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

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. 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+19]. Zum einen ist die Eigenlokalisierung. 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. 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 präzis ist. Das am IfF (Institut für Fahrzeugtechnik) bereits bestehende Framwork 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 maßgebliche Umfeldmodelle bauen auf er-

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

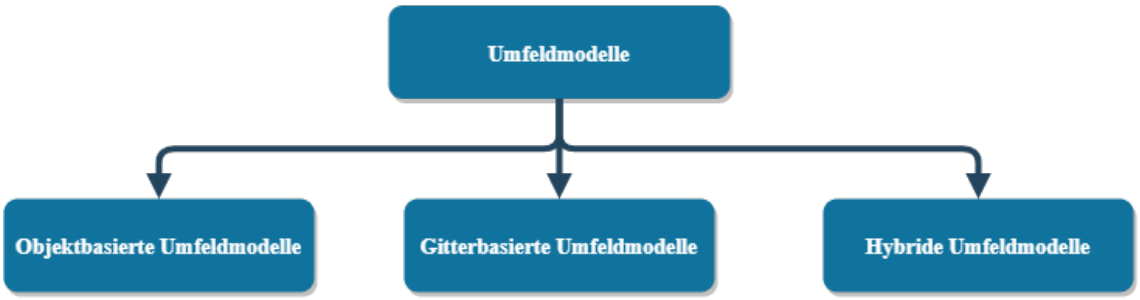

Abbildung 2.1: Mögliche Umfeldmodelle für autonome Fahrzeuge

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 im Praxis als bestimmte Objektklasse beschrieben. Jede Klasse wird mit hervorragenden und maßgeblichen Eingenschaften 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 präzis und einfach beschrieben werden können. Hierbei sind die Genauigkeit gegen der Zeit- und Speicheraufwand abzuwägen. Ein Beispiel ist das in Abblidung 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)

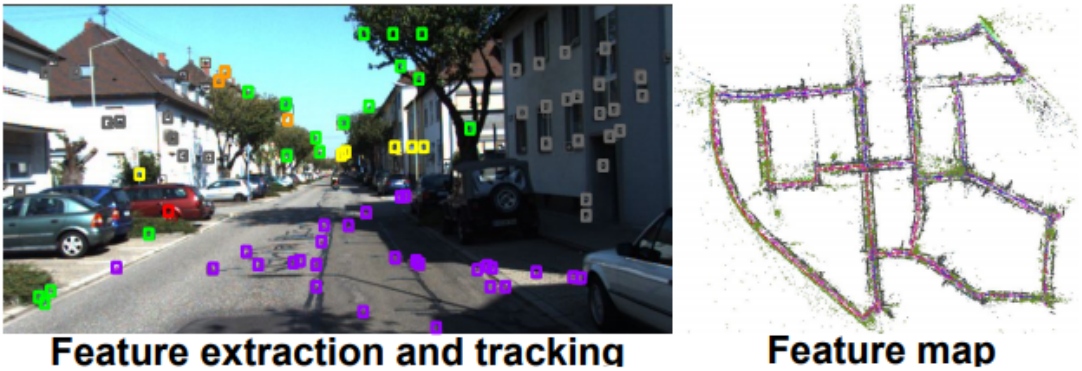

Abbildung 2.2: Ein objektbasiertes Umfeldmodell [FHR+20]

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 diejenige 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].

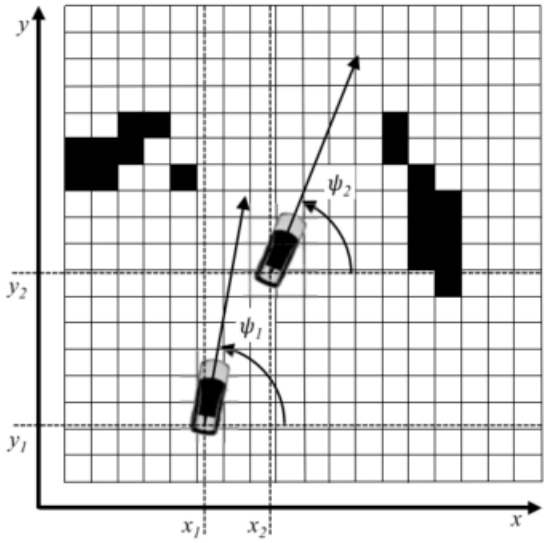

Abbildung 2.4: Raumdiskretisierung der Umgebung um Fahrzeug [WHLS15]

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 Anfangsbedienungen. 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 Ocuupancy Grid

Der Bayessche Filter ist zuerst zu beleuchten, mit dem die Funktionen und Bedeutungen der anderen Bestandteilen 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[1], 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

[1]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].

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

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.

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

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

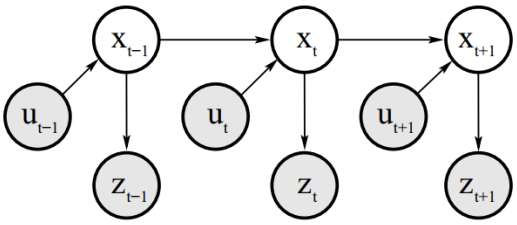

Abbildung 2.6: Hidden Markov Model (HMM) [TBF05]

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 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 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 berüchtigt 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

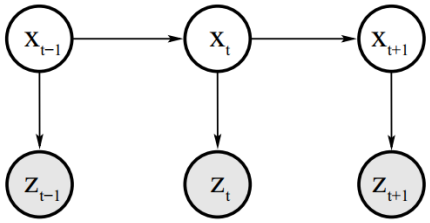

Abbildung 2.7: Das reduzierte HMM [TBF05]

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.

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

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.

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

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

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

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 die Gleichung 2.7 wird zu

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

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 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 50% 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

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

mit

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

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 aufgefordert. 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

Zweifellos dient Wahrnehmung bzw. Erfassung als eine wichtige Voraussetzung für Umfeldmodelleirung, 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 Eingenlokalisierung 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+20] sind die Vorteilen neben der 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-Senor 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+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

2.2.2 Funktionsweise des Lasersensors

Ist ein angemessenes Sensormodell zu finden, ist es sinnvoll das Mechanismus, die Eingenschaften 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].

$$d = \frac{c \cdot t}{2} \tag{2.12}$$

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.

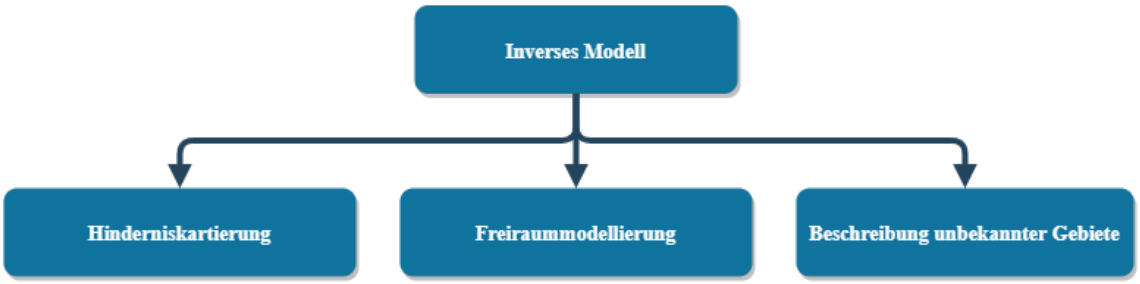

Abbildung 2.8: Bestandteile des Prinzips von Inverses Sensormodell

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. Beispielweise 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 Zell 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 voraussetzt, 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 $0,25°$ oder 0,004363(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 einziges 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-

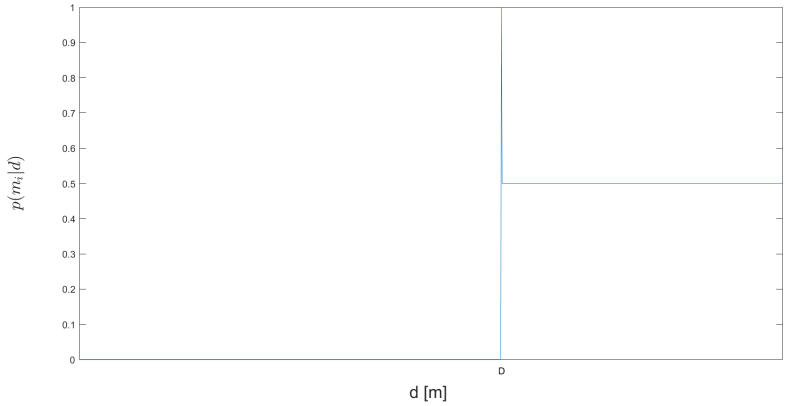

Abbildung 2.9: Das ideale Sensormodell

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

besitzt 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 100%. Hinter dem Hindernis sind alle Zelle verborgen, weshalb die Wahrscheinlichkeit unverändert bleibt und als A-priori-Wahrscheinlichkeit von 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 inverses 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 50% zugewiesen. Hierbei wird die mit größerer Entfernung entstehende

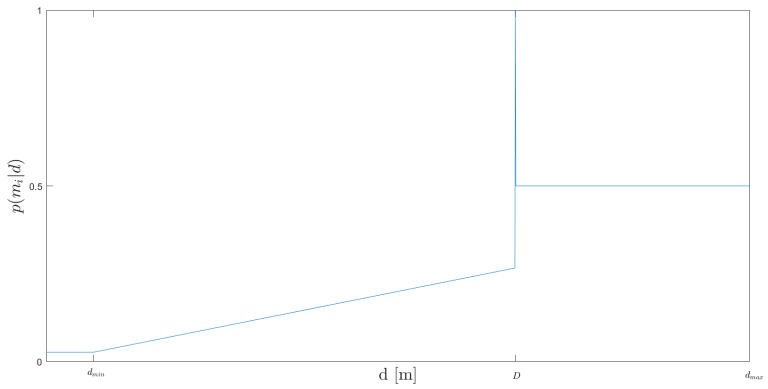

Abbildung 2.10: Das inverses Sensormodell, wenn ein Hindernis detektiert wird

Messunsicherheit 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

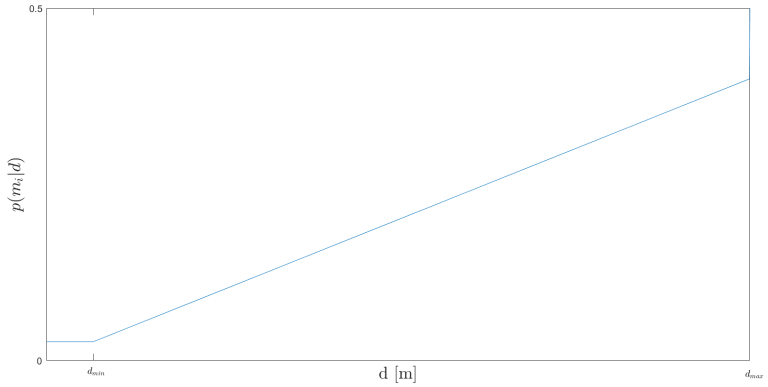

Abbildung 2.11: Das inverses Sensormodell, wenn kein Hindernis detektiert wird

Fall steigt die Wahrscheinlichkeitsfunktion im Bereich der maximalen Erfassungsentfernung und der minimalen Entfernung monoton an. Bei maximaler Entfernung steigt der Wahrscheinlichkeitswert auf 50%.
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

Basierend auf den theoretischen Grundlagen von Kapitel 3.4 liegt der Schwerpunkt dieses Kapitels auf der tatsächlichen Implementierung des Umfeldmodells im Ros-System. Darüber hinaus werden einige Features für die Erweiterung des System oder die Integration von anderen Funktionen hinzugefügt.

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 3.1 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.

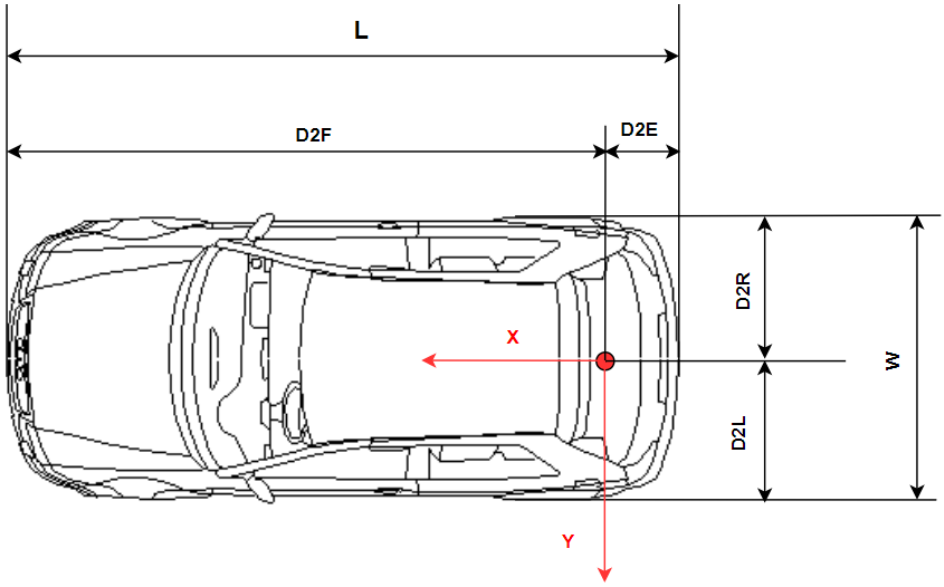

Abbildung 3.1: Dimension von Versuchsfahrzeug

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

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 3.1 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 Winkelbereich 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.

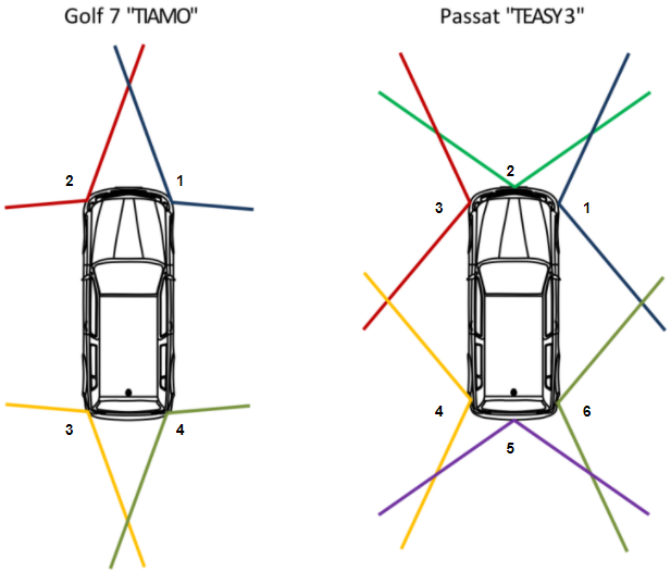

Abbildung 3.2: Einbauposition und Erfassungsbereich der Ibeo-Laserscanner

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

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

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 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 3.3 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

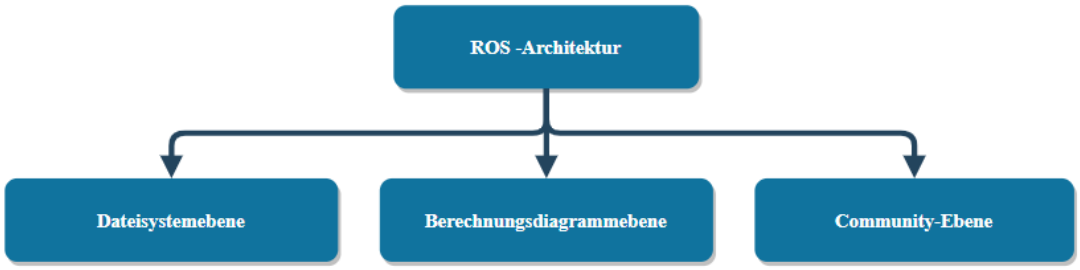

Abbildung 3.3: ROS-Architektur

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-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 3.4 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 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 Nachrich-

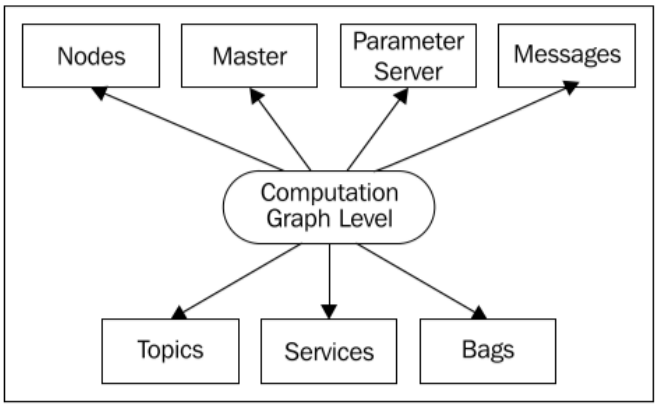

Abbildung 3.4: Wesentliche Grundkonzepte von Berechnungsdiagrammebene

tentypen 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

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

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 3.5 gezeigt, wenn Map value von einer Gitterzelle -1 ist, wird die Zellenfläche ausgegraut dargestellt. Wenn Map Value von 0 auf 100 steigt, verändert sich die entsprechende Zelle in einem Farbverlauf von Weiß zu Schwarz.

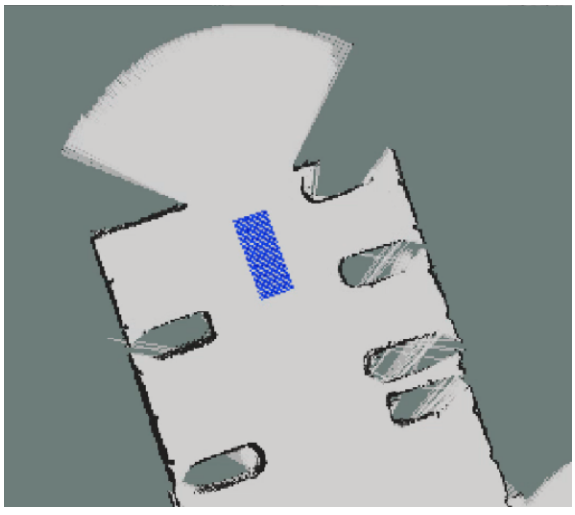

Abbildung 3.5: Display mit Map

Die zweite Möglichkeit der Visualisierung eines Umfeldmodells in RVIZ ist die Ver-

wendung 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 Graustufendarstellung 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 3.6 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 ge-

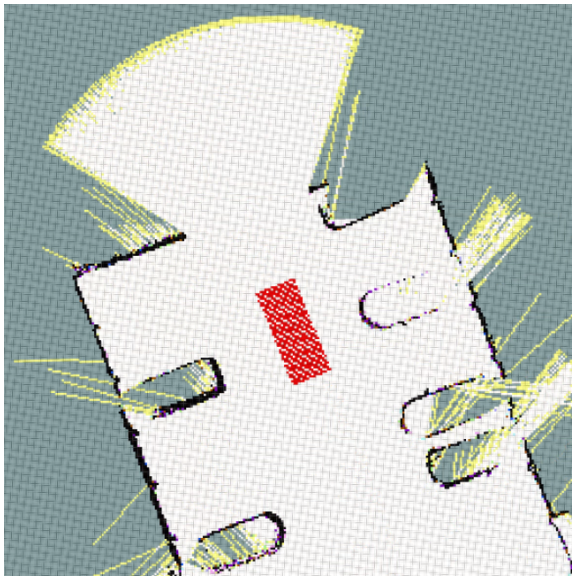

Abbildung 3.6: Display mit GridCell

genüber 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 3.7 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

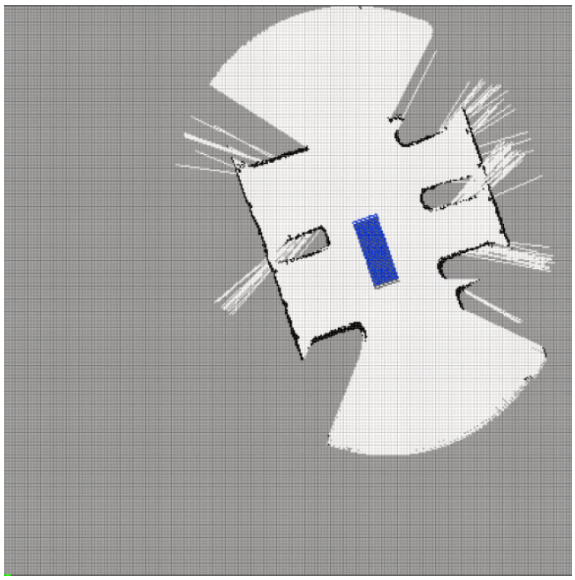

Abbildung 3.7: Display mit GridCell nach Binärisierung

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.

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 3.8 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).

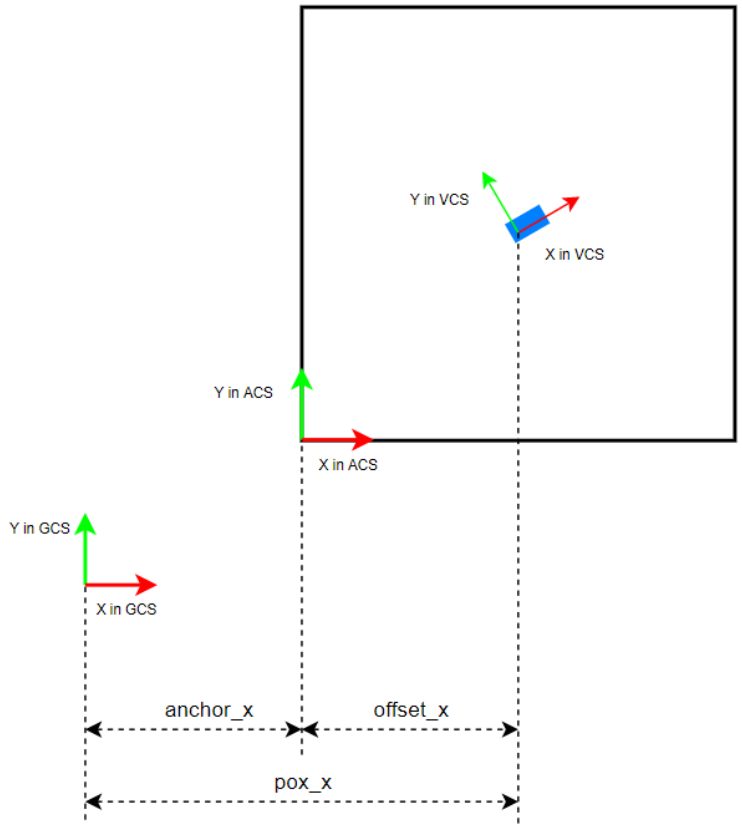

Abbildung 3.8: 3 wesentliche Koordinatensysteme im Umfeldmodell

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 3.8 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 3.8 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 3.9. Der Ursprung des ACS ist der Ausgangspunkt der in Abschnitt 2.1.3 erwähnten Diskretisierung und auch die 0-Stelle des Index. Abbildung 3.9 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.

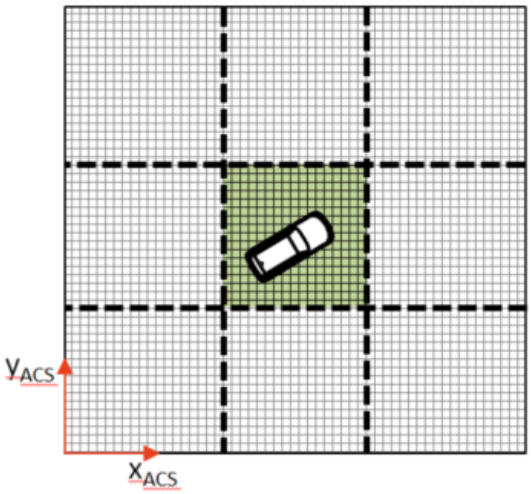

Abbildung 3.9: Anordnung der Position des Autos auf der Karte [Heg18]

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 3.8 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 3.10 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 3.9. 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 3.11 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.

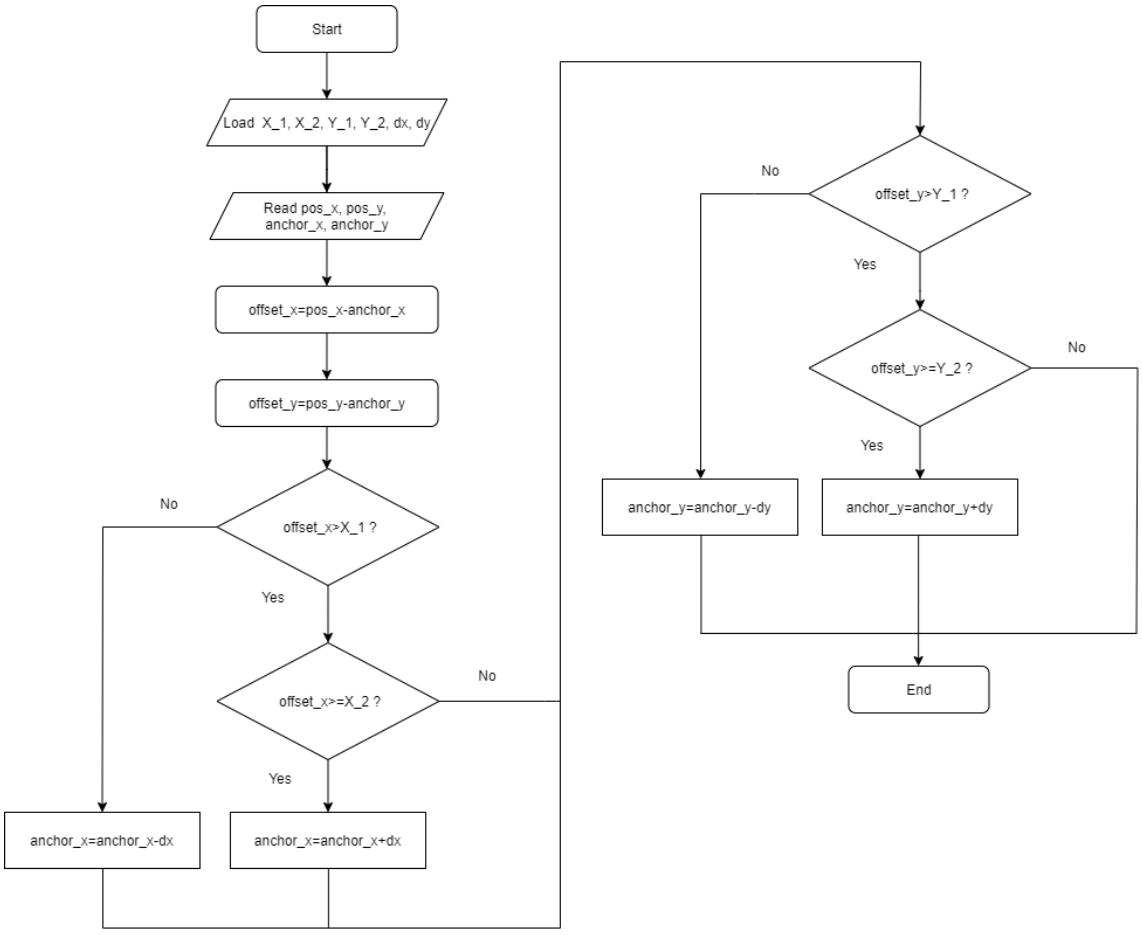

Abbildung 3.10: Programmablaufplan der Aktualisierung von ACS

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-

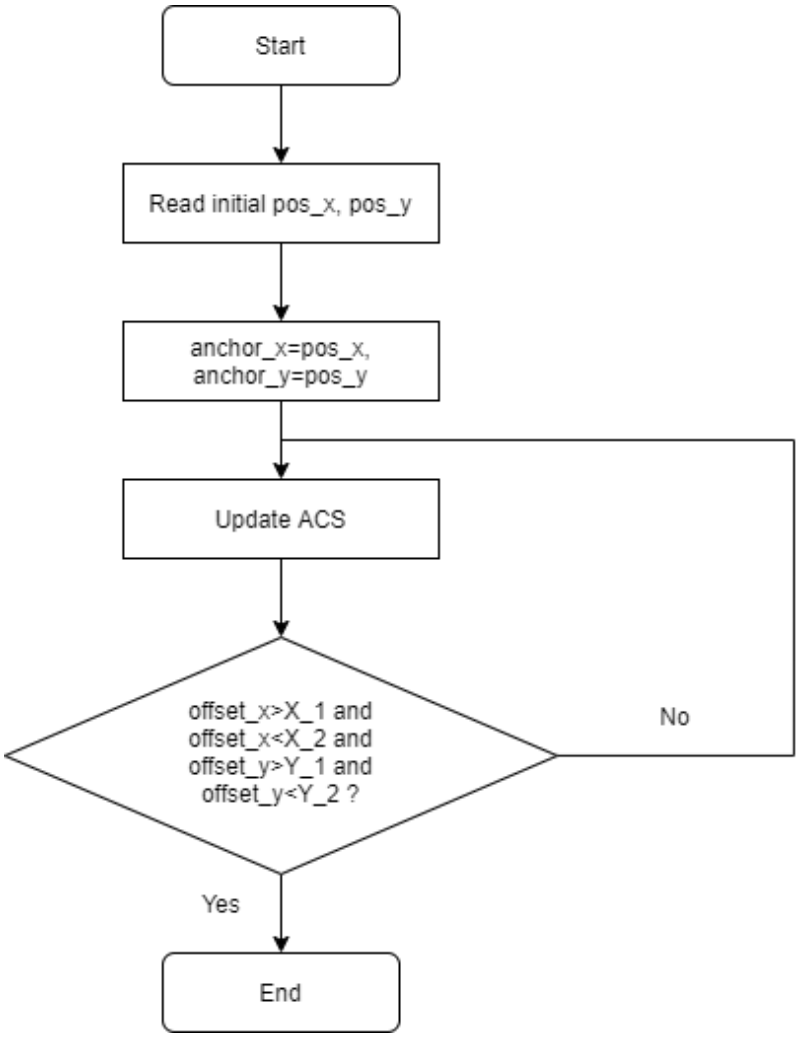

Abbildung 3.11: Initialisierung von ACS

bildung 3.12 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

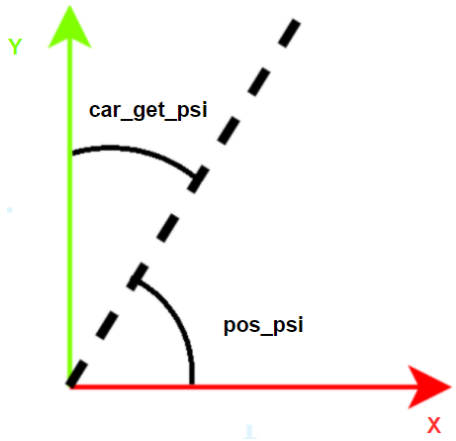

Abbildung 3.12: Umrechnung des Orientierungswinkels in GCS

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 3.13 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

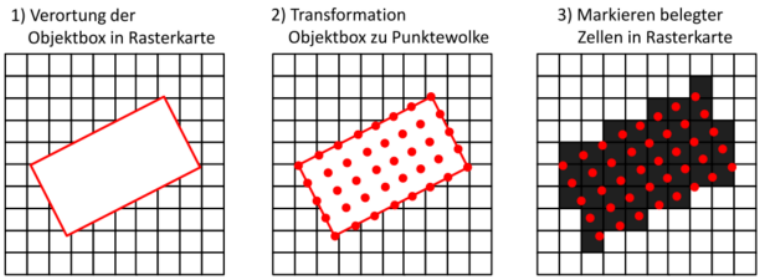

Abbildung 3.13: Kartierungsalgorithmus mit Objektinformation von Laserscanner [Heg18]

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 Punkte

als kontinuierliches Hindernis falsch eingeschätzt. Außerdem werden die Scanpunkte langer gerader Objekte hart durch einen Box dargestellt und daher in kleinere Boxen aufgeteilt oder gar nicht nicht ausgegeben[Heg18]. 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 3.14 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 3.2) 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),

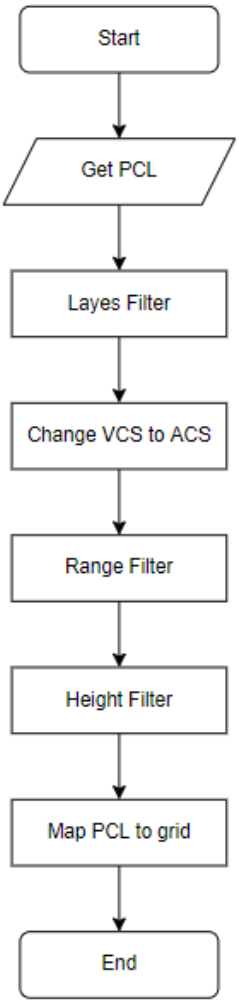

Abbildung 3.14: Programmablaufplan der Laserscannerdaten

wie in Abbildung 3.15 gezeigt, erstellt. Dann lässt sich die Koordinatenumwandlung in 2 Schritte zerlegen. Der erste Schritt besteht darin, das Referenzkoordinatensystem jedes Punktes von ACS in HCS umzuwandeln. Die mathematische Beschreibung dieses Schritts ist in Gleichung 3.2 gezeigt.

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

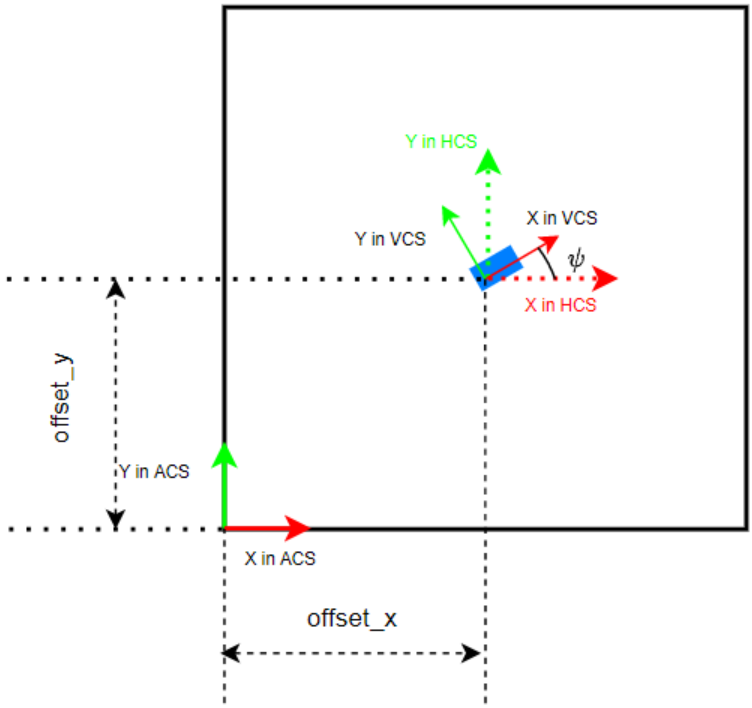

Abbildung 3.15: Darstellung eines Hilfskoordinatensystem

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.

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

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

${}^{H}\mathbf{R}_{V}$ ist als Drehmatrix oder Rotationsmatrix genannt und ihre konkrete Beschreibung befindet sich in Gleichung 3.5. Darunter wird der Wert von *pos_psi* direkt für ψ verwendet, da GCS und ACS immer in die gleiche Richtung zeigen.

$$ {}^{H}\mathbf{R}_{V} = \begin{pmatrix} cos(\psi) & -sin(\psi) \\ sin(\psi) & cos(\psi) \end{pmatrix} \tag{3.5} $$

Der zweite Schritt ist die Umwandlung von HCS in ACS, die durch einfache Vektoraddition erhalten wird. Die mathematische Formel befindet sich in 3.6. 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.

$$ {}^{A}\mathbf{P} = {}^{H}\mathbf{P} + \mathbf{D} \tag{3.6} $$

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 3.7 bzw. 3.8 berechnet werden.

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

$$ {}^{A}y = sin(\psi) \times {}^{V}x + cos(\psi) \times {}^{V}y + offset\_y \tag{3.8} $$

Funktionsblock *Range Filter* in Abbildung 3.14 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 dreidimen-

sionale 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 3.9 und 3.10 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.

$$Index\_x \approx {}^{A}x/GS \tag{3.9}$$

$$Index\_y \approx {}^{A}y/GS \tag{3.10}$$

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 sicherzustel-

len, 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 Sensordatenverarbeitung 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 Rahme dieser Arbeit wird das Umfeldmodell zur Anpassung an das IfF-Framework als 400×400-Gitter beschrieben, wobei jede Gitterzelle 0,1×0,1 Quadrat ist. Dies bedeutet, dass der Fahrzeugerkennungsbereich eine Fläche von 40 x 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 *hitory_grid* des nächsten Frames verwendet.

3.5.2 Probabilistischer Ansatz

Der erste Schritt in 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 3.16 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 3.17 gezeigt, wird beispielsweise der aktuell verarbeitete Lichtstrahl durch eine blaue durchgezogene Linie dargestellt. Der

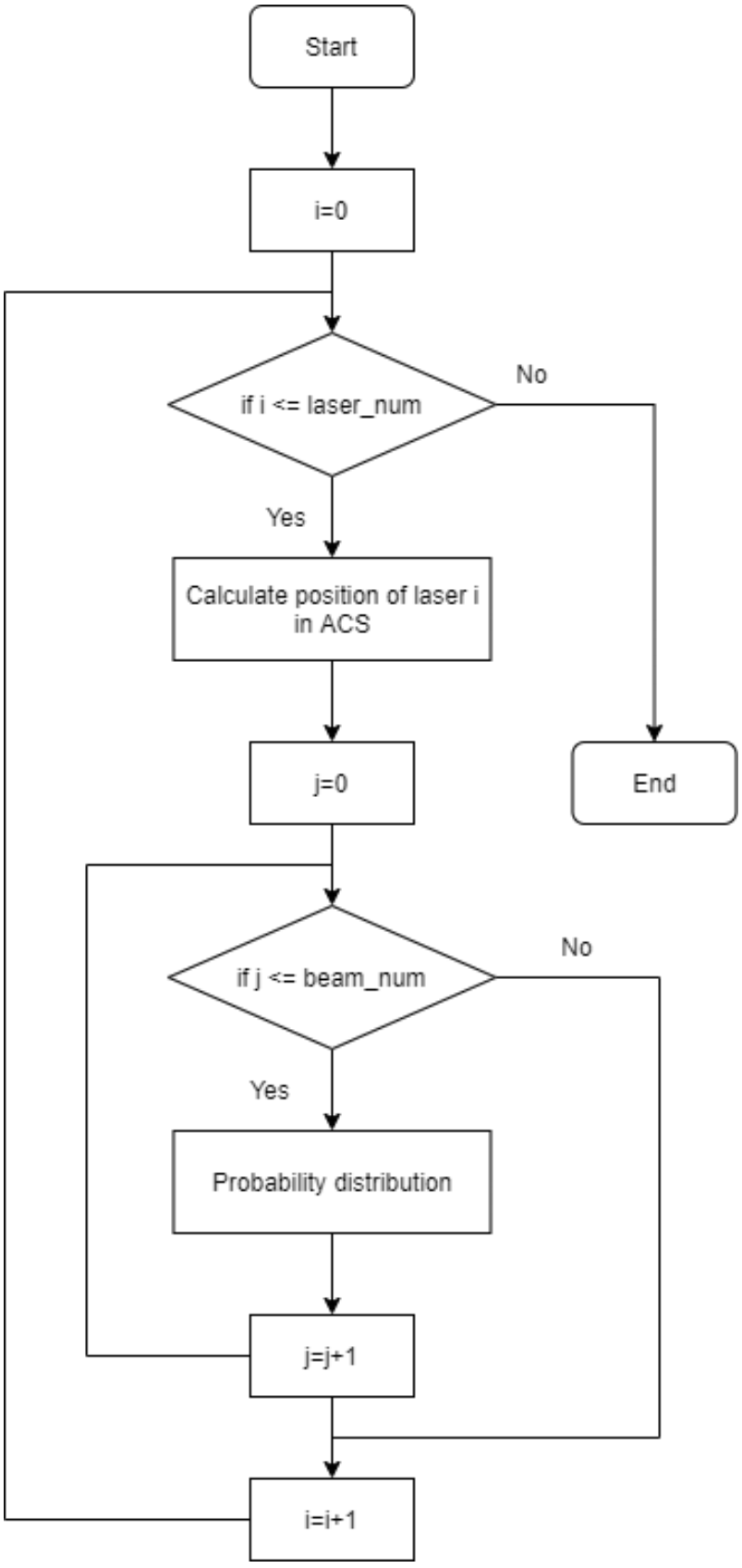

Abbildung 3.16: Implementierung des inversen Sensormodells

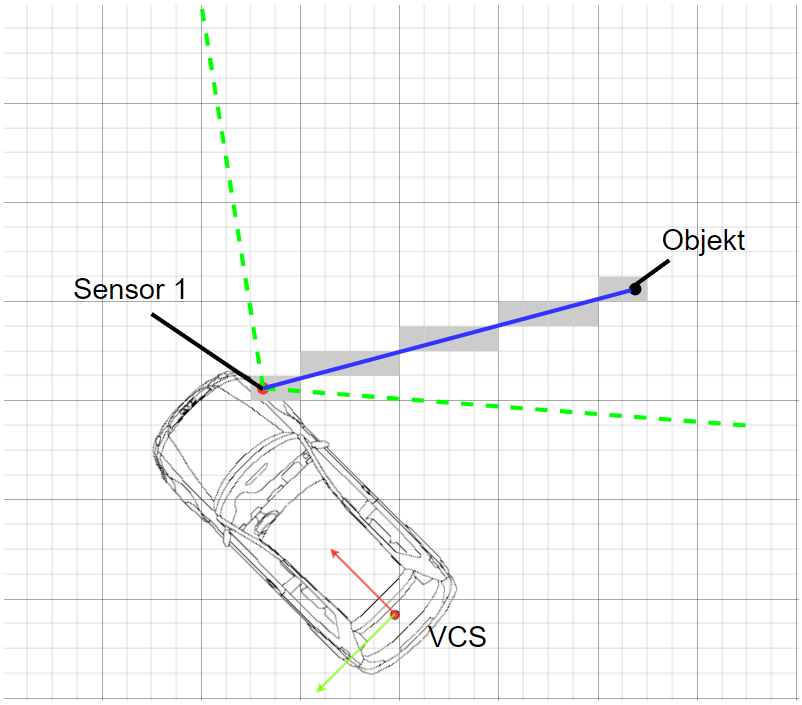

Abbildung 3.17: Raycasting mit Bresenham-Algorithmus

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_grird*, das den Zellen entspricht, die in dem in Abschnitt 3.4.2 genannten *pcl_grid* als belegt markiert sind, den Wert von p_fill. Wird beispielsweise die Position $(3, 4)$ von *pcl_grid* als belegt markiert, wird der Wert p_fill dem Element $(3, 4)$ des Arrays von *current_grid* zugewiesen. Den Zellen innerhalb des minimal erkennbaren Abstands wird *p_clear* 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_clear* und p_fill 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. Anderen Zellen in der Karte wird ein Wahrscheinlichkeitswert von 50% als Bereiche zugewiesen, die vom Sensor nicht erkannt werden können.
Nach Abschluss der Zuweisung aller Elemente im *current_grid* besteht der nächste Schritt in dieser Ebene darin, den binären Bayes-Filter zu verwenden, um die Belegungswahrscheinlichkeit jeder Zelle vom aktuellen Zeitstempel zu aktualisieren. Hierbei wird das Array *history_grid* verwendet, um die Belegungswahrscheinlichkeiten des letzten Zeitstempels zu speichern. Der Berechnungsprozess wird durch Gleichung 2.11 realisiert. Für jede Zelle repräsentiert der Term $p(m_i|z_t)$ den in *current_grid* gespeicherten Wert und der Term $p(m_i|z_{1:t-1})$ den in *history_grid* gespeicherten Wert. Der berechnete Term $bel_t(m_i)$ wird als der Wert in *history_grid* des nächsten Zeitstempels verwendet. Es ist erwähnenswert, dass jedem Element von *history_grid* im Ausgangszustand 50% zugewiesen werden, da keine Vorkenntnisse über die Umgebung vorliegen. Bevor der binäre Bayes-Filter angewendet wird, muss außerdem überprüft werden, ob die Karte verschoben wurde. Wenn der Ort der Karte bzw. der Ankerpunkt nicht mit dem Ort des vorherigen Zeitstempels übereinstimmt, muss das Array *history_grid* entsprechend der Bewegungsrichtung und dem Schritt der Bewegung aktualisiert werden. Wie in Abbildung 3.18 gezeigt, wird ein 16×16-Raster verwendet, um dieses Problem zu veranschaulichen. Der obere Teil der Abbildung zeigt die Karte mit dem Zeitstempel t und der untere Teil zeigt die verschobene Karte mit dem Zeitstempel t + 1. In der oberen linken Ecke, der unteren rechten Ecke und der Mitte dieser Karte befinden sich Hindernisse, die durch schwarze Quadrate gekennzeichnet sind. Gleichzeitig wird ein 16×16-Array erstellt und Wahrscheinlichkeitswerte zugewiesen. Es wird angenommen, dass zum Zeitpunkt $t + 1$ die Karte 4 Zellen nach rechts verschoben hat. Dann entspricht der Bereich, der durch Elemente dargestellt wird, deren x im neuen *history_array* gleich 0 bis 11 ist, dem Bereich, der durch Elemente von 4 bis 15 im alten *history_grid* dargestellt wird. Die grün markierten Bereiche werden im Modell nicht mehr berücksichtigt. Im rot ange-

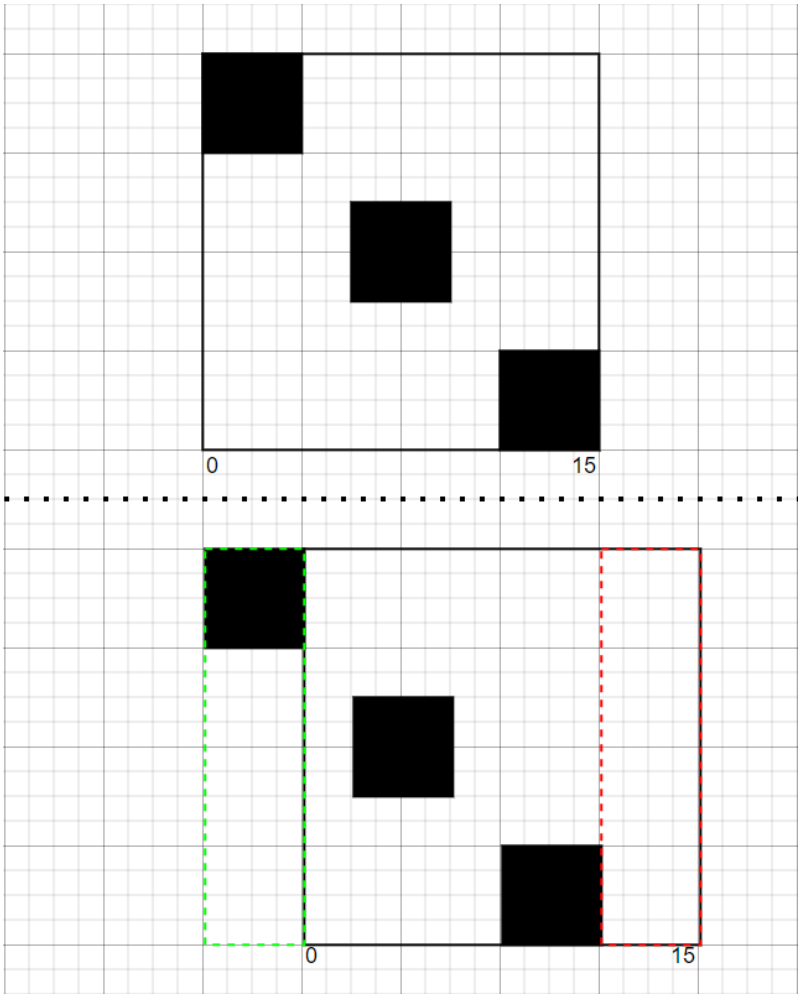

Abbildung 3.18: Aktualisierung vom Array *Hisotry_grid* aufgrund der Verschiebung der Karte

zeigten Bereich sind keine Sensordaten zu diesem Zeitpunkt vorhanden sind. Daher sollten die Elemente in diesem Bereich mit einem Wahrscheinlichkeitswert von 50% initialisiert werden. Der eigentliche Prozess muss die vier Bewegungsrichtungen und Bewegungsentfernungen berücksichtigen.

3.6 Zusätzliche Features

Nach der Erstellung des Umfeldmodells werden dem System einige zusätzliche Funktionen für die spätere Erweiterung und den Informationsaustausch mit anderen Sys-

temen hinzugefügt.

3.6.1 Ausgangsinformationsfluss

Am Ausgang des Systems sind, wie in Abbildung 3.19 dargestellt, zwei wichtige Ausgabedaten geplant. Die erste besteht darin, das Array auszugeben, das Occupancy

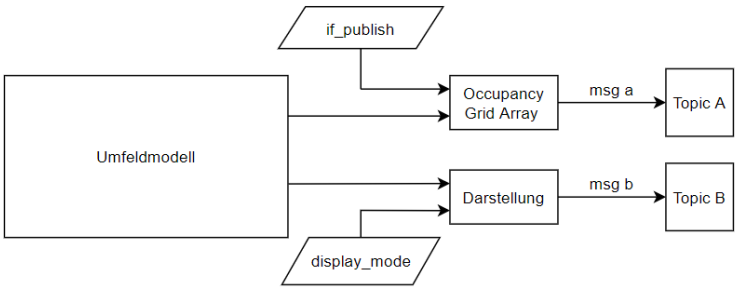

Abbildung 3.19: Ausgangsinformationsfluss

Grid im aktuellen Modell darstellt. Jedes Element des Arrays hat seine Belegungswahrscheinlichkeit. Aufgrund der Eigenschaften des ROS-message ist das Ausgabearray ein eindimensionales Array. Daher muss in diesem Schritt das im System vorhandene zweidimensionale Array in ein eindimensionales Array eingekapselt werden. In einem anderen ROS-node, der die Daten empfängt, können das wieder in ein zweidimensionales Array umgerechnet werden. Zu diesem Zweck muss die Größe jeder Dimension des Arrays angegeben werden. 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. Die Variable $if\_publish$ wird verwendet, um zu entscheiden, ob die Ros-message im entsprechenden Ros-topic geliefert werden soll. Die zweite ausgegebene Information wird verwendet, um sie in einer verwandten ROS-topic zur Visualisierung in ROS-Rviz zu liefern. Der Parameter *display_mode* legt fest, wie das umfeldmodell visualisiert wird. Wie in Abschnitt 3.2.2 erwähnt, umfassen die Visualisierungstypen *Gridcell* und *binarized Gridcells*.

3.6.2 Visualisierung des Fahrzeugs

Die Bestimmung der räumlichen Position des vom Fahrzeug abgedeckten Bereichs im Modell ist nicht Teil des Umgfeldmodells, aber hilfreich für die spätere kollisionsfreie Navigation. Gleichzeitig trägt die Visualisierung dieses Bereichs zur Vollständigkeit der Umfeldmodellvisualisierung bei. Um den vom Fahrzeug abgedeckten Bereich zu beschreiben, wird gemäß den Daten in Tabelle 3.1 und der Position des Ursprungs des Fahrzeugkoordinatensystems ein Rechteck erstellt, das den Fahrzeugbereich abdeckt. Dann wird das Rechteck in Punkte diskretisiert, um eine Punktwolke zu bilden. Wie die Position der Punktwolke von VCS in ACS umrechnet und dann kartiert wird, wurde in Abschnitt 3.4.2 erläutert. In ähnlicher Weise kann das Fahrzeug auch durch einige nahe gelegene Zellen dargestellt werden, die mittels einer einzigen ROS-topic in ROS-Rviz visualisiert werden können. Das tatsächliche Visualisierungsergebnis sind in Abbildung 3.5 als den blauen Bereich dargestellt.

3.6.3 Visualisierung von Bewegungspfaden

Für die Bedürfnisse der nachfolgenden Navigation wird im Rahmen dieser Arbeit auch die Visualisierung des Bewegungspfades realisiert. Die Aufgabe besteht darin, eine Funktion zu kapseln, deren Eingabeparameter einen Vektor der UTM-Koordinatenpositionen des Fahrzeugs ist. Entsprechend den Parametern dieses Vektors wird der durch diese Positionen bestimmte Pfad visualisiert. Unter dem ROS-System wird $nav :: path$ für die Implementierung verwendet. Die Pose des Fahrzeugs wird hier als Parameter eingeführt. Da der Winkel von $nav :: path$ durch die Quaternion bestimmt wird, muss hierbei zusätzlich ein Algorithmus zur Konvertierung vom Gierwinkel in die Quaternion verwendet werden. Da keine Planungsdaten in dieser Arbeit für die Navigation vorhanden sind, werden die vom Fahrzeug zurückgelegten Pose-Informationen aufgezeichnet und als Parameter an die Funktion zur Visualisierung des Pfades übergeben. Das tatsächliche Ergebnis ist in Abbildung 3.20 dargestellt. Diese Funktion kann als Werkzeug verwendet werden, das während der tatsächlichen Pfadplanung aufgerufen wird.

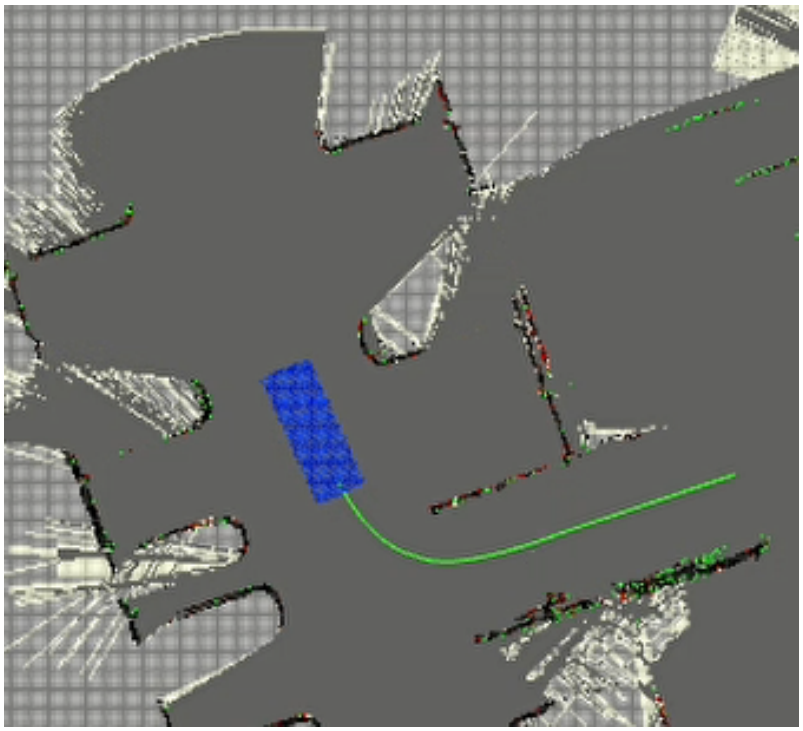

Abbildung 3.20: Visualisierung von Bewegungspfaden

4 Evaluation

Die Implementierung des Modells ist die Kombination aus Theorie und Ros-basierter Programmierung. Die tatsächlichen Ergebnisse dieses Modells in ROS werden in diesem Kapitel evaluiert. Die Evaluierung des Modells orientiert sich hier hauptsächlich an zwei Kriterien. Das erste ist die Qualität der gitterbasierten Karte. Die zweite ist die Laufgeschwindigkeit des Modells.

4.1 Qualität der gitterbasierten Karte

Im Allgemeinen kann das Modell eine statische Umgebung relativ gut beschreiben. Dieser Teil wurde übrigens erklärt, während die Implementierung in Kapitel 3 erläutert wurde. Hier werden hauptsächlich einige kleine Details erläutert, die auf der Karte angezeigt werden. Hierbei wird hauptsächlich auf einige kleine Details beschränkt, die auf der gitterbasierten Karte angezeigt werden.

4.1.1 Freiraum hinter Hindernisse

Theoretisch sollte sich hinter dem Hindernis ein unbekannter Bereich befinden. Das Ergebnis der Visualisierung ist jedoch, wie in Abbildung 4.1 gezeigt, dass der Bereich links vom Hindernis im Bild als Freiraum bezeichnet wird. Der Grund liegt darin, dass die Informationen eines einzelnen Punktes der Punktwolke instabil sind. Wenn zu Beginn einige Punktwolken in der Mitte des durchgehenden Hindernisses nicht erkannt werden, wird der Freiraum gemäß dem Algorithmus durch den Spalt entlang der Strahlrichtung eingerichtet. Nachdem die Informationen vollständig sind, wird das Hindernis vollständig erkannt und das Modell aktualisiert nur den Abstand zwischen dem Fahrzeug und dem Hindernis. Dies bildet einen isolierten Freiraum, der nicht mehr aktualisiert wird. Diese Situation spiegelt sich im unteren Teil des Bildes wider. Es ist jedoch geplant, dass das Fahrzeug während der Navigation in einem befahrbaren Bereich fährt. Da die Hindernisse schnell verbessert werden, hat dies keine Auswirkungen auf die tatsächlichen Navigationsanforderungen. Das Modell wird also als funktionsfähig bewertet.

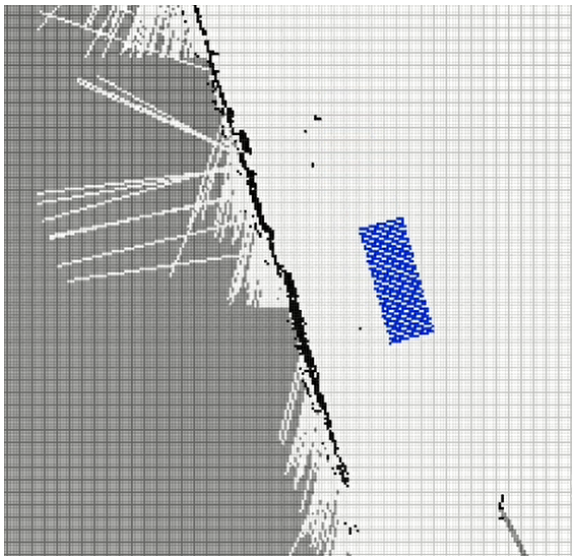

Abbildung 4.1: Freiraum hinter Hindernisse

4.1.2 Situation beim Datenverlust

Im Verlauf des Experiments wurde festgestellt, dass an bestimmten Stellen Punktwolkeninformationen verloren gingen. In diesem Fall berücksichtigt das Modell, dass keine Hindernisse erkannt werden können. Daher wird ein Freiraum um das Fahrzeug herum gebaut, wie in Abbildung 4.2 gezeigt. Nach der Überprüfung in ROS wird festgestellt, dass in diesem Datenrahmen mit Ausnahme von Informationen wie Zeit und Rahmen die gesamten Punktwolkeninformationen vollständig verloren gehen. Daher wird dem ursprünglichen Modell ein Filter hinzugefügt, und die Daten werden nur verwendet, wenn gültige Punktwolkeninformationen vorhanden sind. Nach experimentellen Ergebnissen wurde festgestellt, dass diese Situation angesprochen wurde. Wie in Abbildung 4.3 gezeigt, tritt dieses Phänomen nicht mehr an derselben Position auf. Der Filter wird derzeit als einsatzfähig eingestuft. In der zukünftigen Forschung müssen jedoch die Situation des Datenverlusts eingehend untersucht werden. Beispielweise ist die Beurteilung der Gültigkeit von Daten zu beachtens , wenn ein Teil der Daten verloren geht.

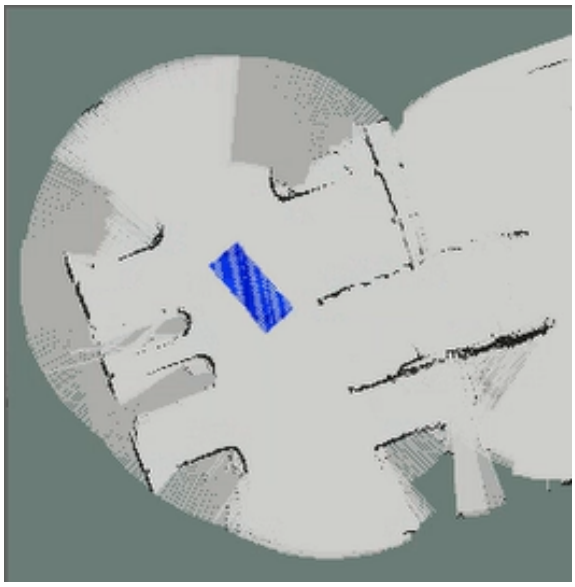

Abbildung 4.2: Modellierung beim Datenverlust

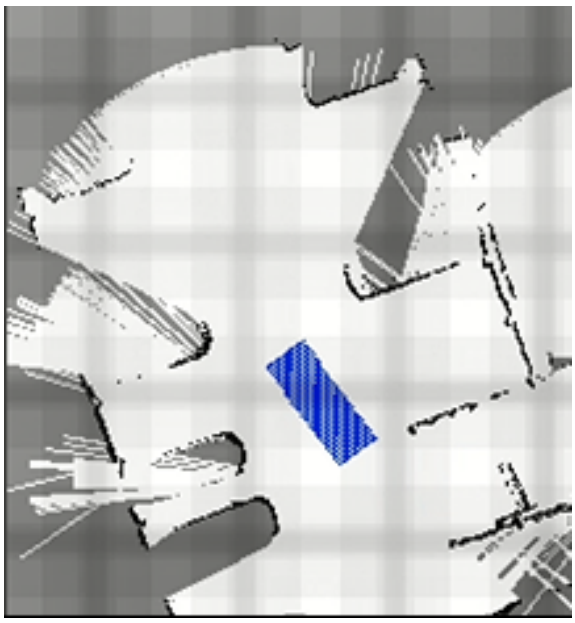

Abbildung 4.3: Modellierung mit Filter beim Datenverlust

4.1.3 Situation über die Rampe

Eine unangenehme Situation trat auf, als das Fahrzeug bergab fuhr. In diesem Fall erfassen die Laserscanner aufgrund der Gesamtneigung des Fahrzeuges den Raum näher am Boden. Zu diesem Zeitpunkt wurden aufgrund der zweidimensionalen Eigenschaften des Modells, wie in Abbildung 4.4 gezeigt, Hindernisse vor dem Fahrzeug erkannt, obwohl dies tatsächlich ein befahrbarer Bereich ist. Dieses Phänomen setzt

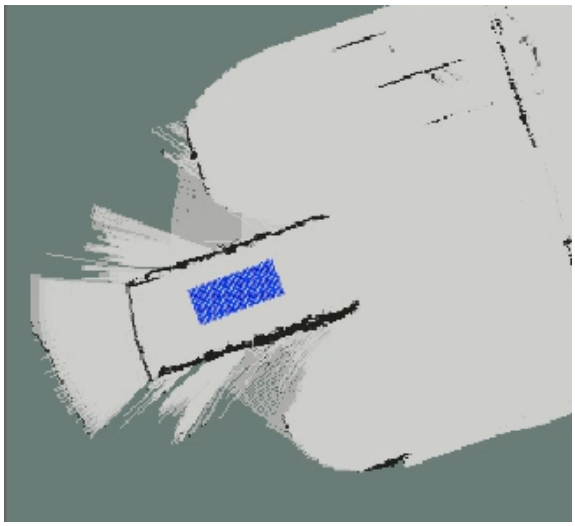

Abbildung 4.4: Situation über die Rampe

sich fort, bis das Fahrzeug beginnt, wieder horizontal zu werden. Durch Experimente kann eine einfache Anpassung der in Abschnitt 3.5.2 genannten Sensormodellparameter wie p_fill und p_clear das Problem nicht beseitigen, ohne die Qualität des Modells in anderen Situationen zu beeinträchtigen. Daher führt das Modell den Nickwinkel des von CAN eingegebenen Fahrzeugs ein, um die Haltung des Fahrzeugs zu überprüfen. Wenn festgestellt wird, dass sich das Fahrzeug auf der Rampe befindet, wird der Laserscanner mit ID 2 an der Vorderseite des Autos ausgeschaltet, um das Problem zu beheben. Wenn der Absolutwert von Nickwinkel des Fahrzeugs unter einem bestimmten Wert liegt, wird der Laserscanner wieder eingeschaltet. Natürlich muss die Einführung der Daten mit anderen Informationen, wie in Abschnitt 3.4.3 erwähnt, synchronisiert werden. Das Ergebnis des Modells mit hinzugefügten Nickwinkel-Informationen ist in Abbildung 4.5 dargestellt. Es gibt keine Fehleinschätzung von Hindernissen an derselben Position und das Modell wird als ef-

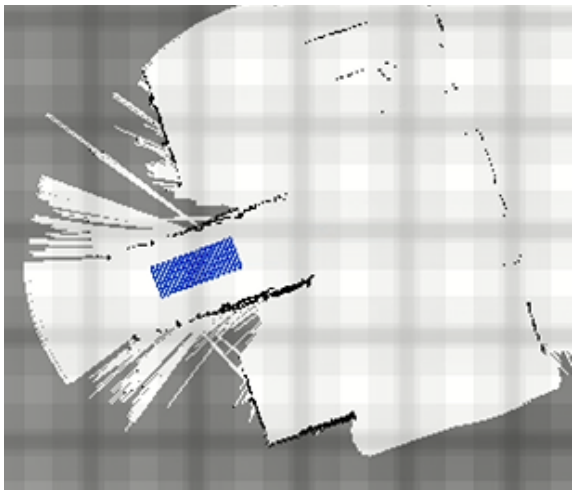

Abbildung 4.5: Situation über die Rampe, wenn Nickelwinkel zusätzlich berücksichtigt wird

fektiv bewertet. Aufgrund der Einführung neuer Sensordaten hängt die tatsächliche Leistung und Genauigkeit des Modells jedoch von der Genauigkeit der Daten von Nickelwinkel ab.

4.2 Laufgeschwindigkeit des implementierten Modells

Da ROS über Tools verfügt, mit denen die Frequenz von Topic überprüft werden kann, wird die Berechnungszeit oder die Berechnungsgeschwindigkeit des Modells hier indirekt durch die Frequenz beschrieben. Wie in Abbildungen 4.6 4.7 4.8 gezeigt, gehört die niedrigste Frequenz zu den Informationen aus der Punktwolke, die etwa 25 Hz beträgt. Die begrenzt den Maximalwert der Laufgeschwindigkeit des Modells. Um die Laufgeschwindigkeit zu messen, wird die Frequenz der Modellausgangsdaten gemessen. Wie in Abbildung A gezeigt, beträgt die Ausgangsfrequenz etwa 12,5 Hz. Der Grund ist die in Abschnitt 3.4.2 genannte Filterung der Daten. Die Daten, die die Layers 4 bis 7 enthalten, werden nicht verwendet, sodass die tatsächlich verwendeten Daten um 12 Hz übertragen werden. Diese Ausgangsfrequenz bedeutet, dass das System Daten in Intervallen von 0,083 Sekunden ausgibt. In der Parkplatzszene beträgt die maximale Abweichung bei einer Geschwindigkeit von $10km/h$ $0,23m$, was ungefähr der doppelten Auflösung von dem Gitter entspricht. Diese Geschwindigkeit gilt als zufriedenstellend und funktionsfähig.

```
tao@tao:~$ rostopic hz /as_tx/point_cloud
subscribed to [/as_tx/point_cloud]
average rate: 24.948
        min: 0.036s max: 0.044s std dev: 0.00214s window: 25
average rate: 24.969
        min: 0.030s max: 0.049s std dev: 0.00337s window: 50
average rate: 25.014
        min: 0.030s max: 0.049s std dev: 0.00336s window: 75
average rate: 25.243
        min: 0.018s max: 0.049s std dev: 0.00439s window: 101
average rate: 25.177
        min: 0.018s max: 0.049s std dev: 0.00404s window: 126
average rate: 25.324
        min: 0.011s max: 0.055s std dev: 0.00494s window: 152
average rate: 25.283
        min: 0.011s max: 0.055s std dev: 0.00471s window: 177
average rate: 25.238
        min: 0.011s max: 0.055s std dev: 0.00451s window: 202
average rate: 25.232
        min: 0.011s max: 0.055s std dev: 0.00429s window: 228
```

Abbildung 4.6: Frequenz von Information über Punktwolke

```
tao@tao:~$ rostopic hz /VehiclePoseFusionUTM
subscribed to [/VehiclePoseFusionUTM]
average rate: 59.977
        min: 0.015s max: 0.018s std dev: 0.00076s window: 58
average rate: 59.992
        min: 0.015s max: 0.018s std dev: 0.00080s window: 119
average rate: 59.994
        min: 0.015s max: 0.018s std dev: 0.00082s window: 179
average rate: 60.010
        min: 0.015s max: 0.018s std dev: 0.00084s window: 239
average rate: 60.008
        min: 0.015s max: 0.018s std dev: 0.00085s window: 299
average rate: 59.994
        min: 0.013s max: 0.020s std dev: 0.00092s window: 359
average rate: 59.996
        min: 0.013s max: 0.021s std dev: 0.00097s window: 419
^Caverage rate: 59.995
        min: 0.013s max: 0.021s std dev: 0.00096s window: 469
```

Abbildung 4.7: Frequenz von Information über UTM-Koordinaten

```
tao@tao:~$ rostopic hz /can_2_ros_gps
subscribed to [/can_2_ros_gps]
average rate: 100.115
        min: 0.008s max: 0.012s std dev: 0.00076s window: 96
average rate: 100.054
        min: 0.008s max: 0.012s std dev: 0.00075s window: 196
average rate: 100.039
        min: 0.008s max: 0.012s std dev: 0.00077s window: 297
average rate: 100.028
        min: 0.008s max: 0.012s std dev: 0.00078s window: 397
average rate: 100.022
        min: 0.008s max: 0.012s std dev: 0.00078s window: 497
average rate: 99.998
        min: 0.008s max: 0.012s std dev: 0.00077s window: 597
average rate: 100.016
        min: 0.008s max: 0.012s std dev: 0.00077s window: 697
average rate: 100.017
        min: 0.008s max: 0.012s std dev: 0.00076s window: 797
```

Abbildung 4.8: Frequenz von Information über Nickelwinkel

```
average rate: 11.654
        min: 0.060s max: 0.151s std dev: 0.01980s window: 39
average rate: 11.877
        min: 0.060s max: 0.151s std dev: 0.01769s window: 52
average rate: 11.972
        min: 0.060s max: 0.151s std dev: 0.01622s window: 64
average rate: 12.074
        min: 0.060s max: 0.151s std dev: 0.01520s window: 77
average rate: 12.104
        min: 0.060s max: 0.151s std dev: 0.01459s window: 89
average rate: 12.155
        min: 0.060s max: 0.151s std dev: 0.01411s window: 101
average rate: 12.203
        min: 0.060s max: 0.151s std dev: 0.01352s window: 114
average rate: 12.240
        min: 0.060s max: 0.151s std dev: 0.01301s window: 127
average rate: 12.235
        min: 0.060s max: 0.151s std dev: 0.01279s window: 139
```

Abbildung 4.9: Frequenz der Modellausgangsdaten

5 Diskussion

Nach der Implementierung und Evaluation des Umfeldmodells werden in diesem Kapitel basierend auf den Einschränkungen, die bei der Entwicklung dieses Artikels auftreten, die Möglichkeit einer zukünftige Modellverbesserung und -erweiterung durchgeführt.
Unter dem aktuellen Rahmen werden die Punktwolkeninformationen aller im Fahrzeug installierten Laserscanner einfach zusammengeführt und dem Modell in Form von ROS-Topic als Ganzes zur Verfügung gestellt. Unter der Annahme, dass zwei oder mehrere Sensoren Stelle A erfassen können, wird Stelle A, selbst wenn sie nur von einem der Sensoren als belegt erkannt wird, als belegt betrachtet. Eine Möglichkeit, dieses Problem zu lösen, besteht darin, die Daten verschiedener Sensoren separat zu verarbeiten und jedem Sensordaten ein Gewicht hinzuzufügen, um die Zuverlässigkeit der Daten auszudrücken. Die spezifische Implementierungsmethode kann sich auf [APML12]beziehen. Eine andere Möglichkeit ist die Verwendung der Dempster-Shafer-Theorie zur Beschreibung der Unsicherheit. Occupancy Grid verarbeitet tatsächlich Unsicherheit, die nicht unbedingt direkt durch die Belegungswahrscheinlichkeit behandelt werden. Die mit Hilfe dieser Theorie in [Pie13] eingerichteten Evidenztheoretische Occupancy Grids sind ein Weg, um das oben genannte Problem zu lösen. Darüber hinaus kann diese Methode zwischen Unsicherheit und Unwissenheit unterscheiden. Wenn beispielsweise die Belegungswahrscheinlichkeit 50% beträgt, kann es sich um ein unbekanntes Gebiet handeln, oder es kann festgestellt werden, dass nach dem Einschätzung mittels Messdaten eine Belegungswahrscheinlichkeit von 50% besteht. Evidenztheoretische Occupancy Grids beschreibt explizit die Wahrscheinlichkeit des Zellzustands: belegt, frei und unbekannt.
In diesem Modell wird die Fixierung der Rasterauflösung als traditionelle Methode übernommen. Dieser Ansatz bietet tatsächlich viel Raum für Verbesserungen. Beispielsweise ist in bestimmten Szenen oder schmalen Räumen, die Präzision erfordern, eine sehr hohe Auflösung erforderlich. In anderen riesigen Räumen mit sehr wenigen Hindernissen bringt eine sehr hohe Auflösung jedoch unnötige Rechenzeit mit sich. Daher ist es wirtschaftlich und vernünftig, eine adaptive Auflösung des Raster effektiv an die Szene anzupassen. Eine der Implementierungsmethoden ist der Nd-Tree von [EC10]. Wenn die neuen Daten mit dem aktuellen Zustand in Kon-

flikt stehen, wird die betroffene Zelle mit hoher Auflösung unterteilt. Zellen mit ähnlichen Belegungswerten in angrenzenden Bereichen werden zusammengeführt, um den Platzbedarf und den Rechenaufwand zu verringern.
Darüber hinaus liefert der verwendete Ibeo-Lux-Sensor tatsächlich dreidimensionale Daten von Punktwolkeninformationen. Um diese Daten optimal zu nutzen, kann eine dreidimensionale Karte verwendet werden, um eine umfassendere Beschreibung der Umgebung des Fahrzeugs zu erhalten. Die aktuelle 3D-Darstellung enthält Voxel und OctoMap.
Ähnliche wie bei dem in Abschnitt 4.1.1 erwähnten Probleme sind darauf zurückzuführen, dass sich ein hochdimensionaler Raum in eine Reihe eindimensionaler Gitterzelle zerlegen lässt. Und machen Sie den Zustand zwischen der Zelle und der Zelle unabhängig voneinander, alle beeinflussen. Dies hat gewisse Einschränkungen bei der Darstellung der Fakten. Daher ist die Suche nach einer neuen statistischen Methode zur Suche nach einem praktischeren Sensormodell der Weg, um dieses Problem zu lösen. Zudem wird es angenommen, dass die Zustände zwischen der Zelle und der Zelle unabhängig voneinander sind und sich nicht gegenseitig beeinflussen. Daher ermöglicht die Untersuchung nach einer neuen statistischen Methode zur Erstellung eines praktischeren Sensormodells dieses Problem zu lösen. Eine Möglichkeit besteht, wie in [Thr03] beschrieben, mit Hilfe von Erwartungs-Maximierungs-Algorithmen ein Vorwärtssensormodell (engl. forward sensor model) zu erstellen.
Außerdem ist das Sensormodell der Schlüssel zum Umfeldmodell, und seine Genauigkeit und Effizienz bestimmen die Genauigkeit und Leistung des Umfeldmodells. Sich auf künstliche Intelligenz und maschinelles Lernen zu verlassen, ist heute eine Idee, um verschiedene technische Probleme zu lösen. Die Einführung des maschinellen Lernens bietet eine neue Richtung für die Erstellung eines genaueren und effizienteren Sensormodells. Auf dieser Grundlage erforschte Arbeiten wie [BKE21] [BKE19] sind bereits im Gange.
Bei den verschiedenen oben genannten Möglichkeiten zur Erweiterung und Verbesserung des Modells müssen in der Praxis unterschiedlichen Kriterien berücksichtigt werden, z. B. tatsächliche Anwendungsszenarien, Entwicklungskosten, Betriebskosten und Systemleistung.

6 Zusammenfassung

Um das ober

Literatur

[.0619] *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 06/12/2019 - 08/12/2019 . – ISBN 978–1–7281–6321–5

[APML12] Adarve, Juan D. ; Perrollaz, Mathias ; Makris, Alexandros ; Laugier, Christian: Computing occupancy grids from multiple sensors using linear opinion pools. In: *2012 IEEE International Conference on Robotics and Automation*, IEEE, 14/05/2012 - 18/05/2012. – ISBN 978–1–4673–1405–3, S. 4074–4079

[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)

[BKE19] Bauer, Daniel ; Kuhnert, Lars ; Eckstein, Lutz: Deep, spatially coherent Inverse Sensor Models with Uncertainty Incorporation using the evidential Framework. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 09/06/2019 - 12/06/2019. – ISBN 978–1–7281–0560–4, S. 2490–2495

[BKE21] Bauer, Daniel ; Kuhnert, Lars ; Eckstein, Lutz: Deep Inverse Sensor Models as Priors for evidential Occupancy Mapping. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 24/10/2020 - 24/01/2021. – ISBN 978–1–7281–6212–6, S. 6032–3067

[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

[EC10] E. Einhorn ; C. Schröter: Building 2D and 3D adaptive-resolution occupancy maps using Nd-Trees, 2010

[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 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+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

[Thr03] Thrun, Sebastian: Learning Occupancy Grid Maps with Forward Sensor Models. In: *Autonomous Robots* 15 (2003), Nr. 2, S. 111–127. http://dx.doi.org/10.1023/A:1025584807625. – DOI 10.1023/A:1025584807625. – ISSN 1573–7527

[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:

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