



# **Gitterbasierte Umfeldmodellierung mittels Laserscanner für das automatisierte Fahren**

## **Studienarbeit**

erstellt von cand. mach.  
Braunschweig, im Januar 2021

Technische Universität Braunschweig  
Institut für Fahrzeugtechnik  
Direktor: Prof. Dr.-Ing. Ferit Küçükay  
Betreuer: M.Sc. Marcel Mascha



Aufgabenstellung (Original bzw. Kopie)

## **Sperrklausel**

Die Ausgabe der vorliegenden Studienarbeit mit dem Titel „Gitterbasierte Umfeldmodellierung mittels Laserscanner für das automatisierte Fahren “ ist ausschließlich unter Genehmigung der Institutsleitung zulässig.

Braunschweig, den 04.Januar 2021

## Eidesstattliche Erklärung

Ich versichere an Eides statt, dass ich die vorliegende Studienarbeit mit dem Titel „Gitterbasierte Umfeldmodellierung mittels Laserscanner für das automatisierte Fahren“, ohne unerlaubte fremde Hilfe oder Beratung und nur unter Verwendung der angegebenen wissenschaftlichen Hilfsmittel angefertigt habe.

Braunschweig, den 04. Januar 2021

\_\_\_\_\_  
Xiantao Chen

## **Kurzfassung**

# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>IX</b>
<b>Tabellenverzeichnis</b>	<b>X</b>
<b>Abkürzungsverzeichnis</b>	<b>XI</b>
<b>Symbolverzeichnis</b>	<b>XII</b>
<b>1 Einleitung</b>	<b>1</b>
1.1 Einführung . . . . .	1
1.2 Zielstellung . . . . .	2
1.3 Aufbau . . . . .	2
<b>2 Theoretische Grundlagen</b>	<b>3</b>
2.1 Umfeldmodellierung . . . . .	3
2.1.1 Mögliche Umfeldmodelle . . . . .	3
2.1.2 Objektbasierte Modelle . . . . .	4
2.1.3 Gitterbasierte Modelle . . . . .	5
2.1.4 Hybride Umfeldmodelle . . . . .	12
2.2 Sensorik und ihr inverses Sensormodell . . . . .	13
2.2.1 Überblick über die verschiedenen Sensoren zur Umfeldmodellierung . . . . .	13
2.2.2 Funktionsweise des Lasersensors . . . . .	17
2.2.3 Theorie des inversen Sensormodells . . . . .	18
2.2.4 Technische Details von Ibeo-LUX . . . . .	19
2.2.5 Das zu verwendende Sensormodell . . . . .	20
<b>3 Implementierung</b>	<b>24</b>
3.1 Versuchsfahrzeug . . . . .	24
3.1.1 Dimension über Versuchsfahrzeug . . . . .	24
3.1.2 Einbauposition der Ibeo-Laserscanner . . . . .	25
3.2 Framework ROS zur Implementierung . . . . .	26
3.2.1 Grundlagen der ROS-Architektur . . . . .	27
3.2.2 Visualisierung des Umfeldmodells in ROS . . . . .	30

---

3.3	Koordinatensysteme . . . . .	34
3.3.1	Global Coordinate System (GCS) . . . . .	35
3.3.2	Vehicle Coordinate System (VCS) . . . . .	35
3.3.3	Anchor Coordinate System (ACS) . . . . .	36
3.3.4	Zusammenhang zwischen Koordinatensystemen . . . . .	37
3.4	Verarbeitung von Sensordaten . . . . .	38
3.4.1	GPS-Information . . . . .	39
3.4.2	Laserscanner-Information . . . . .	40
3.4.3	Synchronisation der Sensordaten . . . . .	46
3.5	Implementierung von Occupancy Grid . . . . .	46
3.5.1	Raumdiskretisierung . . . . .	47
3.5.2	Probabilistischer Ansatz . . . . .	47
<b>4</b>	<b>Implementierung</b>	<b>52</b>
4.1	Unterkapitel 1 . . . . .	52
4.1.1	Unterunterkapitel1 . . . . .	52
4.1.2	Das zu verwendendes Sensormodell . . . . .	53
	<b>Literatur</b>	<b>56</b>
	<b>Anhang</b>	<b>59</b>



## Abbildungsverzeichnis

2.1	Mögliche Umfeldmodelle für autonome Fahrzeuge . . . . .	4
2.2	Ein objektbasiertes Umfeldmodell [FHR <sup>+</sup> 20] . . . . .	5
2.3	Zwei Ebenen von Occupancy Grid . . . . .	6
2.4	Raumdiskretisierung der Umgebung um Fahrzeug [WHLS15] . . . . .	6
2.5	Bestandteile des Modells Ocuupancy Grid . . . . .	7
2.6	Hidden Markov Model (HMM) [TBF05] . . . . .	8
2.7	Das reduzierte HMM [TBF05] . . . . .	10
2.8	Bestandteile des Prinzips von Inverses Sensormodell . . . . .	18
2.9	Das ideale Sensormodell . . . . .	21
2.10	Das inverse Sensormodell, wenn ein Hindernis detektiert wird . . . . .	22
2.11	Das inverse Sensormodell, wenn kein Hindernis detektiert wird . . . . .	23
3.12	Dimension von Versuchsfahrzeug . . . . .	25
3.13	Einbauposition und Erfassungsbereich der Ibeo-Laserscanner . . . . .	26
3.14	ROS-Architektur . . . . .	28
3.15	Wesentliche Grundkonzepte von Berechnungsdiagrammebene . . . . .	28
3.16	Display mit Map . . . . .	31
3.17	Display mit GridCell . . . . .	33
3.18	Display mit GridCell nach Binärisierung . . . . .	33
3.19	3 wesentliche Koordinatensysteme im Umfeldmodell . . . . .	34
3.20	Anordnung der Position des Autos auf der Karte [Heg18] . . . . .	37
3.21	Programmablaufplan der Aktualisierung von ACS . . . . .	39
3.22	Initialisierung von ACS . . . . .	40
3.23	Umrechnung des Orientierungswinkels in GCS . . . . .	41
3.24	Kartierungsalgorithmus mit Objektinformation von Laserscanner [Heg18] . . . . .	41
3.25	Programmablaufplan der Laserscannerdaten . . . . .	43
3.26	Darstellung eines Hilfskoordinatensystem . . . . .	44
3.27	Implementierung des inversen Sensormodells . . . . .	50
3.28	Raycasting mit Bresenham-Algorithmus . . . . .	51
4.29	Folgefahrt Beispielbild . . . . .	52

## Tabellenverzeichnis

2.1	Der Algorithmus des Bayes-Filters [TBF05] . . . . .	9
2.2	Vorteile und Nachteile von Kamera, fernes Infrarotsensor, Radar-, Ultraschall- und Lidarsensor [MAA <sup>+</sup> 20] . . . . .	14
2.3	Technische Details von Ibeo LUX . . . . .	20
3.1	Abmessung von Versuchsfahrzeuge . . . . .	25
3.2	Werte der Einbauposition und des Winkels des Anfangsstrahls jedes Sensors bei Golf 7 (TIAMO)) . . . . .	26
3.3	Werte der Einbauposition und des Winkels des Anfangsstrahls jedes Sensors bei Passat (TEASY 3) . . . . .	27
3.4	Display-Typen und ihre entsprechenden Message-Typen in RVIZ . . .	31
4.1	Tabular Umgebung: Simpel formatierte Tabelle . . . . .	53
4.2	Vorteile und Nachteile von Kamera, Radar-, Lidar- und Ultraschall- sensor . . . . .	54
4.3	Tabular technische Details von Ibeo LUX 8L . . . . .	55
4.4	Tabular Umgebung: Etwas komplexere Tabelle . . . . .	55

---

## **Abkürzungsverzeichnis**

**IfF**      Institut für Fahrzeugtechnik

## Symbolverzeichnis

$\Delta h_{\text{ax}}$	$[\text{m/s}^2]$	Überschwingweite
------------------------	------------------	------------------

# 1 Einleitung

Im ersten Kapitel wird auf die Einführung des Themas, das zu erreichende Ziel und der Aufbau dieser Arbeit eingegangen.

## 1.1 Einführung

Das Thema „Autonomes Fahren“ ist in den vergangenen Jahren immer weiter in den Vordergrund gerückt. Selbstfahrende Autos verlagerten sich allmählich von Laborentwicklungs- und Testbedingungen auf öffentliche Straßen. Die Konzept, dass Autos autonom fahren können, beschäftigte die Menschen schon in den 1920er-1930er-Jahren.

Um das Konzept praxistauglich zu machen, braucht es das kombinierte Wissen unter anderem aus Informatik, Fahrzeugtechnik, Sensorik, Mechatronik. Unter dem Begriff „Autonomes Fahren“ werden Fahrzeuge gefasst, die ohne Eingriff und Überwachung durch einen Menschen selbstständig sowie ziel gerichtet im Straßenverkehr fahren können.

Ingenieure für selbstfahrende Autos verfolgen aktuell zwei wesentliche und unterschiedliche Herangehensweisen für autonome Entwicklung, welche aus Robotik-Ansatz und Deep-Learning-Ansatz bestehen. In den vergangenen zwei Jahrzehnten hat der Robotik-Ansatz mit einer Menge von Beiträgen der zahlreichen Wissenschaftler und Technik sowohl in Akademie als auch in technologisch führenden Unternehmen großen Fortschritt gemacht.

Im Rahmen dieser Arbeit handelt es sich um ein statisches Umfeld. Sollte die Echtzeitanforderung berücksichtigt, ist der Deep-Learning Ansatz zu verzichten, denn es ist mit vielen Literaturen bewiesen, dass ein statisches Umfeld mit Robotik-Ansatz zutreffend und effizient modelliert werden kann.[TBF05]

Die zentrale Idee des Robotik-Ansatzes ist, dass die Unsicherheit in der Robotik sich unter Verwendung der Wahrscheinlichkeitsrechnung darstellen lässt.

## **1.2 Zielstellung**

Um das automatisierte Fahren zu verwirklichen, wird die Umfelderkennung bzw. die Umfeldmodellierung des Fahrzeugs zugrunde gelegt. Das Ziel der vorliegenden Arbeit ist auf Modellierung und dazu ihre Implementierung gerichtet. Es handelt sich bei dieser Arbeit um statische Umgebungen in städtischen Räumen.

## **1.3 Aufbau**

Nach dieser Einleitung wird im Kapitel 2 die technische Grundlagen, welche sich eng mit Erfassung und Modellierung eines statischen Umfelds beziehen.

## 2 Theoretische Grundlagen

Um das in Abschnitt 1.2 erwähnte Ziel zu erreichen, sollten zunächst die theoretische Grundlagen geschaffen werden, bevor die praktische Umsetzung bzw. Implementierung stattfindet. Im Folgenden werden Theorien zur Umfeldmodellierung vorgestellt. Offensichtlich erfordert die Umfeldmodellierung die Wahrnehmung der Umgebung, weshalb als nächstes das Sensorsystem für die Umgebungswahrnehmung erörtert wird.

### 2.1 Umfeldmodellierung

Für Perzeption eines autonomen Fahrzeuges gibt es im wesentlichen zwei Bestandteilen [BGC<sup>+</sup>19]. Zum einen ist die Eigenlokalisierung. Die aktuelle Pose, das heißt die Position und Orientierung des eigenen Fahrzeuges, ist hierbei zu ermitteln. Zum anderen ist die Umgebung um das Fahrzeug zu erfassen. Dies wird als Umfeldmodellierung genannt. Diese beiden Aspekte werden in vielen Forschungen und Anwendungen zu einem Problem zusammengefasst. Dies ist als SLAM (Simultaneous Localization And Mapping) bekannt. Jedoch trennen sich diese beiden Aspekte im Rahmen dieser Arbeit und die Aufmerksamkeit ist auf Umfeldmodellierung zu lenken. Hierbei wird eine Annahme getroffen, dass die Eigenpose des Fahrzeugs selbst ohne Information der Umfeldmodell ausreichend präzise ist. Das am IfF (Institut für Fahrzeugtechnik) bereits bestehende Framework verwendet bestimmte Algorithmen wie das Kalman-Filter, um genaue GPS-Informationen des Fahrzeugs zu erhalten. In Bezug auf Robotik und autonomes Fahren dient Umfeldmodellierung als ein kompaktes Verfahren, mit dem die Umgebung des Fahrzeugs mathematisch beschrieben werden kann [Pie13]. Dies hat eine unverzichtbare solide Grundlage für das Design und die anschließende Implementierung aller fortschrittlichen autonomen Fahrfunktionen gelegt.

#### 2.1.1 Mögliche Umfeldmodelle

Die zur Modellierung der Umgebung verwendeten Methoden basieren auf verschiedenen entsprechenden Umfeldmodellen. Die maßgeblichen Umfeldmodelle bauen auf er-

folgreichen Erfahrungen in dem Themenbereich der Robotik auf [Pie13]. Wie in Ab-

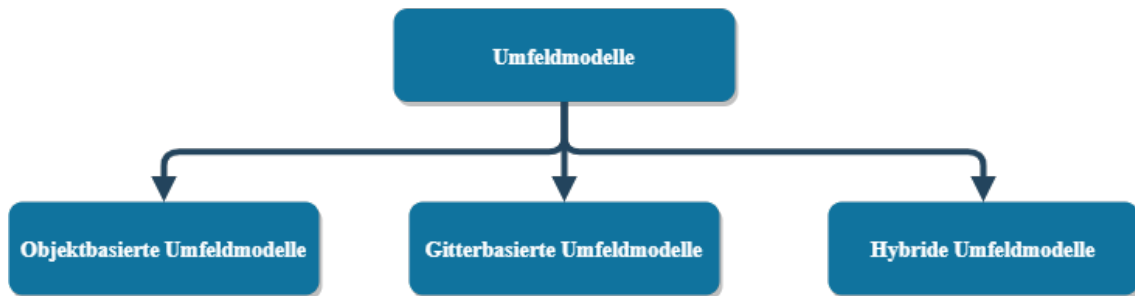


Abbildung 2.1: Mögliche Umfeldmodelle für autonome Fahrzeuge

bildung 2.1 dargestellt, stehen aktuell drei Hauptumweltmodelle in der Forschungs- und Automobilindustrie zur Verfügung [Heg18]. Sie sind objektbasierte Umfeldmodelle, gitterbasierte Umfeldmodelle und hybride Umfeldmodelle. In den folgenden Abschnitten werden sie vorgestellt.

### 2.1.2 Objektbasierte Modelle

Objektbasierte Modelle werden auch als merkmalsbasierte Umfeldmodelle (engl. landmark-based oder feature map) bezeichnet [TBF05]. Hierbei werden die Merkmalen, die für Erfassung der Umgebung günstig sind, durch Modelle beschrieben werden [WSG10]. Außerdem sollten die Merkmalen zuverlässig beobachtet und gemessen werden [Bus05]. Die Modelle werden in der Praxis als bestimmte Objektklasse beschrieben. Jede Klasse wird mit hervorragenden und maßgeblichen Eigenschaften versehen, die deutlich mittels zutreffenden Sensoren beobachtbar sind [WHLS15]. Für die Klassen lassen sich darüber hinaus die zu Objekt passende Geometrieformen bestimmen [Pie13]. Daher ist die Performance des Modells abhängig davon, ob die zu modellierten Objekte ausreichend präzise und einfach beschrieben werden können. Hierbei sind die Genauigkeit gegen den Zeit- und Speicheraufwand abzuwägen. Ein Beispiel ist das in Abbildung 2.2 gezeigte Umfeldmodell. Auf der linken Seite sind Merkmale dargestellt und das Bild rechts zeigt das generierte Umfeldmodell.

Objektbasierte Umfeldmodelle sind vor allem geeignet für Szenen, wo die Objekte entweder im großen offenen Raum mit vordefinierten Merkmalen (z.B. Autobahn) oder dynamisch und einfach modelliert (z.B. Fußgänger oder Fahrzeuge)



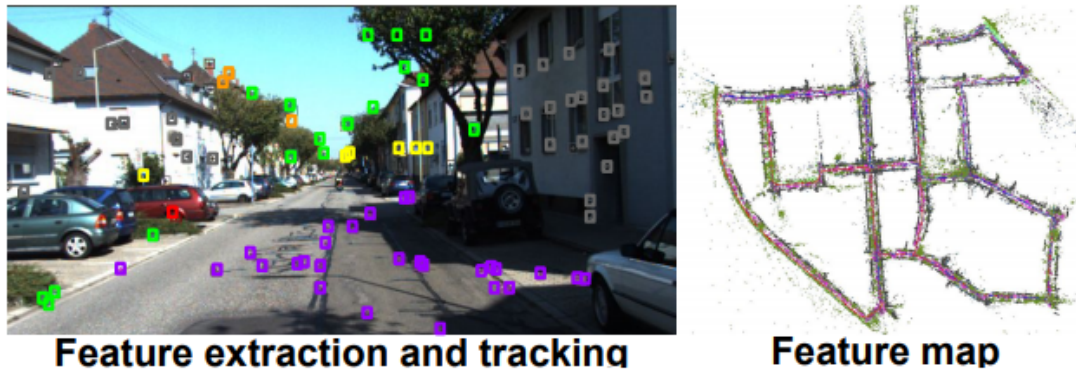


Abbildung 2.2: Ein objektbasiertes Umfeldmodell [FHR<sup>+</sup>20]

sind [Heg18] [WSG10]. Da es im Rahmen dieser Arbeit ein statisches Umfeld im urbanen Raum angeht, ist dieses Modell unangemessen und sollte daher verzichtet werden.

### 2.1.3 Gitterbasierte Modelle

In gitterbasierten Modellen wird die zentrale Idee der probabilistischen Robotik eingeführt, um die Wahrscheinlichkeitsverteilung unbeobachteter Zustände eines Systems mit gegebenen Beobachtungen und Messungen zu abschätzen. Zur Modellierung des Umfelds ist der zentrale und entscheidende Zustand der sogenannte Belegungszustand. Der Belegungszustand erweist sich, ob eine Gitterzelle belegt ist. Daraus ergibt sich eine Zufallsvariable  $X$ , den Belegungszustand repräsentiert. Die Variable  $X$  kann zwei Werte annehmen, welche 0 und 1 sind. Die entsprechen dabei, dass die Gitterzelle frei und belegt ist.

Das Modell, welches auf der obenerwähnten Idee aufbaut, wird als Occupancy Grid beschrieben. Das Occupancy Grid ist ein mehrdimensionales Zufallsfeld, das zum Speichern einer stochastischen Schätzung des Belegungszustands jeder Zelle im räumlichen Gitter verwendet wird [Elf89]. Wie in Abbildung 2.3 gezeigt, lässt sich Occupancy Grid in zwei Ebenen zerlegen.

Auf der ersten Ebene wird die Raumdiskretisierung ausgeführt. Im Rahmen dieser Arbeit wird das Umfeld als zweidimensionaler Raum dargestellt. Die Diskretisierung des Raums bedeutet hierbei, dass die Fahrzeugumgebung durch ein Raster diskreter und fester Größe dargestellt wird [Heg18]. Die grafische Darstellung ist in

1	Raumdiskretisierung
2	Probabilistischer Ansatz

Abbildung 2.3: Zwei Ebenen von Occupancy Grid

Abbildung 2.4 gezeigt. Zu jeder Zelle wird außerdem zusätzliche Informationen hinzugefügt. In Hinsicht auf Umfeldmodellierung ist die entscheidende Information der Belegungszustand der Zelle. Zur Vereinfachung der Modellierung, wird eine Annahme auf dieser Ebene zugleich getroffen, dass der Belegungszustand einer Gitterzelle ist unabhängig von derjenigen der Nachbarzellen. Obwohl diese Annahme mit der Realität unvereinbar zu sein scheint, ist es erwiesen, dass Occupancy Grid Modell in der Praxis unter dieser Annahme robust ist und zudem die Rechenkomplexität reduzieren kann [TBF05].

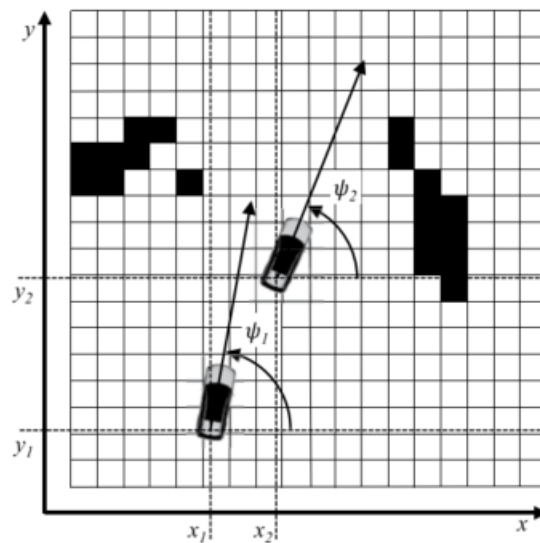


Abbildung 2.4: Raumdiskretisierung der Umgebung um Fahrzeug [WHLS15]

Auf der zweiten Ebene ist der probabilistische Ansatz eingeführt, welcher aus dem Themengebiet von Robotik stammt [Pie13]. Der Ansatz beruht auf Raumdiskretisierung und wird in der Praxis für jede einzelne Zelle eingesetzt. Der Ansatz lässt sich, wie in Abbildung 2.5 gezeigt, in viele Komponenten unterteilen. Im Zentrum

des Modells steht ein Algorithmus-Framework, das als binärer Bayes-Filter oder der binäre Bayessche Filter (engl. Binary Bayes Filter) bekanntlich ist. Die Eingaben des Frameworks umfassen Messungswerte, inverses Sensormodell und Anfangsbedingungen. Die Ausgabe ist A-posteriori-Wahrscheinlichkeit von dem Zufallsereignis, die Möglichkeit repräsentiert, dass eine Zelle als besetzt betrachtet wird. Um die Konzept von Occupancy Grid zu verdeutlichen, wird im Folgenden auf jede Komponente vertieft eingegangen.



Abbildung 2.5: Bestandteile des Modells Occupancy Grid

Der Bayessche Filter ist zuerst zu beleuchten, mit dem die Funktionen und Bedeutungen der anderen Bestandteile des Modells eng verbunden sind. Der Bayes-Filter ist eine rekursive Berechnungsvorschrift zur Schätzung von Wahrscheinlichkeitsverteilungen unbeobachteter Zustände eines Systems mit gegebenen Beobachtungen und Messungen [TBF05]. Die Zustandsschätzung befasst sich mit dem Problem der Schätzung von Größen, die nicht direkt beobachtbar sind, sondern abgeleitet aus den Sensordaten werden können. Das Ziel ist den Zustand  $x$  eines Systems zu schätzen<sup>1</sup>, wenn Beobachtung  $z$  und Kontrolle  $u$  angegeben sind. Das heißt, die in Gleichung 2.1 gezeigte mathematische Formel sollte bestimmt werden. Hierbei entspricht  $x_t$  dem Zustand zum Zeitpunkt  $t$ .  $z_{1:t}$  bezeichnet die Beobachtungen bzw. die Messgrößen von den Zuständen, die sich von Zeitstempel 1 bis  $t$  erstrecken. Zudem gibt  $u_{1:t}$  die Kontrollen an, die sich von Zeitstempel 1 bis  $t$  erstrecken. Die linke Seite der Gleichung  $bel(x_t)$  verkörpert den Glauben (engl. belief) der Wahrheit, dass der Zustand

<sup>1</sup>Präzis sollte die Formel  $X = x$  verwendet, um den Zustand zu beschreiben, wobei  $X$  eine Zufallsvariable ist und  $x$  ist der spezifische Wert, den  $X$  in Echtzeit annimmt. Zur Vereinfachung der Notation wird die Formel  $X = x$  im Rahmen dieser Arbeit sowie in vielen Literaturen als  $x$  bezeichnet.

$x_t$  ist [TBF05].

$$bel(x_t) = p(x_t | z_{1:t}, u_{1:t}) \quad (2.1)$$

Um die Wahrscheinlichkeitsverteilung einzugehen und eine rekursive Form zu entdecken, wird der stochastische Prozess von Zustandsänderung als Hidden Markov Model (HMM) betrachtet, das auch als dynamic Bayes network (DBN) bezeichnet wird. Unter Anwendung dieser Annahme gilt: Die Wahrscheinlichkeit eines Zustands bei dem Zeitstempel  $t$  ist einschließlich abhängig von Zustand bei dem Zeitstempel  $t - 1$  und nicht von Zuständen bei den Zeitstempeln, die früher als  $t - 1$  sind. Mathematisch wird HMM als eine Gleichung in 2.2 beschrieben.

$$p(x_t | x_{1:t}) = p(x_t | x_{t-1}) \quad (2.2)$$

Außerdem beschreibt Hidden Markov Model, wie in Abbildung 2.6 dargestellt, die vereinfachte Zusammenhang zwischen dem Zustand  $x$ , der Messgröße  $z$  und der Kontrolle  $u$ . Der Zustand zum Zeitpunkt  $t$  ist abhängig von dem Zustand zum Zeitpunkt  $t - 1$  und der Kontrolle  $u_t$ . Die Messgröße  $z_t$  hängt stochastisch vom Zustand zum Zeitpunkt  $t$  ab. Zusätzlich wird unter Anwendung des Satzes von Bayes und

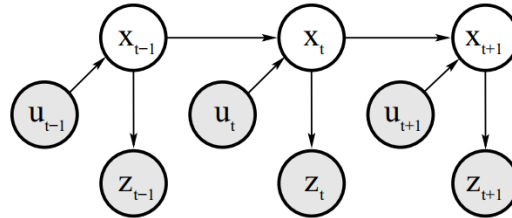


Abbildung 2.6: Hidden Markov Model (HMM) [TBF05]

des Gesetzes der totalen Wahrscheinlichkeit wird die Formel 2.1 in einen rekursiven Ausdruck abgeleitet, was als Bayes-Filter bekanntlich ist. Die Ableitung findet sich in [TBF05] und wird hier weggelassen. Der daraus resultierte Algorithmus befinden sich in Tabelle 2.1. Der Algorithmus besteht aus 2 Schritten, die als Prädiktion (engl. prediction) und Korrektur (engl. Correction oder update) bekanntlich sind. Der erste Schritt wird dadurch ausführt, dass der Glaube bzw. die Wahrscheinlichkeitsverteilung über den Zustand  $x_t$  - basierend auf dem vorherigen Glauben über den Zustand  $x_{t-1}$  und Kontrolle  $u_t$  - berechnet werden sollte. Der zweite Schritt ist eine Korrektur bzw. ein Update des prognostizierte Glaubens, indem die beob-

Tabelle 2.1: Der Algorithmus des Bayes-Filters [TBF05]

---

**Algorithm Bayes Filter**( $bel(x_{t-1}), u_t, z_t$ ):  
 for all  $x_t$  do  
 $\overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1})bel(x_{t-1})dx_{t-1}$   
 $bel(x_t) = \eta p(z_t|x_t)\overline{bel}(x_t)$   
 endfor  
 return  $bel(x_t)$

---

achte Informationen des Zustands bzw. die Messgrößen der Sensoren fusioniert und berücksichtigt werden. Der Wert  $bel(x_t)$  wird als A-posteriori-Verteilung bezeichnet. Im Gegensatz zu A-priori-Verteilung verkörpert A-posteriori-Verteilung den theoretisch präziseren Glauben, nachdem die Messgrößen von Sensoren durch ein Sensormodell zur Genauigkeit des Glaubens beitragen. Dieser Algorithmus bildet eine Grundlage, auf der viele weitere Algorithmen für bestimmte Szenarien entwickelt werden. Dazu gehören bekannte Gauß-Filter mit ihren Varianten, Particle-Filter und der diskrete Bayes-Filter. Der binäre Bayes-Filter, der in dieser Arbeit eine große Rolle spielt, ist ein spezieller Fall des Bayes-Filters bzw. des diskreten Bayes-Filter.

Der binäre Bayes-Filter, der durch den grundlegenden Bayes-Filter eingeführt wird, erfordert eine Annahme. Es wird angenommen, dass der Zustand einschließlich zwei Möglichkeiten besitzen. Das heißt, die Zufallsvariable, die den Zustand repräsentiert, kann nur zwei Werte annehmen. Bei Occupancy Grid kann der wichtigste und auch einzige Zustand der Belegungszustand sein, der lediglich zwei Fälle - belegt oder frei - hat. Aus diesem Grund ist der binäre Bayes-Filter dafür zutreffend. Darüber hinaus wird eine weitere Annahme getroffen, dass der Belegungszustand bei Occupancy Grid statisch ist. Dies bedeutet, dass sich der Belegungszustand im Laufe der Zeit nicht ändert und die Kontrolle  $u$  hat keine Auswirkung auf den Belegungszustand. Somit wird das Schema des stochastischen Prozesses der Zustandsveränderung von Abbildung 2.6 nach Abbildung 2.7 vereinfacht, indem die Kontrollen ausgeklammert werden. Nach diesen beiden Annahmen ist klar, dass der Algorithmus des Bayes-Filters in Occupancy Grid den Schritt Prädiktion nicht mehr enthält. Die A-posteriori-Verteilung der Zustand  $bel(x_t)$  ist einschließlich berechnet mit Informationen von Messdaten und dem vorherigen Zustand  $bel(x_{t-1})$ .

Anhand der Vereinfachung des Modells sollte der grundlegende Bayes-Filter entsprechend vereinfacht werden. Außerdem sollte der vereinfachte Algorithmus ein inverses

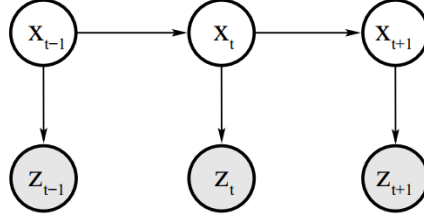


Abbildung 2.7: Das reduzierte HMM [TBF05]

Sensormodell verwenden. Das inverse Sensormodell gibt anstatt  $p(z_t|x)$  eine Verteilung über die binäre Zustandsvariable als Funktion der Messung  $p(x|z_t)$  an [TBF05]. Ein Grund für die Verwendung des inversen Sensormodells ist die Leichtigkeit, eine Funktion zu entwickeln, die die Wahrscheinlichkeitsverteilung von Sensordaten berechnet. Es ist zum Beispiel relativ simpel ein Modell zu entwerfen, womit die Wahrscheinlichkeit des Belegungszustands einer Zelle oder mehrerer Zellen anhand der Sensordaten bestimmt werden kann. Ein Vorwärtssensormodell ist dagegen in diesem Fall erstaunlich schwierig. Mit dem Ziel, einen vereinfachten Algorithmus zu finden, der unter Verwendung eines inversen Sensormodells das Glauben  $bel(x_t)$  berechnen kann, sollte die mathematische Ableitung nach [TBF05] [WSD07] [Heg18] folgend vorgestellt.

Da der Belegungszustand statisch ist und die Kontrolle  $u$  somit ignoriert wird, kann die Gleichung 2.1 in eine vereinfachte Gleichung 2.3 umgewandelt werden.

$$bel(x_t) = p(x_t|z_{1:t}, u_{1:t}) = p(x_t|z_{1:t}) \quad (2.3)$$

Bei Occuancy Grid ist der interessierende Zustand der Belegungszustand. Zur Vereinfachung der Notation wird das Glauben, dass die Zelle  $i$  des Gitters zum Zeitpunkt  $t$  belegt ist, als Gleichung 2.4 bezeichnet.

$$bel_t(m_i) = p(m_i|z_{1:t}) \quad (2.4)$$

Analog dafür ergibt sich das Glauben, dass die Zelle  $i$  des Gitters zum Zeitpunkt  $t$  frei ist, aus Gleichung 2.5.

$$bel_t(\overline{m_i}) = p(\overline{m_i}|z_{1:t}) \quad (2.5)$$

Mit Hilfe des bedingten Satzes der Bayes ergibt sich die Gleichung 2.4 zu

$$bel_t(m_i) = p(m_i|z_{1:t}) = \frac{p(z_t|m_i, z_{1:t-1})p(m_i|z_{1:t-1})}{p(z_t|z_{1:t-1})} \quad (2.6)$$

Unter Annahme eines Hidden-Markov-Modells wird die Gleichung 2.6 zu

$$bel_t(m_i) = \frac{p(z_t|m_i, z_{1:t-1})p(m_i|z_{1:t-1})}{p(z_t|z_{1:t-1})} = \frac{p(z_t|m_i)p(m_i|z_{1:t-1})}{p(z_t|z_{1:t-1})} \quad (2.7)$$

Zur Wahrscheinlichkeit  $p(z_t|m_i)$  wird der Satz des Bayes wiederum eingesetzt, womit die Gleichung 2.7 wird zu

$$bel_t(m_i) = \frac{p(z_t|m_i)p(m_i|z_{1:t-1})}{p(z_t|z_{1:t-1})} = \frac{p(m_i|z_t)p(z_t)p(m_i|z_{1:t-1})}{p(m_i)p(z_t|z_{1:t-1})} \quad (2.8)$$

Analog dazu hat der gegenteilige Zustand den Glauben

$$bel_t(\overline{m_i}) = 1 - bel_t(m_i) = \frac{p(z_t|\overline{m_i})p(\overline{m_i}|z_{1:t-1})}{p(z_t|z_{1:t-1})} = \frac{p(\overline{m_i}|z_t)p(z_t)p(\overline{m_i}|z_{1:t-1})}{p(\overline{m_i})p(z_t|z_{1:t-1})} \quad (2.9)$$

Dividiert Gleichung 2.8 durch der Gleichung 2.9, so ergibt sich

$$\frac{bel_t(m_i)}{1 - bel_t(m_i)} = \frac{p(m_i|z_t)}{1 - p(m_i|z_t)} \frac{p(m_i|z_{1:t-1})}{1 - p(m_i|z_{1:t-1})} \frac{1 - p(m_i)}{p(m_i)} \quad (2.10)$$

Die Gleichung 2.10 bietet eine perfekte mathematische Darstellung bzw. Erklärung an, was die Angaben und die Ausgabe des binären Bayes-Filters sind. Die Ausgabe ist das Glauben bzw. die A-posteriori-Wahrscheinlichkeit des Ereignisses, dass die Zelle  $m_i$  belegt ist, welches den Term  $bel_t(m_i)$  in Gleichung 2.10 entspricht. Der Term  $p(m_i|z_t)$  ist, wie oben erzählt, das inverse Sensormodell. Es ist ersichtlich, dass eine bedeutende Beziehung zwischen dem Sensormodell bzw. der Beschreibung des Sensormodells und dem Sensortyp. Darauf wird in Abschnitt 2.2 mit der Erfassung der Umgebung vertieft eingegangen. Der Term  $p(m_i|z_{1:t-1})$  beweist dabei das wichtige Merkmal des binären Bayes-Filters, dass das Verfahren der Schätzung auf Rekursion beruht. Die Anfangsbedingung bzw. die A-priori-Wahrscheinlichkeit wird als Term  $p(m_i)$  in Gleichung 2.10 bezeichnet. Die A-priori-Wahrscheinlichkeit gibt den Glauben an, vordem alle Messdaten von Sensorik berücksichtigt werden. Typischerweise wird bei Occupancy Grid anfänglich  $p(m_i)$  ein Wert von 0,5 angegeben, weil es zu

Beginn keine Information über den Belegungszustand gibt. Die Wahrscheinlichkeit, dass eine Gitterzelle belegt ist, ist die gleiche wie die Wahrscheinlichkeit, dass sie nicht belegt ist. Daher wird die Gleichung 2.10 zu

$$bel_t(m_i) = \frac{Y}{Y + 1} \quad (2.11)$$

mit

$$Y = \frac{p(m_i|z_t)}{1 - p(m_i|z_t)} \frac{p(m_i|z_{1:t-1})}{1 - p(m_i|z_{1:t-1})}$$

Zusammenfassend ist das gitterbasierte Modell eignet für eine statische Umgebung im urbanen Raum. Außerdem bietet das Modell einen Vorteil, einen Freiraum neben belegten Objekten zu modellieren, was eine Grundlage der darauffolgenden Navigation schaffen.

#### 2.1.4 Hybride Umfeldmodelle

Um ein Umfeld zu modellieren, welches komplizierte Szenarien repräsentieren kann, werden selbstverständlich Hybride Umfeldmodelle aufgefördert. Dadurch werden die Einschränkungen der beiden vorherigen Basismodelle eliminiert und ihre Vorteilen ausgenutzt. Es gibt unterschiedliche Kriterien und Methoden zum Erstellen eines hybriden Umfeldmodells. Das [0619] betreffende Modell, ist ein beliebtes Beispiel von hybriden Umfeldmodellen. Im Prinzip ist das Modell eine Kombination von einem objektbasierten Umfeldmodell und einem gitterbasierten Umfeldmodell. Hierbei beschreibt das objektbasierte Umfeldmodell dynamische Objekte, wohingegen das gitterbasierten Umfeldmodell bzw. Occupancy Grid den statischen Raum darstellen. Auf diese Weise ist das fusionierte Modell generell in der Lage eine Umgebung, wo sich dynamische und statischen Objekte befinden, vollständig und zutreffend zu beschreiben. Darüber hinaus kann die objektbasierte Umfeldmodellierung mit aktueller Technik z.B. Deep-Learning präziser und effizienter durchgeführt werden, während die semantische Navigation weiterhin direkt im gitterbasierten Umfeldmodell verlaufen kann[0619]. Da das Umfeld im Rahmen dieser Arbeit als statische Umgebung betrachtet wird, wird ein Hybrides Umfeldmodell aufgrund seiner Komplexität und Redundanz nicht verwendet.



## 2.2 Sensorik und ihr inverses Sensormodell

Zweifelloos dient Wahrnehmung bzw. Erfassung als eine wichtige Voraussetzung für Umfeldmodellierung, weil sie die Informationsquelle bietet. Die Wahrnehmung eines Fahrzeugs umfasst im Wesentlichen die Bestimmung der eigenen Position samt Orientierung und die Abtastung der Umgebung um das Fahrzeug. Die Eingenkalisierung ist im Rahmen dieser Arbeit nebensächlich und durch die vorgegebene GPS-Information bestimmt. Darauf soll in dieser Arbeit nicht näher eingegangen werden. In diesem Abschnitt liegt der Schwerpunkt auf den Sensoren, mit denen die Umgebung modelliert wird. Darüber hinaus werden die von IFF-Versuchsfahrzeug verwendeten Laserscanner und das in Abschnitt 2.1.3 erwähnte inverse Sensormodell erläutert.

### 2.2.1 Überblick über die verschiedenen Sensoren zur Umfeldmodellierung

Um ein zuverlässiges Umfeldmodell zu entwickeln und das Modell danach in der Praxis effektiv umzusetzen, spielt die Auswahl der erfassenden Sensoren entweder in der Akademie oder in der Industrie eine große Rolle. Die wichtigsten und am weitesten verbreiteten Fahrzeugsensoren zur Wahrnehmung der Umgebung sind Kameras, fernes Infrarot- (engl. Far-infrared, als FIR abgekürzt) Radar-, LiDAR (Light Detection and Ranging)- und Ultraschallsensoren. Nach [MAA<sup>+</sup>20] sind die Vorteilen neben den Nachteilen in Tabelle 2.2 aufgelistet.

Es ist erwähnenswert, dass Kameras aufgrund ihrer niedrigen Preise akademisch und industriell beliebt sind. Es wird häufig verwendet, um objektbasierte Umfeldmodelle mithilfe von Computer Vision zu erstellen. Immer mehr Algorithmen werden entwickelt, um die Tiefeninformation von Bildern zu berechnen. Obwohl die Erkennungsqualität begrenzt ist, wurde aus historischen Gründen eine große Anzahl von Radargeräten an Testfahrzeugen angebracht, sodass die Radar-basierte Forschung und Entwicklung noch nicht abgeschlossen ist. Der Preis von LiDAR-Sensor ist relativ hoch, aber sein absoluter Vorteil liegt in der Genauigkeit und Auflösung von Tiefeninformationen und Positionsinformationen, und es eignet sich für die Erstellung von Occupancy Grid.

Im Rahmen dieser Arbeit ist die am IFF bereits bestehende Fahrzeugarchitektur mit

Ibeo-LUX-Laserscanner versehen. Daher werden folgend in Abschnitt 2.2.2 der Mechanismus des Laserscanners und das daraus resultierende Sensormodell erläutert. Außerdem werden die technische Details bei Einführung der inversen Sensormodellierung vorgestellt, weil die realen technischen Größen dabei ein wichtiger Faktor sind.

Tabelle 2.2: Vorteile und Nachteile von Kamera, fernes Infrarotsensor, Radar-, Ultraschall- und Lidarsensor [MAA<sup>+</sup>20]

Sensor	Vorteile	Nachteile
Kamera	<ul style="list-style-type: none"> <li>• eine hohe Auflösung und Farbskalen über das gesamte Sichtfeld haben</li> <li>• eine farbenfrohe Perspektive der Umgebung bieten</li> <li>• eine 3D-Geometrie von Objekten bei Stereokameras bereitstellen</li> <li>• kostengünstig Im Vergleich zu Lidar sind</li> </ul>	<ul style="list-style-type: none"> <li>• ein leistungsfähiges Berechnungssystem benötigen, um nützliche Daten zu extrahieren</li> <li>• empfindlich auf starken Regen, Nebel und Schneefall reagieren</li> <li>• eine 3D-Geometrie von Objekten bei Stereokameras bereitstellen</li> <li>• begrenzte Reichweite besitzen</li> </ul>

FIR-Sensor	<ul style="list-style-type: none"><li>• nicht von der Lichtbedingungen und Objektoberflächenmerkmale beeinflusst werden können</li><li>• eine bessere Sicht durch Staub, Nebel und Schnee als Kameras haben</li><li>• eine horizontale Erfassungsbereichweite bis zu 200m oder mehr abdecken</li><li>• im Vergleich zu Lidar billiger und kleiner sind</li></ul>	<ul style="list-style-type: none"><li>• anspruchsvolle Rechenquellen und robuste Algorithmen erfordern</li><li>• schwierig Ziele in Szenarien mit kaltem Klima zu unterscheiden</li><li>• niedrigere Auflösung im Vergleich zur sichtbaren Kamera haben</li><li>• keine Information über Entfernung bieten</li></ul>
Radarsensor	<ul style="list-style-type: none"><li>• lange Strecken bei schlechten Sichtverhältnissen vor dem Auto sehen</li><li>• klein, leicht und erschwinglich sind</li><li>• weniger Strom als ein Lidar-Sensor benötigen</li><li>• im Vergleich zu Lidar robuster gegen Ausfälle sind</li></ul>	<ul style="list-style-type: none"><li>• eine geringe Genauigkeit und Auflösung bieten</li><li>• begrenzte Informationen (z. B. weder genaue Form noch Farbinformationen) bekommen</li><li>• das Problem wegen der gegenseitigen Beeinflussung von Radarsensoren haben</li><li>• schlechte Azimut- und Höhenauflösung verfügen</li><li>• ohne einer Erhöhung der Leistung Radardämpfung zeigen</li></ul>

Ultraschall-sensor	<ul style="list-style-type: none"> <li>• lange Strecken bei schlechten Sichtverhältnissen vor dem Auto sehen</li> <li>• klein, leicht und erschwinglich sind</li> <li>• weniger Strom als ein Lidar-Sensor benötigen</li> <li>• im Vergleich zu Lidar robuster gegen Ausfälle sind</li> </ul>	<ul style="list-style-type: none"> <li>• eine geringe Genauigkeit und Auflösung bieten</li> <li>• begrenzte Informationen (z. B. weder genaue Form- noch Farbinformationen) bekommen</li> <li>• das Problem wegen der gegenseitigen Beeinflussung von Radarsensoren haben</li> <li>• schlechte Azimut- und Höhenauflösung verfügen</li> <li>• ohne einer Erhöhung der Leistung Radardämpfung zeigen</li> </ul>
LiDAR-Sensor	<ul style="list-style-type: none"> <li>• große Entfernungen vor dem Auto bei guten Sichtverhältnissen erfassen</li> <li>• volle 360°- und 3D-Punktwolken bieten</li> <li>• eine gute Genauigkeit und Auflösung haben</li> <li>• keine signifikanten Interferenzen bei mehreren Lidarsensoren haben</li> </ul>	<ul style="list-style-type: none"> <li>• teurer als Radar und Kamera sind</li> <li>• kleine Objekte (wie Drähte und Stangen) nicht entdecken können</li> <li>• eine schlechte Kontrastunterscheidung bei der Erkennung nasser Oberflächen haben</li> <li>• durch unterschiedliche klimatische Bedingungen beeinflusst werden</li> </ul>

### 2.2.2 Funktionsweise des Lasersensors

Ist ein angemessenes Sensormodell zu finden, ist es sinnvoll das Mechanismus, die Eigenschaften und die Gründe der Unsicherheit zu untersuchen. Die Angemessenheit hierbei bedeutet, dass eine Abwägung bzw. ein Kompromiss immer nach verschiedenen Anwendungsszenarien gemacht werden muss.

Laserscanner beruht auf dem Prinzip ToF (Time-of-flight). Nach diesem Prinzip wird die Entfernung zwischen dem zu erfassenden Ziel und Laserscanner dadurch berechnet, dass die Zeit gemessen wird, die ein Lichtimpuls benötigt, um von der Lichtquelle zum beobachteten Ziel und dann zum Detektor (normalerweise zusammen mit der Lichtquelle) zu gelangen [SK08]. Im Prinzip ist Lasersensor ähnlich wie Radarsensor, wobei nur Infrarot-, Ultraviolett- oder Strahlen aus dem Bereich des sichtbaren Lichts anstelle von Mikrowellen eingesetzt werden [WHLS15]. Die mathematische Beschreibung des Prinzips lässt sich in Gleichung 2.12 darstellen. Dabei bezeichnet  $d$  den Abstand zwischen Lasersensor und dem zu erfassenden Objekt. Das Lichtgeschwindigkeit wird als  $c$  dargestellt. Bei einigen hochpräzisen Lasersensoren wird Lichtausbreitungsmedium auch berücksichtigt und dazu wird  $c$  der Umgebung gemäß kompensiert. Zeit  $t$  benötigt das Licht um die Ausbreitungsstrecke zu decken, die doppelte Entfernung zwischen Laserscanner und Objekt ist. Die Zeit wird in der Tat gemessen und in Abstand  $d$  übergeführt. Der Sensor sendet periodisch Lichtimpulse aus und berechnet die durchschnittliche Zielentfernung basierend auf der Zeit des Rückimpulses [SK08].

$$d = \frac{c \cdot t}{2} \quad (2.12)$$

Im Bereich des selbstfahrenden Fahren wird der Lichtpuls mit nicht nur eine Ausrichtung ausgestrahlt, weil ein relativ großer Beobachtungsbereich mehrere Lichtstrahlen erfordert, die in verschiedene Richtungen emittiert werden. Aktuell gibt es zwei verschiedene LiDAR-Systeme. Zum einen ist das feststehende Sensor, in dem mehrere Sende-/ Empfangseinheiten in unterschiedlichen Ausrichtungen angeordnet werden [Eff09]. Zum anderen wird das rotierende LiDAR-System dadurch realisiert, dass der Lichtimpuls mittels einer drehbaren Spiegeleinheit abgelenkt wird [Eff09]. Der in dieser Arbeit verwendete Laserscanner Ibeo-LUX gehört zu dem zweiten Sensorsystem.

### 2.2.3 Theorie des inversen Sensormodells

Die Aufgabe des inversen Sensormodells besteht darin, den in 2.1.3 erwähnten mathematischen Ausdruck  $p(m_i|z)$  zu finden, der die Wahrscheinlichkeitsverteilung des Belegungszustands der Zelle mit dem Index  $i$  beschreibt. Nach [Pie13] lässt sich das inverse Sensormodell im Prinzip in drei Bestandteile zerlegen. Wie in Abbildung 2.8 gezeigt, handelt es sich hierbei um Hinderniskartierung, Freiraummodellierung und Beschreibung unbekannter Gebiete.

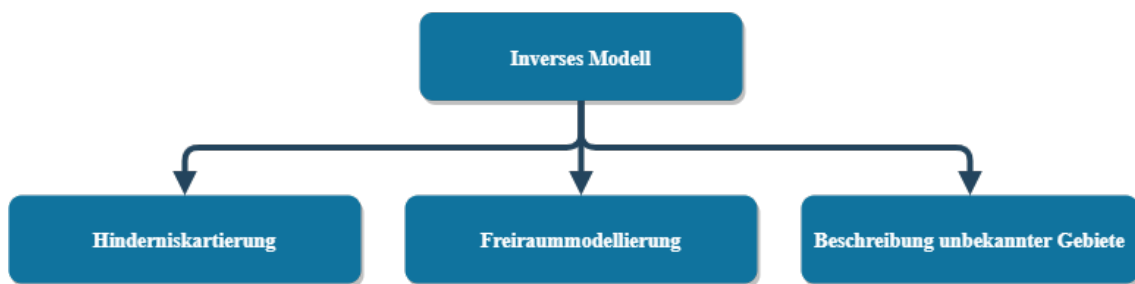


Abbildung 2.8: Bestandteile des Prinzips von Inverses Sensormodell

Unter Hinderniskartierung wird das Verfahren verstanden, dass die vom Sensor detektierten Objekte in das gitterbasierte Umfeldmodell bzw. das Occupancy Grid Map übertragen werden [Pie13]. Zwei Probleme sind in diesem Bestandteil des inversen Sensormodells zu lösen. Zum einen gewinnt es an Relevanz, welche Form für das detektierte Objekt bzw. das Hindernis angenommen wird [Pie13]. Die Form (z.B. Linie oder Ellipse), die das Hindernis repräsentiert, ist theoretisch eng mit der Unsicherheitsquellen des Laserscanners verbunden. Jedoch wird die Form in der Praxis ausgewählt unter Berücksichtigung der Ausführlichkeit bzw. Genauigkeit und Echtzeitanforderung. Zum anderen ist das Problem, mit welchem Zahlenwert die Wahrscheinlichkeit der kartierten Zelle erfüllt ist. Diese beiden Probleme werden nachstehend bei der tatsächlichen Sensormodellierung ausführlicher erörtert.

Freiraum ist bei dem Umfeldmodell der beobachtbare Bereich zwischen dem Sensor und dem erfassten Objekt. Analog zur Hinderniskartierung steht die Bestimmung der Form des Freiraums und der Wahrscheinlichkeit der betreffenden Zelle in Zentrum der Freiraummodellierung. Jedoch hängt die Form des Freiraums von der oben bestimmten Hindernisform. Beispielsweise wird der Freiraum als Linie oder Dreiecke modelliert, wenn die Hindernisse punktförmig oder linieförmig beschrieben werden.

Bei Wahrscheinlichkeitsverteilung über den Freiraum spielt die Ursache der Unsicherheit eine große Rolle. Mit zunehmender Entfernung wird beispielsweise die vom Laserscanner erkannte Entfernung zum Hindernis weniger zuverlässig. Aus diesem Grund ist dabei ein Modell zu entwickeln, die Unsicherheit in gewissem Grade widerspiegeln kann. In der Anwendung sind ebenso die Korrektheit und die Zeitaufwand zu gewichten.

Der Raum innerhalb der Reichweite des Laserscanners, der weder ein Hindernis noch ein freier Raum ist, ist ein unbekannter Bereich des Modells. Ein typisches Beispiel dafür ist der Raum hinter hinter Hindernissen. Im Allgemeinen wird der Wahrscheinlichkeit einer unbekannten Zelle als die A-priori-Wahrscheinlichkeit zugewiesen. Der Grund liegt darin, dass keine neuen Informationen über den Belegungszustand eingegeben werden. Typischerweise wird die A-priori-Wahrscheinlichkeit bzw. die Wahrscheinlichkeit einer unbekannten Zelle als 0,5 zugewiesen.

Die obigen drei Teile werden nicht getrennt, sondern gleichzeitig in der tatsächlichen Implementierung ausgeführt. Das inverse Sensormodell ist einer der Schlüsselbestandteile von Occupancy Grid. Seine Genauigkeit und Einfachheit bestimmen die Qualität des Modells und den Zeitaufwand des Systems. In der Technik wird es häufig in Kombination mit tatsächlichen Anwendungsszenarien und Sensoreigenschaften erstellt.

#### 2.2.4 Technische Details von Ibeo-LUX

In der am IfF bereits bestehenden Fahrzeugarchitektur werden Laserscanner Ibeo-LUX von Ibeo Automotive Systems GmbH für Erfassung der Umgebung um das selbstfahrende Fahrzeug verwendet. Das Golf 7 ist mit vier IBEO-LUX-4L ausgestattet. Neben 4 IBEO-LUX-4L Laserscannern sind auf der Vorder- und Rückseite des Passat IBEO-LUX-8L-Laserscanner installiert. Um ein angemessenes Sensormodell für den Laserscanner zu entwickeln, ist es vorausgesetzt, dass die technische Details des Sensors zur Verfügung stehen. Nach [Ibe17] sind die relevanten technischen Parameter in Tabelle 2.3 aufgelistet.

Tabelle 2.3: Technische Details von Ibeo LUX

Technische Daten	Wert
Reichweite	50m mit 10% Remission
Genauigkeit	10cm
Entfernungsauflösung	4cm
Horizontaler Öffnungswinkel	110° (50° bis −60°)
Vertikaler Öffnungswinkel	6,4° (LUX-8L) / 3,2° (LUX-4L)
Horizontale Winkelauflösung	0,25°
Vertikale Winkelauflösung	0,8°
Bildrate	25 Hz
Multi-Layer	8 (LUX-8L) / 4 (LUX-4L)
Ausgabe	Punktwolke und Objektdaten
Abmaße (B×T×H)	164,5×93,2×88mm
Gewicht	998,7g

### 2.2.5 Das zu verwendende Sensormodell

Nach Einführung der Theorie des inversen Sensormodells und der technischen Daten des verwendeten Ibeo-LUX-Laserscanners wird in diesem Abschnitt ein geeignetes Sensormodell entwickelt, um die Wirkung der Sensorinformation auf Glauben des Zellenbelegungszustands zu beschreiben. Hierbei wird ein Kompromiss zwischen Genauigkeit und Effizienz geschlossen. Aufgrund der Messunsicherheit und der in Tabelle 2.3 aufgelisteten Laserscanner-Spezifikationen erfolgt die Sensormodellierung in zwei Schritten. Im ersten Schritt wird die Form des Hindernis mitsamt des entsprechenden Freiraums festgestellt. Darauffolgend wird es bestimmt, welcher Wahrscheinlichkeit in welchem Bereich zugewiesen wird [Pie13].

Ein genaues Modell sollte die Wahrscheinlichkeit jeder Zelle in Abhängigkeit von der Position auf der Karte, der Strahlbreite und dem Abstand zum Zentrum des Strahls berechnen [HKOB10]. Wenn die Positionsunsicherheit beträchtlich ist, sollte eine kreis- oder ellipsenförmige Form für Hindernis unter Anwendung einer zweidimensionalen Gaußfunktion angenommen werden [Pie13]. Steht eine niedrige Winkelauflösung eines Sensor zur Verfügung, kann die Form zu einer Linie vereinfacht werden. Wenn der Sensor ausreichend genaue Tiefeninformationen hat, lässt sich die Form weiter zu einem Punkt vereinfachen.

Im Rahmen dieser Arbeit wird die Auflösung der Zelle als  $0,1m$  zugewiesen. Außerdem ist es sinnvoll, die maximale erfasste Entfernung  $d_{max}$  des Laserscanners



zu beschränken, obwohl laut Tabelle 2.3 eine Reichweite 50m besteht. Dies kann die durch die große Entfernung verursachte Unsicherheit verringern. Nach dem Test wird der Wert von  $d_{max}$  als 10m bestimmt. Die horizontale Winkelauflösung  $\Delta\theta$  beträgt nach Tabelle 2.3  $0,25^\circ$  oder  $0,004363(\text{rad})$ . Durch Multiplizieren von  $d_{max}$  und  $\Delta\theta$  beträgt der Wert des Kreisbogens  $0,043$ , der die wegen der Strahlbreite entstehende Divergenz beschreibt. Da  $0,043 < 0,1$  gilt, lässt sich der Einfluss von Strahlbreite vernachlässigen. Zudem wird angenommen, dass die Genauigkeit von  $0,1m$  ausreichend genau ist. Damit ist es deutlich, dass Hindernisse im Rahmen dieser Arbeit aufgrund der Sensorspezifikation als Punkte beschrieben werden können. Daraus lassen sich herleiten, dass das entsprechende Freiraummodell als Linie dargestellt wird. Um ein Objekt zu detektieren, dessen Abmessung größer als Auflösung ist, ist der bekannte Begriff Raycasting zu einsetzen. Hierbei werden die einzelnen physikalischen Strahlen des Laserscanners zwischen dem Sensor und einem Objekt nachgebildet. Basierend darauf wird der nächste Schritt der Modellierung eines Sensors simplifiziert und es wird nur erwartet, dass das inverse Sensormodell einzigen Laserstrahls beschreiben kann. Während der Implementierung werden alle Laserstrahlen mit demselben Sensormodell behandelt. Das inverse Modell eines einzelnen Strahls ist eindimensional und das von einem idealen Laserscanner verwendete Modell ist in Abbildung 2.9 dargestellt. Das Diagramm veranschaulicht die ideale Zusammen-

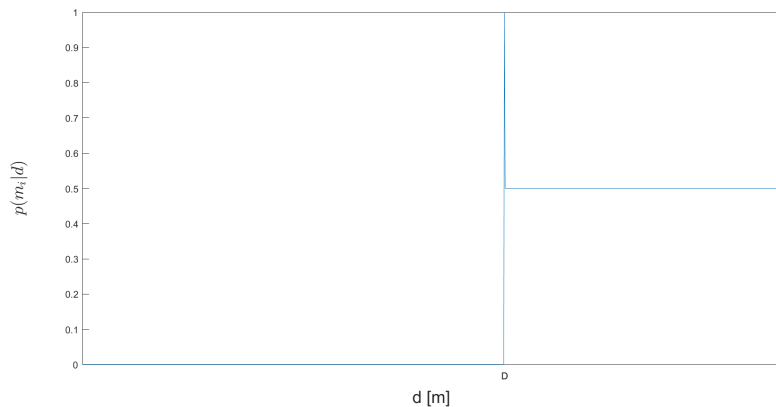


Abbildung 2.9: Das ideale Sensormodell

hang zwischen der Belegungswahrscheinlichkeit der Zelle mit dem Index  $i$  und dem Abstand vom Laserscanner. Es ist angenommen, dass sich ein punktförmiges Objekt in einer Entfernung von  $D$  befindet. Ohne Messabweichung bzw. Unsicherheit be-

sitzt das ideale Modell eine besonders einfache Funktion. Vor dem Objekt bzw. dem Hindernis ist es angenommen, dass es kein anderes Hindernis existiert. Die Wahrscheinlichkeit der Zelle, die von Laserscanner mit einem Abstand von  $D$  entfernt, beträgt 1. Hinter dem Hindernis sind alle Zelle verborgen, weshalb die Wahrscheinlichkeit unverändert bleibt und als A-priori-Wahrscheinlichkeit von 0,50 angegeben wird.

Jedoch unterliegt jede Messung aus verschiedenen Gründen einer gewissen Messunsicherheit [Heg18], weshalb das ideale Sensormodell eine bescheidene Genauigkeit bietet. Unter Berücksichtigung der Messunsicherheit kann das inverse Sensormodell von ideal vereinfachend bis stark komplex sein [Pie13]. Allerdings wird die Funktion, die das inverse Sensormodell definiert, in der Praxis nach der bisherigen Erfahrungen bestimmt. Basierend auf [WSD07] [Pie13] [Heg18] wird im Rahmen dieser Arbeit je nach Situation zwei bestimmte abschnittsweise definierte Funktionen angewendet. Die beide Funktionen weisen darauf hin, dass die Zellen innerhalb des minimalen erfassbaren Abstands mit einer niedrigen und konstanten Wahrscheinlichkeit hinzugefügt werden. Das inverse Sensormodell ist eine Erweiterung des idealen Sensormodells, wenn ein Hindernis detektiert wird. Wie in Abbildung 2.10 dargestellt, ist der Freiraum vor dem detektierten Hindernis durch eine monoton ansteigende lineare Funktion zu beschreiben. Der Belegungswahrscheinlichkeit des als Hindernis erkannten Quadrats wird ein höherer Wert zugewiesen, z. B. 0,9. Wie beim idealen Modell wird auch bei diesem Modell hinter dem Hindernis ein Wahrscheinlichkeitswert von 0,5 zugewiesen. Hierbei wird die mit größerer Entfernung entstehende Mes-

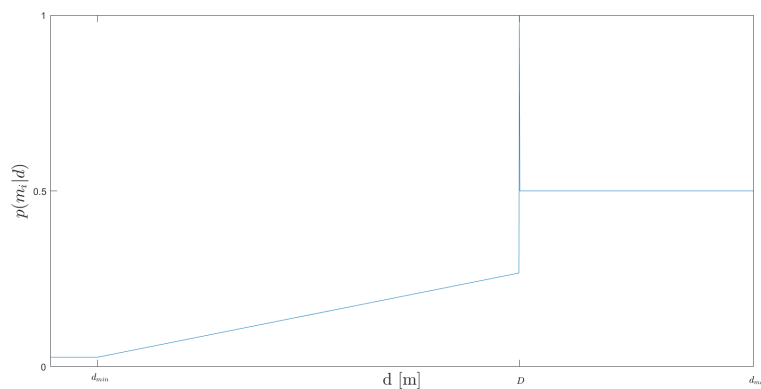


Abbildung 2.10: Das inverse Sensormodell, wenn ein Hindernis detektiert wird

sunsicherheit abgebildet, indem die Belegungswahrscheinlichkeit mit zunehmender Distanz von dem Laserscanner erhöht wird [Heg18]. Besteht kein Hindernis, wird die Funktion in die in Abbildung 2.11 gezeigte Beschreibung umgewandelt. In diesem

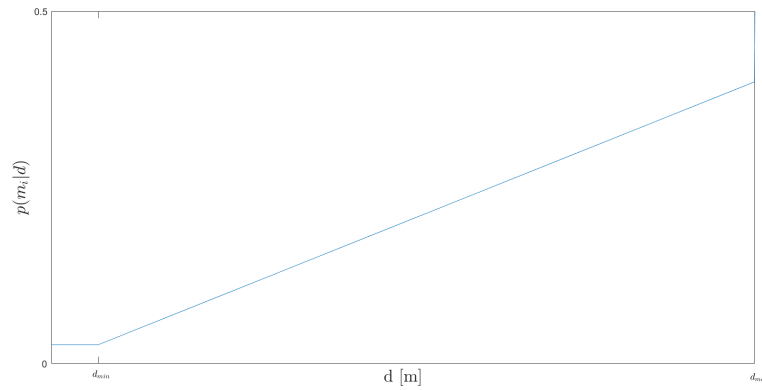


Abbildung 2.11: Das inverse Sensormodell, wenn kein Hindernis detektiert wird

Fall steigt die Wahrscheinlichkeitsfunktion im Bereich der maximalen Erfassungsentfernung und der minimalen Entfernung monoton an. Bei maximaler Entfernung steigt der Wahrscheinlichkeitswert auf 0,5.

Da diese Funktionen linear und einfach sind, wird die Echtzeitanforderung in gewissem Maß gewährleistet. Darüber hinaus werden die Steigung, der minimale bzw. maximale Abstand und die Wahrscheinlichkeit der Zelle mit dem Abstand parametrisiert und bei der folgenden Implementierung justiert. Zusätzlich werden die Steigung, der minimale oder maximale Abstand und die Wahrscheinlichkeit der Zelle, die sich mit diesen beiden Abständen befindet, so parametrisiert, dass sie je nach Anwendung variieren können.

## 3 Implementierung

Um das ober

### 3.1 Versuchsfahrzeug

Bevor mit Implementierung des in Kapitel 3.4 entwickelten Umfeldmodells begonnen wird, werden die relevanten Informationen über das Versuchsfahrzeug mitsamt die darin eingebauten Sensoren dokumentiert und in der eigentlichen Implementierung parametrisiert.

#### 3.1.1 Dimension über Versuchsfahrzeug

Bei Implementierung im Rahmen dieser Arbeit ist es auch bedeutungsvoll, die Position bzw. den belegten Raum des Versuchsfahrzeugs zu modellieren und dokumentieren, was einen konkreten Beitrag zur kollisionsfrei Navigation leistet. Außerdem ist die Information über die Anordnung der Lasersensoren eng verbunden mit der Abmessung des Fahrzeugs. Daher wird die Dimension des Fahrzeugs als ein wichtiges Element betrachtet. Die Abbildung 3.12 zeigt, dass die wichtige Größen von Abmessung des Fahrzeugs parametrisiert werden. Obwohl die Zeichnungsbemäßung eigentlich redundant ist, wird sie mit Absicht angewendet, um die Darstellung der wichtigen Größen sichtbar zu machen. Der Rot Punkt bezeichnet hierbei die Koordinatenursprung des Fahrzeugkoordinatensystem und befindet sich mittig auf der Hinterachse [Heg18]. Die X-Achse des Fahrzeugkoordinatensystem zeigt die Längsrichtung des Fahrzeugs nach vorne [Heg18]. Die Y-Achse verläuft senkrecht zur X-Achse und zeigt nach links des Fahrtrichtung. Die Koordinatenursprung dient als ein Bezugspunkt und die Größen, z.B. die Lage eines Sensors sowie die Position eines detektierten Objekts, werden nur relativ zu dem Bezugssystem bzw. Fahrzeugkoordinatensystem angegeben.

In ifF stehen Golf7 (TIAMO) und Passat Alltrack (TEASY 3) als Versuchsfahrzeuge zur Verfügung[Heg18]. Die der Abbildung 3.12 entsprechenden Abmessungen von diesen Versuchsfahrzeugen werden in Tabelle 3.1 aufgelistet.

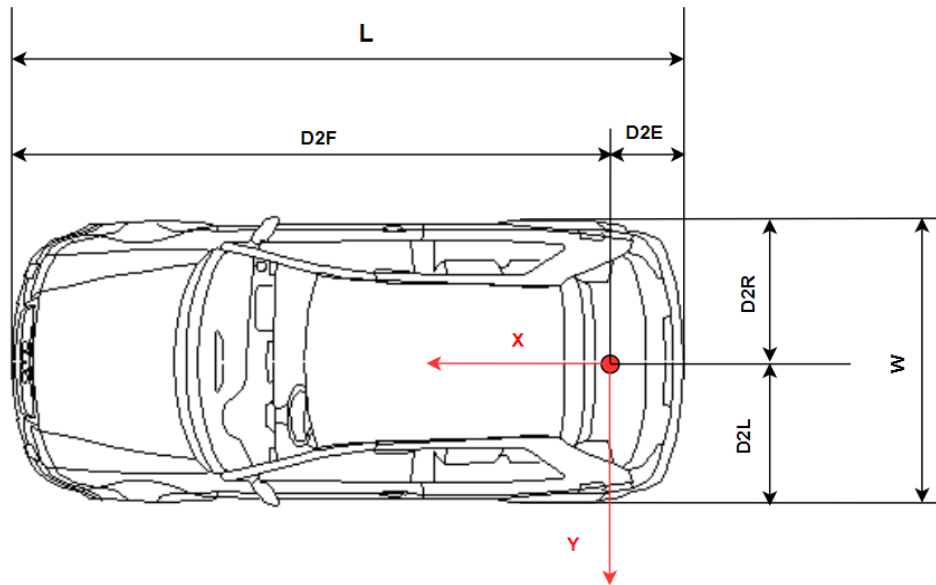


Abbildung 3.12: Dimension von Versuchsfahrzeug

Tabelle 3.1: Abmessung von Versuchsfahrzeuge

Abmessung	Golf 7 (TIAMO)	Passat (TEASY 3)
L	4.3	4.6
W	1.8	1.6
D2F	3.5	3.6
D2E	0.8	1.0
D2L	0.9	0.8
D2R	0.9	0.8

### 3.1.2 Einbauposition der Ibeo-Laserscanner

Die Anzahl und die Anordnung der im Versuchsfahrzeug installierten Ibeo-Laserscanner dienen auch als wichtigen Parametern bei Implementierung, denn diese Informationen liefern den Startpunkt des Strahls jedes Sensors. In Abbildung 3.12 sind die Einbauposition und der Erfassungsbereich jedes Sensors dargestellt. Dazu werden die tatsächlichen Werte in Tabelle 3.2 und Tabelle 3.3 gegeben. In den Tabellen bezeichnet  $x$  die x-Koordinate im Fahrzeugkoordinatensystem und  $b$  die y-Koordinate. Der Winkel  $\theta$  beschreibt die ausgesandte Richtung des Anfangsstrahls. Der Anfangsstrahl jedes Laserscanners ist gegen den Uhrzeigersinn zur Endstrahl. Der Winkel-

bereich des Erfassungsraum des Sensors ist nach der Tabelle 2.3 auf  $110^\circ$  begrenzt. Dieser Wert wird in der Praxis entsprechend der Performance des Umfeldmodells angepasst.

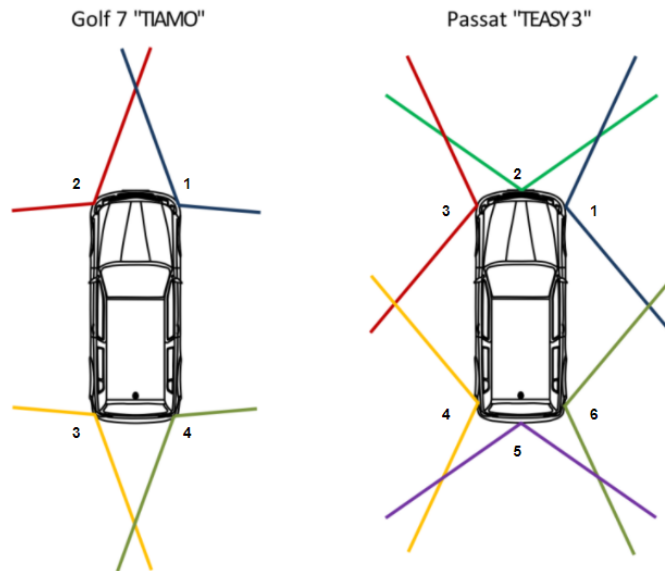


Abbildung 3.13: Einbauposition und Erfassungsbereich der Ibeo-Laserscanner

Tabelle 3.2: Werte der Einbauposition und des Winkels des Anfangsstrahls jedes Sensors bei Golf 7 (TIAMO))

Sensor ID	x (m)	y (m)	Winkel $\theta$ des Anfangsstrahls ( $^\circ$ )
1	3	-0.9	-10
2	3	0.9	80
3	-0.7	0.9	170
4	-0.7	-0.9	-100

## 3.2 Framework ROS zur Implementierung

Eine der zentralen Aufgaben dieser Arbeit handelt sich um die Konvertierung von dem am IfF bereits bestehenden MATLAB/Simulink-Modell nach Robot Operating System (ROS). Das Robot Operating System (ROS) ist ein Framework zum Schreiben von Robotersoftware. Es handelt sich um eine Sammlung von Tools, Bibliotheken und Konventionen, die darauf abzielen, die Erstellung komplexer und robuster

Tabelle 3.3: Werte der Einbauposition und des Winkels des Anfangsstrahls jedes Sensors bei Passat (TEASY 3)

Sensor ID	x (m)	y (m)	Winkel $\theta$ des Anfangsstrahls ( $^{\circ}$ )
1	3.3	-0.8	-35
2	3.6	0	45
3	3.3	0.8	125
4	-0.5	0.8	145
5	-1	0	-135
6	-0.5	-0.8	-55

Roboterverhalten auf einer Vielzahl von Roboterplattformen zu vereinfachen [QGS15]. Obwohl diese Idee aus dem Bereich der Robotik stammt, machen ihre verschiedenen guten Eigenschaften ihre Investition in den Bereich des autonomen Fahrens sehr bedeutsam. Um eine klare Programmstruktur und eine genaue und effiziente Umsetzung des in Kapitel 3.4 genannten Umfeldmodells zu erhalten, ist eine kurze Einführung in die ROS-Grundlagen und Funktionsmodule in Bezug auf diesen Artikel erforderlich.

### 3.2.1 Grundlagen der ROS-Architektur

Die ROS-Architektur, die in Abbildung 3.14 dargestellt, wurde entworfen und in drei Abschnitte oder Konzeptebenen unterteilt, welche sich um die Dateisystemebene (engl. The Filesystem level), die Berechnungsdiagrammebene (engl. The Computation Graph level) und die Community-Ebene (engl. The Community level) handeln [Fer15].

Auf der Dateisystemebene wird eine Gruppe von Konzepten verwendet, um zu erklären, wie ROS intern gebildet wird. Ähnlich wie bei einem Betriebssystem ist ein ROS-Programm in Ordner unterteilt, und diese Ordner enthalten Dateien, die ihre Funktionen beschreiben [Fer15]. Hierbei sind die wichtigen Konzepte zu diesem Artikel Package und Metapackage. Das Package ist die zentrale und grundlegende Dateiorganisationseinheit, die Programmierfunktionen in ROS vollständig realisieren kann. Es enthält im Allgemeinen ROS Laufzeitprozess (engl. runtime process), Quellcode (engl. Sourcecode), Konfigurationsdateien (engl. configuration files) und das Package-manifest, das zur Bereitstellung von Informationen von build-

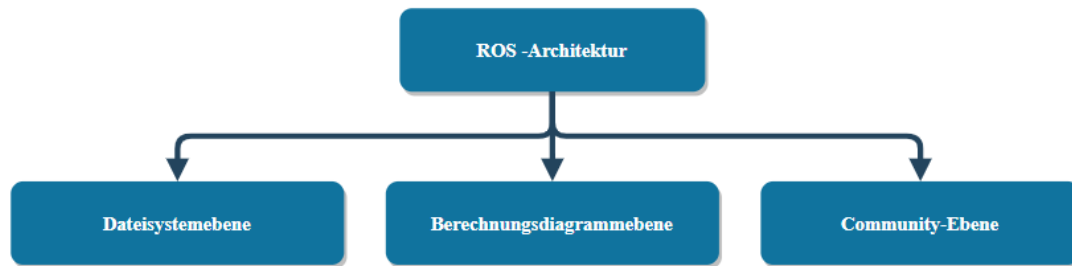


Abbildung 3.14: ROS-Architektur

dependencies, run-dependencies und Lizenz verwendet wird. Metapackages werden in der Regel nach einer ähnlichen Funktionalität gruppiert. Andere Grundkonzepte und Begriffe auf dieser Ebene sind aufgrund der Länge des Artikels nicht detailliert und finden sich in [Fer15][Kou16].

Die Berechnungsdiagrammebene ist die relevanteste Ebene für diese Arbeit, auf der die Kommunikation zwischen Prozessen und Systemen stattfindet. Die Grundkonzepte auf dieser Ebene sind, wie in Abbildung 3.15 dargestellt, Nodes, ROS-Master, Parameter Server, Messages, Topics, Services und ROS-Bags, die alle Daten auf unterschiedliche Weise für das Diagramm bereitstellen [Fer15]. Nodes sind ausführbare

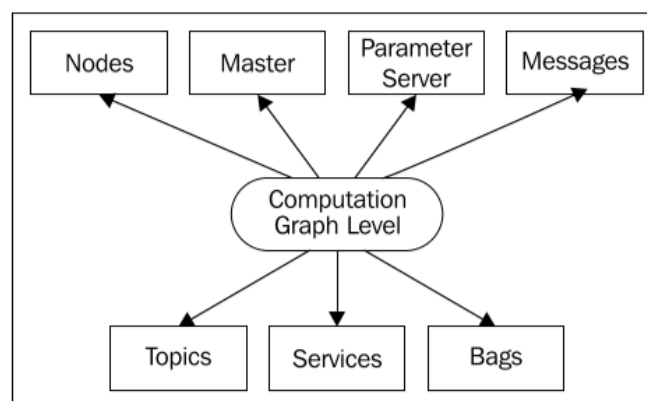


Abbildung 3.15: Wesentliche Grundkonzepte von Berechnungsdiagrammebene

Dateien (engl. executables) in ROS und vervollständigen die erwartete Funktion und die zugehörigen Berechnungen. Die Nodes können miteinander kommunizieren und



Daten übertragen. Daher gibt es im Allgemeinen mehrere Nodes in einem System, die unterschiedliche Funktionen ausgeführt haben. Der Datenaustausch zwischen Nodes erfolgt über Messages. ROS verwendet eine vereinfachte Nachrichtenbeschreibungssprache, um die Datenwerte zu beschreiben, die von Nodes publiziert (engl. published) [Fer15]. Damit kann ROS den richtigen Quellcode für diese Nachrichtentypen in mehreren Programmiersprachen (z.B. C++ oder Python) generieren. Zahlreiche vordefinierte Messages in ROS können direkt zum Übertragen von Daten oder zum Erstellen neuer aufgabenorientierter Messages verwendet werden. Dies erfolgt durch Definieren einer Datei mit .msg-Extension. Wenn ein Node Daten sendet, heißt es, dass das Node eine Topic publizieren. Ein anderer Knoten kann die Topic abonnieren (engl. subscribe), um die Daten abzurufen. Ein Node kann eine Topic nur abonnieren, wenn es denselben Message-Typ hat. Eine Topic kann verschiedene Subsribers und auch verschiedene Publishers haben. Wenn die Kommunikation zwischen Nodes empfangen und beantwortet (engl. receive and reply) werden muss, sollten Services anstelle von Topics verwendet werden. Services geben den Entwicklern die Möglichkeit, mit Nodes zu interagieren. Mit Parameter Server ist es möglich, Schlüssel zu verwenden, um Daten an einem zentralen Ort zu speichern und Nodes während der Ausführung zu konfigurieren oder die Nodes der Knoten zu ändern [Fer15]. Die oben genannte Kommunikation garantiert ROS-Master, der jede Nodes verwalten. Nodes werden zuerst beim Master registriert, und dann integriert der Master Nodes in das gesamte ROS-Programm. Auf diesem Grund besteht der erste Schritt darin, den Master zu starten, wenn das ROS-Programm gestartet wird. ROS-Bag ist ein Format zum Speichern und Wiedergeben aller Informationen der Messages, Topics und Services, die gewünscht werden. In dieser Arbeit wird ROS-Bag verwendet, um die Sensordaten von dem Versuchsfahrzeug zu speichern. Wenn das ROS-Bag wiedergegeben ist, simuliert es die Datenwerte von Sensoren zu messen und erfassen, was ist praktisch zum Debuggen von Implementierungsalgorithmus. Die Konzepte auf ROS-Community-Ebene sind die ROS-Ressourcen, die es separaten Communities ermöglichen, Software und Wissen auszutauschen [Fer15]. Zu den Ressourcen gehören unter anderem ROS-Repositories, ROS-Distributions und ROS-Wiki. Jedoch hat diese Ebene für diesen Artikel nur eine sehr geringe Relevanz. Auf diesem Grund ist die Auseinandersetzung damit im Rahmen dieser Arbeit zu verzichten.

Aufgrund des oben erwähnten Mechanismus und der Philosophie von ROS hat der

Aufbau der Implementierung des Umfeldmodell auf ROS einen starken Vorteil. Die dezentrale Kommunikationsmethode macht das Implementierungssystem klarer und einfacher. Außerdem sind Fehler im System leichter zu finden und sortieren. Die Aufteilung zwischen verschiedenen Funktionen erleichtert die spätere Systemerweiterung, z.B. Navigation bzw. kollisionsfreie Pfadplanung. Im Rahmen dieser Arbeit ist für die Implementierung ROS-Kinetic-Kame mit Ubuntu 16.04 (Xenial) in Benutzung.

### 3.2.2 Visualisierung des Umfeldmodells in ROS

Die Visualisierung des Modells ist ebenso wichtig wie seine Einrichtung und Implementierung. Eine gute Visualisierung spiegelt den tatsächlichen Betriebsstatus des Modells hervorragend wider. Dies hilft bei der Behebung von Programmfehlern und beim Datenaustausch mit anderen Funktionsmodulen oder Modellen im nachfolgenden Systemerweiterungsprozess. Das ROS-System bietet eine Vielzahl von Tools zur Datenvisualisierung und zum Debuggen. Das wichtigste und am weitesten verbreitete ist RVIZ. rviz ist ein 3D-Visualisierungswerkzeug von ROS, mit dem Sensordaten und Statusinformationen visualisiert werden. RVIZ unterstützt umfangreiche Datentypen, die durch Laden verschiedener Display-Typen visualisiert werden. Jeder Display hat einen eindeutigen Namen. Wichtige Display-Typen und ihre entsprechenden Message-Typen im Bereich des autonomen Fahrens sind in Tabelle 3.4 aufgeführt. Aufgrund der Philosophie des verteilten Software-Frameworks von ROS muss das RVIZ-Visualisierungstool nur den passenden Message-Typ und die passende Topic auswählen, wenn Daten auf einer Topic visualisiert werden sollen.

Für das auf diesen Artikel bezogene Umgebungsmodell gibt es zwei grundlegende Visualisierungsoptionen. Zum einen ist Display-Typ Map, das nav\_msgs/OccupancyGrid message anzeigt. Die Informationen in nav\_msgs/OccupancyGrid umfassen die Koordinaten des ursprünglichen Standorts, die Auflösung sowie die Länge und Breite der Karte und die Kartendaten (engl. Map data) in jeder Gitterzelle. Map data werden in zwei Situationen betrachtet. Einer ist, dass der Belegungszustand unbekannt ist und der Wert in diesem Fall -1 ist. Die andere ist, dass die Wahrscheinlichkeit der Belegung bekannt ist und der Wert in diesem Fall 0 bis 100 beträgt. Wie in Abbildung 3.16 gezeigt, wenn Map value von einer Gitterzelle -1 ist, wird die Zellenfläche ausgegraut dargestellt. Wenn Map Value von 0 auf 100 steigt, verändert

Tabelle 3.4: Display-Typen und ihre entsprechenden Message-Typen in RVIZ

Display-Typ	Message-Typ	Beschreibung
Grid Cells	nav_msgs/GridCells	Zeichnet Zellen aus einem Raster
Point Cloud 2	sensor_msgs/PointCloud2	Zeigt Daten aus einer Punktwolke
Map	nav_msgs/OccupancyGrid	Zeigt eine Karte in der Grundebene
Path	nav_msgs/Path	Zeigt einen Pfad
Pose	geometry_msgs/PoseStamped	Zeichnet eine 3D-Pose
Pose Array	geometry_msgs/PoseArray	Zeichnet mehrere Posen
Image	sensor_msgs/Image	Erstellt ein neues Rendering-Bild
Laser Scan	sensor_msgs/LaserScan	Zeigt Daten von einem Laserscan
Odometry	nav_msgs/Odometry	Sammelt Kilometerzähler-Posen aus der Zeit
TF	tf2_msgs/TFMessage	Zeigt die Koordinatentransformationshierarchie

sich die entsprechende Zelle in einem Farbverlauf von Weiß zu Schwarz.

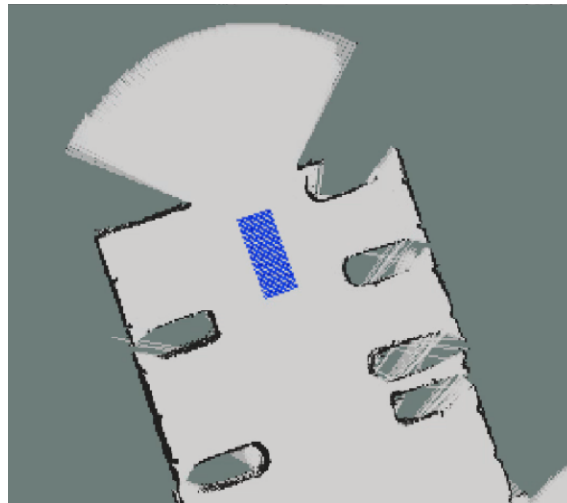


Abbildung 3.16: Display mit Map

Die zweite Möglichkeit der Visualisierung eines Umfeldmodells in RVIZ ist die Verwendung von GridCells. Dies darstellt die Daten von Message-Typ `nav_msgs/GridCells`. Darin handelt sich um die Informationen über die Länge und Breite sowie die Koordinaten jeder Zelle. GridCells-Display ist nur für die Visualisierung des vom Entwickler angegebenen Bereichs verantwortlich. Es ist von der Belegungswahrscheinlichkeit getrennt und wird einfach, leicht und flexibel. Je nach Aufgabe des Entwicklers oder Debugging-Anforderungen können unterschiedliche Wahrscheinlichkeitsbereiche angezeigt werden. Darüber hinaus ermöglicht es einen starken Kontrast von Farben mit unterschiedlichen Belegungswahrscheinlichkeiten. Im Gegensatz dazu ist die Grau-

stufendarstellung von dem oben erwähnten Map nicht offensichtlich und für die Programmentwicklung und die Erkennung der Datenkorrektheit nicht geeignet. Ein Anwendungsbeispiel besteht darin, wie in Abbildung 3.17 gezeigt, Gitterzellen mit unterschiedlichen Belegungswahrscheinlichkeiten in verschiedenen Topics zu organisieren und dann die verschiedenen Topics mit verschiedenen offensichtlichen Farben zu visualisieren. Neben der Flexibilität bietet GridCells-Display einige gute Vorteile gegenüber Map-Display. Zunächst kann die jeder Gitterzelle zugewiesenen zusätzlichen Informationstypen und -werten selbst definiert werden, was eine direktere Bedingung für die zukünftige Erweiterung und Verbesserung des Modells darstellt. Selbst wenn nur die Belegungswahrscheinlichkeit zu berücksichtigen ist, kann die Wahrscheinlichkeit (0 bis 100) als Ganzes betrachtet werden, anstatt den Wert -1 allein zu verwenden bzw. umrechnen, um das Unbekannte auszudrücken. Darüber hinaus erleichtert die Verwendung von GridCells-Display die anschließende Binärisierung von Werten und Bildern. Wie in Abbildung 3.18 gezeigt, kann die Binärisierung durch Einstellen des Schwellenwerts, der durch Experimente oder Deep-Learning erhalten wurde, leicht erzielt werden. Daher im Rahmen dieser Arbeit wird GridCells-Display verwendet, um eine Visualisierung zu erreichen. Es gibt aber ein kleines Problem, das bei der Verwendung von GridCells besondere Aufmerksamkeit und Lösung erfordert. Durch tatsächliche Experimente ist bekannt, dass bei sehr großen Positionskoordinaten von GridCells (z. B. 10 bis 6 Potenzen) die Darstellung von Gitterzellen in RVIZ deformiert wird oder sogar verschwindet. Daher können bei der Implementierung des Modells die vom GPS erhaltenen UTM-Koordinateninformationen nicht direkt als Koordinaten für die Anzeige der Gitterzellen verwendet werden. Vor der eigentlichen Visualisierung werden zwei Abweichungen `X_VISUAL_OFFSET` und `Y_VISUAL_OFFSET` so eingestellt, dass der Koordinatenwert der Zellen nahe am Ursprung liegt, wodurch die Genauigkeit der Visualisierung sichergestellt wird. Dieses Abweichungspaar wird durch die anfänglichen Fahrzeugkoordinateninformationen bestimmt, die bei der Initialisierung des Modells erhalten werden, was sich in der nächsten Erläuterung des Funktionsblocks widerspiegelt.

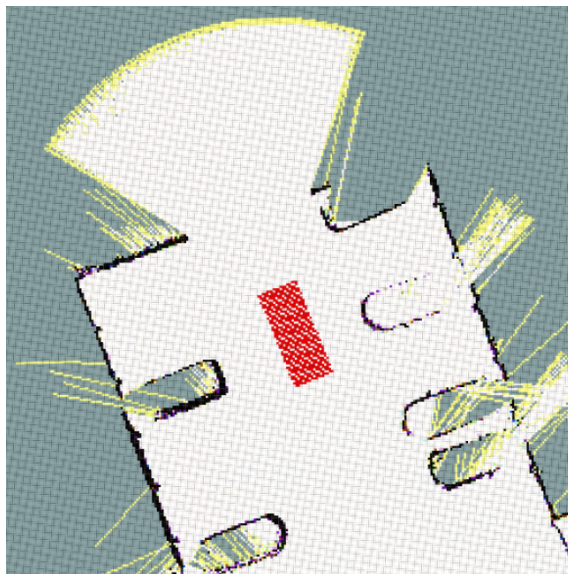


Abbildung 3.17: Display mit GridCell

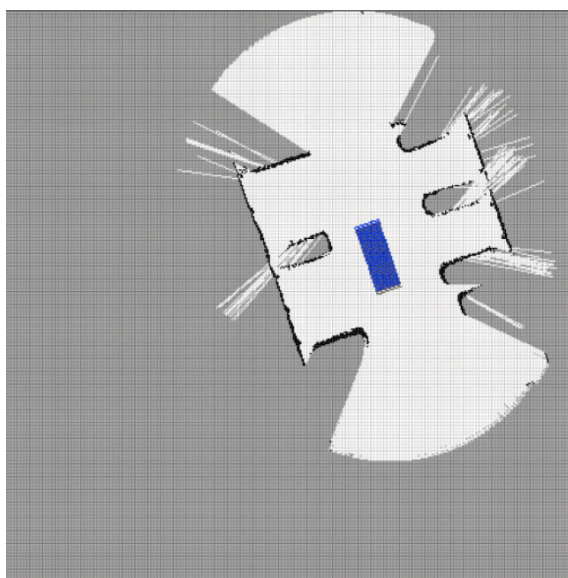


Abbildung 3.18: Display mit GridCell nach Binärisierung

### 3.3 Koordinatensysteme

Daten von verschiedenen Informationsquellen bzw. Sensoren sind häufig mittels unterschiedlichen Koordinatensystemen gegeben. Auf diesem Grund wird die Konvertierung zwischen verschiedenen Koordinatensystemen bei der Realisierung des Umfeldmodells oft durchgeführt. Hierbei gibt es 3 wesentliche Koordinatensysteme, deren Klärung für das Verständnis der nachfolgenden Funktionsbausteine dieser Arbeit sehr hilfreich ist. Wie in Abbildung 3.19 gezeigt, sind diese 3 Koordinatensysteme Weltkoordinatensystem (engl. Global Coordinate System, als GCS abgekürzt), Ankerkoordinatensystem (engl. Anchor Coordinate System, als ACS abgekürzt) und Fahrzeugkoordinatensystem (engl. Vehicle Coordinate System, als VCS abgekürzt).

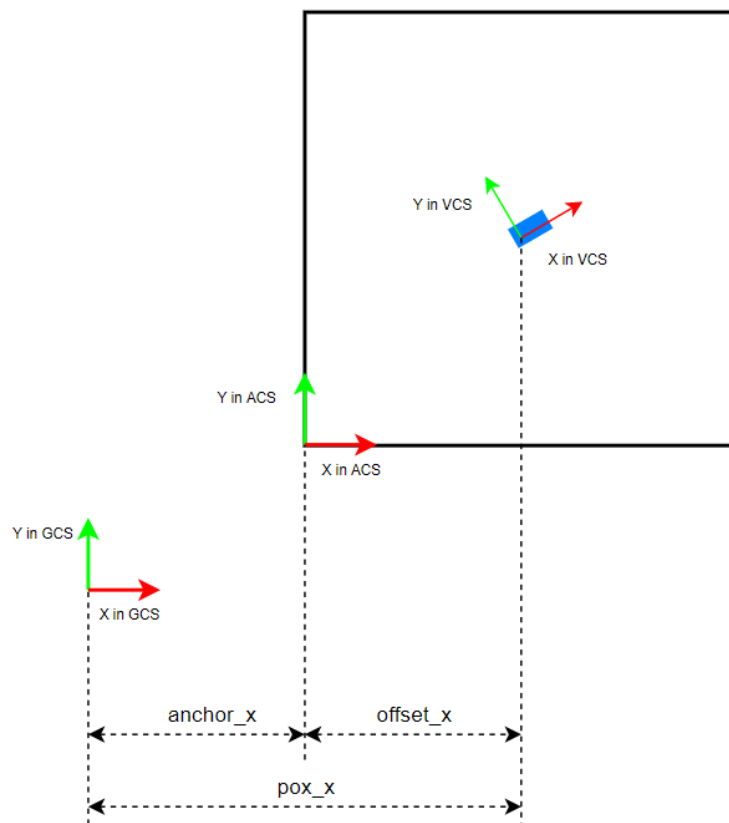


Abbildung 3.19: 3 wesentliche Koordinatensysteme im Umfeldmodell

### 3.3.1 Global Coordinate System (GCS)

Das Weltkoordinatensystem ist das grundlegendste Koordinatensystem. Die vom GPS-Sensor erhaltenen Informationen zur Fahrzeugpose basieren auf dem Weltkoordinatensystem. Im tatsächlichen Gebrauch sind dafür zwei Umrechnungen erforderlich. Das erste ist die Notwendigkeit, die Abweichung zwischen dem Ursprung des Fahrzeugkoordinatensystems (der Mitte der Hinterachse des Fahrzeugs) und dem tatsächlichen Standort der GPS-Antenne (einer bestimmten Position auf dem Dach) zu kompensieren. Außerdem ist die Beschreibung bzw. die Berechnung der geographischen Koordinaten mittels UTM-Koordinatensystem (von englisch Universal Transverse Mercator coordinate system) notwendig. Diese beiden Berechnungen werden nach [Heg18] im Vorverarbeitungsprozess unter Verwendung einiger Algorithmen von iff abgeschlossen und werden hier nicht ausführlich erläutert. Außerdem ist das UTM-Koordinatensystem tatsächlich ein nordweisendes, rechtsdrehendes Koordinatensystem. Jedoch wird näher in Abschnitt 3.4.1 in ein gebräuchliches, linksdrehendes Koordinatensystem umgerechnet. Im Rahmen dieser Arbeit beziehen sich die Koordinaten im Weltkoordinatensystem auf die verarbeitete bzw. umgerechnete UTM-Koordinateninformationen.

### 3.3.2 Vehicle Coordinate System (VCS)

In Abbildung 3.19 wird das blaue Rechteck verwendet, um das Fahrzeug einfach darzustellen. Wie in Abschnitt 3.1.1 erwähnt, liegt der Ursprung des Fahrzeugkoordinatensystems in der Mitte der Hinterachse des Fahrzeugs. Die X-Achse des VCS zeigt die Längsrichtung des Fahrzeugs nach vorne. Die Y-Achse verläuft senkrecht zur X-Achse und zeigt nach links der Fahrtrichtung. In dieser Arbeit sind die mittels VCS angegebenen Originaldaten die Punktwolkenpositionsinformationen des Laserscanners, und der Ausdruck der vom Fahrzeug eingenommenen Position. Darüber hinaus erfordert die Darstellung des vom Fahrzeug abgedeckten Raums auch die Hilfe von Fahrzeugkoordinatensystem. Hierbei ist zu beachten, dass die Koordinateninformation der Punktwolke jedes Laserscanners tatsächlich auf dem unabhängigen Koordinatensystem jedes Laserscanners basiert. Unter dem bestehenden Rahmen von iff wird jedoch die Umrechnung zwischen jedem Sensorkoordinatensystem und dem Fahrzeugkoordinatensystem somit die Kombination aller Sensordaten während der

Vorverarbeitung abgeschlossen. Schließlich wird in Form von ROS-Bag die Punktwolke aller Sensoren basierend auf den Koordinateninformationen des Fahrzeugkoordinatensystems bereitgestellt.

### 3.3.3 Anchor Coordinate System (ACS)

Ankerkoordinatensystem ist ein Hilfskoordinatensystem, das auf den Erfahrungen von [WSD07] [Pie13] basiert. Aufgrund des Speicherbedarfs und der Performance ist es unmöglich und auch unnötig, einen sehr großen Bereich von Umgebungsinformationen aufzuzeichnen und zu aktualisieren. Daher ist ein Wahrnehmungsbereich des Fahrzeugs, wie das schwarze Quadrat in Abbildung 3.19 geplant. Dieser Wahrnehmungsbereich befindet sich im engen Raum des Fahrzeugs und bewegt sich mit der Änderung der Positionsinformationen des Fahrzeugs. Um die Position und Größe des Bereichs vollständig anzuzeigen, wird neben der Länge und Breite des Bereichs auch ein Ankerpunkt benötigt. Normalerweise wird dieser Ankerpunkt in der unteren linken Ecke des Wahrnehmungsbereichs eingerichtet. Das mit diesem Ankerpunkt als Ursprung festgelegte Koordinatensystem wird als Ankerkoordinatensystem bezeichnet. Es ist jedoch anzumerken, dass dieses Koordinatensystem nur mit der Position des Fahrzeugs verschoben wird. Die Richtung seiner Koordinatenachse ändert sich nicht, da das rotierende Koordinatensystem Aliasing und geringe Qualität des Umfeldmodells verursacht [WSD07] [Heg18]. Zusätzlich wird innerhalb dieses Bereichs der Raum in eine Gitterzelle diskretisiert, siehe Abbildung 3.20. Der Ursprung des ACS ist der Ausgangspunkt der in Abschnitt 2.1.3 erwähnten Diskretisierung und auch die 0-Stelle des Index. Abbildung 3.20 zeigt auch die Einschränkungen des Umfeldmodells hinsichtlich der Position des Fahrzeugs auf der Karte. Wenn sich die Position des Fahrzeugs nicht wesentlich ändert, muss die Position des Ursprungs des ACS nicht jederzeit aktualisiert werden. Um den Fahrbereich des Fahrzeugs weiter einzuschränken, wird er im Allgemeinen nach [WSD07] [Heg18] als mittlerer Teil der Karte festgelegt. Wenn das Fahrzeug den Bereich verlässt, wird das ACS aktualisiert, wodurch sich der Rechenaufwand verringert. Darüber hinaus wird in dieser Arbeit der Grenzwert des Bereichs parametrisiert und als Schnittstelle für die spätere Verwendung und Weiterentwicklung bereitgestellt.



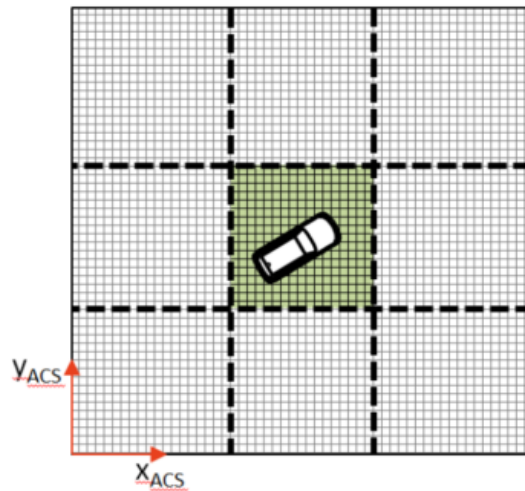


Abbildung 3.20: Anordnung der Position des Autos auf der Karte [Heg18]

### 3.3.4 Zusammenhang zwischen Koordinatensystemen

Zwischen den oben erläuterten Koordinatensystemen besteht ein Zusammenhang und die Umrechnung zwischen GCS, ACS und VCS gewinnt bei Implementierung des Umfeldmodells große Bedeutung. Wenn die  $x$ -Koordinate des Ankerpunkts  $anchor\_x$  und die  $x$ -Koordinate des Fahrzeugs  $pos\_x$  im GCS bekannt sind, wie in Abbildung 3.19 gezeigt, kann die  $x$ -Koordinate des Fahrzeugs im ACS durch die Formel  $offset\_x = pos\_x - anchor\_x$  erhalten werden. Dabei ist die Umrechnung auf der  $y$ -Achse analog zur  $x$ -Achse.

Darüber hinaus gibt es zwei Punkte, die besondere Aufmerksamkeit erfordern. Das erste ist die Aktualisierung bzw. Initialisierung von ACS. Im Modell wird auch die Zeit diskretisiert, um sich an Computerberechnungen anzupassen. Zu jedem einzelnen Zeitpunkt wird das ACS getestet, ob es aktualisiert werden muss und wie es sich bewegt. Dieser Prozess kann durch das in Abbildung 3.21 gezeigte Programmablaufdiagramm dargestellt werden. Dabei repräsentieren  $X\_1$  und  $X\_2$  jeweils die linke und rechte Grenze der  $X$ -Achse des grün befahrbaren Bereichs in Abbildung 3.20.  $Y\_1$  und  $Y\_2$  repräsentieren jeweils die unteren und oberen Grenzen des Bereichs. Außerdem geben  $dx$  und  $dy$  als positive Werte die Entfernung an, um die der Ursprung des ACS verschoben werden muss. Diese Werte sind so parametrisiert, dass sie je nach Anwendungsszenario jederzeit geändert werden können. Dabei beschreiben

$pox\_x$ ,  $pox\_y$  und  $offset\_x$ ,  $offset\_y$  die Position des Fahrzeugs im Weltkoordinatensystem und im Ankerkoordinatensystem. Der Kern des Algorithmus besteht darin, zu überprüfen, ob die Position des Fahrzeugs eine bestimmte Grenze überschritten hat, und sich entsprechend zu bewegen. Wenn beispielsweise  $offset\_x > X\_2$  gilt, bedeutet dies, dass die Position des Fahrzeugs die Grenze des befahrbaren Bereichs berührt oder überschritten hat. In diesem Fall bewegt sich der Anker nach rechts, indem der Wert der x-Koordinate erhöht wird, sodass das Fahrzeug immer in der Mitte der Rasterkarte bleibt. Dieser ganze Prozess wird als Funktionsmodul mit der Bezeichnung Update ACS betrachtet und zum Entwerfen der Initialisierung von ACS verwendet, wie in Abbildung 3.22 dargestellt. Die Initialisierung des ACS erfolgt gleichzeitig mit der Initialisierung des gesamten Umfeldmodells. Wenn gültige Fahrzeugpositionsinformationen erhalten werden, werden die Anfangskordinaten des Fahrzeugs auch der Anfangsposition des Ankers zugewiesen, wodurch die Anzahl der Bewegungen des ACS verringert wird. Anschließend wird mit dem Aktualisierungsmodul die Position des Ankers automatisch angepasst, bis sich die Fahrzeugposition innerhalb des eingestellten Fahrbereichs befindet.

Der zweite Punkt ist, dass Punktwolkeninformationen und die Visualisierung der Fahrzeugkarosseriestruktur von Koordinaten unter VCS in Koordinaten unter ACS umrechnet werden müssen, da der Ausgangspunkt der Diskretisierung Anker ist. Dieser Punkt wird im nächsten Abschnitt zur Verarbeitung von Sensorinformationen ausführlich erläutert.

### 3.4 Verarbeitung von Sensordaten

Für das Umfeldmodells in dieser Arbeit sind die beiden wichtigsten Sensorinformationen GPS-Informationen von dGPS-Moduls und Hindernisinformationen von Laserscannern. Unter Verwendung des vorhandenen Frameworks und Algorithmus in IfF werden GPS-Informationen in Form von UTM-Koordinaten angegeben. Wie in Kapitel erwähnt, sind im Rahmen dieser Arbeit die beiden Themen Eigenlokalisierung und Umfeldmodellierung entkoppelt, und der Schwerpunkt liegt auf der Umfeldmodellierung. Daher wird hier in Hinsicht auf die Erfassung und Verarbeitung der Daten Laserscannern vertieft eingegangen.

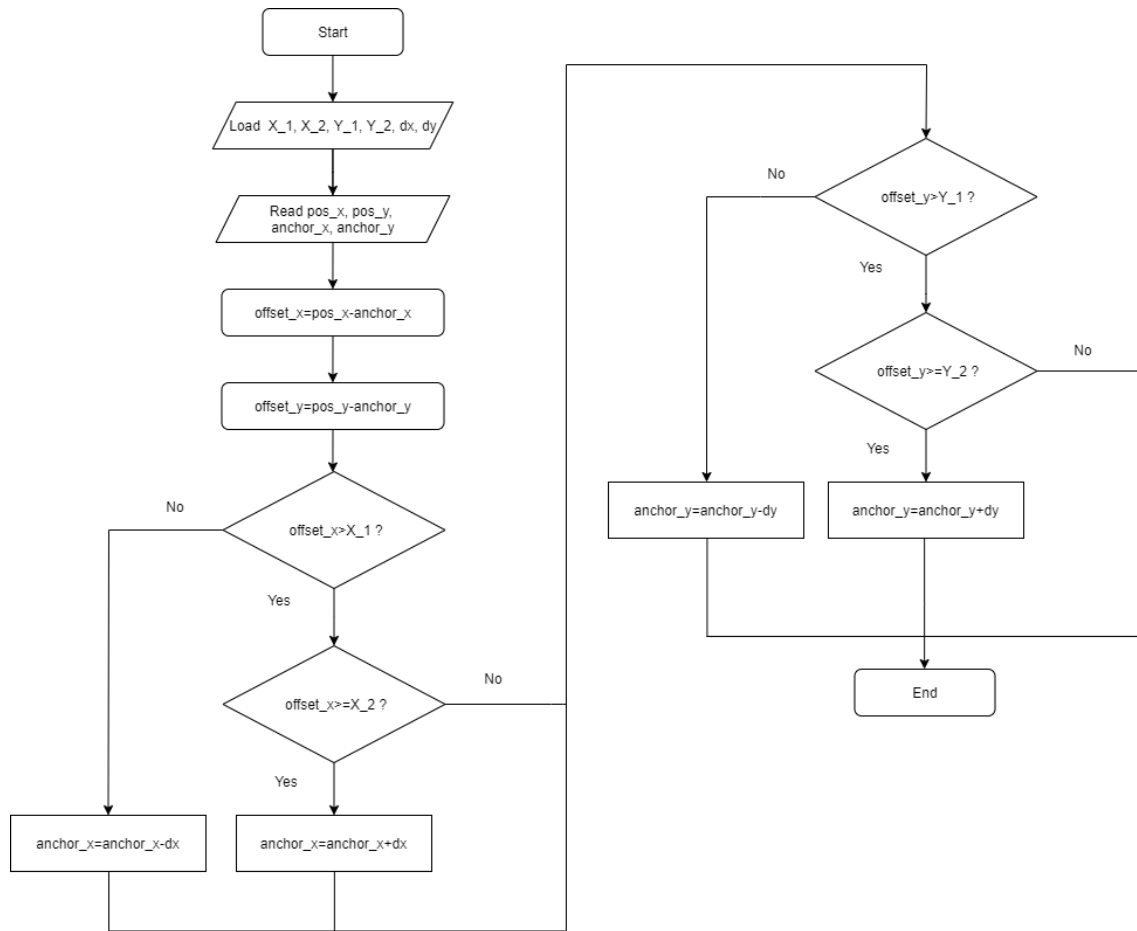
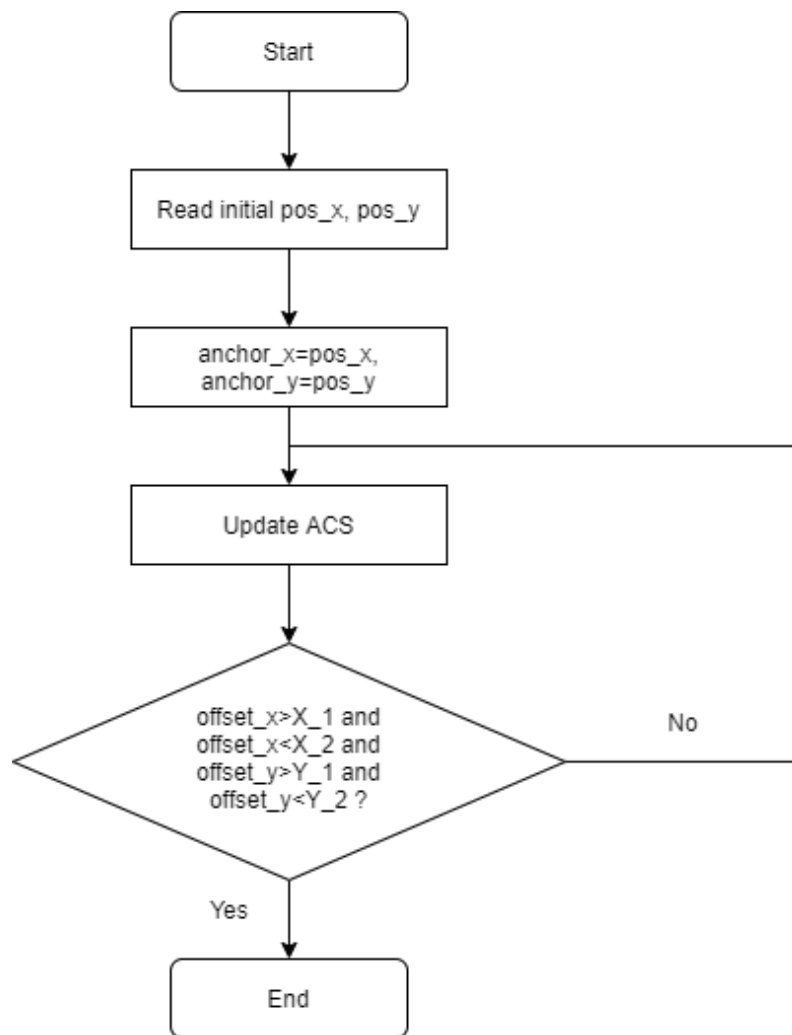


Abbildung 3.21: Programmablaufplan der Aktualisierung von ACS

### 3.4.1 GPS-Information

Die wesentliche Information, die GPS liefert, ist die Pose des Fahrzeugs, die die Positionsinformation  $pos_x$  mit  $pos_y$  und Orientierungsinformation  $pos_psi$  enthält. Hierbei ist aber zu beachten, dass das ausgewählte UTM-Koordinatensystem ein nordweisendes, rechtsdrehendes Koordinatensystem ist [Heg18]. Daher ist es bei der tatsächlichen Verarbeitung erforderlich, den Richtungswinkel in dem in Abschnitt 3.3.1 erwähnten linksdrehendes Koordinatensystem GCS durch Berechnung mittels Formel 3.1 zu berechnen. Dabei bezeichnet  $car\_get\_psi$  die ursprüngliche Datengröße des Fahrzeugrichtungswinkels. Diese Umrechnungsbeziehung kann auch durch Abbildung dargestellt werden. Diese Umrechnungsbeziehung kann durch Ab-



Abbildungung 3.22: Initialisierung von ACS

bildung 3.23 visuell dargestellt werden.

$$pos\_psy = -car\_get\_psi + 90^\circ \quad (3.1)$$

### 3.4.2 Laserscanner-Information

Die Daten von Ibeo-Laserscanner haben zwei Ausgabeformate. Eines sind Rohdaten, die auf einer Punktwolke basieren, und das andere sind Objektinformationen nach

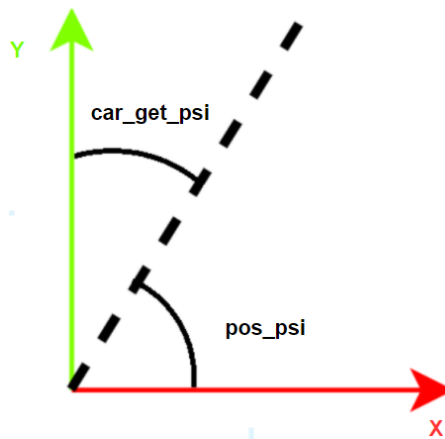


Abbildung 3.23: Umrechnung des Orientierungswinkels in GCS

der Verarbeitung von Rohdaten. Die geometrische Form des Objekts ist ein Rechteck. Das vorhandene Framework in IFF verwendet hauptsächlich Objektinformation, um ein Umfeldmodell bzw. eine Rasterkarte zu erstellen. In [Heg18] werden die Positions- und Größeninformationen von Objekten verwendet, gefolgt von Kartierung statischer Hindernisse. Wie in Abbildung 3.24 gezeigt, besteht der Kernschritt des Algorithmus darin, die Pose des Objekts in der Rasterkarte zu bestimmen, es als Punktwolkeninformation zu diskretisieren bzw. umrechnen und schließlich die belegten Zellen zu markieren. Die Verwendung dieses Ansatzes weist jedoch mehrere



Abbildung 3.24: Kartierungsalgorithmus mit Objektinformation von Laserscanner [Heg18]

Nachteile auf. Wie in Kapitel 3.4 erläutert, besteht einer der Vorteile der Verwendung des gitterbasierten Modells darin, dass es Hindernisse beliebiger Form ausdrücken kann. Bei Verwendung der Objektinformationen werden Hindernisse in diesem Fall jedoch immer durch Rechtecke dargestellt, wodurch dieser Vorteil zunichte gemacht wird. Zweitens werden in tatsächlichen Anwendungen z.B. mehrere diskrete Punk-

te als kontinuierliches Hindernis falsch eingeschätzt. Dies führt zu Ungenauigkeiten und geringer Qualität des Modells. Schließlich erfordert die Verwendung von Objektinformationen einen weiteren Schritt zur Umwandlung in eine Punktwolke, was den Rechenaufwand erhöht. Es ist direkter und natürlicher, die Punktwolkeninformationen des Laserscanners direkt zu verwenden.

Als nächstes wird die Verarbeitung von Punktwolkeninformationen mittels des in Abbildung 3.25 gezeigte Flussdiagramm ausführlich erläutert. Der Ibeo-Laserscanner liefert über den ROS-Treiber verschiedene Dateninformationen und publiziert diese zu den entsprechenden Topics. Das wichtigste ist, dass Topic *as\_tx/point\_cloud* Information liefert, deren Messagety *sensor\_msgs/PointCloud2* ist. Es ist anzumerken, dass diese Daten tatsächlich vom Datentyp  $\langle pcl :: PointXYZL \rangle$  der PCL-Standardbibliothek gekapselt und geliefert werden. Daher wird in der tatsächlichen Codeimplementierung Zeiger (engl. pointer) verwendet, um die 4 Beschreibungsinformationen der Punktwolke in  $\langle pcl :: PointXYZL \rangle$  zu lesen. Sie handelt sich um X-, Y- und Z-Koordinaten des Fahrzeugkoordinatensystems und der Schicht (engl. layer), in der sich die Punktwolke befindet.

Im Versuchsfahrzeug Passat (siehe Abbildung 3.13) sind Sensor 2 und Sensor 5 am Fahrzeug mit Ibeo-LUX-8L ausgestattet. Die restlichen Laserscanner sind Ibeo-LUX-4L. Ibeo-LUX-8L wird tatsächlich durch die Drehung des Objektivs konstruiert, um den vertikalen Erfassungsbereich zu verdoppeln. Wenn es in Kombination mit Ibeo-LUX-4L verwendet wird, ist der Erfassungsbereich zu zwei benachbarten Zeitpunkten bei derselben Frequenz inkonsistent. Beispielsweise ist die bei Zeitpunkt  $t_1$  erfasste Punktwolke nur in den Schichten 0 bis 3 verteilt, während die bei  $t_2$  erfasste Punktwolke sich einschließlich in den Schichten 4 bis 7 befinden. Diese Inkonsistenz kann durch Binär-Bayes-Filter die Korrektheit und Stabilität des Modells beeinträchtigen. Aus diesem Grund wird im Funktionsblock *LayerFilter* das Datenframe, das 4 bis 7 Schichten von Punktwolkeninformationen enthält, verworfen. Diese Methode ist direkt und einfach und verbessert nachweislich die Stabilität des Modells.

Im Funktionsblock *ChangeVCStoACS* wird die im Fahrzeugkoordinatensystem vorliegende Position jedes Punkt von Punktwolke in Ankerkoordinatensystem umgerechnet. Um die Umrechnung durchzuführen, ist ein Hilfskoordinatensystem (HCS), wie in Abbildung 3.26 gezeigt, erstellt. Dann lässt sich die Koordinatenumwandlung in 2 Schritte zerlegen. Der erste Schritt besteht darin, das Referenzkoordinatensys-

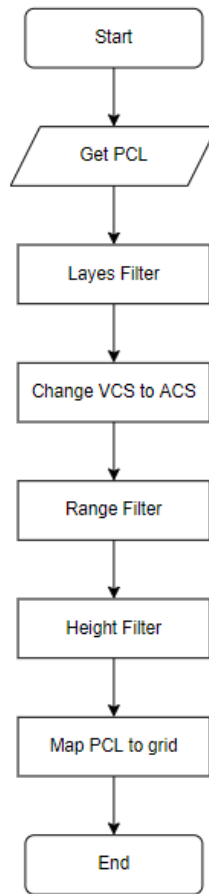


Abbildung 3.25: Programmablaufplan der Laserscannerdaten

tem jedes Punktes von ACS in HCS umzuwandeln. Die mathematische Beschreibung dieses Schritts ist in Gleichung 3.5 gezeigt. Dabei bezeichnet  ${}^V\mathbf{P}$  und  ${}^H\mathbf{P}$  den Koordinatenvektor eines bestimmten Punktes in VCS bzw. HCS. Die lassen sich mit Gleichung 3.3 und 3.4 beschreiben.  ${}^H\mathbf{R}_V$  ist als Drehmatrix oder Rotationsmatrix genannt und ihre konkrete Beschreibung findet sich in Gleichung ???. Darunter wird der Wert von *pos\_psi* direkt für  $\psi$  verwendet, da GCS und ACS immer in die gleiche Richtung zeigen.

$${}^H\mathbf{P} = {}^H\mathbf{R}_V {}^V\mathbf{P} \quad (3.2)$$

$${}^V\mathbf{P} = \begin{pmatrix} {}^Vx \\ {}^Vy \end{pmatrix} \quad (3.3)$$

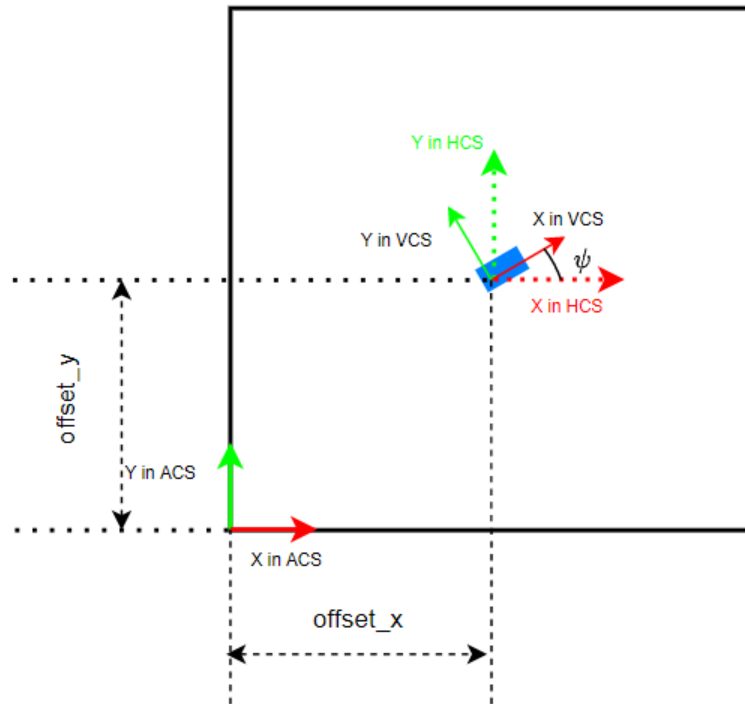


Abbildung 3.26: Darstellung eines Hilfskoordinatensystem

$${}^H\mathbf{P} = \begin{pmatrix} {}^Hx \\ {}^Hy \end{pmatrix} \quad (3.4)$$

Der zweite Schritt ist die Umwandlung von HCS in ACS, die durch einfache Vektoraddition erhalten wird. Analog bezeichnet  ${}^A\mathbf{P}$  den Positionsvektor in ACS. Offset-Vektor  $\mathbf{D}$  stellt die Abweichung von ACS und HCS dar, sodass kein bestimmtes Koordinatensystem angegeben werden muss.

$${}^A\mathbf{P} = {}^H\mathbf{P} + \mathbf{D} \quad (3.5)$$

mit

$$\mathbf{D} = \begin{pmatrix} offset\_x \\ offset\_y \end{pmatrix}$$

und

$${}^A\mathbf{P} = \begin{pmatrix} {}^Ax \\ {}^Ay \end{pmatrix}$$



Zusammenfassend kann die x-Koordinate und die y-Koordinate des Punktes im ACS unter Verwendung der Gleichungen 3.6 bzw. 3.7 berechnet werden.

$$^Ax = \cos(\psi) \times ^Vx - \sin(\psi) \times ^Vy + offset\_x \quad (3.6)$$

$$^Ay = \sin(\psi) \times ^Vx + \cos(\psi) \times ^Vy + offset\_y \quad (3.7)$$

Funktionsblock *Range Filter* in Abbildung 3.25 handelt sich um, dass alle Punkte von Punktwolken außer Erfassungsbereich bzw. Umfeldmodelldimension ausgefiltert werden. In Übereinstimmung mit dem IFF-Framework verfügt die Umfeldmodell bzw. Rasterkarte über 400×400 Gitterzellen. Jede Gitterzelle ist 0,25 m×0,25 m groß, daher wird auch die Auflösung der Rasterkarte als 0,25 m bezeichnet. In diesem Fall werden alle erfasste Punkte herausgefiltert, deren x- oder y-Koordinate 100 m im ACS-Koordinatensystem überschreitet. Natürlich werden auch die Anzahl der Gitterzellen und die Auflösung der Karte so parametrisiert, dass sie sich entsprechend den tatsächlichen Anwendungsanforderungen ändern können. Durch das Anordnen des Funktionsblocks *Range Filter* vor der Kartierung des Hindernis können unnötige Daten im Voraus verworfen und nutzlose Berechnungen vermieden werden.

Obwohl das Umfeldmodell in dieser Arbeit ein zweidimensionales Occupancy-Grid ist, enthält die vom Ibeo-Laserscanner erhaltene Punktwolke tatsächlich dreidimensionale Informationen. Daher hat es die Höheninformationen des Punktes  $z$ . Im Funktionsblock *Height Filter* ist die zu erkennende Höhe begrenzt und zu hohe oder zu niedrige Daten werden herausgefiltert. Dies kann erstens die abnormalen Punktwolkeninformationen beseitigen und zweitens die Anzahl von Punktwolken unterschiedlicher Höhe an derselben Stelle verringern, wodurch die Berechnungslast verringert wird. Der Grund liegt daran, dass der Beitrag von Punktwolken am selben Ort und in unterschiedlichen Höhen zum 2D-Umfeldmodell gleich und redundant ist. Im Funktionsblock *Map PCL to grid* wird die Punktwolke in den diskretisierten Gitterzellen weiter abgebildet. In der Implementierung wird ein zweistelliges 400×400-Array erstellt, und jedes ihrer Gitterzelle hat einen Index in x- und y-Richtung. Jede Punktwolkeninformation wird durch Gleichung 3.8 und 3.9 in einen Gitterindex umgewandelt, wobei dieses Gitter als belegt markiert wird. Im Rahme dieser Arbeit wird das Ergebnis in einem zweidimensionalen Array namens *pcl\_grid* gespeichert.

chert.

$$Index_x \approx {}^Ax/GS \quad (3.8)$$

$$Index_y \approx {}^Ay/GS \quad (3.9)$$

In der Gleichungen ist GS (Grid Spacing) die Auflösung der Rasterkarte. Die Rundung in den Gleichungen bedeutet, dass die berechneten Daten abgerundet werden, bevor sie als Indexparameter verwendet werden können. Darüber hinaus hat das wiederholte Markieren eines Quadrats keine Auswirkung.

### 3.4.3 Synchronization der Sensordaten

Unterschiedliche Sensordaten stammen aus in ROS unterschiedlichen Kanälen bzw. Themen. Die Sicherstellung der Zeitsynchronisation dieser Daten ist für die Echtzeitgenauigkeit des Modells sehr wichtig. Beispielsweise bleiben die Sensorinformationen von Lidar hinter den Informationen von GPS zurück, was dazu führt, dass die Hindernisinformationen um das Fahrzeug nicht rechtzeitig aktualisiert werden und das Modell daher ungenau ist. In ROS wird jede Informationsfreigabe von einem Zeitstempel (engl. timestamp) begleitet. Der *ApproximateTime Policy* Algorithmus in der Bibliothek (engl. library) von *message\_filters* wird verwendet, um sicherzustellen, dass die Zeitstempel von Sensorinformationen aus verschiedenen Datenquellen sehr nahe oder fast gleich sind, z. B. 10 Femtosekunde (fs). Informationen, die diesen Schwellenwert überschreiten, werden als ungültig betrachtet und verworfen. Dieses Verfahren stellt nicht nur die Synchronisation von Informationen sicher, sondern kann auch die aktuellen Rahmendaten filtern, wenn eine bestimmte Datenquelle abnormal ist, wodurch die Genauigkeit der Daten sichergestellt wird.

## 3.5 Implementierung von Occupancy Grid

Nach der Einführung von Koordinatensystemen und der Beschreibung der Sensor-datenverarbeitung wird in diesem Abschnitt der Kern des Modells, die Implementierung von Occupancy Grid, erläutert. Sie umfasst die Implementierung von 2 Ebenen, wie in Abbildung 2.3 dargestellt.

### 3.5.1 Raumdiskretisierung

Auf dieser Ebene wird die Umgebung des Fahrzeugs diskretisiert und als 2D-Rasterkarte beschrieben. Im ersten Schritt werden die Länge und Breite des Erfassungsbereichs bestimmt, und im zweiten Schritt wird die Auflösung der Rasterkarte bzw. die Größe jeder Gitterzelle festgestellt. Daraus ergibt sich die Anzahl der Zelle im gesamten Modell. Diese Werte sind parametrisiert und bieten eine Schnittstelle für nachfolgende Änderungen gemäß verschiedenen Anwendungsszenarien. Im Rahmen dieser Arbeit wird das Umfeldmodell zur Anpassung an das IfF-Framework als  $400 \times 400$ -Gitter beschrieben, wobei jede Gitterzelle  $0,1 \times 0,1$  Quadrat ist. Dies bedeutet, dass der Fahrzeugerkennungsbereich eine Fläche von  $40 \times 40$  m beträgt. Es ist sehr natürlich, ein zweidimensionales Array in C++ zu verwenden, um dieses Raster zu beschreiben, das eine bestimmte Zelle im Raum direkt indizieren und den entsprechenden zusätzlichen Wert lesen kann. Hierbei werden zwei zweidimensionale Arrays erstellt, nämlich *current\_grid* und Verlauf *history\_grid*. Ersteres wird verwendet, um die vom inversen Sensormodell erhaltenen Wahrscheinlichkeitsverteilungen zu speichern, nämlich  $p(m_i|z_t)$  in Gleichung 2.11. Letzteres wird verwendet, um die historisch akkumulierten A-posteriori-Wahrscheinlichkeit nach der Bayes-Filterung zu speichern, die in Gleichung 2.11 als  $p(m_i|z_{1:t-1})$  bezeichnet wird. Im aktuellen Frame wird der berechnete  $bel_t(m_i)$  als Daten im *history\_grid* des nächsten Frames verwendet. Da die Wahrscheinlichkeitsgenauigkeit von 1% ausreichend ist, ist die jeder Zelle zugewiesene Wahrscheinlichkeit eine Ganzzahl zwischen 0 und 100. In diesem Fall wird das Array aufgrund Speicherbedarf als Uint8-Datentyp erstellt.

### 3.5.2 Probabilistischer Ansatz

Der erste Schritt auf dieser Ebene ist die Zuweisung von *current\_grid*, bei der die aktuelle Belegungswahrscheinlichkeit jeder Zelle gemäß den Sensorinformationen ermittelt wird. Dies bedeutet, dass die Implementierung des in Abschnitt 2.2.5 genannten inversen Sensormodells. Wie in Abschnitt 2.2.5 erwähnt, wird das in dieser Arbeit verwendete Sensormodell durch zwei eindimensionale abschnittsweise definierte Funktionen beschrieben und für einen einzelnen Strahl verwendet. Die verschiedenen Lichtstrahlen, die von jedem Sensor emittiert werden, werden von demselben Modell beschrieben.

Der Implementierungsalgorithmus kann durch das in Abbildung 3.27 gezeigte Flussdiagramm dargestellt werden. Hier werden zwei Schleifen verwendet. Die äußere Schleife repräsentiert die Behandlung verschiedener Laserscanner am Fahrzeug. Die Zahl *laser\_num* repräsentiert die Anzahl der Sensoren im Fahrzeug. Die innere Schleife bezeichnet die Verarbeitung verschiedener emittierter Lichtstrahlen von einem einzelnen Sensor. Dabei ist *beam\_num* die Anzahl dieser Strahlen ist, die durch Dividieren des horizontalen Öffnungswinkels des Laserscanners durch die horizontale Winkelauflösung erreicht wird.

Der Funktionsblock *Calculate position of laser i in ACS* dient zur Bestimmung der Ortskoordinaten des Laserscanners, dessen ID *i* ist. Die Positionsinformationen des Sensors als Eingabeparameter des Umfeldmodell werden mit dem Fahrzeugkoordinatensystem als Referenzsystem angegeben. Die Erstellung des Umfeldmodell basiert auf dem Ankerkoordinatensystem, daher müssen die Koordinaten der Sensorposition auch in ACS bestimmt werden. Das Wesentliche dieses Problems ist immer noch das in Abschnitt 3.4.2 erwähnte Problem der Konvertierung von VCS in ACS, daher wird die Erörterung hierbei nicht wiederholt.

Den Zellen in Strahlrichtung werden je nach Abstand zum Laserscanner unterschiedliche Wahrscheinlichkeitswerte im Funktionsblock *Probability distribution* zugeordnet. Als eine Linie wird der eindimensionale Strahl vom berühmten Bresenham-Algorithmus realisiert. Wie in Abbildung 3.28 gezeigt, wird beispielsweise der aktuell verarbeitete Lichtstrahl durch eine blaue durchgezogene Linie dargestellt. Der rote Punkt zeigt die aktuelle Position des zu verarbeitenden Sensors an und dient als Startpunkt des Strahls. Der schwarze Punkt repräsentiert die Position des Objekts und dient als Ende des Strahls. Unter Anwendung von dem Bresenham-Algorithmus werden die Zellen, denen tatsächlich Wahrscheinlichkeitswerte zugewiesen werden, grau dargestellt. Der grün gestrichelte Teil stellt den Bereich dar, den der Sensor scannen kann. Die grüne gestrichelte Linie stellt den Bereich dar, den der Sensor scannen kann. In diesem Bereich wird in verschiedene Richtungen durch die oben erwähnte innere Schleife mit dem Schritt von der Winkelauflösung abgetastet. Dies wird auch als Raycasting bezeichnet. In Bezug auf den spezifischen Wahrscheinlichkeitswert haben die Elemente von *current\_grid*, das den Zellen entspricht, die in dem in Abschnitt 3.4.2 genannten *pcl\_grid* als belegt markiert sind, den Wert von *p\_max*. Wird beispielsweise die Position (3, 4) von *pcl\_grid* als belegt markiert, wird der Wert *p\_max* dem Element (3, 4) des Arrays von *current\_grid* zugewiesen. Den

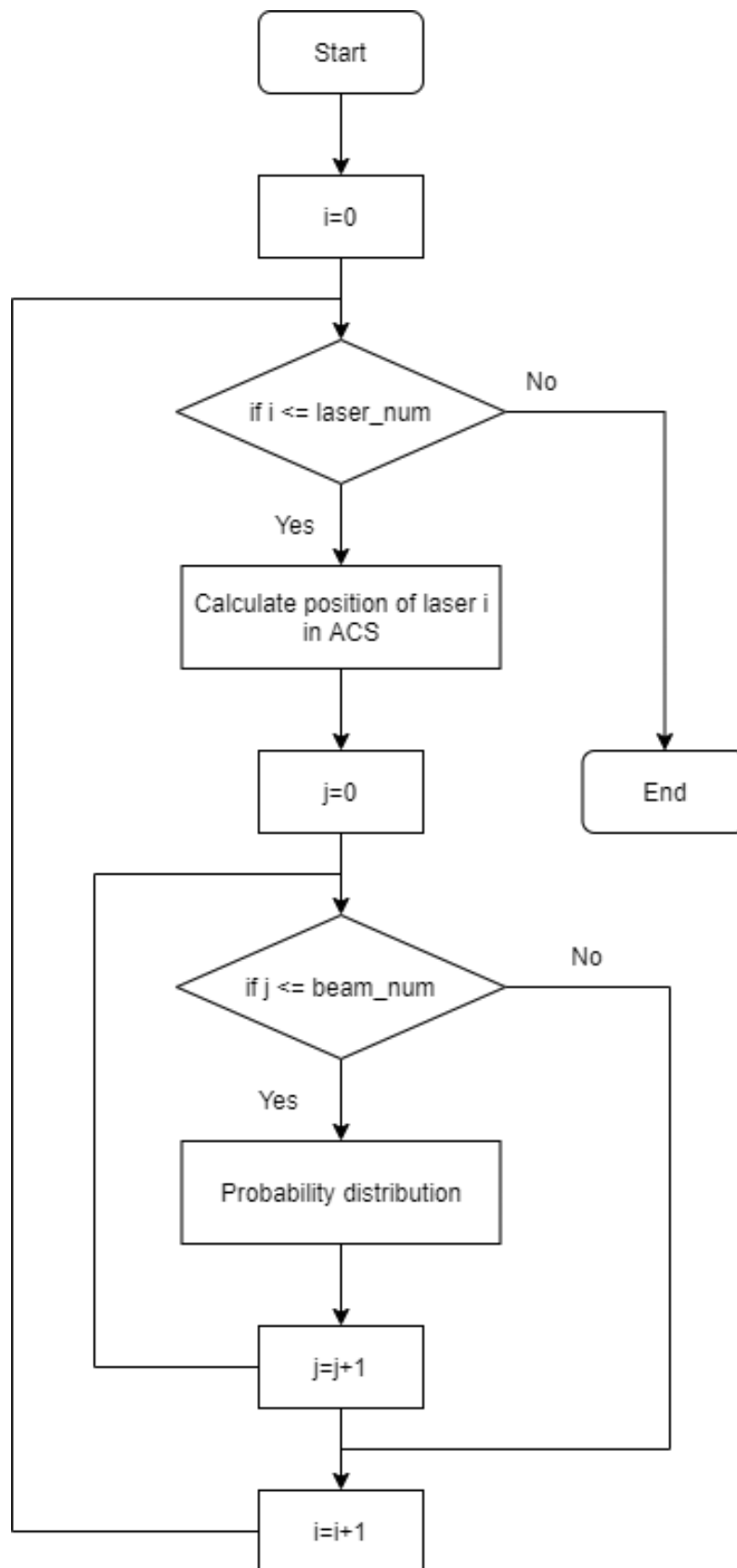


Abbildung 3.27: Implementierung des inversen Sensormodells

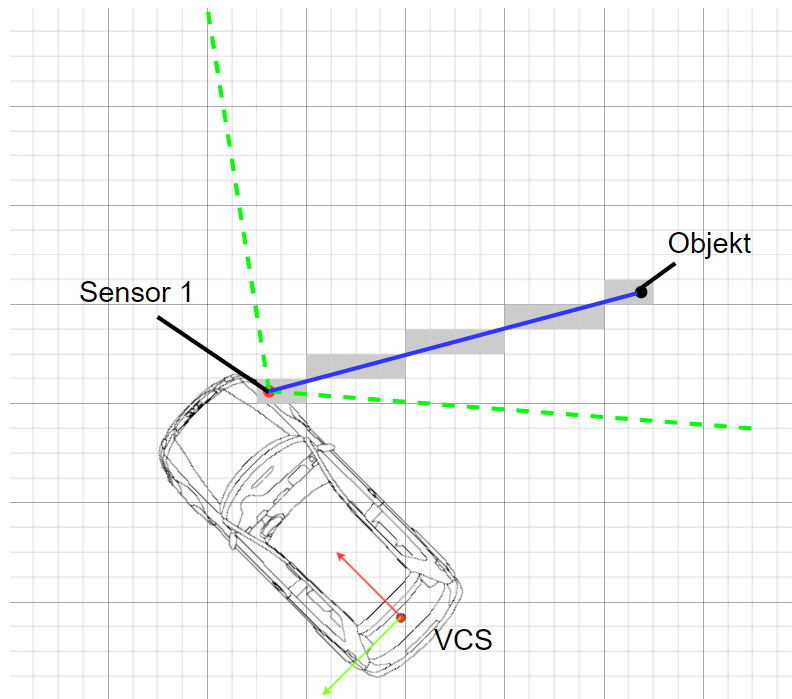


Abbildung 3.28: Raycasting mit Bresenham-Algorithmus

Zellen innerhalb des minimal erkennbaren Abstands wird  $p_{min}$  zugewiesen. Die Wahrscheinlichkeitsverteilung der in der Mitte bestehenden Zellen werden durch eine lineare Funktion dargestellt, und ihre Steigung wird durch eine Variable  $p_{slope}$  dargestellt. Der Sensor erkennt, dass sich an einem bestimmten Ort ein Hindernis befindet, oder der Sensor liefert die Information, dass sich kein Hindernis befindet. Diese beiden Situationen weisen eine unterschiedliche Genauigkeit auf. Die Variable  $p_{max}$  und  $p_{min}$  sind jeweils ein Indikator für die Genauigkeit dieser beiden Situationen. Die Variable  $p_{slope}$  verkörpert, wie weit die Sensordaten von der Entfernung zwischen dem Objekt und dem Laserscanner beeinflusst werden. Die oben genannten drei Variablen müssen entsprechend den tatsächlichen Anwendungen und Szenarien sowie den Eigenschaften und der Qualität der verwendeten Sensoren angepasst werden.

## 4 Implementierung

Um das ober

### 4.1 Unterkapitel 1

Beispiel für eine Fußnote<sup>2</sup>.

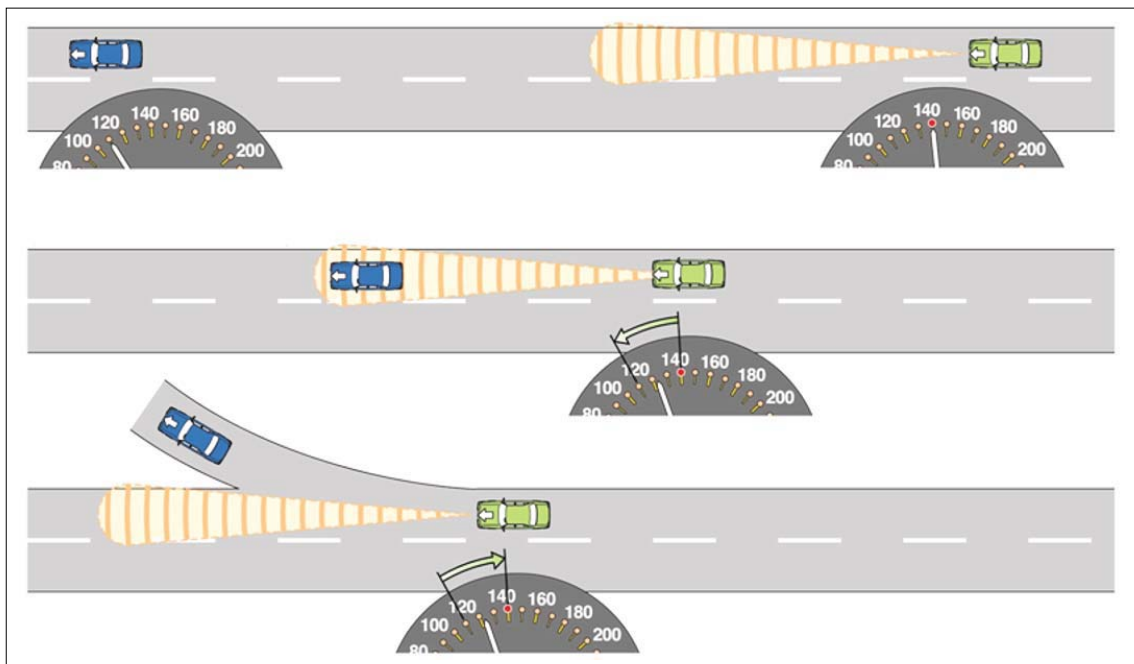


Abbildung 4.29: Folgefahrt Beispielbild

Wie in 4.1 gezeigt, ist es möglich, Kapitel zu [?]verlinken. Abkürzungen wie das Institut für Fahrzeugtechnik (IfF) werden so dargestellt. Quellen übrigens so [?].

Abbildung 4.29 soll ein Beispielhaftes Bild zeigen.

#### 4.1.1 Unterunterkapitel1

**Möglichkeit zur weiteren Unterteilung – Teilkapitel** Mehr als drei Unterteilungsebenen sollen vermieden werden!

---

<sup>2</sup>Fußnoten sind gut.

**Eingerücktes Teilkapitel** Derartige Dinge wie eingerückte Teilkapitel gibt es auch. Falls Du sie brauchen solltest.

Tabelle 4.1: Tabular Umgebung: Simpel formatierte Tabelle

Zwei miteinander verbundene Zellen	
Punkt 1	Adaptive Cruise Control
Punkt 2	Lane Keeping Assist
Schon wieder zwei miteinander verbundene Zellen	
Punkt 3	Elektronisches Stabilitätsprogramm

### 4.1.2 Das zu verwendendes Sensormodell

Nachfolgend das Beispiel für eine Aufzählung mit Itemize.

- Reaktionszeit  $\Delta t_{\text{react}}$
- Im oberen Punkt sieht man auch schön die Einbindung von Formeln in den Text.

$v_{Obj.} = 120 \text{ km/h}$

Einheiten sollten nicht kursiv dargestellt werden und werden vom Zahlenwert durch ein schmales Leerzeichen (Backslash und Komma) getrennt.

Es gibt eine Vielzahl von Gleichungsumgebungen. Equation ist eine davon. Die Gleichung ergibt natürlich keinen tieferen Sinn:

$$x_{obj} = \frac{s^2 \cdot a \cdot \sqrt{b}}{234 \cdot h_{ego}} \tag{4.1}$$



Tabelle 4.2: Vorteile und Nachteile von Kamera, Radar-, Lidar- und Ultraschallsensor

Sensor	Vorteile	Nachteile
Kamera	<ul style="list-style-type: none"> <li>• eine hohe Auflösung und Farbskalen über das gesamte Sichtfeld haben</li> <li>• eine farbenfrohe Perspektive der Umgebung bieten</li> <li>• eine 3D-Geometrie von Objekten bei Stereokameras bereitstellen</li> <li>• kostengünstig Im Vergleich zu Lidar sein</li> </ul>	<ul style="list-style-type: none"> <li>• ein leistungsfähiges Berechnungssystem benötigen, um nützliche Daten zu extrahieren</li> <li>• empfindlich auf starken Regen, Nebel und Schneefall reagieren</li> <li>• eine 3D-Geometrie von Objekten bei Stereokameras bereitstellen</li> <li>• begrenzte Reichweite besitzen</li> </ul>
Radarsensor	<ul style="list-style-type: none"> <li>• lange Strecken bei schlechten Sichtverhältnissen vor dem Auto sehen</li> <li>• klein, leicht und erschwinglich sind</li> <li>• weniger Strom als ein Lidar-Sensor benötigen</li> <li>• im Vergleich zu Lidar robuster gegen Ausfälle sein</li> </ul>	<ul style="list-style-type: none"> <li>• eine geringe Genauigkeit und Auflösung bieten</li> <li>• begrenzte Informationen (z. B. weder genaue Form noch Farbinformationen) bekommen</li> <li>• das Problem wegen der gegenseitigen Beeinflussung von Radarsensoren haben</li> <li>• schlechte Azimut- und Höhenauflösung verfügen</li> <li>• ohne einer Erhöhung der Leistung Radardämpfung zeigen</li> </ul>
Ultraschallsensor	<ul style="list-style-type: none"> <li>• lange Strecken bei schlechten Sichtverhältnissen vor dem Auto sehen</li> <li>• klein, leicht und erschwinglich sind</li> <li>• weniger Strom als ein Lidar-Sensor benötigen</li> <li>• im Vergleich zu Lidar robuster gegen Ausfälle sein</li> </ul>	<ul style="list-style-type: none"> <li>• eine geringe Genauigkeit und Auflösung bieten</li> <li>• begrenzte Informationen (z. B. weder genaue Form noch Farbinformationen) bekommen</li> <li>• das Problem wegen der gegenseitigen Beeinflussung von Radarsensoren haben</li> <li>• schlechte Azimut- und Höhenauflösung verfügen</li> <li>• ohne einer Erhöhung der Leistung Radardämpfung zeigen</li> </ul>

Tabelle 4.3: Tabular technische Details von Ibeo LUX 8L

Technische Daten	Wert
Reichweite	50m mit 10% Remission
Genauigkeit	10cm
Entfernungsauflösung	4cm
Horizontaler Öffnungswinkel	110°?50° <i>bis</i> – 60°?
Vertikaler Öffnungswinkel	6.4°
Horizontale Winkelauflösung	0.25°
Vertikale Winkelauflösung	0.8°
Bildrate	25.0 Hz
Multi-Layer	8 (2 Paare von 4 Layers)
Ausgabe	Roh- und Objektdaten
Abmaße (B×T×H)	164,5×93,2×88mm
Gewicht	998,7g

Tabelle 4.4: Tabular Umgebung: Etwas komplexere Tabelle

	Fahrzeug-CAN	GPS-System	ESP-Cluster	Correvit	Lichtschränke	Radar	Lidar	Kreiselpattform	Webcam, Leuchtdiode
Subjekt									
Setzgeschwindigkeit									x
relative Größen									
Triggersignal		x			x				x
Objekt									
Längsgeschwindigkeit	x	x		x					

## Literatur

- [.0619] *2019 IEEE International Conference on Robotics and Biomimetics (RO-BIO)*. IEEE, 06/12/2019 - 08/12/2019 . – ISBN 978–1–7281–6321–5
- [.1307] *2007 IEEE Intelligent Vehicles Symposium*. IEEE, 13/06/2007 - 15/06/2007 . – ISBN 1–4244–1067–3
- [BGC<sup>+</sup>19] BADUE, Claudine ; GUIDOLINI, Rânik ; CARNEIRO, Raphael V. ; AZEVEDO, Pedro ; CARDOSO, Vinicius B. ; FORECHI, Avelino ; JESUS, Luan ; BERRIEL, Rodrigo ; PAIXÃO, Thiago ; MUTZ, Filipe ; VERONESE, Lucas ; OLIVEIRA-SANTOS, Thiago ; SOUZA, Alberto Ferreira D.: *Self-Driving Cars: A Survey*. (2019)
- [Bus05] BUSCHKA, Pär: *Örebro studies in technology*. Bd. 20: *An investigation of hybrid maps for mobile robots*. Örebro : Universitetsbiblioteket, 2005. – ISBN 91–7668–454–7
- [Eff09] EFFERTZ, Jan: *Autonome Fahrzeugführung in urbaner Umgebung Autonome Fahrzeugführung in urbaner Umgebung durch Kombination objekt- und kartenbasierter Umfeldmodelle*, Technischen Universität Carolo-Wilhelmina zu Braunschweig, Dissertation, 2009
- [Elf89] ELFES, Alberto: Using occupancy grids for mobile robot perception and navigation - Computer. In: *IEEE* 22, No. 6 (1989), S. 46–57
- [Fer15] FERNÁNDEZ, Enrique: *Learning ROS for robotics programming: Your one-stop guide to the Robot Operating System / Enrique Fernández [and three others]*. Birmingham, UK : Packt Publishing, 2015 (Community experience distilled). – ISBN 978–1–78398–759–7
- [FHR<sup>+</sup>20] FAWAD AHMAD ; HANG QIU ; RAY EELLS ; FAN BAI ; RAMESH GOVINDAN: *CarMap: Fast 3D Feature Map Updates for Automobiles*. In: *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. Santa Clara, CA : USENIX Association, 2020. – ISBN 978–1–939133–13–7, 1063–1081

- [Heg18] HEGERHORST, Torben: *Entwicklung eines Umfeldmodells für das automatisierte Fahren*. Braunschweig, Technische Universität Braunschweig, Masterarbeit, 2018
- [HKOB10] HOMM, Florian ; KAEMPCHEN, Nico ; OTA, Jeff ; BURSCHKA, Darius: Efficient occupancy grid computation on the GPU with lidar and radar for road boundary detection. In: *2010 IEEE Intelligent Vehicles Symposium*, IEEE, 21/06/2010 - 24/06/2010. – ISBN 978–1–4244–7866–8, S. 1006–1013
- [Ibe17] IBEO AUTOMOTIVE SYSTEMS GMBH: *ibeo LUX 4L / ibeo LUX 8L / ibeo LUX HD DATA SHEET*. <https://cdn.www.ibeo-as.com/f4b928ce695cc32b6e93db216f2e1bee56ddaa26/ibeo%2BLUX%2BFamily%2BData%2BSheet.pdf.pdf>. Version: 2017
- [Kou16] KOUBÂA, Anis: *Studies in computational intelligence, 1860-949x*. Bd. volume 625: *Robot operating system (ROS): The complete reference. Volume 1 / Anis Koubâa, Editors*. First edition. Cham : Springer, 2016 <http://link.springer.com/BLDSS>. – ISBN 978–3–319–26054–9
- [MAA<sup>+</sup>20] MOHAMMED, Abdul S. ; AMAMOU, Ali ; AYEVIDE, Folliivi K. ; KELOUWANI, Soussou ; AGBOSSOU, Kodjo ; ZIOUI, Nadjat: The Perception System of Intelligent Ground Vehicles in All Weather Conditions: A Systematic Literature Review. In: *Sensors (Basel, Switzerland)* 20 (2020), Nr. 22. <http://dx.doi.org/10.3390/s20226532>. – DOI 10.3390/s20226532
- [Pie13] PIERINGER, Christian: *Modellierung des Fahrzeugumfelds mit Occupancy Grids*. Passau, Universität Passau, Masterarbeit, 2013
- [QGS15] QUIGLEY, Morgan ; GERKEY, Brian ; SMART, William D.: *Programming robots with ROS*. First edition. Sebastopol, CA : O'Reilly Media, 2015. – ISBN 978–1–4493–2551–0
- [SK08] SICILIANO, Bruno ; KHATIB, Oussama: *Springer handbook of robotics*. Berlin and London : Springer, 2008. – ISBN 978–3–540–23957–4

- [TBF05] THRUN, Sebastian ; BURGARD, Wolfram ; FOX, Dieter: *Probabilistic robotics*. Cambridge, Mass. and London : MIT, 2005 (Intelligent robotics and autonomous agents). – ISBN 978-0-262-20162-9
- [WHLS15] WINNER, Hermann ; HAKULI, Stephan ; LOTZ, Felix ; SINGER, Christina: *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort / Hermann Winner, Stephan Hakuli, Felix Lotz, Christina Singer (Hrsg.)*. 3., überarbeitete und ergänzte Auflage. Wiesbaden : Springer Vieweg, 2015 (ATZ/MTZ-Fachbuch). – ISBN 978-3-658-05734-3
- [WSD07] WEISS, Thorsten ; SCHIELE, Bruno ; DIETMAYER, Klaus: Robust Driving Path Detection in Urban and Highway Scenarios Using a Laser Scanner and Online Occupancy Grids. In: *2007 IEEE Intelligent Vehicles Symposium*, IEEE, 13/06/2007 - 15/06/2007. – ISBN 1-4244-1067-3, S. 184–189
- [WSG10] WURM, Kai M. ; STACHNISS, Cyrill ; GRISETTI, Giorgio: Bridging the gap between feature- and grid-based SLAM. In: *Robotics and Autonomous Systems* 58 (2010), Nr. 2, S. 140–148. <http://dx.doi.org/10.1016/j.robot.2009.09.009>. – DOI 10.1016/j.robot.2009.09.009. – ISSN 09218890

## Anhang

### Hier sind zusätzliche Infos einzubringen

Das Beispiel für eine Tabbing Umgebung zeigt, dass es möglich ist, mehrere Zeilen mit dem gleichen Einzug darzustellen:

$$v_{Start} = 120 \text{ km/h} = 33,3 \text{ m/s}$$

$$v_{End} = 80 \text{ km/h} = 22,2 \text{ m/s}$$

$$v_{Diff} = 40 \text{ km/h} = 11,1 \text{ m/s}$$