um zwischen den beiden Hindernissen die Fahrspuren zwei Mal zu wechseln, so dass die Fahrt auf der rechten Spur verlängert werden muss. Stehen die beiden Hindernisse weiter auseinander, muss zwischen ihnen der Ausweichvorgang für das Hindernis 2 beendet werden und für 3 neu gestartet. Die Hindernissituationen der Hindernisse 1, 2 und 3, 4 wurden mit dem Mindestabstand von einem Meter getestet. Dabei wurde darauf geachtet, dass der Ausweichvorgang bei vollen und nahezu leeren Akkus erfolgreich ablief. Die Distanzwerte der maxAvoideDistance und minAvoidDistance wurden hierbei empirisch ermittelt.

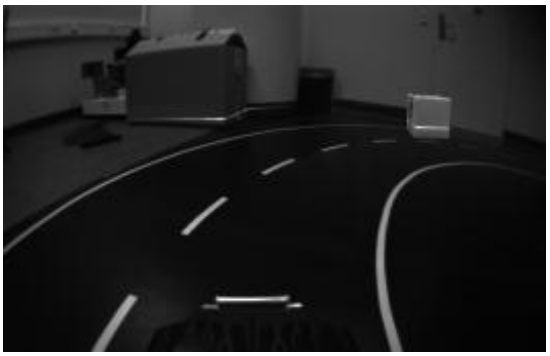

Abb. 5-2: Kamerabild einer Rechtskurve mit einem 1m vom Fahrzeug entfernt, auf der linken Fahrbahn positioniertem Hindernis

Im Kurvenbereich wurde für Links- und Rechtskurven jeweils getestet, wie sich das Fahrzeug bei einem links und einem rechts stehenden Hindernis verhält. Hierfür wurde das Hindernis jeweils an den Kurvenanfang, die Kurvenmitte und das Kurvenende positioniert. Dabei fiel auf, dass trotz korrekter Belichtungseinstellungen die Zuordnung von Fahrbahnmarkierung und Hindernissen nicht immer korrekt war. Dies liegt daran, dass POLARIS die Hindernisse als Fahrbahnmarkierungen fehlinterpretiert, wenn diese wie im Kapitel 1.5.1 beschrieben, nah an der Fahrbahnmarkierung liegen. Ist darüber hinaus von dem Hindernis, wie in Abb. 5-2, lediglich die Seite mit obsLaserRight zu erkennen, kann es dazu kommen, dass das Hindernis der rechten Fahrspur zugeordnet wird. Da dieses Phänomen selten auftrat und der Grund, weshalb das Hindernis falsch zugeordnet wurde, erst nach dem Carolo-Cup 2010 festgestellt werden konnte, wurde die Wahrscheinlichkeitsberechnung aus Kapitel 3.1 erst später entwickelt und bislang nur in Matlab implementiert und getestet. Durch die Wahrscheinlichkeitsberechnung soll dieser Fehler minimiert werden, da ausgehend von allen zur Verfügung stehenden Polynomen eine Wahrscheinlichkeit dafür ermittelt wird, auf welcher Position sich das Hindernis befindet.

Im Kreuzungsbereich wurde getestet, ob der Ausweichvorgang funktioniert, wenn ein Hindernis auf der rechten Spur einen Meter vor oder hinter der Kreuzung steht. Bei diesem Test wurde der Ausweichvorgang erfolgreich durchgeführt. Das dynamische Hindernis wurde bei der Entwicklung des Ausweichverfahrens und der auf das Ausweichverfahren abgestimmten Karte nicht berücksichtigt. In den vorfahrtsberechtigten Kreuzungen fährt das dynamische Hindernis quer über die Fahrbahn. Dies hat zur Folge, dass, nachdem der Kreuzungsbereich wieder frei ist, das Fahrzeug nicht fährt, da laut Karte beide Spuren belegt sind. Im Carolo-Cup 2010 wurde dies kurzfristig durch ein Resetten der Karte gelöst, wenn sich das Fahrzeug in einer vorgegebenen Zeit nicht bewegt.

6 Diskussion

Aufgrund des teilweisen Fehlens von Grundvoraussetzungen für einen Ausweichvorgang, wie z.B. der Kenntnis von Hindernissen vor dem Fahrzeug und deren Position auf der Fahrbahn, sowie einer Karte zur Umgebungsanalyse, wurden im Rahmen dieser Arbeit einige einfache Verfahren entwickelt um eine Grundlage zu schaffen, so dass das Fahrzeug einen Ausweichvorgang durchführen kann. Diese Grundlagen und das Ausweichverfahren werden in diesem Kapitel auf ihre Nachhaltigkeit diskutiert.

6.1 Erkennen von Hindernissen und deren Positionsbestimmung im Raum

Die Hindernisse im Bereich vor dem Fahrzeug werden durch die Kombination einer Kamera und eines Linienlasers realisiert. Der Linienlaser ist so montiert, dass er eine Laserlinie parallel zum Boden projiziert. Hierdurch werden nur Hindernisse vom Linienlaser angestrahlt und nicht die Fahrbahn. Die Idee dahinter ist, dass das Finden der Hindernisse durch die Laserlinie erleichtert wird, und die Laserlinie zusätzlich zur Positionsbestimmung der Hindernisse im Raum genutzt werden kann. Der Laserstrahl projiziert zwar eine helle Linie auf die Hindernisse, jedoch ist diese nicht konstant hell. Das Abschwächen des Lichts der Laserlinie zu den Seiten hin erschwert das Finden von Hindernissen in Kurven.

Für die Positionsbestimmung der Hindernisse im Raum anhand des Kamerabildes wird eine projektive Transformation genutzt. Dabei wird die Ebene des Bildes auf die Ebene, die der Linienlaser erzeugt, abgebildet. Um die projektive Transformation mit der Ebene der Laserlinie durchzuführen, muss diese zusätzlich zur Ebene der Fahrbahn kalibriert werden. Die bei der Kalibrierung der beiden Ebenen entstandenen Fehler summieren sich durch die getrennte Kalibrierung auf. Zusätzlich werden durch ein Neigen oder Kippen des Fahrzeugs die Ebenenverhältnisse zwischen Kamera und Fahrbahn sowie zwischen Kamera und Laserlinie gestört, so dass die Umwandlung von Bild- in Weltpunkte fehlerhaft ist.

Aufgrund dieser beiden Aspekte ist es fraglich, ob bei einer Verbesserung der Hinderniserkennung im Bild, weiter auf das Hilfsmittel der Laserlinie zurückgegriffen werden sollte. Wird beispielsweise die Unterkante der Hindernisse gefunden, kann auch für die Positionsbestimmung der Hindernisse die projektive Transformation der Fahrbahn genutzt werden. So entstehen beim Vergleich zwischen dem Hindernis und den Fahrspuren keine zusätzlichen Fehler.

Um die Berechnungszeit der Bildverarbeitung so gering wie möglich zu halten, wird das Bild in interessante Betrachtungsbereiche ROIs (*Region of Interests*) unterteilt. Hierzu werden die oberen Bildzeilen, in denen keine ausreichend genauen Angaben über die Fahrbahn möglich ist, entfernt. Dies geschieht auch mit den unteren Bildzeilen, in denen die Motorhaube des Fahrzeugs abgebildet ist. Eine Einschränkung des zu untersuchenden Bildbereichs in der Breite erfolgt nicht, da die Angaben der Fahrbahn in Kurven zum Entwicklungszeitpunkt zu fehlerhaft waren. Zur weiteren Einschränkung werden lediglich einzelne Bildspalten mit einem vordefinierten Abstand über die gesamte Bildbreite verteilt untersucht.

Um die Berechnungszeit so gering wie möglich zu halten, wurde das Finden der Laserlinie durch ein Schwellwertverfahren realisiert, welches den Vorteil der schnellen Umsetzung und einfachen Handhabung hat. Jeder untersuchte Pixel wird lediglich mit dem Schwellwert verglichen und kann dann klassifiziert werden. Da zwischen den Fahrspurmarkierungen und der Laserlinie unterschieden werden muss, werden zwei Schwellwerte eingesetzt. Der Nachteil liegt darin, dass für eine Unterscheidung von Fahrbahnmarkierung und Laserlinie die Einstellungen der Kamera präzise der jeweiligen Belichtung angepasst sein müssen. Bei einer stärkeren Reduktion des Bildbereichs wird Rechenzeit eingespart, die in rechenintensivere, dafür jedoch konsistentere Algorithmen zur Objekterkennung fließen kann. Da nur die Fahrbahn für die Hinderniserkennung interessant ist, könnte diese, bei einer stabileren Fahrspurerkennung, zur Reduktion des Bildbereichs genutzt werden.

6.2 Zuordnung der Hindernisse einer Fahrspur

Das vorgestellte Verfahren zur Zuordnung der Hindernisse zu einer Fahrspur ist das nach dem Carolo-Cup verbesserte Positionsbestimmungsverfahren. Anhand jedes zur Verfügung stehenden Polynoms wird eine Wahrscheinlichkeit nach dem Bayesschen Entscheidungstheorem berechnet, ob sich das Hindernis auf der linken oder rechten Fahrspur befindet, und anschließend zusammengefasst. Für die Berechnung der Wahrscheinlichkeit wird davon ausgegangen, dass die Mitte der Hindernisfront bekannt ist. Anhand der Entfernung von dieser und der linken bzw. rechten Fahrspurmitte auf Höhe des Hindernisses, wird jeweils eine Wahrscheinlichkeit berechnet, da sich die Hindernisse in der Regel mittig der Fahrbahn befinden. Es gibt jedoch Situationen, wie beispielsweise in Kurven, bei denen neben der Hindernisfront zusätzlich eine Hindernisseite zu erkennen ist. Derzeit wird die Seite als Hindernisfront gewählt, die näher am Fahrzeug ist, was dazu führen kann, dass die Hindernisseite als Hindernisfront fehlinterpretiert wird. Dies hat zur Folge, dass die zur Positions- und Entfernungsbestimmung gewählte Mitte nicht mehr mittig der Fahrspur liegt, sondern nahe einer Fahrspurmarkierung. Je näher das Hindernis der Fahrspurmarkierung positioniert ist, desto wahrscheinlicher ist eine Fehlinterpretation der Positionszuordnung zu einer Fahrspur. Es muss untersucht werden, ob durch eine geschickte Einstellung der Varianz (in der Wahrscheinlichkeitsberechnung) diese Hindernisse

trotzdem ausreichend genau der richtigen Fahrspur zugewiesen werden. Dieses Verfahren wurde bereits in Matlab implementiert und kann nun in die Kartenerstellung integriert werden.

Eine weitere Möglichkeit der Fehlinterpretation der Fahrspurzuordnung von Hindernissen ist das Einführen eines Zuverlässigkeitswertes. Je näher ein Hindernis dem Fahrzeug ist, desto eher wird die Hindernisfront und nicht eine Hindernisseite erkannt. Zusätzlich bilden die Fahrspurpolynome die Fahrbahnmarkierungen in der Nähe des Fahrzeugs korrekter ab, als in der Ferne. Durch diese beiden Eigenschaften gilt, je näher ein Hindernis ist, desto zuverlässiger ist die Wahrscheinlichkeitsangabe.

6.3 Karte

Zur Beobachtung der Umgebung wurde eine Karte erstellt, die auf das Aufgabengebiet zugeschnitten ist. Da sich das Fahrzeug lediglich auf der Fahrbahn befindet, wird die Karte auf die Informationen der linken und rechten Fahrspur minimiert. Die Fahrspuren werden als Spalten abgebildet, auf denen zeilenweise Hindernisse und das Fahrzeug eingetragen werden. Die Karte ist um das Fahrzeug herum aufgebaut und starr positioniert, so dass drei Spalten notwendig sind, um die beiden Fahrspuren abzubilden. Das hat den Vorteil, dass der Aufruf zum Überprüfen, ob sich ein Hindernis vor dem Fahrzeug befindet, immer gleich ist und nicht abhängig von der gerade befahrenen Fahrspur. Das hat zur Folge, dass die Spalten der Karte nicht immer dieselbe Fahrspur repräsentieren und die Einträge bei einem Fahrspurwechsel verschoben werden müssen. Das Verschieben der Karte bleibt dem Nutzer jedoch verborgen. Da ein Fahrspurwechsel der Hindernisse derzeit nicht vorgesehen ist, wird zum Einfügen von Hindernissen lediglich die Fahrspurspalte überprüft, die die Fahrspur, auf der das Hindernis steht, repräsentiert. Für dynamische Hindernisse und das Fehlinterpretieren der Hindernispositionen, sollten jedoch alle Zellen der Karte berücksichtigt werden.

Wird die Karte dahingehend verbessert, sollte überprüft werden, ob die Rechenzeitersparnis, die eine zwei-spaltige Karte gegenüber der drei-spaltigen Karte hat, die Vorteile des Kartenzugriffs aufwiegen. Die Rechenzeitersparnis bei der Kartenerstellung setzt sich aus zwei Vereinfachungen zusammen. Zum einen entfällt das Verschieben der Hindernisse, es muss lediglich das Fahrzeug verschoben werden. Zum anderen müssen zum Einfügen, bzw. Aktualisieren eines Hindernisses nur acht anstelle von zwölf Zellen untersucht werden.

6.4 Ausweichen

Für den Ausweichvorgang wurde ein Automat entwickelt, der durch eine Situationsanalyse der Umgebung entscheidet, ob ein Ausweichvorgang eingeleitet werden muss. Dabei wird anhand

einer Karte überprüft, ob die Hindernisse die nötige Entfernung haben, um gegebenenfalls einen Fahrspurwechsel einzuleiten. Für den Fahrspurwechsel selbst wird die bestehende Fahrspurführung, die das Fahrzeug in der Fahrspur halten soll, benutzt. Dies hat den Vorteil, dass kein weiteres Modul in die Fahrzeugführung eingreift. Der Nachteil liegt jedoch darin, dass es die Aufgabe der Fahrspurführung ist, eine möglichst ruhige Fahrt, mit kleinen Lenkwinkeln, zu erzeugen. Bei einem Ausweichvorgang sollte der Fahrspurwechsel gegebenenfalls schnell erfolgen, was durch große Lenkwinkel realisiert wird, um eine Kollision mit Hindernissen zu vermeiden. Durch einen speziell für den Ausweichvorgang entwickelten Fahrspurwechsel könnte dieser mit einer höheren Geschwindigkeit und kürzerer Strecke realisiert werden.

Ein in der Robotik gängiges Verfahren zum Ausweichen von Hindernissen ist die Potentialfeldmethode ([19]). Hierbei wird dem Fahrzeug zu jedem Zeitpunkt eine Richtung und Geschwindigkeit zugewiesen, um auf einer optimalen Strecke zu einem vorgegebenen Ziel zu gelangen. Während einer Fahrt zwischen Hindernissen, die einen engen Korridor bilden, kommt es zu oszillierenden Bewegungen ([1]). Die Fahrbahnmarkierungen des Carolo-Cups bilden einen solchen Korridor, weshalb das Verfahren nicht geeignet ist. In dieser Situation ist eine Erweiterung der Potentialfeldmethode durch ein *Vector Field Histogram* (VFH, [2]) möglich. Das VFH bietet gegenüber der Potentialfeldmethode den Vorteil, dass das Fahrzeug durch enge Passagen und Korridore mit hoher Geschwindigkeit und ohne Schlingern fahren kann. Für beide Verfahren sind die Kenntnis von exakter Geschwindigkeit und das korrekte Einstellen des Lenkwinkels notwendig.

Vor dem Einleiten des Ausweichvorgangs kann keine Abschätzung über dessen Länge getroffen werden, da das Ende des Hindernisses erst bekannt ist, sobald die Fahrzeugfront das Hindernis passiert hat. Da die statischen Hindernisse im Carolo-Cup mindestens einen Meter auseinander stehen, ist das Ausweichverfahren für den Wettbewerb geeignet. Für einen Ausweichvorgang an dynamischen Hindernissen ist eine Abschätzung über die Länge des Ausweichvorgangs notwendig, um zu erkennen, ob genügend Platz auf der Gegenfahrbahn ist. Zudem müssten bei dynamischen Hindernissen dessen Fahrinformationen mit in der Berechnung der Länge des Ausweichvorgangs berücksichtigt werden. Des Weiteren muss für einen kompletten Ausweichvorgang an dynamischen Hindernissen der Bereich hinter diesem bekannt sein, so dass ein Wiedereinscheren gewährleistet werden kann, um die Gegenfahrspur nicht unnötig lange zu blockieren. Durch den Einsatz einer globalen Karte, die den gesamten Rundkurs und die darauf befindlichen statischen Hindernisse beinhaltet, wäre diese Voraussetzung gegeben.

Der Ausweichvorgang ist beendet, wenn das Fahrzeug sich genügend weit auf der rechten Fahrspur befindet. Derzeit wird ein Timer verwendet, wobei die Timerzeit empirisch aus dem Fahrspurwechsel auf die linke Fahrspur ermittelt wurde. Die Timerzeit ist so gewählt, dass bei jeder Kurvensituation das Fahrzeug genügend weit auf der rechten Fahrspur befindet. Es wurde auf einen Timer zurückgegriffen, da zum Zeitpunkt der Entwicklung des Ausweich-Automaten, kein Verfahren verfügbar war, um festzustellen, wie weit sich das Fahrzeug auf der rechten Fahrspur befindet. POLARIS liefert einen Abstand der Fahrzeugmitte zur lateralen Fahrspur-

mitte. Da es sich um einen lateralen Abstand handelt, kann in Kurven keine genaue Angabe gemacht werden wie weit es sich zur tatsächlichen Fahrspurmitte befindet.

7 Fazit

Komplexere Ausweichverfahren benötigen eine genaue Bestimmung von Geschwindigkeit und Lenkwinkeln, welche hier nicht zu jeder Zeit gegeben ist. Je nach verbleibender Akkuleistung reicht beispielsweise die Kraft der Servomotoren für den vorgegebenen Wert der Lenkung nicht aus, um den gewünschten Lenkwinkel einzustellen. Zudem gibt es keine Rückmeldung, auf wie viel Grad die Lenkung tatsächlich eingeschlagen ist. An dieser Stelle besteht weiterer Untersuchungsbedarf. Ähnliche Problemstellungen ergeben sich in Bezug auf die Geschwindigkeit. Des Weiteren ist eine Darstellung der Umgebung notwendig, um eine Route durch diese zu planen.

Während der Entwicklungsphase des Ausweichverfahrens war keine Darstellung der Umgebung verfügbar. Um diese darstellen zu können, ist eine Identifikation von Hindernissen vor dem Fahrzeug und deren Position auf der Fahrbahn erforderlich. Da diese Grundvoraussetzungen für Ausweichverfahren nicht gegeben waren, musste für die Teilnahme am Carolo-Cup 2010 an der dynamischen Disziplin *Rundstrecke mit Hindernissen* schnell eine Lösung gefunden werden. Aus diesem Grunde wurde im Rahmen dieser Arbeit die Identifikation von Hindernissen anhand von Kamerabildern und deren Positionszuordnung auf der Fahrbahn sowie die Karte zur Darstellung der Umgebung entwickelt und implementiert.

Da die Entwicklung der Karte viel Zeit in Anspruch genommen hat, wurde für den Ausweichvorgang in dieser Arbeit lediglich ein einfacher Automat entwickelt, für den die genaue Bestimmung von Geschwindigkeit und Lenkwinkel jedoch nicht relevant ist. Die Karte dient im Ausweichverfahren der Umgebungsanalyse, damit gegebenenfalls ein Ausweichvorgang eingeleitet werden kann. Um den Fahrspurwechsel innerhalb des Ausweichens durchzuführen, wird die bereits vorhandene Fahrspurführung genutzt. Das Ziel des Ausweichverfahrens ist es, statischen Hindernissen ordnungsgemäß auszuweichen. Dieses Ziel konnte im Zuge dieser Arbeit erreicht werden. Von Vorteil ist, dass der Automat so entwickelt wurde, dass er ohne Umstände erweitert werden kann.

Quellenverzeichnis

[1] BORENSTEIN, J.; KOREN, Y.: *Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation*. April 1991, IEEE International Conference on Robotics and Automation, pp. 1398-1404

[2] BORENSTEIN, J.; KOREN, Y.: *The vector field histogram – fast obstacle avoidance for mobile robots*". Juni 1991, IEEE Journal of Robotics and Automation Vol 7, No 3, pp. 278-288.

[3] Duden-online: *ausweichen*. URL: http://www.duden.de/zitieren/10028245/1.8 (2012-01-10)

[4] DRESCHLER-FISCHER, L.: *Mastermodul „Bildverarbeitung 2" Bildverarbeitung in der Fernerkundung*. Universität Hamburg, 2011, Seite 504ff.

[5] Eos Systems Inc.: *Photomodeler*. 2008. URL: http://www.photomodeler.com/index.htm (2011-11-01)

[6] *FAUST Fahrerassistenz und autonome Systeme*. Hochschule für Angewandte Wissenschaften Hamburg. URL: http://www.informatik.haw-hamburg.de/faust.html (2012-01-13)

[7] IDS Imaging Development Systems GmbH: *USB UI-1226LE Datenblatt*. URL: http://www.ids-imaging.de/frontend/products.php?cam_id=12&sc=2 (2011-11-01)

[8] JENNING, E.: *Systemidentifikation eines autonomen Fahrzeugs mit einer robusten, kamerabasierten Fahrspurerkennung in Echtzeit*. Hochschule für Angewandte Wissenschaften Hamburg, 2008

[9] Kakos Bros Solutions: *Robot Path Planning*. URL: http://www.kakos.com.gr/page_1145700674781.html (2011-11-01)

[10] MANSKE, N.: *Kamerabasierte Präzisionsnavigation mobiler Systeme im Indoor-Bereich*. Hochschule für Angewandte Wissenschaften Hamburg, 2008

[11] MEINEL, C.; MUNDHENK, M.: *Mathematische Grundlagen der Informatik*. 5. Auflage, Wiesbaden: Vieweg+Teubner Verlag, 2011, Seite. 53f.

[12] MÜLLER-FONFARA, R.: *Mathematik verständlich*. Weltbild, 2006.

[13] NEHMZOW, U: *Mobile Robotik: Eine praktische Einführung*. Berlin: Springer, 2002

[14] NIKOLOV, I.: *NFQ zur Optimierung eines Lenkungsreglers*. Hochschule für Angewandte Wissenschaften Hamburg, 2010

[15] NIKOLOV, I.: *Verfahren zur Fahrbahnverfolgung eines autonomen Fahrzeugs mittels „Pure Persuit“ und „Follow-the-carrot“*. Hochschule für Angewandte Wissenschaften Hamburg, 2009

[16] Picotronic: *Red Line Lasermodule LH650-16-3(14x45)*. URL: http://media.picotronic.de/datasheets/LH650-16-3%2814x45%29.pdf (2011-11-01)

[17] Robotikhardware: *SRF10 – Ultraschall Entfernungsmessung – Dokumentation in deutsch und englisch*. Brall Software GmbH. URL: http://www.robotikhardware.de/download/srf10doku.pdf (2011-11-13)

[18] SHARP Microelectronics Europe: *GP2D12-DATA-SHEET*. SHARP, 2006. URL: http://www.sharpsme.com/webfm_send/1203 (23011-11-01)

[19] SICILIANO, B.; KHATIB, O. (Hrgs.): *Springer Handbook of Robotics*. Berlin-Heidelberg: Springer, 2008, Seite 839 f.

[20] Technische Universität Carolo-Wilhelmina zu Braunschweig (Hrsg.):*„Carolo-Cup“ Regelwerk 2010*. Braunschweig, 2009. URL: http://www.carolo-cup.de/nc/wettbewerb/2010/regelwerk/ (2011-11-05)

[21] Vishay: *TCUT1300X01 Datasheet*. Vishay Semiconductors, 2011. URL: http://www.vishay.com/optical-sensors/list/product-84756/ (2011-11-28)

Glossar

Bezeichnung	**Beschreibung**
carFront	Bereich vor dem Fahrzeug
maxAvoidDistance	Die maximal benötige Ausweichdistanz, um bei gegebener Geschwindigkeit zwischen zwei Hindernissen mit einem Meter Entfernung, die Fahrspur zu wechseln, ohne diese zu berühren (100cm)
carLeft	Bereich links neben dem Fahrzeug
carLength	Länge des Fahrzeugs in Zentimetern (47cm)
Carolo-Cup	Wettbewerb der Universität Carolo-Wilhelmina zu Braunschweig
carRear	Bereich hinter dem Fahrzeug
carRight	Bereich rechts neben dem Fahrzeug
carWidth	Breite des Fahrzeugs in Zentimetern (22cm)
COI	Column of interest; zu untersuchende Spalte im Kamerabild
detectRangeIR	Maximaler Entfernungswert der IR-Sensoren, um das erkannte Objekt als relevantes Hindernis zu klassifizieren (40cm)
FAUST	Fahrerassistenz- und Autonome Systeme; Projekt der HAW Hamburg
FAUSTcore	Softwarearchitektur innerhalb des Projektes FAUST.
FAUSTplugins	Die FAUSTplugins beschreiben die einzelnen Module des FAUSTcore
IObsWidthMin	Mindestbreite der Hindernisse im Kamerabild (in Pixeln) in der Entfernung WDMax
IOL	Bildkoordinate des am weitesten links liegenden, bekannten Randpunktes einer Laserlinie im Kamerabild
IOR	Bildkoordinate des am weitesten rechts liegenden, bekannten Randpunktes einer Laserlinie im Kamerabild
IRFrontPos	Abstand von der Fahrzeugfront zur Position der vorderen IR-

	Sensoren in Zentimetern
IRRearPos	Abstand von der Fahrzeugfront zur Position der hinteren IR-Sensoren in Zentimetern
IR-Sensor	Infrarot-Sensor
IXMax	x-Wert der Auflösung des Kamerabildes in Pixeln (752)
IYMax	y-Wert der Auflösung des Kamerabildes in Pixeln (480)
laneMiddle	Bezeichnung für die Mittelstreifenmarkierung im Kamerabild
laneOuterLeft	Bezeichnung für die linke Außenmarkierung im Kamerabild
laneOuterRight	Bezeichnung für die rechte Außenmarkierung im Kamerabild
laserLineObstacleDetection	Modul zur Hinderniserkennung durch Bildverarbeitung
LBrightMin	Helligkeit des dunkelsten Pixels, das einer Laserlinie zugewiesen werden kann
leftEdge	Fahrspurpolynom für die linke Fahrbahnmarkierung der von POLARIS beobachteten Fahrspur
mapRange	Reichweite der Karte obstacleMap in Zentimetern (200cm)
maxDistPerCyc	Entfernungsangabe in Zentimetern für die maximal möglich zurückgelegte Strecke des Fahrzeugs in einem Taktzyklus
maxBackwardSpeed	Geschwindigkeit der Rückwärtsfahrt, während der in die Fahrspurführung eingreifenden Phase des Ausweichautomaten
maxForwardSpeed	Geschwindigkeit der Vorwärtsfahrt, während der in die Fahrspurführung eingreifenden Phase des Ausweichautomaten
minUSRange	Die Mindestentfernung der hinteren US-Sensoren für ein Hindernis. Sie wird durch den Sensor und seine Befestigungsposition am Fahrzeug vorgegeben
obsLaserFront	Bezeichnung für die Laserlinie auf der Vorderseite der Hindernisse im Kamerabild
obsLaserLeft	Bezeichnung für die Laserlinie auf der linken Seite des Hindernisses im Kamerabild
obsLaserRight	Bezeichnung für die Laserlinie auf der rechte Seite des Hindernisses im Kamerabild
obstacleAvoidance	Modul für den Ausweichvorgang

obstacleMap	Modul für die Kartenerstellung
obstacles	LaserObstacleVector, in dem im Bildverarbeitungsschritt der Hinderniserkennung, für jedes Hindernis die beiden Bildkoordinaten IOL und IOR abgespeichert sind
P_L	Wahrscheinlichkeit dass das Hindernis auf der linken Fahrspur steht, ohne Berücksichtigung der Entfernungsungenauigkeit
POLARIS	Modul zur Fahrbahndetektion per Bildverarbeitung
PolarisLaneSide	Von POLARIS zur Verfügung gestellten Datentyps, der die Werte leftLane und rightLane annehmen kann, und somit die beobachtete Fahrbahnseite angibt
P_R	Wahrscheinlichkeit dass das Hindernis auf der rechten Fahrspur steht, ohne Berücksichtigung der Entfernungsungenauigkeit
rightEdge	Das Fahrspurpolynom für die rechte Fahrbahnmarkierung der von POLARIS beobachteten Fahrspur
ROI	Region of Interest; Interessanter Bildbereich
roiXMax	Die rechte Grenze der ROI im Kamerabild
roiXMin	Die linke Grenze der ROI im Kamerabild
roiYMax	Die untere Grenze der ROI im Kamerabild
roiYMin	Die obere Grenze der ROI im Kamerabild
scan	Vektor im Bildverarbeitungsschritt der Hinderniserkennung, in dem für jede COI ein Bildkoordinatenpaar für einen Hindernispunkt abgelegt ist.
scanHits	Mindestanzahl an COIs die ein Hindernis treffen müssen, damit dieses als ein Hindernis erkannt wird
SLAM	simultaneous localization and mapping ([19] Kapitel 37.2)
steeringControl	Modul zur Fahrspurführung
stopDistance	Die minimal benötige Distanz für ein Anhalten vor dem Hindernis, so dass dieses nicht berührt wird (20cm)
minAvoidDistance	Die minimal benötige Ausweichdistanz, um bei gegebener Geschwindigkeit zwischen zwei Hindernissen mit einem Meter Entfernung, die Fahrspur zu wechseln, ohne diese zu berühren (70cm)

US-Sensor	Ultraschall-Sensor
WDMax	Maximal gültige Entfernung in Zentimetern, in der die Fahrbahn- und Hinderniserkennung stabil ist
WObsWidthMin	Mindestbreite der Hindernisse in der Welt (in Zentimetern)
WOL	Korrespondierender Weltpunktkoordinate zu IOL, welche des am weitesten links liegenden, bekannten Randpunktes einer Laserlinie entspricht
WOR	Korrespondierender Weltpunktkoordinate zu IOR, welche des am weitesten rechts liegenden, bekannten Randpunktes einer Laserlinie entspricht
Δscan	Abstand zwischen zwei COIs

A Darstellung eines im Carolo-Cup möglichen Rundkurses

Die folgende Abbildung stellt Rundkurs des Carolo-Cups 2010 dar. Für die dynamischen Disziplinen: *Rundstrecke ohne Hindernisse* und *Rundstrecke mit Hindernissen*, wurden teilweise Fahrspurmarkierungen entfernt und Hindernisse auf der Fahrbahn positioniert.

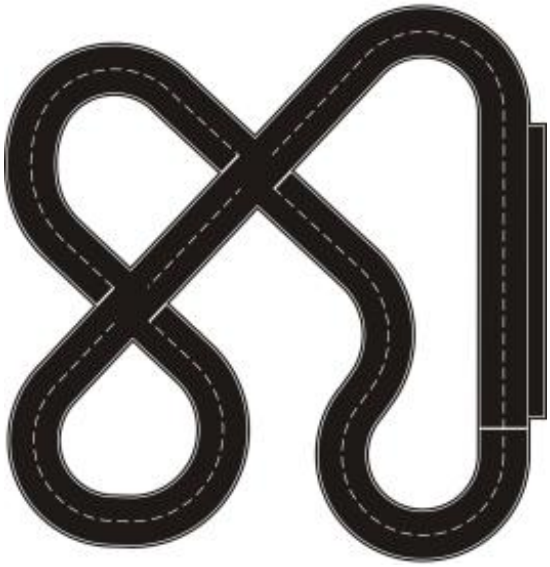

Abb. A-1: Rundkursaufbau des Carolo-Cups 2010 ([20] Kapitel 5.9)

B Beschreibung der Klassen zur Kommunikation mit dem Modul POLARIS

In der PolarisLane stellt POLARIS Informationen über die Fahrspuren zur Verfügung.

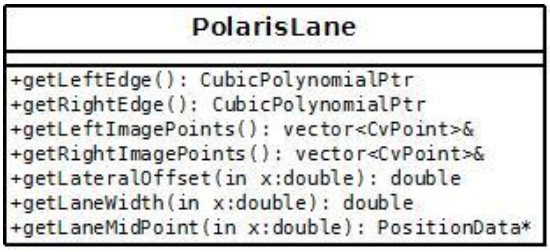

Abb. B-1: PolarisLane - Methoden

„getLeftEdge“ liefert das Polynom der approximierten linken Fahrspurmarkierung. Es entspricht der linken äußeren Fahrbahnmarkierung, wenn die linke Fahrspur überwacht wird, oder der Mittelstreifenmarkierung, wenn die rechte Fahrspur überwacht wird.

„getRightEdge“ liefert das Polynom der approximierten rechten Fahrspurmarkierung. Es entspricht der Mittelstreifenmarkierung, wenn die linke Fahrspur überwacht wird, oder der rechten äußeren Fahrbahnmarkierung, wenn die rechte Fahrspur überwacht wird.

„getLeftImagePoints“ liefert die zum linken Polynom linken Fahrspurmarkierung gehörenden Bildpunkte.

„getRightImagePoints“ liefert die zum rechten Polynom linken Fahrspurmarkierung gehörenden Bildpunkte.

„getLateralOffset“ liefert den lateralen Abstand in der Entfernung x zwischen dem vorderen Fahrzeugmittelpunkt und der Mitte der beobachteten Fahrspur in cm. Ist der Rückgabewert negative, befindet sich das Fahrzeug links der lateralen Mitte. Ist der Rückgabewert positiv, befindet sich das Fahrzeug rechts der lateralen Mitte.

„getLaneWitdh“ liefert den Abstand zwischen den beiden Polynomen in der Entfernung x.

„getLaneMidPoint“ liefert die x-y-Position der Mitte zwischen den beiden Fahrspurpolynomen an der gegebenen Entfernung x zurück.

Die PolarisControlData dient dazu, um POLARIS zu zwingen die linke oder rechte Fahrspur zu beobachten, sowie ein Reset durchzuführen.

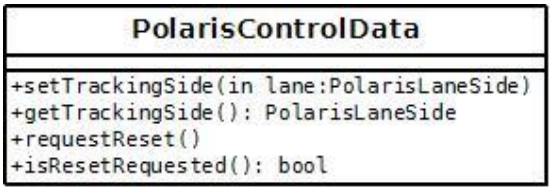

Abb. B-2: PolarisControlData - Methoden

„setTrackingSide“ setzt die Fahrbahnseite, die beobachtet werden soll. Wird von anderen Modulen als POLARIS gesetzt.

„getTrackingSide“ liefert die gesetzte trackingSide. (wird von POLARIS erfragt, und von map)

„requestReset“ wird diese Methode aufgerufen, wird ein privates resetFlag gesetzt, dass POLARIS dazu anregt sich zu resetten.

„isResetRequested“ liefert true zurück, falls das private resetFlag gesetzt wurde. Andernfalls wird false zurückgegeben.

C Auszug der zur Verfügung stehenden Komponenten des Fahrzeugs Onyx

Die hier vorgestellten Komponenten des Fahrzeugs Onyx, aus dem Projekt FAUST der HAW Hamburg ([6]), entsprechen dem Stand des Fahrzeugs zu der Entwicklung der Masterarbeit aus dem Jahr 2010. Im Anhang C.1 sind die technischen Daten der zur Verfügung stehenden Recheneinheit angegeben. Im Anhang C.2 werden die genutzten Sensoren, sowie deren Position am Fahrzeug und ihr Einsatzgebiet für diese Arbeit beschrieben.

C.1 Recheneinheit

Firma+Bezeichnung	Acer: Aspire One
Technische Daten:	
Prozessor	Atom N270 mit 1,6 GHz
Chipsatz	Intel 945GSE
Festplatte	Flash-Speicherdisc mit 8 GByte
RAM	512MB
Laufwerke	keine optischen, dafür ein 5-in-1 Kartenleser
Anschlüsse	3x USB, 1x RJ45 (100MBit), VGA-Port, 54-MBit-WLan
Betriebssystem	Linux
Energieverbrauch	11,1W - 18,8W

C.2 **Sensoren**

Kamera ([7])

Firma+Bezeichnung	iDS: UI-1226LE-M-GL
Position	Oben
Ausrichtung	Vorne
Einsatzgebiet	zur Identifizierung der Fahrbahn und Hindernissen
Erfassungswinkel	140° (durch Weitwinkelobjektiv)
Probleme	Linsenverzeichnung

Ultraschallsensoren ([17])

Firma+Bezeichnung	Devantech: SRF10
Position	2 Vorne, 2 Hinten
Ausrichtung	Längsachse des Fahrzeugs
Einsatzgebiet	Zur Identifizierung von Hindernissen und Parklücken
Erfassungswinkel	ca. 70°
Funktionsprinzip	Ein Schallgeber wird für kurze Zeit angeregt, und versendet kegelförmige Schallwellen. Von den empfangenen Schallimpulsen wird durch dessen Laufzeit auf die Entfernung des Objektes geschlossen.
Probleme	Großer Erfassungswinkel.
	Crosstalk: Bei gleichzeitigem Betrieb von mehreren Ultraschallsensoren kann ein Sensor das Echo einer Welle empfangen, die von einem anderen Sensor gesendet wurde.
	Totalreflektion: Objekt reflektiert die Welle so, dass das Echo am Sensor vorbei geleitet wird.

Linienlaser ([16])

Firma+Bezeichnung	PicoTronic: LH650-16-3

Position	Vorne
Ausrichtung	paralleler Strahl zum Boden
Einsatzgebiet	Zur Markierung von Hindernissen
Probleme	Lichtstrahl verblasst und wird schmaler zu den Seiten hin.

Infrarotsensor([18])

Firma+Bezeichnung	Sharp: GP2D12
Position	2 Vorne, 2 Hinten
Ausrichtung	Seitlich, links und rechts
Einsatzgebiet	Zur Identifizierung von Parklücken und parallelen Hindernissen
Erfassungswinkel	Ca. 4,5°
Reichweite	10-80cm
Probleme	Ungünstige Reichweite und Erfassungswinkel für den Einsatz im Bereich vor dem Fahrzeug.

Inkrementalgeber ([21])

Firma+Bezeichnung	Vishay: TCUT-1300
Position	Vorderräder: links + rechts
Einsatzgebiet	Zur Entfernungsmessung

D Anhang zur Positionsbestimmung der Hindernisse

D.1 Umwandlung des Steigungsdreiecks:

Da sich die Mittelpunkte M_L und M_R auf der Gerade *n(x)* befinden (siehe Abb. 3-1), ist die Steigung zwischen H und M_R bekannt, nämlich m_n. Zur Umformung nach y_r wird das Steigungsdreieck zur Hilfe genommen:

$$m_n = \frac{y_r - y_m}{x_r - x_m}$$
$$m_n * (x_r - x_m) = y_r - y_m$$
$$y_r = m_n * (x_r - x_m) + y_m$$

D.2 Zuordnung des Operators aus Gleichung (3.10)

$f_l(x)$	M_L	M_R
$m_n \geq 0$	+	+
$m_n < 0$	-	-
d	20= halbe linke Fahrspurbreite	62= linke Fahrspurbreite + Breite der Mittelstreifenmarkierung + halbe rechte Fahrspurbreite

$f_m(x)$	M_L	M_R
$m_n \geq 0$	-	+
$m_n < 0$	+	-
d	20= halbe linke Fahrspurbreite	20= halbe rechte Fahrspurbreite

$f_r(x)$	M_L	M_R
$m_n \geq 0$	-	-
$m_n < 0$	+	+
d	62= rechte Fahrspurbreite + Breite der Mittelstreifenmarkierung + halbe linke Fahrspurbreite	20= halbe rechte Fahrspurbreite

E Kartenerstellung

Beispielcode für das Verschieben der Karte bei einem Fahrspurwechsel von der rechten auf die linke Fahrspur:

```
if(obsMap->isOccupied(front2nd)){
   obsMap->setCell(rightFront2nd, obsMap->getObstacle(front2nd));
}else{
   obsMap->clearCell(rightFront2nd);
}

if(obsMap->isOccupied(ObstacleMap::leftFront2nd)){
   obsMap->setCell(front2nd, obsMap->getObstacle(leftFront2nd));
}else{
   obsMap->clearCell(front2nd);
}
obsMap->clearCell(leftFront2nd);
```

Beispielcode für die Aktualisierung der Karte mittels Inkrementalgeberdaten:

```
ieLOld = ieLNew;
ieLNew = sensorValues->getDistanceLeft();
ieLClock = ieLNew - ieLOld;

ieROld = ieRNew;
ieRNew = sensorValues->getDistanceRight();
ieRClock = ieRNew - ieROld;

calcDist = (ieLClock/2)+(ieRClock/2);
```

Beschreibung der Klasse ObstacleObject

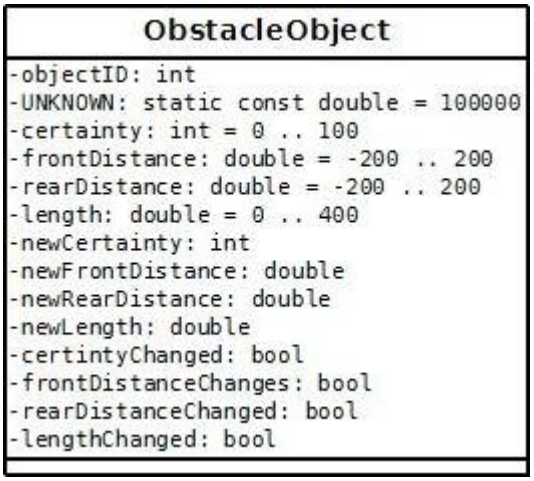

Abb. E-1: ObstcleObject - Attribute

Die Abb. E-1 zeigt die wichtigsten Attribute des ObstacleObjects.

Die „objectID“ dient der Identifikation eines Hindernisses. Sie wird beim Erzeugen eines neuen Hindernisobjektes generiert und zugewiesen.

Im Datenfeld „certainty“ ist für eine Wahrscheinlichkeit für die korrekte Positionsangabe vorgesehen, jedoch noch nicht implementiert.

Die Abstände „distanceFront“ und „distanceRear“ haben einen Wertebereich von -200cm bis 200cm, was jeweils der mapRange entspricht. Die Entfernungsangaben entstammen dem Fahrzeugkoordinatensystem. Die vordere Stoßstange des Fahrzeugs markiert die Nulllinie. Hindernisse vor dem Fahrzeug haben einen positiven Abstandswert (Abb. 3-3). distanceFront gibt den Abstand von der Fahrzeugfront zur Vorderkante des Hindernisses an. distanceRear gibt den Abstand von der Fahrzeugfront zur Hinterkante des Hindernisses an. Beide Felder sind mit dem Wert UNKNOWN initialisiert.

Die Länge der Hindernisse wird im Datenfeld „length“ abgespeichert. Es ist mit dem Wert UNKNOWN initialisiert. Sind distanceFront und distanceRear bekannt, kann die Länge berechnet und ein gültiger Wert in length abgespeichert werden.

Zu jedem der öffentlichen Datenfelder (außer objectID) des ObstacleObjects gibt es zwei zusätzliche Felder, die für eine Aktualisierung der Karte notwendig sind. Das erste zusätzliche Feld ist mit new gekennzeichnet und vom selben Typ wie das Ausgangsfeld (z.B. double newDistanceFront). Alle Änderungen der öffentlichen Datenfelder werden zunächst temporär im entsprechenden new-Feld abgespeichert. Erst nachdem alle Sensoren überprüft wurden, wird die Karte durch eine update-Funktion aktualisiert. Dabei werden die temporären Daten festgeschrieben. Das zweite zusätzliche Datenfeld ist ein Boolean und mit Changed gekennzeichnet (z.B. DistanceFrontChanged). Wurde ein Datenwert durch einen Sensor verändert, wird das entsprechende Changed-Feld automatisch auf true gesetzt, ansonsten bleibt es false.

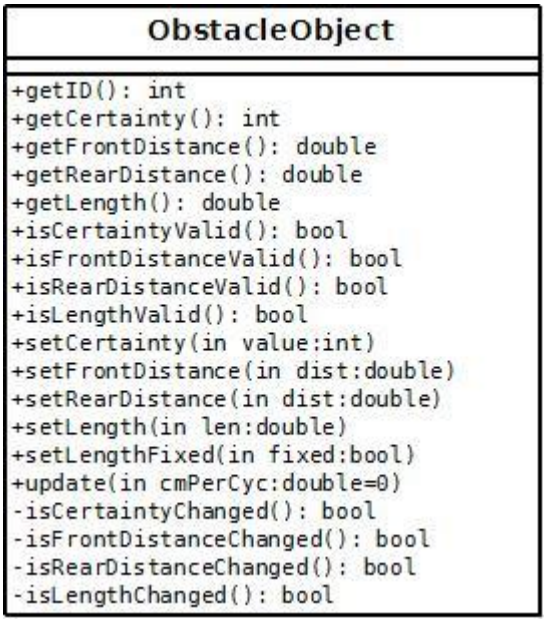

Abb. E-2: ObstacleObject - Methoden

Die Abb. E-2 zeigt die wichtigsten Methoden des ObstacleObjects. Eine Beschreibung folgt:

„get*AttributName*“ liefert den fest abgespeicherten Wert des entsprechenden Attributs, nicht jedoch die temporär gespeicherten Daten. Ist der Wert nicht gültig, wird ein Wert UNKNOWN zurückgegeben.

„is*AttributName*Valid“ liefert true zurück, falls im entsprechenden Attribut ein gültiger Wert abgespeichert ist. Die objectID besitzt keine isValid-Methode.

Mit „set*AttributName*“ wird jedes Attribut zunächst temporär durch das zugehörige new-Attribut erneuert. Zusätzlich wird das entsprechende Changed-Attribut auf true gesetzt. Die objectID besitzt keine set-Methode.

„update“ Alle Attribute erhalten die temporären Daten (Ausgenommen ist objectID). Ist das entsprechende Changed-Attribut von distanceFront, distanceRear oder length nicht gesetzt, wurden diese nicht durch einen Sensor verändert. distanceFront und distanceRear werden durch den Übergabeparameter cmPerCyc aktualisiert. Das Feld length wird angepasst.

„is*AttributName*Changed“ liefert den Wert des entsprechenden Changed-Attributs.

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 16. Januar 2012 _____

Ort, Datum Unterschrift