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

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

IfF Institut für Fahrzeugtechnik

Symbolverzeichnis

Δh_{ax}	$[m/s^2]$	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 vergangen 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 Umfelderfassung bzw. die Umfeldmodellierung des Fahrzeugs zugrunde gelegt. Das Ziel der vorliegende Arbeit ist auf Modellierung und dazu ihre Implementierung gerichtet.

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 Einleitung 1.2 erwähnte Ziel zu erreichen, sollten zunächst die theoretische Grundlagen geschaffen, bevor die praktische Umsetzung bzw. Implementierung stattfindet. Im Folgenden werden die für Umfeldmodellierung relevanten Theorie vorgestellt. Es ist offensichtlich, dass für die Umfeldmodellierung eine Wahrnehmung über das Umfeld gefordert, weshalb die für Umfeldwahrnehmung Sensorik darauffolgend gesprochen wird.

2.1 Umfeldmodellierung

Für Perzeption eines autonomen Fahrzeuges gibt es im wesentlichen zwei Bestandteilen [BGC+19]. Zum einen ist die Lokalisation. Die aktuelle Pose, das heißt die Position und Orientierung des eigenen Fahrzuges, ist hierbei zu ermitteln. Zum anderen ist die Umgebung um das Fahrzeug zu erfassen. Dabei wird es als Umfeldmodellierung genannt. Diese zwei Aspekte werden in vieler Forschung und Anwendung als ein Problem gekoppelt. Dies ist als SLAM (Simultaneous localization and mapping) bekannt. Jedoch trennen sich diese zwei Aspekte im Rahmen dieser Arbeit und die Aufmerksamkeit ist auf Umfeldmodellierung zu lenken. Hierbei wird eine Annahme getroffen, dass die Eigenpose des Fahrzeugs ohne Information der Umfeldmodell genug präzis ist. Im bestehenden ifF-Framework wird die Eigenpose mit GPS-Daten unter Anwendung des Kalmann-Filters übergeben.
In Bezug auf Robotik und autonomes Fahren dient Umfeldmodellierung als ein kompaktes Verfahren zu der mathematischen Repräsentation von der Umgebung um ein Auto.[Pie13] Das schafft eine unentbehrliche solide Grundlage, auf der der Entwurf und die darauffolgende Verwirklichung aller höheren automatisierten Fahrfunktion.

2.1.1 Mögliche Umfeldmodelle

Methoden zur Modellierung eines Umfelds basieren sich auf verschiedene entsprechenden Umfeldmodelle. Die maßgebliche Umfeldmodelle bauen auf erfolgreichen Erfahrung in dem Themenbereich der Robotik[Pie13]. Wie in Abbildung 2.1 ge-

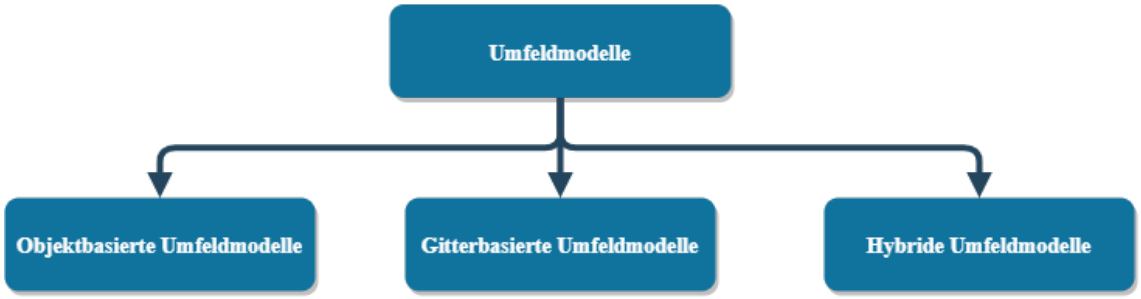

Abbildung 2.1: Mögliche Umfeldmodelle für autonome Fahrzeuge

zeigt, stehen aktuell in Forschung und Automobilindustrie meistens drei mögliche Umfeldmodelle zur Verfügung[Heg18].

2.1.2 Objektbasierte Modelle

Objektbasierte Modelle werden auch als merkmalbasierte Umfeldmodelle (engl. landmark-based oder feature map) genannt[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 im Praxis als bestimmte Objektklasse beschrieben. Jede Klasse wird mit hervorragenden und maßgeblichen Eingenschaften versehen, die deutlich durch die zutreffende Sensorik beobachtbar ist[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 genug präzis und einfach beschrieben werden können. Hierbei sind die Genauigkeit gegen der Zeit- und Speicheraufwand abzuwägen. Ein Beispiel ist ein in Abblidung 2.2 gezeigtes Umfeldmodell. Auf der linken Seite sind Merkmale dargestellt und das Bild rechts zeigt das generierte Umfeldmodell.
Objektbasierte Umfeldmodelle sind vor allem eignet für Szenen, wo die Objekte entweder im großen offenen Räume mit vordefinierten Merkmalen (z.B. Autobahn) oder dynamisch und einfach modelliert (z.B. Fußgänger oder Fahrzeuge) sind[Heg18][WSG10]. Da es im Rahmen dieser Arbeit ein statisches Umfeld im urbanen Raum angeht, ist dieses Modell ungeeignet und daher zu verzichten.

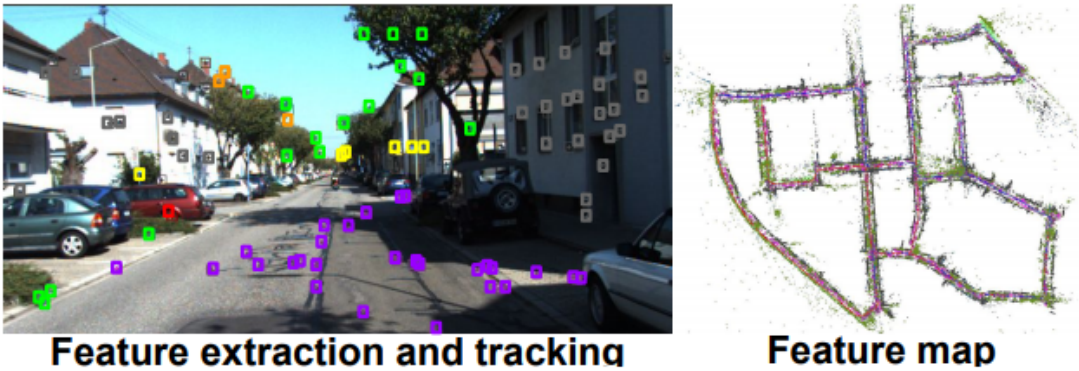

Abbildung 2.2: Ein objektbasiertes Umfeldmodell. [FHR+20]

2.1.3 Gitterbasierte Modelle

In gitterbasierten Modellen wird die zentrale Idee der probabilistischen Robotik eingeführt, die Wahrscheinlichkeitsverteilung unbeobachteter Zustände eines Systems bei gegebenen Beobachtung und Messungen zu schätzen. Zur Modellierung des Umfelds ist der zentrale und entschiedene 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 das Gitter frei und belegt ist.
Ein Modell, welches auf der obenerwähnten Idee aufbaut, wird als Occupancy Grid beschrieben. Das Occupancy Grid ist ein mehrdimensionales Zufallsfeld, das stochastische Schätzungen des Belegungszustands der Zellen in einem räumlichen Gitter halten [Elf89]. Wie in Abbildung 2.3 gezeigt, lässt Occupancy Grid sich in zwei Ebenen zerlegen.

1	Raumdiskretisierung
2	Probabilistischer Ansatz

Abbildung 2.3: Zwei Ebenen von Occupancy Grid

Auf der ersten Ebene wird die Raumdiskretisierung ausgeführt. Im Rahmen dieser

Arbeit wird das Umfeld als zweidimensionalen Raum dargestellt. Die Diskretisierung des Raumes bedeutet hierbei, dass die Fahrzeugumgebung durch Gitter diskreter und fester Größe dargestellt wird [Heg18]. Die grafische Darstellung ist in Abbildung 2.4 gezeigt. Außerdem wird jeder Zelle zusätzliche Informationen hinzugefügt. In Hinsicht auf Umfeldmodellierung ist die entschiedene Information die Belegungszustand der Zelle. Zur Vereinfachung der Modellierung, wird eine Annahme auf dieser Ebene zugleich getroffen, dass die Belegungszustand einer Gitterzelle ist unabhängig von diejenige der Nachbarzellen. Obwohl die Annahme anders als Realität aussieht, ist es erwiesen, dass Occupancy Grid Modell in Praxis unter dieser Annahme robust ist und zudem die Rechenkomplexität reduzieren kann [TBF05].

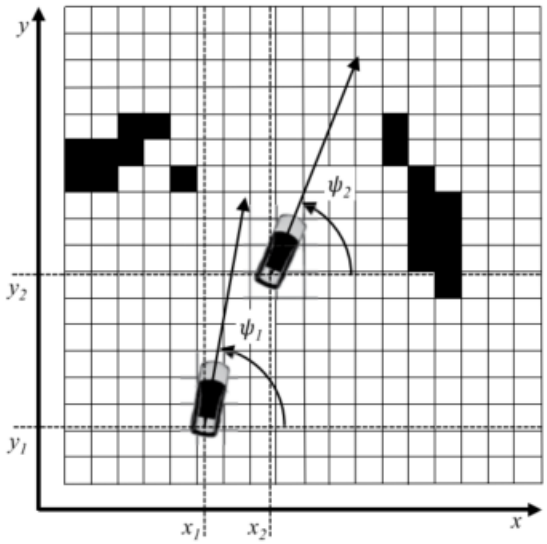

Abbildung 2.4: Raumdiskretisierung der Umgebung um Fahrzeug [WHLS15]

Auf der zweiten Ebene ist der probabilistische Ansatz eingeführt, welcher aus Themengebiet von Robotik stammen [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 aufteilen. Im Zentrum des Modells steht ein Algorithmus-Framework, der als binärer Bayes-Filter (engl. Binary Bayes Filter) bekanntlich ist. Die Eingaben von dem Framework bestehen aus Messungswerte, inverses Sensormodell und Anfangsbedienungen. Die Ausgabe ist A-posteriori-Wahrscheinlichkeit von dem Zufallsereignis, wenn die Zelle belegt ist.

Um die Konzept von Occupancy Grid zu beleuchten, wird im Folgenden auf jede Komponente vertieftet eingegangen.

Abbildung 2.5: Bestandteile des Modells Ocuupancy Grid

Der Binary-Baye-Filter ist zuerst zu beleuchten, mit dem die Funktionen und Bedeutungen von anderen Bestandteilen des Modells eng verbunden sind. Der Bayes-Filter ist ein rekursives Berechnungsvorschrift zur Schätzung von Wahrscheinlichkeitsverteilungen unbeobachteter Zustände eines Systems bei gegebenen Beobachtungen und Messungen [TBF05]. Die Zustandsschätzung befasst sich mit dem Problem der Schätzung von Größen aus Sensordaten, die nicht direkt beobachtbar sind, aber abgeleitet werden können. Das Ziel ist den Zustand x eines Systems zu schätzen[1], wenn Beobachtung z und Kontrolle u gegeben sind. Das heißt, die in Gleichung 2.1 gezeigte mathematische Formulierung sollte bestimmt werden. Hierbei entspricht x_t 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 x_t ist [TBF05].

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

Um die Wahrscheinlichkeitsverteilung einzugehen und eine rekursive Form zu entdecken, ist der stochastische Prozess der Zustandsänderung als Hidden Markov Model (HMM), auch dynamic Bayes network (DBN) genannt, zu betrachten. Unter Anwen-

[1] Präzis wird den Zustand $X = x$ formuliert, wobei X eine Zufallsvariable ist und x ist der spezifische Wert, den X in Echtzeit annimmt. Zur Vereinfachung der Notation wird die Formulierung $X = x$ im Rahmen dieser Arbeit sowie in vieler Literaturen als x bezeichnet.

dung dieser Annahme gilt: Die Wahrscheinlichkeit eines Zustandes bei Zeitstempel t (x_t) ist einschließlich abhängig von Zustand bei Zeitstempel $t-1$ (x_{t-1}) und nicht von Zuständen bei Zeitstempel, die früher als $t-1$ sind. Mathematisch wird HMM als eine Gleichung in 2.2 beschrieben.

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

Außerdem beschreibt Hidden Markov Model, wie in Abbildung 2.6 dargestellt, die vereinfachte Zusammenhang zwischen Zustand x, Messgröße z und 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. Unter zusätzliche Anwendung des Satzes von Bayes und des Gesetzes der totalen

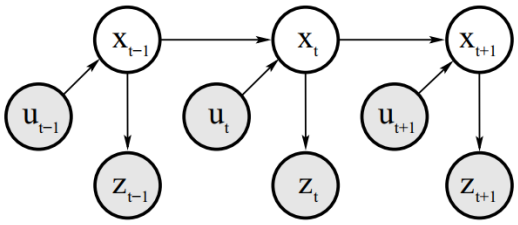

Abbildung 2.6: Hidden Markov Model (HMM) [TBF05]

Wahrscheinlichkeit wird die Formulierung 2.1 in einen rekursiven Ausdruck abgeleitet, was als Bayes-Filter bekanntlich ist. Die Ableitung findet sich in [TBF05]. Der daraus resultierte Algorithmus lautet in Tabelle 2.1: Der Algorithmus umfasst

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 $bel(x_t)$

2 Schritte, welche als Prädiktion (engl. prediction) und Korrektur (engl. Correction oder update) bekanntlich sind. Der erste Schritt wird dadurch ausführt, dass der Glauben 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 Glauben, indem die beobachte Informationen des Zustandes bzw. die Messgrößen der Sensorik fusioniert und berüchtigt wird. Der Wert $bel(x_t)$ wird als A-posteriori-Verteilung genannt. Im Gegensatz zu A-priori-Verteilung verkörpert A-posteriori-Verteilung den theoretisch präziseren Glauben, nachdem die Messgrößen von Sensorik durch ein Sensormodell die Genauigkeit des Glaubens beiträgt. Dieser Algorithmus bildet eine Grundlage, auf der viele weitere Algorithmen für bestimmte Szenarien entwickelt werden. Dazu gehören bekanntlich Gauß-Filter mit ihren Varianten, Particle-Filter und der diskrete Bayes-Filter. Der binäre Bayes-Filter, der eine große Rolle in dieser Arbeit spielt, ist ein spezieller Fall des Bayes-Filters bzw. des diskreten Bayes-Filter.

Eingeführt von dem zugrunde liegenden Bayes-Filters, benötigt der binäre Bayes-Filter eine Annahme. Es ist angenommen, dass der Zustand einschließlich zwei Möglichkeiten besitzen. Das heißt, der Zufallsvariable, der den Zustand repräsentiert, kann nur zwei Werte annimmt. Bei Occupancy Grid kann der wichtige und auch einzige Zustand der Belegungszustand, welche lediglich zwei Fälle - belegt oder frei - sein. 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. Das heißt, dass der Belegungszustand sich nicht im Laufe der Zeit verändert und die Kontrolle u hat keine Wirkung auf den Belegungszustand. Somit ist das Schema des stochastischen Prozesses der Zustandsveränderung von Abbildung 2.6 auf Abbildung 2.7 reduziert, indem die Kontrollen ausgeklammert werden. Es ist nach dieser zwei Annahme deutlich, 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 Information von Messdaten und der vorherige 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 Sensormodell. 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, das ein inverses Sensormodell eingesetzt wird, 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

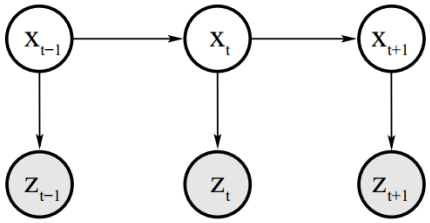

Abbildung 2.7: Das reduzierte HMM [TBF05]

der Belegungszustand einer Zelle oder mehrerer Zellen anhand der Sensordaten bestimmt werden kann. Ein Vorwärtsensormodell ist dagegen in diesem Fall erstaunlich schwierig. Mit dem Ziel, ein vereinfachter Algorithmus, der ein inverses Sensormodell verwendet, zur Berechnung des Glauben $bel(x_t)$ zu finden, sollte die mathematische Ableitung nach [TBF05] [WSD07] [Heg18] folgend vorgestellt.
Da Belegungszustand statisch ist und Kontrolle u somit ignoriert wird, kann die Zielgleichung 2.1 zur vereinfachten Gleichung 2.3.

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

Bei Occuancy Grid ist der interessierte 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.

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

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

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

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

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

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

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

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

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

Analog dazu hat der gegenteilige Zustand den Glauben

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

Dividiert Gleichung 2.8 durch der Gleichung 2.9, so ergibt sich

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

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 eines Ereignis, 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 inverses Sensormodell. Es ist ersichtlich, dass eine bedeutende Zusammenhang zwischen dem Modell bzw. der Beschreibung des Modells und den Arten der Sensoren. Darauf wird in Abschnitt 2.2 mit der Erfassung des Umgebung vertieftet eingegangen. Der Term $p(m_i|z_{1:t-1})$ beweist dabei die wichtige Eingenschaft des binären Bayes-Filters, dass das Verfahren der Schätzung auf Rekursion beruht. Die Anfangsbedingung bzw. die A-priori-Wahrscheinlichkeit ist als der Term $p(m_i)$ in Gleichung 2.10 bezeichnet. Die A-priori-Wahrscheinlichkeit gibt den Glauben an, vordem alle Messdaten von Sensorik berüchtigt werden. Typischerweise wird bei Occupancy Grid anfänglich $p_(m_i)$ den Wert 0,5 gegeben, weil es keine Information über die Belegungszustand gibt. Die Wahrscheinlichkeit, dass eine Gitterzelle belegt ist, ist die gleichen wie die, dass sie nicht belegt ist. Daher wird die Gleichung 2.10 zu

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

mit

$$Y = \frac{p(m_i|z_t)}{1 - p(m_i|z_t)} \frac{p(m_i|z_{1:t-1})}{1 - p(m_i|z_{1:t-1})} \tag{2.12}$$

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

2.1.4 Hybride Umfeldmodelle

Um ein Umfeld zu modellieren, welches komplizierte Szenarien repräsentieren kann, ist selbstverständlich Hybride Umfeldmodelle aufgefordert. Dadurch lassen sich die Einschränkung von jedem grundlegenden Modell eliminieren und die Vorteilen ausnutzen. Es gibt unterschiedliche Kriterien und Methoden um ein hybrides Umfeldmodell zu bilden. Das Modell, das [.0619] angeht, ist ein populäres 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-Map den statischen Raum darstellen. Auf dieser 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 noch direkt im gitterbasierten Umfeldmodell verlaufen kann[.0619]. Wird die Umgebung im Rahmen dieser Arbeit als ein statisches Umfeld betrachtet, wird ein Hybrides Umfeldmodell wegen Komplexität und Überflüssigkeit nicht angewandt.

2.2 Sensorik und ihr inverses Sensormodell

Zweifellos dient eine Wahrnehmung bzw. eine Erfassung als eine wichtige Voraussetzung zur Umfeldmodelleirung, weil sie die Informationsquelle bietet. Wahrnehmung eines Fahrzeugs besteht wesentlich aus die Bestimmung der eigenen Position samt

Orientierung und die Erfassung der Umgebung um das eigenen Fahrzeug. Die Lokalisierung ist im Rahmen dieser Arbeit nebensächlich und durch die vorgegebene GPS-Information bestimmt. Darauf soll an dieser Arbeit nicht näher eingegangen werden.

2.2.1 Überblick über verschiedene Sensoren zur Umfeldmodellierung

Um ein zuverlässiges Umfeldmodell zu entwickeln und danach das Modell in Praxis effektiv umzusetzen, spielt einer Auswahl der erfassenden Sensoren entweder in Akademie oder in Industrie eine große Rolle. Die wichtigsten und am weitverbreitetsten 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+20] sind die Vorteilen neben der Nachteilen in Tabelle 2.2 aufgelistet.

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

Sensor	**Vorteile**	**Nachteile**
Kamera	• eine hohe Auflösung und Farbskalen über das gesamte Sichtfeld haben • eine farbenfrohe Perspektive der Umgebung bieten • eine 3D-Geometrie von Objekten bei Stereokameras bereitstellen • kostengünstig Im Vergleich zu Lidar sind	• ein leistungsfähiges Berechnungssystem benötigen, um nützliche Daten zu extrahieren • empfindlich auf starken Regen, Nebel und Schneefall reagieren • eine 3D-Geometrie von Objekten bei Stereokameras bereitstellen • begrenzte Reichweite besitzen

FIR-Sensor	• nicht von der Lichtbedingungen und Objektoberflächenmerkmale beeinflusst werden können • eine bessere Sicht durch Staub, Nebel und Schnee als Kameras haben • eine horizontale Erfassungsreichweite bis zu 200m oder mehr abdecken • im Vergleich zu Lidar billiger und kleiner sind	• anspruchsvolle Rechenquellen und robuste Algorithmen erfordern • schwierig Ziele in Szenarien mit kaltem Klima zu unterscheiden • niedrigere Auflösung im Vergleich zur sichtbaren Kamera haben • keine Information über Entfernung bieten
Radarsensor	• lange Strecken bei schlechten Sichtverhältnissen vor dem Auto sehen • klein, leicht und erschwinglich sind • weniger Strom als ein Lidar-Sensor benötigen • im Vergleich zu Lidar robuster gegen Ausfälle sind	• eine geringe Genauigkeit und Auflösung bieten • begrenzte Informationen (z. B. weder genaue Formnoch Farbinformationen) bekommen • das Problem wegen der gegenseitigen Beeinflussung von Radarsensoren haben • schlechte Azimut- und Höhenauflösung verfügen • ohne einer Erhöhung der Leistung Radardämpfung zeigen

Ultraschall-sensor	• lange Strecken bei schlechten Sichtverhältnissen vor dem Auto sehen • klein, leicht und erschwinglich sind • weniger Strom als ein Lidar-Sensor benötigen • im Vergleich zu Lidar robuster gegen Ausfälle sind	• eine geringe Genauigkeit und Auflösung bieten • begrenzte Informationen (z. B. weder genaue Form- noch Farbinformationen) bekommen • das Problem wegen der gegenseitigen Beeinflussung von Radarsensoren haben • schlechte Azimut- und Höhenauflösung verfügen • ohne einer Erhöhung der Leistung Radardämpfung zeigen
LiDAR-Sensor	• große Entfernungen vor dem Auto bei guten Sichtverhältnissen erfassen • volle 360°- und 3D-Punktwolken bieten • eine gute Genauigkeit und Auflösung haben • keine signifikanten Interferenzen bei mehreren Lidarsensoren haben	• teurer als Radar und Kamera sind • kleine Objekte (wie Drähte und Stangen) nicht entdecken können • eine schlechte Kontrastunterscheidung bei der Erkennung nasser Oberflächen haben • durch unterschiedliche klimatische Bedingungen beeinflusst werden

Im Rahmen dieser Arbeit ist die am ifF (Institut für Fahrzeugtechnik) bereits bestehende Fahrzeugarchitektur mit Ibeo LUX Laserscanner versehen. Daher wird folgend in Abschnitt 2.2.2 auf der Mechanismus und das darauf resultierende Sensormodell des Laserscanners eingegangen. Außerdem werden die technische Details bei Einführung der inversen Sensormodellierung vorgestellt, weil die reale technische Größen dabei ein wichtiger Faktor sind.

2.2.2 Funktionsprinzip des Lasersensors

Ist ein angemessenes Sensormodell zu finden, ist es sinnvoll das Mechanismus, die Eingenschaften und die Gründe des 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 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 Laserscanner ähnlich wie Radarsensor, wobei nur anstatt Mikrowellen beim Laserscanner Infrarot-, Ultraviolett- oder Strahlen aus dem Bereich des sichtbaren Lichts eingesetzt werden [WHLS15]. Mathematische Beschreibung des Prinzips lässt sich in Gleichung 2.13 darstellen. Dabei bezeichnet d den Abstand zwischen Laserscanner und dem detektierten Objekt. Das Lichtgeschwindigkeit wird als c dargestellt. In 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. Es wird in der Tat gemessen und in Abstand d überführt. Der Sensor sendet periodisch Lichtimpulse aus und berechnet eine durchschnittliche Zielentfernung von der Zeit der zurückkehrenden Impulse [SK08].

$$d = \frac{c \cdot t}{2} \tag{2.13}$$

Im Bereich des selbstfahrenden Fahren wird der Lichtpuls mit nicht nur eine Ausrichtung ausgestrahlt, weil ein relativ großer Beobachtungsbereich mehrere in unter-

schiedlichen Richtungen ausgesendet Lichtstrahl benötigt. Aktuell gibt es zwei verschiedenen LiDAR-Systeme. Zum einen ist das feststehende Sensor, in dem mehrere Sende- / Empfangseinheiten mit unterschiedlicher Ausrichtungen angeordnet werden [Eff09]. Zum anderen ist das rotierende LiDAR-System dadurch realisiert, dass der Lichtimpuls über eine drehbare Spiegeleinheit abgelenkt [Eff09]. Das in dieser Arbeit verwendete Laserscanner Ibeo-LUX gehört zu dem zweiten Sensorsystem.

2.2.3 Theorie des inversen Sensormodells

Ein inverses Sensormodell hat die Aufgabe, den schon in 2.1.3 erwähnten mathematischen Ausdruck $p(m_i|z)$ zu finden, d.h. die Wahrscheinlichkeitsverteilung des Belegungszustands der Zelle mit Index i zu beschreiben. Nach [Pie13] lässt sich das inverse Sensormodell im Prinzip in drei Bestandteile zerlegen. Die sind, wie in Abbildung gezeigt, Hinderniskartierung, Freiraummodellierung und Beschreibung unbekannter Gebiete.

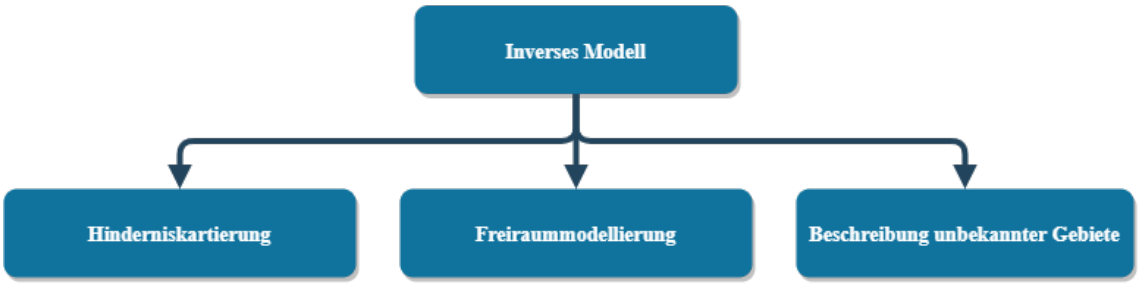

Abbildung 2.8: Bestandteile des Prinzips von Inverses Sensormodell

Unter Hinderniskartierung werden das Verfahren verstanden, dass die vom Sensor detektierten Objekte in das gitterbasierte Umfeldmodell bzw. das Occupancy Grid Map eingebracht werden [Pie13]. Zwei Probleme sind in diesem Bestandteil des inversen Sensormodell zu lösen. Zum einen gewinnt es die 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 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 zwei Probleme werden folgend bei

der tatsächlichen Sensormodellierung vertieftet eingegangen.
Bei Freiraum handelt sich um den beobachtbaren Bereich zwischen Sensor und dem detektieren Objekt. Analog zur Hinderniskartierung stehen die Bestimmung von der Form des Freiraums und der Wahrscheinlichkeit der betreffenden Zelle in Zentrum der Freiraummodellierung. Jedoch ist die Form des Freiraums abhängig von der oben bestimmten Hindernisform. Beispielweise wird der Freiraum als Linie oder Dreiecke modelliert, wenn die Hindernisse punktförmig oder linieförmig beschrieben werden. Bei Wahrscheinlichkeitsverteilung über den Freiraum spielt die Gründe der Unsicherheit eine große Spiele. Zum Beispiel wird die von Laserscanner erfasste Entfernungsinformation von dem Hindernis mit zunehmender Distanz weniger sicher. Auf diesem Grund ist dabei ein Modell zu entwickeln, die Unsicherheit in gewissem Grade widerspiegeln kann. In Anwendung sind ebenso die Korrektheit und die Zeitaufwand zu gewichten.
Der sich innerhalb der Reichweite von Laserscanner befindende Raum, der weder Hindernis noch Freiraum ist, ist der unbekannte Bereich des Modell. Ein typisches Beispiel dafür ist der Raum hinter dem Hindernis. Normalerweise wird der Wahrscheinlichkeit der unbekannten Zell die A-priori-Wahrscheinlichkeit zugewiesen. Der Grund liegt darin, dass keine neue Informationen über den Belegungszustand eingegeben werden. Typischerweise wird die A-priori-Wahrscheinlichkeit bzw. die Wahrscheinlichkeit der unbekannte Zellen als $0,5$ zugewiesen.

2.2.4 Technische Details von Ibeo-LUX

In die am IfF bereits bestehende Fahrzeugarchitektur ist Laserscanner Ibeo-LUX von Ibeo Automotive Systems GmbH für Erfassung der Umgebung um das selbstfahrende Fahrzeug verwendet. Das Golf7 ist mit vier IBEO-LUX-4L ausgestattet. Neben dem 4 IBEO-LUX-4L installierte Passat auch den IBEO-LUX-8L am vorderen und hinteren Ende des Fahrzeugs. Um ein angemessenes Sensormodell für das Laserscanner zu entwickeln, ist es voraussetzt, dass die technische Details von dem Sensor zur Verfügung steht. Nach [Ibe17] werden die relevante technische Größen 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	Roh- und Objektdaten
Abmaße (B×T×H)	164,5×93,2×88mm
Gewicht	998,7g

2.2.5 Das zu verwendendes Sensormodell

Nachdem die Theorie des inversen Sensormodell und technische Daten des verwendenden Laserscanner Ibeo-LUX vorgestellt wurden, wird in diesem Abschnitt ein angemessenes Sensormodell entwickelt, um die Wirkung der Sensorinformation auf Glauben des Zellenbelegungszustands zu beschreiben. Hierbei wird ein Kompromiss zwischen Genauigkeit und Effizienz gemacht. Die Modellierung ist mit zwei Schritte aufgrund der Messunsicherheit und der in Tabelle 2.3 aufgelisteten Laserscanner-Spezifikationen durchgeführt. Im ersten Schritt wird die Form des Hindernis mitsamt des entsprechenden Freiraums festgestellt. Darauffolgend wird es bestimmt, welche Wahrscheinlichkeit konkret in welchen Bereichen zugewiesen werden [Pie13]. Ein genaues Modell sollte die Wahrscheinlichkeit jeder Zelle in Abhängigkeit von der Position in 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 [Pie13]. Steht eine hinreichende Genauigkeit des Sensors zu Verfügung, kann die Form zu einer Linie vereinfacht werden. Wenn die horizontale Winkellösung zugleich niedrig ist, lässt sich die Form weiter zu einem Punkt vereinfachen.

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

zu beschränken, obwohl die Reichweite nach Tabelle 2.3 50m ist. Dies kann die wegen großer Entfernung entstehende Unsicherheit verringern. Nach dem Test wird der Wert von d_{max} als 10m bestimmt. Die horizontale Winkelauflösung $\Delta\theta$ beträgt nach Tabelle 2.3 $0,25°$ oder 0,004363(rad). Durch die Multiplikation mit d_{max} und $\Delta\theta$ ergibt sich der Kreisbogen den Wert $0,043$, der die wegen der Strahlbreite entstehende Divergenz beschreiben. Da $0,043 < 0,1m$ gilt, lässt sich der Einfluss von Strahlbreite vernachlässigen. Zudem wird die Genauigkeit mit 0,1m als ausreichend präzise angenommen. Damit ist es deutlich, dass Hindernisse im Rahmen dieser Arbeit aufgrund der Sensorspezifikation als Punkte zu beschreiben sind. 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 zweite Schritt der Modellierung eines Sensors simplifiziert und es ist nur erwartet, das inverses Sensor einziges Laserstrahls zu beschreiben. Bei Implementierung werden alle Laserstrahlen mit demselben Sensormodell behandelt. Das inverse Modell einziges Strahls ist 1-dimensional und das für idealen Laserscanner eingesetzte Modell ist in Abbildung 2.9 gezeigt. Das Diagramm veranschaulicht die Zusammenhang zwischen der Belegungswahrscheinlichkeit der Zelle mit Index i und dem Abstand von Laserscanner. Es ist angenommen, dass sich ein punktförmiges Objekt mit Abstand D befindet. Ohne Messabweichung bzw. Unsicherheit besitzt ein ideales Modell eine besonders einfache Funktion. Vor dem Objekt bzw. dem Hindernis ist es festgestellt, dass es unmöglich ein anderes Hindernis existiert. Die Zelle, die von Laserscanner mit Abstand D entfernt, ist mit Wahrscheinlichkeit 1 versehen. Hinter dem Hindernis sind alle Zelle verborgen, weshalb die Wahrscheinlichkeit unverändert bleibt und als A-priori-Wahrscheinlichkeit $0,50$ gegeben.
Jedoch unterliegt jede Messung aus verschiedenen Gründen einer gewissen Messunsicherheit [Heg18], weshalb ein ideales Sensormodell bietet eine bescheidene Genauigkeit. Unter Berücksichtigung der Messunsicherheit kann das inverses Sensormodell von ideal vereinfachend bis stark komplex sein [Pie13]. Allerdings werden die Funktion, die das inverses Sensormodell definiert, in der Praxis nach der bisherigen Erfahrungen bestimmt. Basierend auf [WSD07] [Pie13] [Heg18] wird im Rahmen dieser Arbeit je nach Situation eine bestimmte abschnittsweise definierte Funktionen angewendet. Die beide Funktion weisen darauf hin, dass die Zellen innerhalb der mi-

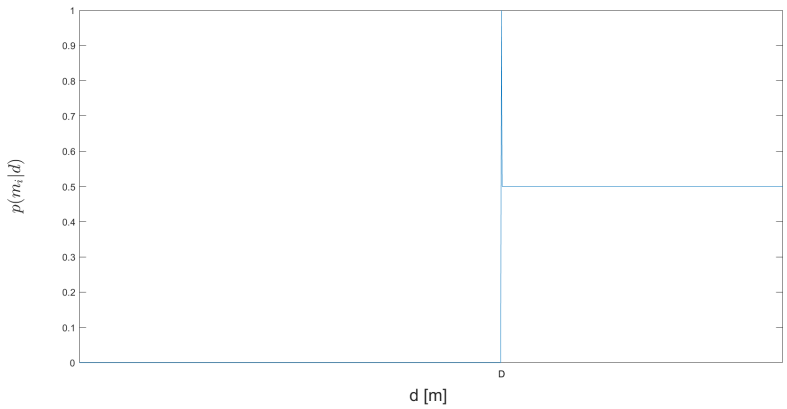

Abbildung 2.9: Das ideale Sensormodell

nimalen erfassbaren Abstand mit einer niedrigen und konstante Wahrscheinlichkeit hinzugefügt. Das inverses Sensormodell ist eine Erweiterung zu dem idealen Sensormodell, wenn ein Hindernis detektiert wird. Der Freiraum vor dem detektierten Hindernis, wie in Abbildung 2.10 dargestellt, ist mit einer monoton ansteigenden linearen Funktion zu beschreiben. Hierbei wird die mit größerer Entfernung entstehenden Messunsicherheit abgebildet, indem sich die Belegungswahrscheinlichkeit mit zunehmender Distanz von dem Laserscanner erhöht [Heg18]. Besteht kein Hindernis, wird die Funktion in in Abbildung 2.11 gezeigte Beschreibung umgewandelt. Dabei wird nur der Freiraum außer dem Bereich innerhalb der minimalen Abstand modelliert. Da die Funktionen linear und einfach sind, wird der 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.

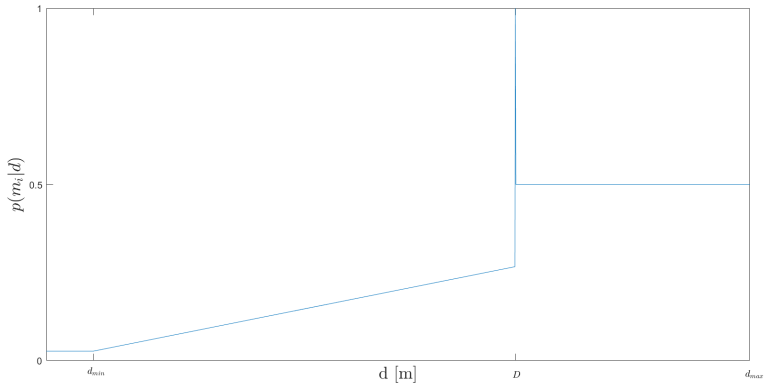

Abbildung 2.10: Das inverses Sensormodell, wenn ein Hindernis detektiert wird

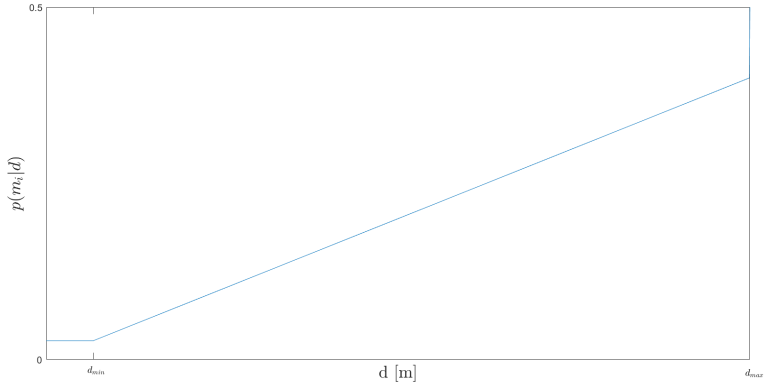

Abbildung 2.11: Das inverses Sensormodell, wenn kein Hindernis detektiert wird

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 parametriert.

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 parametriert werden. Obwohl die Zeichnungsbemaßung 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.

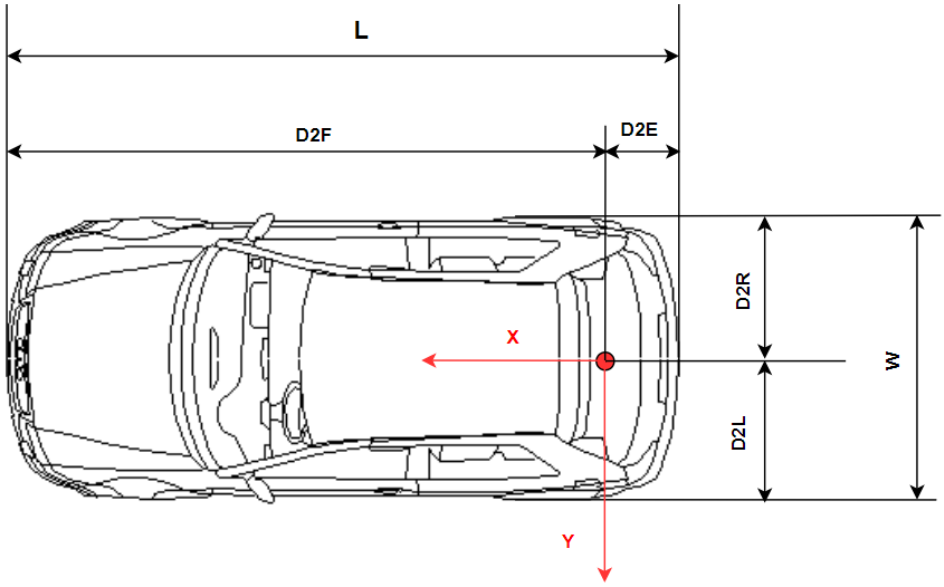

Abbildung 3.12: Dimension von Versuchsfahrzeug

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 θ 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° begrenzt. Dieser Wert wird in der Praxis entsprechend der Performance des Umfeldmodells angepasst.

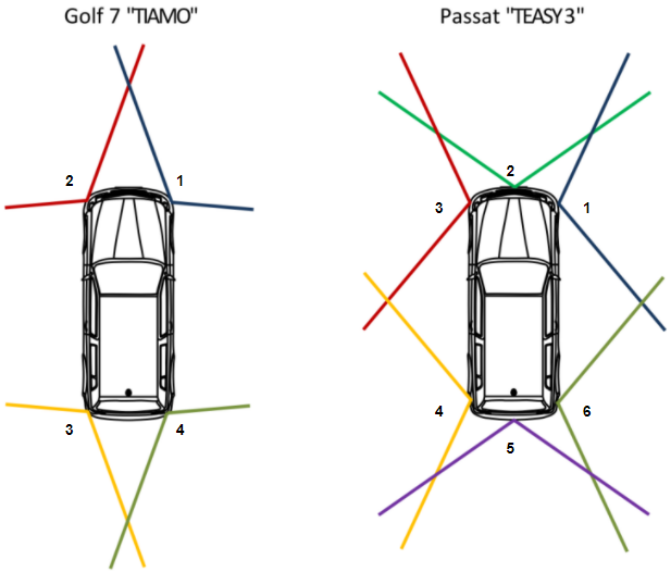

Abbildung 3.13: Einbauposition und Erfassungsbereich der Ibeo-Laserscanner

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

Sensor ID	**x (m)**	**y (m)**	**Winkel θ des Anfangsstrahls (°)**
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 θ des Anfangsstrahls (°)**
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-

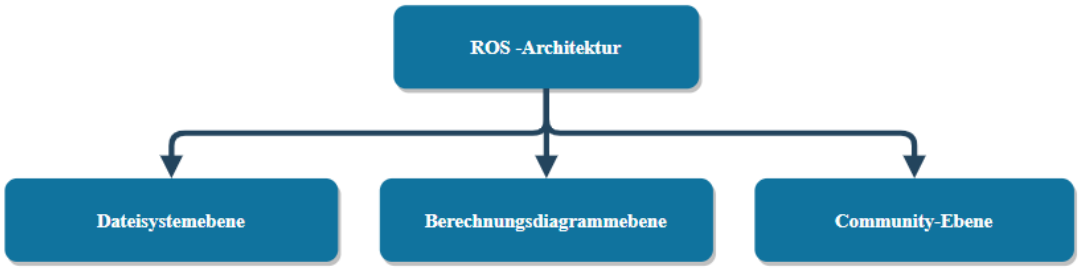

Abbildung 3.14: ROS-Architektur

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

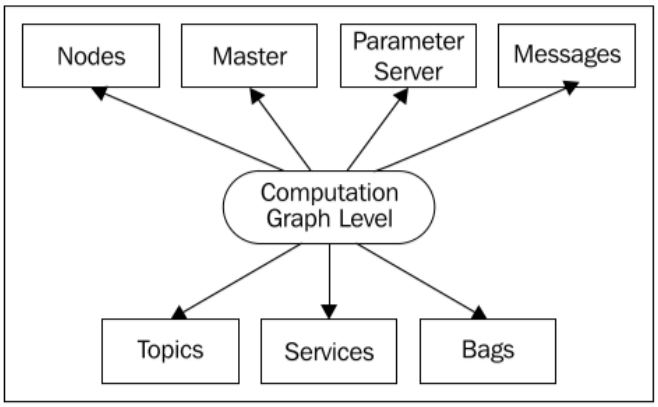

Abbildung 3.15: Wesentliche Grundkonzepte von Berechnungsdiagrammebene

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 Note 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.

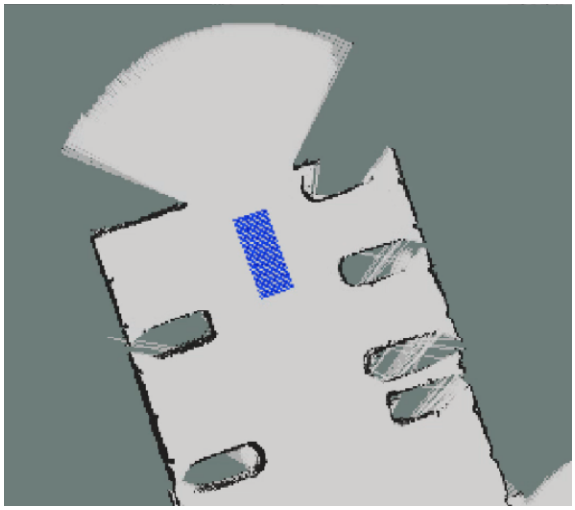

Abbildung 3.16: Display mit Map

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 verschwindt. 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.

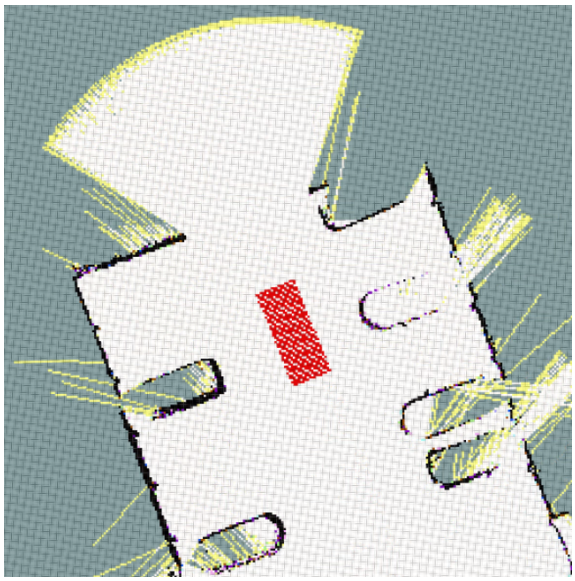

Abbildung 3.17: Display mit GridCell

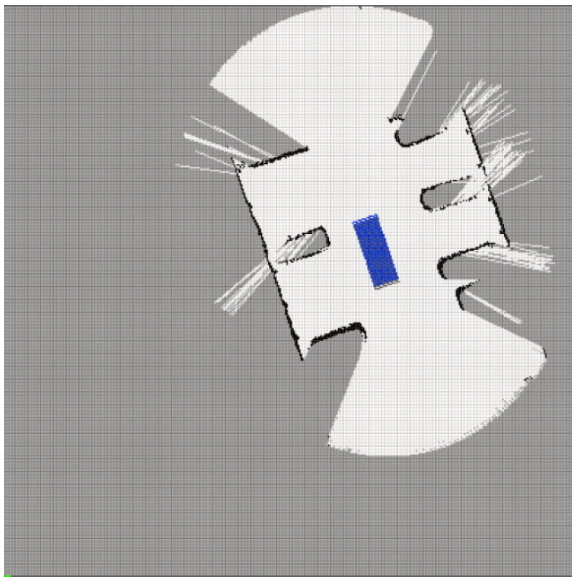

Abbildung 3.18: Display mit GridCell nach Binärisierung

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 ACS abgekürzt).

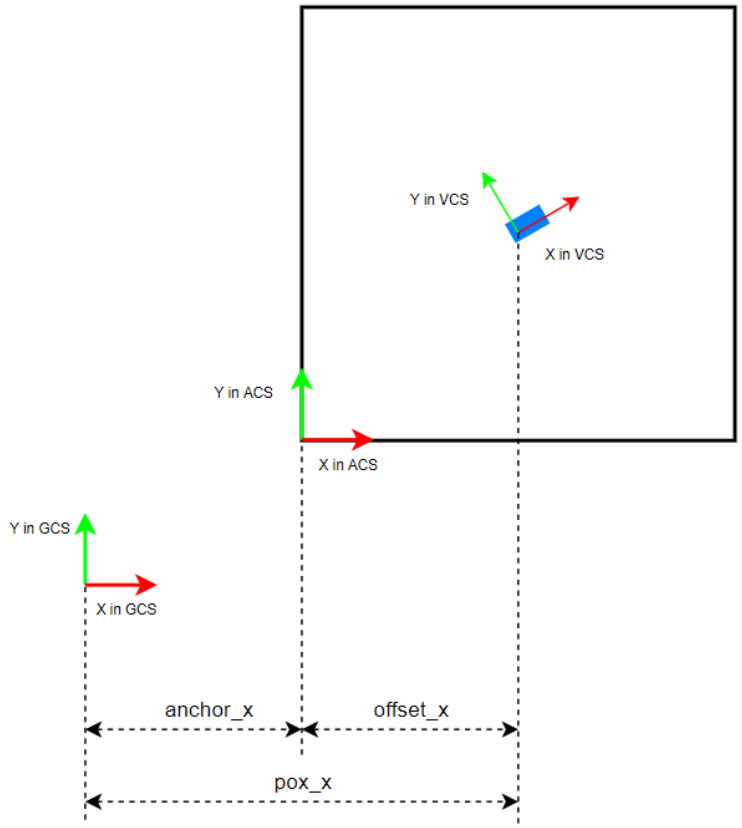

Abbildung 3.19: 3 wesentliche Koordinatensysteme im Umfeldmodell

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 naher 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 des 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 Kombinierung 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 Quadrats 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 verringern. Darüber hinaus wird in dieser Arbeit der Grenzwert des Bereichs parametrisiert und als Schnittstelle für das spätere Verwendung und Weiterentwicklung bereitgestellt.

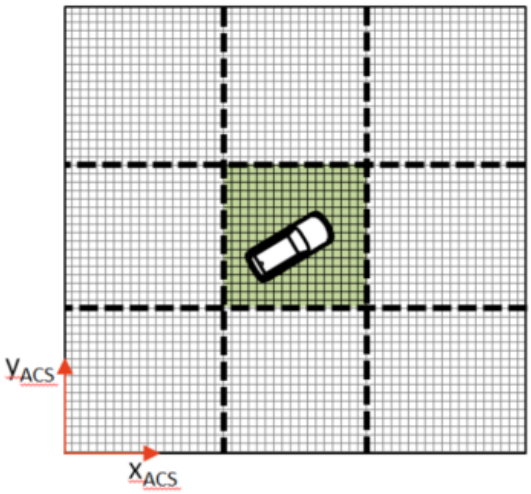

Abbildung 3.20: Anordnung der Position des Autos auf der Karte [Heg18]

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=pox_x-anchor_x erhalten werden. Dabei ist die Umrechnung auf der y-Achse ist 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 parametriert, 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 ist, 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 Anfangskoordinaten 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.

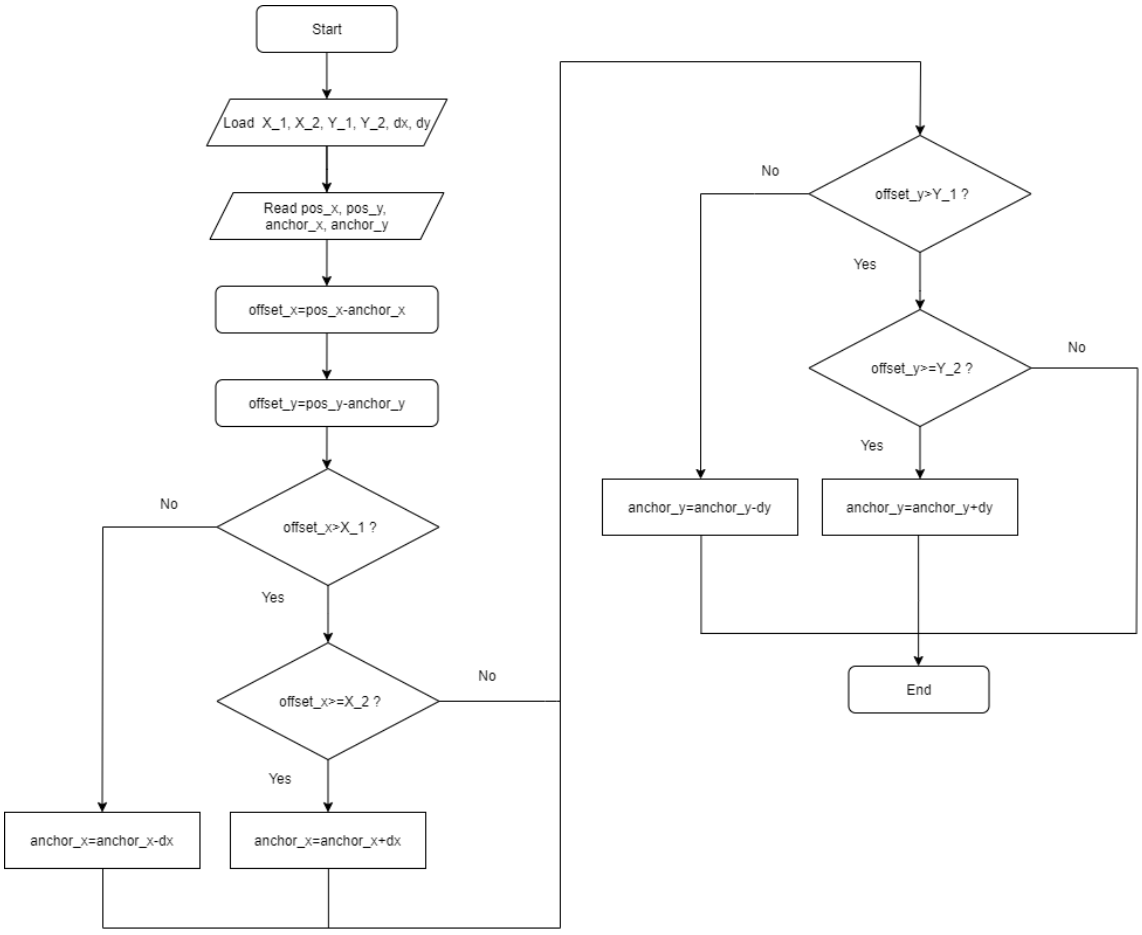

Abbildung 3.21: Programmablaufplan der Aktualisierung von ACS

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-

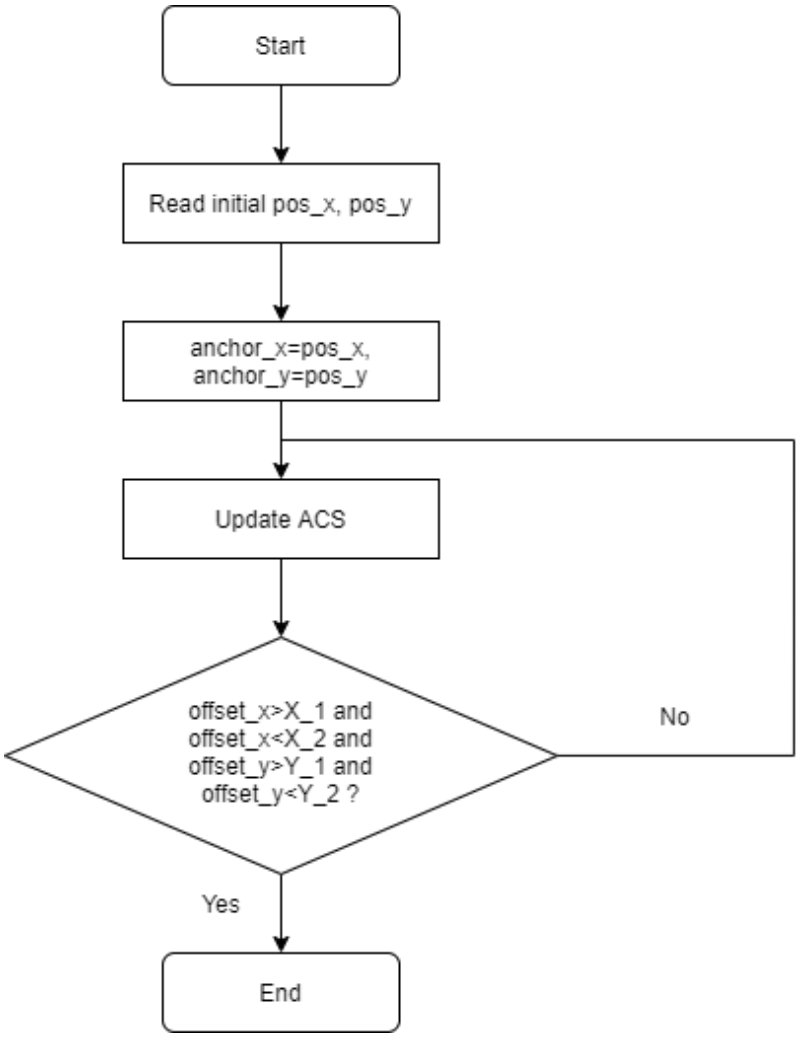

Abbildung 3.22: Initialisierung von ACS

bildung 3.23 visuell dargestellt werden.

$$pos\_psy = -car\_get\_psi + 90° \tag{3.1}$$

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

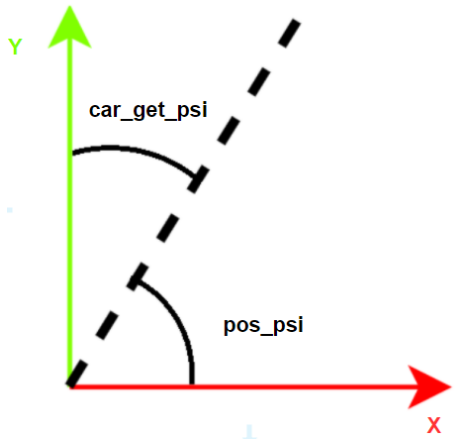

Abbildung 3.23: Umrechnung des Orientierungswinkels in GCS

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

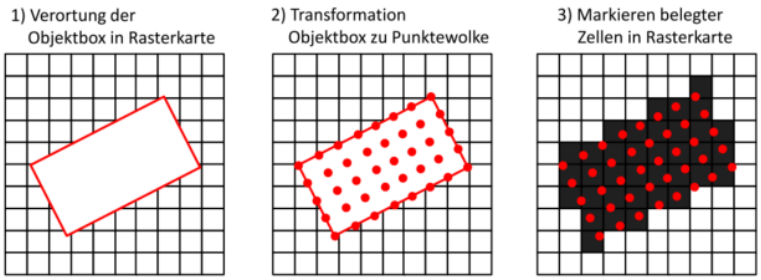

Abbildung 3.24: Kartierungsalgorithmus mit Objektinformation von Laserscanner [Heg18]

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 Messagetyp $sensor\_msgs/PointCloud2$ ist. Es ist anzumerken, dass diese Daten tatsächlich vom Datentyp $<pcl :: PointXYZL>$ der PCL-Standardbibliothek gekapselt und geliefert werden. Daher wird in der tatsächlichen Codeimplementierung Zeiger (engl. pointer) verwendet, um die 4 Beschreibungsinformationen der Punktwolke in $<pcl :: PointXYZL>$ 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 $LayesFilter$ 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-

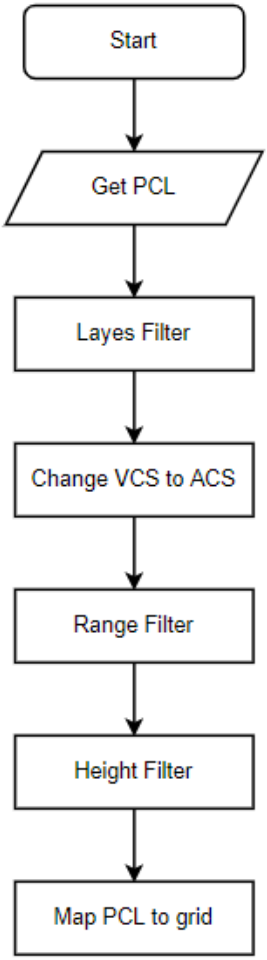

Abbildung 3.25: Programmablaufplan der Laserscannerdaten

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 ψ verwendet, da GCS und ACS immer in die gleiche Richtung zeigen.

$$ {}^{H}\mathbf{P} = {}^{H}\mathbf{R}_{V}\ {}^{V}\mathbf{P} \tag{3.2} $$

$$ {}^{V}\mathbf{P} = \begin{pmatrix} {}^{V}x \\ {}^{V}y \end{pmatrix} \tag{3.3} $$

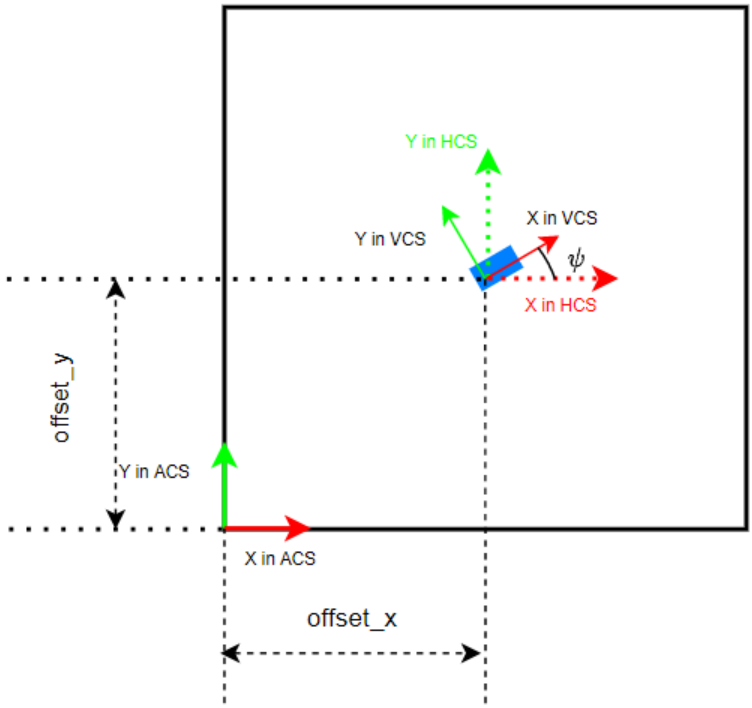

Abbildung 3.26: Darstellung eines Hilfskoordinatensystem

$$ {}^{H}\mathbf{P} = \begin{pmatrix} {}^{H}x \\ {}^{H}y \end{pmatrix} \tag{3.4} $$

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 D stellt die Abweichung von ACS und HCS dar, sodass kein bestimmtes Koordinatensystem angegeben werden muss.

$$ {}^{A}\mathbf{P} = {}^{H}\mathbf{P} + \mathbf{D} \tag{3.5} $$

mit

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

und

$$ {}^{A}\mathbf{P} = \begin{pmatrix} {}^{A}x \\ {}^{A}y \end{pmatrix} $$

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

$$^{A}x = cos(\psi) \times {}^{V}x - sin(\psi) \times {}^{V}y + offset\_x \tag{3.6}$$

$$^{A}y = sin(\psi) \times {}^{V}x + cos(\psi) \times {}^{V}y + offset\_y \tag{3.7}$$

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 Fitler* 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 um-

gewandelt, wobei dieses Gitter als belegt markiert wird.

$$Index\_x \approx {}^{A}x/GS \tag{3.8}$$

$$Index\_y \approx {}^{A}y/GS \tag{3.9}$$

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.

4 Implementierung

Um das ober

4.1 Unterkapitel 1

Beispiel für eine Fußnote[2].

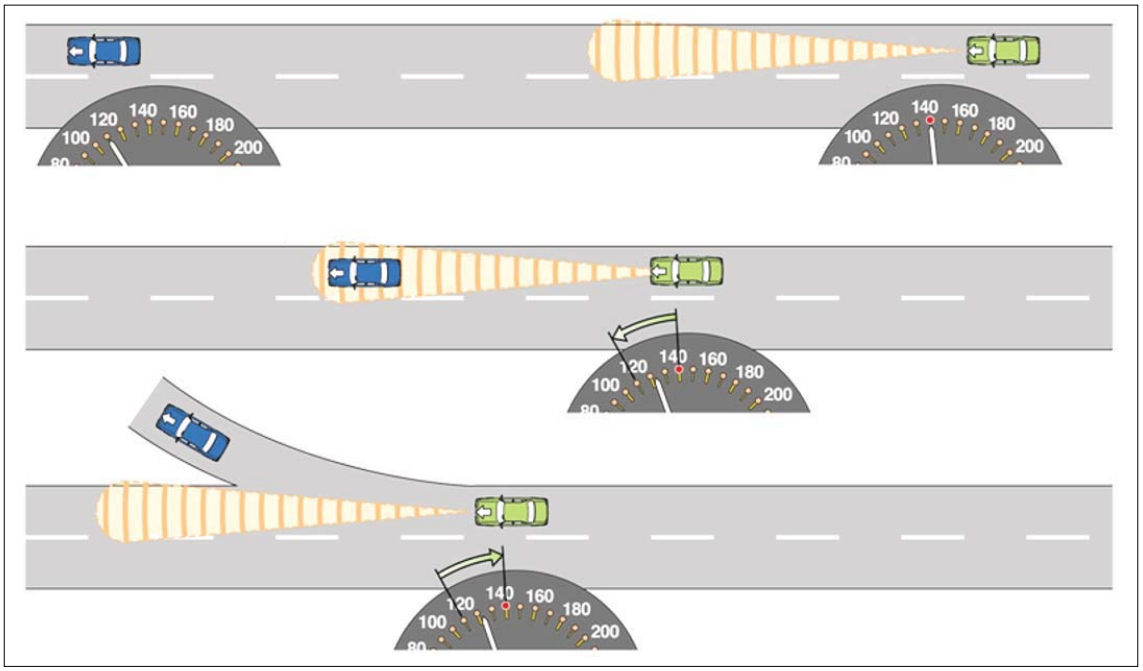

Abbildung 4.27: Folgefahrt Beispielbild

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 4.27 soll ein Beispielhaftes Bild zeigen.

4.1.1 Unterunterkapitel1

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

[2]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 Δt_{react}
- Im oberen Punkt sieht man auch schön die Einbindung von Formeln in den Text.

$v_{Obj.} = 120\,\mathrm{km/h}$
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	• eine hohe Auflösung und Farbskalen über das gesamte Sichtfeld haben • eine farbenfrohe Perspektive der Umgebung bieten • eine 3D-Geometrie von Objekten bei Stereokameras bereitstellen • kostengünstig Im Vergleich zu Lidar sein	• ein leistungsfähiges Berechnungssystem benötigen, um nützliche Daten zu extrahieren • empfindlich auf starken Regen, Nebel und Schneefall reagieren • eine 3D-Geometrie von Objekten bei Stereokameras bereitstellen • begrenzte Reichweite besitzen
Radarsensor	• lange Strecken bei schlechten Sichtverhältnissen vor dem Auto sehen • klein, leicht und erschwinglich sind • weniger Strom als ein Lidar-Sensor benötigen • im Vergleich zu Lidar robuster gegen Ausfälle sein	• eine geringe Genauigkeit und Auflösung bieten • begrenzte Informationen (z. B. weder genaue Form- noch Farbinformationen) bekommen • das Problem wegen der gegenseitigen Beeinflussung von Radarsensoren haben • schlechte Azimut- und Höhenauflösung verfügen • ohne einer Erhöhung der Leistung Radardämpfung zeigen
Ultraschall-sensor	• lange Strecken bei schlechten Sichtverhältnissen vor dem Auto sehen • klein, leicht und erschwinglich sind • weniger Strom als ein Lidar-Sensor benötigen • im Vergleich zu Lidar robuster gegen Ausfälle sein	• eine geringe Genauigkeit und Auflösung bieten • begrenzte Informationen (z. B. weder genaue Form- noch Farbinformationen) bekommen • das Problem wegen der gegenseitigen Beeinflussung von Radarsensoren haben • schlechte Azimut- und Höhenauflösung verfügen • ohne einer Erhöhung der Leistung Radardämpfung

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°bis − 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	Lichtschranke	Radar	Lidar	Kreiselplattform	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 (ROBIO)*. 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+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+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 Universistät 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+20] Mohammed, Abdul S. ; Amamou, Ali ; Ayevide, Follivi K. ; Kelouwani, Sousso ; Agbossou, Kodjo ; Zioui, Nadjet: 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:

$$
\begin{aligned}
v_{Start} &= 120\,\mathrm{km/h} &&= 33{,}3\,\mathrm{m/s} \\
v_{End} &= 80\,\mathrm{km/h} &&= 22{,}2\,\mathrm{m/s} \\
v_{Diff} &= 40\,\mathrm{km/h} &&= 11{,}1\,\mathrm{m/s}
\end{aligned}
$$