Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Andrej Rull

Sensorbasierte Umgebungskartierung mit lokaler Positionskorrektur für autonome Fahrzeuge

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Andrej Rull

Sensorbasierte Umgebungskartierung mit lokaler Positionskorrektur für autonome Fahrzeuge

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. rer. nat. Stephan Pareigis
Zweitgutachter : Prof. Dr. Ing. Andreas Meisel

Abgegeben am 20. August 2008

Andrej Rull

Thema der Bachelorarbeit

Sensorbasierte Umgebungskartierung mit lokaler Positionskorrektur für autonome Fahrzeuge

Stichworte

autonome Fahrzeuge, Weltmodellierung, Positionsbestimmung, Positionskorrektur, Globale- und Lokale Kartierung, Kartenfusionierung

Kurzzusammenfassung

Ein autonomes Fahrzeug, welches sich selbstständig auf einer Strecke orientieren soll, benötigt eine Repräsentation der Einsatzumgebung. So eine Repräsentation muss durch die Sensordatenaufnahme und die Integration der Daten in eine Karte aufgebaut werden. Das Weltmodell unterstützt dabei das Fahrzeug -bei bestimmten Situationen- schneller zu reagieren und eine Wegplanung durchzuführen.
Diese Bachelorarbeit beschäftigt sich mit der Erstellung globaler Weltmodelle, der Skalierung solcher und der Positionsbestimmung in der Karte. Es wird ein Modul entwickelt, welches bei verschiedenen Fahrzeugen zum Einsatz kommen kann.

Andrej Rull

Title of the paper

Sensor-based environment mapping with local position correction for autonomous vehicles

Keywords

autonomous vehicles, world modelling, position regulation, position correction, global and local mapping, map-merging

Abstract

An autonomous vehicle which should orientate itself independently on a distance needs a representation of the application environment. This representation must be built up by the sensor data admission and the integration of the data in a map. Besides, the world model supports the vehicle to react faster to certain situations and to carry out a road planning. This bachelor-document deals with the production of global world models, such the scaling and the position regulation in the map. A Model will developed which can be used on different vehicles.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einführung

Seit Jahren werden Technologien für Fahrzeuge entwickelt, welche die Menschen beim Führen eines Fahrzeugs entlasten und dadurch zur Reduzierung von Verkehrsunfällen führen sollen. Dabei bedient man sich am Vorteil der Sensoren, die -anders als der Mensch- gegenüber Ermüdung, Emotionen, sowie der Missachtung von Verkehrsregeln stabil sind und dadurch zuverlässiger arbeiten.

So findet jedes Jahr in America, die von dem US-amerikanischen Verteidigungsministerium ins Leben gerufene, Darpa Challenge statt. Dabei muss ein Fahrzeug ohne menschliches Eingreifen einen festgelegten Straßenkurs (bspw. in einer Wüste) in einer bestimmten Zeit abfahren. Um von einem Punkt zum anderen zu gelangen orientieren sich die Fahrzeuge anhand einer Karte der befahrenen Umgebung und planen dementsprechend eine Route, die autonom gefahren wird. Nicht anders als beim Menschen ermöglicht die Karte dem Fahrzeug bestimmte Wege (Abkürzungen) zu finden und sich besser in einer Umgebung zu orientieren.

Auch an der HAW werden verschiedene Projekte im Bereich des autonomen Fahrens durchgeführt. Eins davon ist das Carolo-Cup Projekt, bei dem anhand eines Modellfahrzeugs im Maßstab 1:10 Verfahren eines autonomen Fahrens entwickelt werden. Anders als bei der Darpa Challenge muss sich das Carolo-Cup Fahrzeug auf einer aufgeklebten Fahrbahn ohne eine vorgegebene Karte orientieren und dabei Hindernissen ausweichen.

1.1 Zielsetzung der Arbeit

Das Ziel dieser Arbeit ist ein Konzept zu entwickeln, welches die Vorteile einer Karte auch bei Carolo-Cup Fahrzeug ermöglicht. Dazu wird eine autonome globale Kartenmodellierung entwickelt, wo eine Karte während der Fahrt erstellt wird. Diese soll mit der Zeit genauer werden und dem Fahrzeug Informationen über die Umgebung in der es sich befindet, sowie die Information über die Positionierung des Fahrzeugs bereitstellen. Durch so ein Konzept wird dem Fahrzeug das Auffinden der Fahrbahn ermöglicht, wenn es sich verfährt bzw. von der Fahrbahn abkommt. Die Softwarearchitektur der Kartenmodellierung soll dabei nicht nur

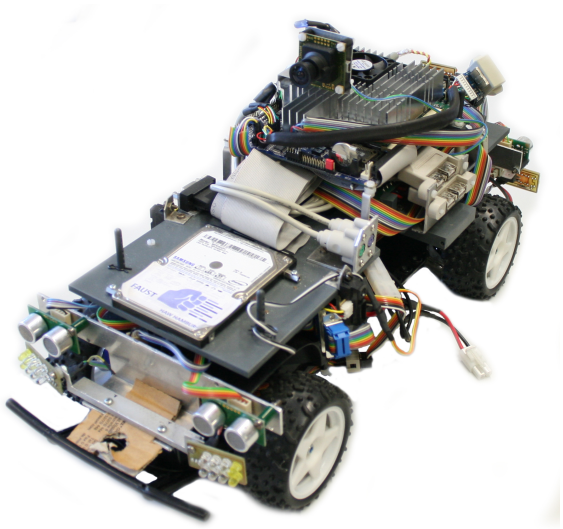

Abbildung 1.1: CaroloCup-Fahrzeug

auf dem Carolo-Cup Fahrzeug betrieben werden. So will man in dieser Arbeit eine Architektur entwickeln, die modular aufgebaut ist und sich auf verschiedenen Plattformen einsetzen lässt.

1.2 Aufbau der Arbeit

Die Arbeit ist in mehrere Abschnitte gegliedert. Zunächst wird im [Kap. 2] auf die Sensorik eingegangen und die verschiedene Methoden erläutert um Sensordaten in einer Karte effektiv einzusetzen. [Kap. 3] gibt ein Einblick in die Aufgaben der Weltmodellierung. Es werden verschiedenen Arten von Karten vorgestellt, welche bei der Kartierung zum Einsatz kommen. Im [Kap. 4] werden die Anforderungen an die Weltmodellierung dieser Arbeit gestellt und Algorithmen für die Kartenkonstruktion, Positionsbestimmung sowie die Skalierung der Karte analysiert und entwickelt. Die Implementierung der globalen Weltmodellierung erfolgt im [Kap. 5], wo der Softwareaufbau und der Kartierungsablauf anhand von UML Diagrammen beschrieben wird. Das letze Kapitel [Kap. 6] beschäftigt sich mit den verschiedenen Testszenarien der Weltmodellierung auf dem Carolo-Cup Fahrzeug, wobei auch die Genauigkeit der Positionskorrektur betrachtet wird.

2 Sensorsysteme und deren Wahrnehmung

Durch die Umwandlung eines nichtelektrischen Eingangssignals in ein elektrisches Ausgangssignal stellen Sensoren die Basis für die Wahrnehmung der Umgebung dar. Dadurch sind sie das Bindeglied zwischen der roboterinternen Informationsverarbeitung und der realen Welt. Solch ein Bindeglied ist bei einem autonomen Betrieb unumgänglich. Damit ein fahrerloses Agieren möglich ist, muss das Fahrzeug die Strecke wie auch die Hindernisse erkennen und bestimmte interne Parameter- wie Geschwindigkeit oder Lenkwinkel- bestimmen können, um bei bestimmten Situationen rechtzeitig und richtig zu reagieren.

2.1 Sensorik

Je nach Messwert werden Sensoren bei autonomen Systemen in interne und externe Sensoren klassifiziert. Interne Sensoren stellen dabei dem Fahrzeug systeminterne Zustände durch Messwerte zu Verfügung. Diese wären bspw. die Position des Fahrzeugs, die Geschwindigkeit oder auch der Lenkwinkel. Anders die externe Sensoren, welche zur Erfassung der Umwelt eingesetzt werden.

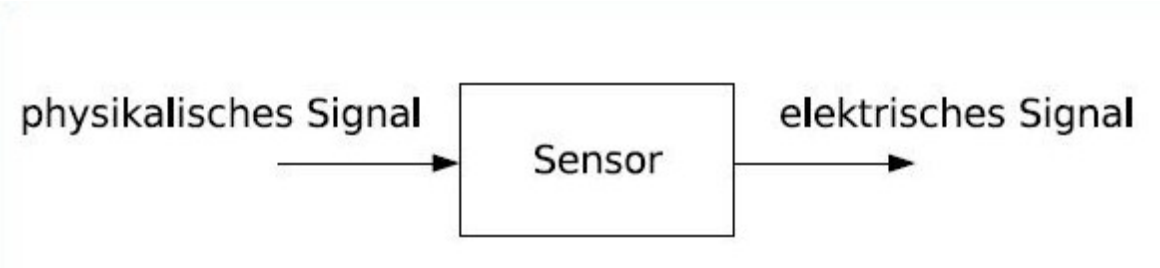

Abbildung 2.1: Sensor

Bedingt durch die Fehleranfälligkeit von Sensoren erfordert die Wahrnehmungsverarbeitung einer realen Umwelt eine Vielzahl von unterschiedlichen Sensortypen. Solch ein multisensorieller Ansatz kann durch die Kombination artfremder Sensoren (visuell, akustisch) und durch eine unterschiedliche Positionierung gleicher Sensoren auf dem System erreicht werden.

Durch diesen Ansatz soll erreicht werden, dass sich Sensoren bei Messfehlern widersprechen und man mit einer gewissen Wahrscheinlichkeit sagen kann welche Messwerte richtig und welche falsch sind.

interne Sensoren	externe Sensoren
Radencoder	Ultraschall
Spannungsüberwachung	Laser
Temperatur	Infrarot
Gyroskop	Kamera

Tabelle 2.1: Interne und externe Sensoren

2.1.1 Akustische Sensoren

Die meisten akustischen Sensoren arbeiten nach dem Prinzip der Laufzeitmessung. Dabei wird ein elektrischer Impuls vom Sendewandler ausgesandt und von der geometrischen Lage der Umwelt reflektiert. Nach einer bestimmten Zeit bekommt der Empfangswandler die Reflexion zurück und kann diese wiederum in ein elektrisches Signal konvertieren. Durch die Laufzeit des Echosignals kann so die Entfernung bestimmt werden. In der Praxis werden dafür gern Ultraschallsensoren genommen, bei welchen sich der Abstand s zwischen dem Reflektor (z.B. ein Hindernis) und dem Sensor im Idealfall durch die Formel

$$s = \frac{1}{2} * tc$$

bestimmen lässt. Wobei t die Laufzeit des Impulses darstellt und c die Schallgeschwindigkeit in der Luft. Diese beträgt laut [Knieriemen (1991), S.27] 33,4 cm/ms.

Da der Sendewandler einen Öffnungswinkel besitzt, durchleuchten akustische Sensoren immer einen bestimmten Bereich. Je höher der Öffnungswinkel ist, desto größer ist auch das Gebiet welches durchleuchtet wird. Dies hat zwar den Vorteil, dass kleinere Objekte in der Umgebung erfasst werden, jedoch den Nachteil, dass bei einem zu großen Öffnungswinkel eine Mehrfachreflexion auftritt. Bei der Mehrfachreflexion werden mehrere Objekte durch den

Sensor erfasst, wodurch man nicht genau sagen kann welches Objekt zu dem Messwert in Relation steht.

Akustische Sensoren haben auch den Nachteil, dass Messergebnisse von den Umwelteigenschaften wie Temperatur, Druck und Luft abhängig sind. So könnten Echosignale verzögert werden oder je nach dem wie der Reflektor (das Objekt in der Umgebung) steht, das Signal in eine falsche Richtung reflektieren und dadurch ein falsches Ergebnis produzieren.

2.1.2 Optische Sensoren

Optische Sensoren werden je nach Signalaufnahme in passive und aktive Systeme eingeteilt. Passive optische Sensoren messen Farbsignale oder auch Lichtintensität in der Umgebung. Diese Eigenschaften werden als Bild aufgenommen. Die Entfernung zu einem Objekt wird dann durch die Korrelation berechnet. Dabei nehmen unterschiedlich positionierte Kameras dieselbe Szane auf. Der Objektpunkt P und die Standpunkte der Kameras bilden ein Dreieck, dessen Basislänge b und dessen Basiswinkel α und β man kennt. Dadurch dass die Basiswinkel bekannt sind, können alle anderen Größen im Dreieck berechnet werden. So wird auch der Entfernungswert zwischen den entsprechenden Punkten der Bilder bestimmt [Abb. 2.2].

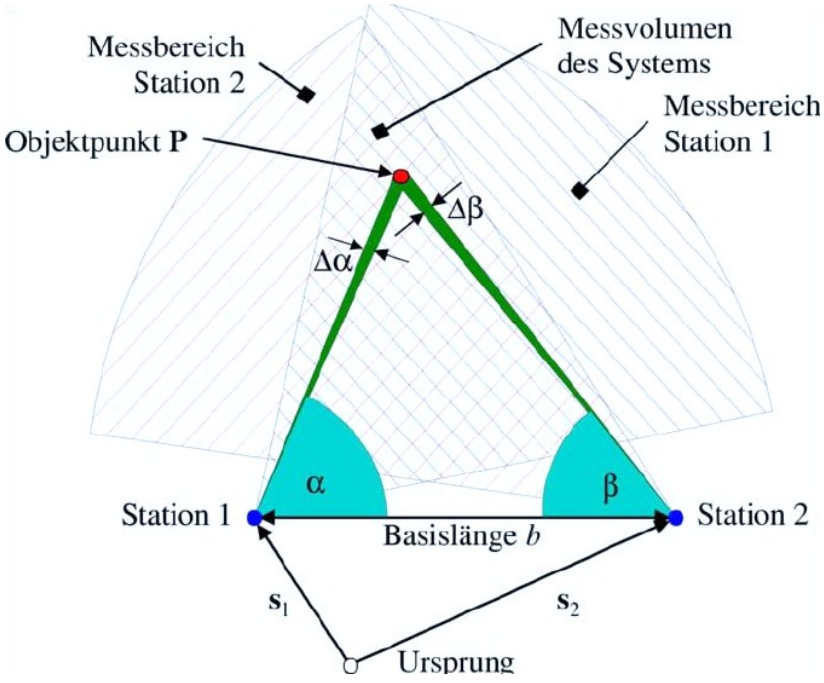

Abbildung 2.2: Triangulation[1]

Da die Korrelation in der Bildverarbeitung sehr Zeitaufwendig ist werden anstelle der passiven, aktive optische Sensoren eingesetzt. Aktive Verfahren nutzen eine Lichtquelle, z.B. einen Laser, welche unter einem bestimmten Winkel das Objekt beleuchtet, dessen Oberfläche erfasst werden soll. Durch die Kenntnis der Strahlrichtung und des Abstandes zwischen Kamera und dem Laser kann der Abstand vom erkannten Objekt zu der Kamera bestimmt werden. Dabei bildet die Verbindung zwischen Kamera, Laser und den Strahlen zum und vom Objekt ein Dreieck. Michal Ebert hat so ein passives System mit einem Linienlaser auf dem CaroloCup Fahrzeug umgesetzt. Weitere Einzelheiten zu dem System werden in seiner Bachelorarbeit [Ebert (2008)] gut veranschaulicht dargestellt.

2.2 Sensordatenverarbeitung

Da Sensoren nur physikalische Eigenschaften aufnehmen, die zunächst vage in einem Zusammenhang mit den Informationen stehen, die ein autonomes Fahrzeug benötigt, müssen die Sensordaten optimal verarbeitet werden.

Die Sensordatenverarbeitung hat dabei die Aufgabe eine problemorientierte Aufarbeitung der Daten durchzuführen, nackte Sensordaten in Systemdaten zu konvertieren und ungültige Messdaten zu eliminieren. [Knieriemen (1991), S.30] beschreibt dies als die Restriktion der Sensoren auf der Eingabeseite und die Anforderungen an das System auf der Ausgabeseite.

Bei der Aufarbeitung der Daten werden die Sensordaten in ein einheitliches Datenformat gepackt, welches unabhängig vom Sensor im ganzen System bekannt ist. Dadurch können Messwerte, die durch Sensoren eingespeist wurden flexibel im ganzen System genutzt werden. Die verarbeiteten Daten können wiederum, je nach Systemanforderung, verschiedene Informationen beinhalten und ermöglichen durch diese Informationsspeicherung zu einem späteren Zeitpunkt einen flexibleren und effektiveren Umgang mit den Daten. So werden keine nackten Sonarsensor, - Infrarotsensor,- oder Kamerdaten im System gespeichert, welche nur die Anwesenheit und Abwesenheit von Objekten angeben, sondern die modifizierten Daten, welche bspw. ein Hindernis kennzeichnen.

Die Eliminierung der Ungültigen Sensordaten ist ein wichtiger Bestandteil der Sensordatenverarbeitung. Dabei wird darauf geauchtet welche Sensoren Messwerte liefern, die im Wiederspruch zueinander stehen. Mit einer gewissen Wahrscheinlichkeit lässt sich somit ermitteln, welcher Sensor einen falschen Wert liefert.

Die unterschiedliche Charakteristik der Sensordaten (z.B. sind Ultraschallsensoren meist nur für großflächige Hinderniserkennung geeignet) wie auch die Unmenge an Datenaufkommen

[1] www.fh-wolfsburg.de/cms/de/pws/justen/studiumlehre/Vorlesungen/Sensor_Aktor/Teil2.pdf

in einem Sensorsystem erschwert die Sensordatenverarbeitung. Diese muss möglichst in Echtzeit durchgeführt werden, damit das System auf die Messwerte rechtzeitig reagieren kann. Ein Lösungsansatz wäre, die Verarbeitung separat auf einem Mikrokontroller bzw. auf einem FPGA Baustein durchzuführen.

Folgende Aufgaben werden von der Sensordatenverarbeitung bearbeitet:

- Fehlereliminierung
- Messwertkompensierung
- Merkmalsextraktion
- Modellabstraktion
- Datenfusion

2.2.1 Fehlereliminierung

Das größte Problem bei Sensoren ist, dass die Genauigkeit der gelieferten Daten durch äußere Einflüsse beeinträchtigt wird. Bei Fahrzeugen hat man bspw. mit der Vibration zu kämpfen, welche ein Rauschen im empfangenen Signal verursacht. Unter Einem Signal wird eine skalare Zeitfunktion der Form

$$\tau = x(t), t_0 \leq t \leq t_1$$

verstanden.

Damit das Signal richtig interpretiert werden kann, muss es zunächst durch verschiedene Filtermechanismen raus gefiltert werden, indem das Rauschen unterdrückt wird. Hier kann man auch von einer Low Level Fehlereliminierung sprechen, da man nicht die Richtigkeit der Messwerte überprüft, sondern das empfangene Signal optimiert, um die Daten besser auslesen zu können. Die High Level Eliminierung erfolgt direkt nach der Low Level Eliminierung. Dabei werden Daten, welche das Signal beinhaltet, analysiert und ausgewertet. Bei dieser Eliminierung wird entweder ein zeitlich, logischer Zusammenhang zwischen den empfangenen Messwerten aufgestellt und die Messwerte, die logisch nicht in den Zusammenhang passen verworfen oder Messwerte unterschiedlicher Quellen durch Abgleich nach ihrer Richtigkeit geprüft. Durch diesen Abgleich lässt sich mit einer gewissen Wahrscheinlichkeit sagen, welche Daten richtig und welche falsch sind.

2.2.2 Messwertkompensierung

Da sich bei einem Multisensorsystem häufig mehrere Sensoren überschneiden, sind nicht alle Messwerte zu gebrauchen. Auch will man die Datenmenge flexibel halten, sodass diese schnell verarbeitet werden kann. Die Messwertkompensierung dient zur Reduktion solcher empfangenen Datenmengen. Mittelwertbestimmung oder auch Histogrammgenerierung sind zwei Verfahren mit welchen die Sensordaten zu charakteristischen Werten komprimiert und auf die Problemstellung reduziert werden.

2.2.3 Merkmalsextraktion

Wie schon bei der Messwertkompensierung wird auch bei der Merkmalsextraktion die Reduzierung der aufkommenden Datenmenge angestrebt. Durch den Gewinn symbolischer Informationen bleibt der Informationsgehalt aber erhalten. Die unterschiedlichen Sensoren werden nach bestimmten geometrischen Kriterien wie Kanten, Linien, Cluster, Kreisbögen, Flächen, Ecken, etc. zusammengefasst und können unabhängig vom Sensortyp betrachtet werden. Die Daten werden dann nicht mehr als Punktdaten weiterverarbeitet, sondern als geometrische Merkmale im System geführt.

So können für Hindernisse in der Umgebung unter anderem Position, Breite, Länge und je nach Anwendung auch die Höhe als Merkmalsattribut abgespeichert werden. Bei autonomen Fahrzeugen, für die Streckenmerkmale wichtig sind können Stopplinien oder auch komplette Kreuzungen als Merkmale durch eine Kamera erfasst und durch verschiedene Algorithmen extrahiert werden. Laut [Knieriemen (1991), S.113] werden die Verfahren zur Generierung der Merkmale aus einer Messwertaufnahme grundsätzlich nach der Betrachtungsweise der Aufnahme und der Vorgehensweise der Bearbeitung klassifiziert. Für die Navigation eines autonomen Fahrzeugs ohne GPRS Empfänger ist eine Merkmalsextraktion unumgänglich, da solch eine Datenaggregation die Basis für eine echtzeitfähige Datenverarbeitung bildet.

2.2.4 Modellabstraktion

Je nach dem welche Anforderung an das System gestellt wird, muss die Sensordatenverarbeitung Daten mit unterschiedlichen Abstraktionsebenen bereitstellen. Sollen später nur geometrische Daten verarbeitet werden, so reicht es, wenn man zu einem Merkmal die Position und Orientierung abspeichert. Will man die Daten später jedoch für die Wegplanung einsetzen, sind topologische Attribute, wie räumliche Beziehung zwischen mehreren geometrischen Merkmalen, zu empfehlen.

2.2.5 Sensordatenfusion

Die Sensordatenfusionierung ist wohl das wichtigste Merkmal der Sensordatenverarbeitung, da hier die Sensormesswerte unterschiedlicher Sensortypen zu einem konsistenten und koinzidenten Datenformat zusammengefasst werden. Die Fusionierung zu einer integrierten Darstellung stellt die Basis für eine Lösung von komplexen Aufgabenstellungen dar und verbessert die Leistungsfähigkeit eines autonomen Fahrzeugs.

Die Quellsensoren müssen dabei nicht zwingend unterschiedlich sein. Auch Sensoren gleichen Typs, können durch verschiedene Positionierung auf einem mulisensoriellem System zur Verbesserung der Sensorqualität führen, indem sie Messwerte aus unterschiedlichen Blickwinkeln bereitstellen, welche durch die Verschmelzung zu einem Datenformat ein genaueres Ergebnis liefern, als ein Sensortyp alleine. Die Leistungsverbesserung erreicht man laut [Isler und Bajcsy (2006), S.372-381] indem man Fehler der einzelnen Sensoren durch die Kombination reduziert und den Sensoreinsatz maximiert.

Doch nicht nur eine reine Sensordatenfusionierung ist bei der Komprimierung zu einem Datenformat möglich. Zusätzlich kann das verschmolzene Datenformat um weitere, aufgabenspezifische Informationen ergänzt werden und diese dem System zu Verfügung stellen. So kann bspw. für die Navigation eines autonomen Fahrzeugs das Datenformat zusätzlich zu den Polarkoordinaten und kartesischen Koordinaten bestimmte Merkmalsinformationen beinhalten, welche die Navigationssteuerung zu einem späteren Zeitpunkt unterstützt und erleichtert.

Laut [Hall und Llinas (1997), S.6-23] gibt es drei unterschiedliche Ebenen der Sensordatenfusionierung:

- data fusion
- feature fusion
- decision fusion

Anders als bei der "`data fusion`", in der reine Sensordaten miteinander verschmelzen, werden bei der "`feature fusion`" zunächst die Sensordaten durch die Merkmalsextraktion bearbeitet und dann die erhaltene Merkmale zu Merkmalsvektoren kombiniert. In der Audiovisuellen Spracherkennung gäbe es z.B. akustische und visuelle Merkmalsvektoren, welche sich aus der Kombination von Sprachlauten und Lippenbewegungen zusammensetzen.

Ein anderes Beispiel wäre das Mapping. Hier könnten bestimmte Kanten durch eine Kamera erkannt werden und dadurch ein Merkmal (wie Kreuzung) bilden. Die Sensoren könnten wiederum durch Polygonbildung das Datenformat um weitere Merkmale (z.B. Hindernisse) aus der Umgebung erweitern. Die Kombination dieser Merkmale würde einen Merkmalsvektor

zu einem Zeitpunkt t und einer Position p bilden. Bei der "`decision fusion`" findet die Zusammenführung der Daten erst statt, nachdem alle Sensordaten für richtig bewertet und alle Merkmale extrahiert wurden.

3 Weltmodellierung

Damit ein Fahrzeug sich selbstständig in der Umgebung oder auf einer Strecke orientieren kann, muss es -genau wie der Mensch- die befahrene Umgebung und deren Merkmale aufnehmen und diese richtig interpretieren können. Dazu wird eine interne Weltdarstellung aufgebaut, welche als Weltmodell bezeichnet wird. Das Weltmodell muss dabei so genau aufgebaut werden, dass es allen Anforderungen, wie z.B. der Hinderniserkennung und der Wegplanung, gerecht wird.

Das Problem dabei ist, dass bei einem genaueren Modell die Komplexität zunimmt, da mehrere Daten von verschiedenen Informationsquellen aufgenommen und verarbeiten werden müssen. Informationsquellen können, wie im vorherigen Kapitel beschrieben, Sensoren, Kamerabilder oder auch Steuermodule sein. Um diese Komplexität in den Griff zu bekommen, werden die Anforderungen durch ein hierarchisches System mit unterschiedlichen Abstraktionsebenen erfüllt [Knieriemen (1991), S.53].

Wie in [Abb. 3.1] zu sehen, ist eine Abstraktionsebene nichts anderes als eine Karte, die -je nach Anforderung- verschiedene Informationen beinhaltet. Der Abstraktionsgrad reicht dabei von der einfachen Sensorkarte, bei der reine Sensormesswerte in die Karte aufgenommen werden, bis hin zur metrischen oder topologischen Darstellung der Karte, welche für komplexe Aufgaben Sensorwerte zusammengefasst betrachten und dadurch mehr Informationen über die Umwelt zur Verfügung stellen. Dabei können die Abstraktionskarten entweder manuell in das System geladen oder autonom durch das Fahrzeug kartographiert werden. Bei der autonomen Kartierung baut sich die Karte Schritt für Schritt auf und wird mit der Zeit immer genauer, da zeitlich immer mehr Informationen in die Karte gemappt werden. Solch eine Generierung der Karte wird auch als Weltmodellierung bezeichnet.

Die Weltmodellierung ist generell zuständig für die Verwaltung systeminterner Darstellung der Einsatzumgebung. Sie hat folgende Aufgaben zu erledigen:

- Abspeichern, Aktualisieren und Erweitern von Daten der Einsatzumgebung
- Generierung von geeigneter Kartendarstellung (Bestimmung des Abstraktionsgrades)
- Positionsbestimmung auf der Karte

Verfahren, die solche Anforderungen erfüllen, werden durch Repräsentation und Abstraktionsebenen des Weltmodells gekennzeichnet.

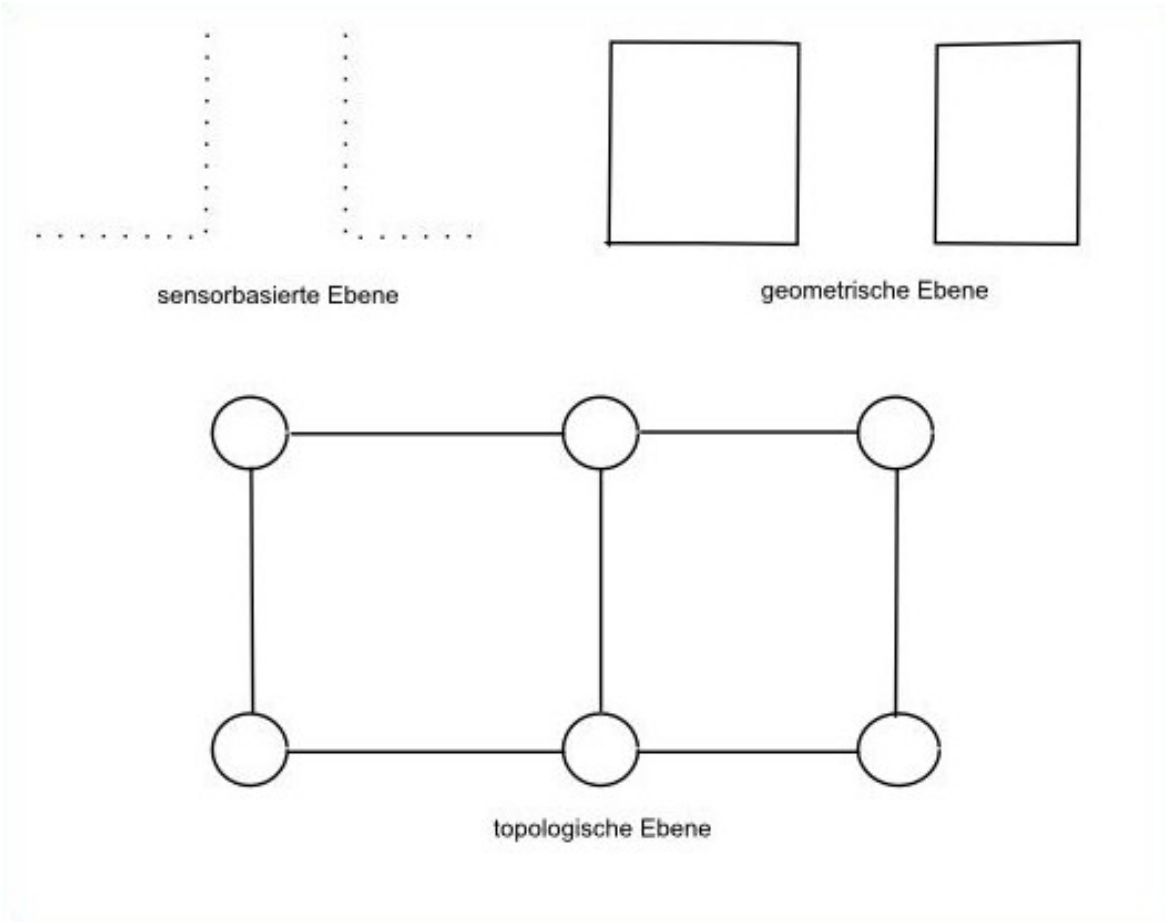

Abbildung 3.1: Modellabstraktion

3.1 Abstraktionsebenen des Weltmodells

Wie im vorherigen Abschnitt bereits erwähnt, wird eine genaue Karte angestrebt, die zwar möglichst viele Informationen beinhaltet, jedoch einfach zu verwalten und zu benutzen ist. Sensorbasierte und datenintensive Informationen werden deshalb aufgenommen und zu kompakten aufgabenorientierten Darstellungen transformiert. Durch diese Transformation erhält man, je nach Anforderung, mehrere Karten, die mehr oder weniger Informationen enthalten. Beispielweise wird für die Visualisierung und Navigation eine geometrische Karte benötigt, in welcher detailliert alle Hindernisse eingezeichnet werden.

Damit kann ein autonomes Fahrzeug unter der Verwendung mehrerer Kartentypen auch unterschiedliche Anforderungen erfüllen. So lässt sich das Ausweichen von Hindernissen oder das Erkennen von Sackgassen durch eine geometrische Karten und die kürzeste Routenberechnung -bei einer Strecke mit verschiedenen Kreuzungen- durch die topologische Karte

realisieren. Topologische Karten bestehen aus Knoten und Kanten, wobei die Knoten unterschiedliche Orte der Umgebung repräsentieren und die Kanten die Verbindung bzw. den befahrenen Weg zwischen diesen. Anders die geometrischen Karten. Diese beinhalten entweder die Sensordaten in Form von Merkmalspolygonen oder Rastern.

Ein Abstraktionsmodell kann auf verschiedene Arten modelliert werden. Laut [Kuipers (2000), S.191-233] wird zunächst ein topologisches Modell aus den Sensordaten aufgebaut. Danach wird eine lokale Landmessung durchgeführt und den Plätzen oder den Verbindungen zugeordnet. Eine lokale Landmessung bedeutet, dass man die Umgebung aus Sicht des Fahrzeugs abscannt und die erhaltenen Daten dem jeweiligen Kartenabschnitt zuordnet. Eine lokale Landmessung wird auch als lokales Mapping bezeichnet. Durch diesen Ansatz erhält man sensorbasierte, topologische und geometrische Ebenen.

In [Elfes (1986), S.1151-1156] und [Elfes (1987), S.249-265] wird eine andere Möglichkeit beschrieben verschiedene Abstraktionsmodelle zu generieren. Dabei ist aus den Sensorwerten zuerst eine geometrische Karte und anschließend eine topologische Karte aufzubauen.Es werden Sensordaten aufgenommen, durch die Sensordatenverarbeitung nach Gültigkeit geprüft und in eine Karte eingetragen. In der neu generierten Darstellung sucht man anschließend nach bestimmten Plätzen (Knoten), die sich durch bestimmte Merkmale unterscheiden und legt eine Verbindung (Kante) zwischen diesen. Der Vorteil solch eines Abstraktionsablaufs ist, dass die Knotenberechnung durch verschiedene Bildverarbeitungsalgorithmen im Bereich der Merkmalsextraktion umgesetzt werden kann. Ein Nachteil ist der Zeitaufwand bei der Suche nach Merkmalen in der geometrischen Karte.

3.2 Sensorbasiertes Weltmodell

Das sensorbasierte Weltmodell ist in der Abstraktionshierarchie ganz unten angesiedelt. Laut [Elfes (1986), S. 1152] ist dieses Weltmodell direkt von der Sensorlesung abgeleitet und ist gewissermaßen eine Beschreibung, die der Welt am nächsten kommt. Es stellt eine Momentaufnahme der Sensoren dar, durch die andere Arten von Darstellungen abgeleitet werden.Dabei wird die Geometrie der momentanen Umgebung durch Informationsquellen (Sensoren, Kamera) erfasst und durch die Sensordatenverarbeitung und die Datenfusionierung modelliert. Die Konstruktion des sensorbasierten Weltmodells erfolgt dabei sukzessiv während der Fahrt.

Sensorbasierte Weltmodellierung sagt nichts über die Form eines erkannten Hindernisses aus. Es werden lediglich Punktkoordinaten eines Sensorwertes in ein Koordinatensystem aufgenommen.

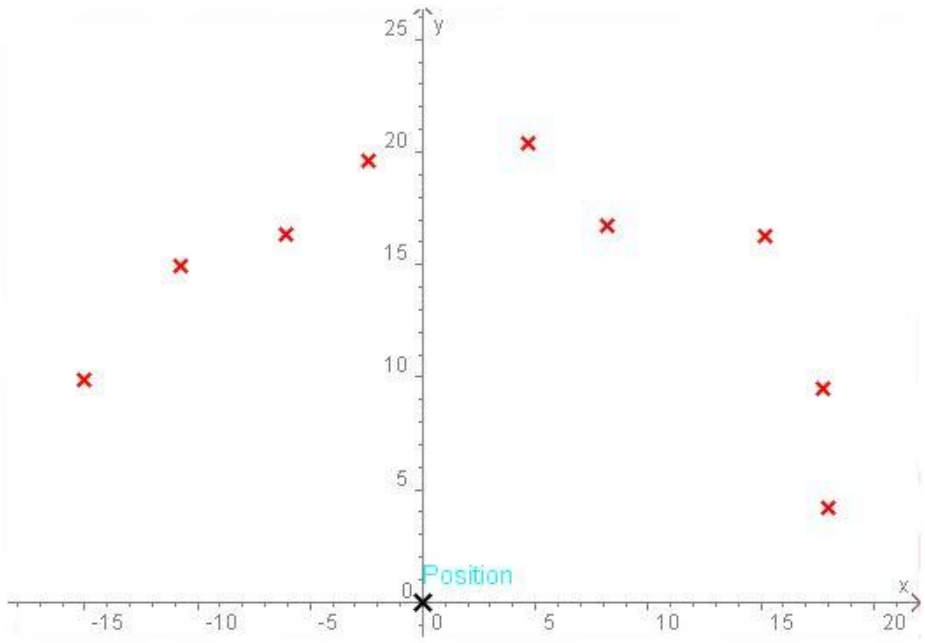

Abbildung 3.2: Sensorbasiertes Weltmodell

Da lediglich Objekte der Punktkoordinaten generiert werden, ist ein sensorbasierter Modellaufbau sehr schnell und einfach zu realisieren und durch den geringen Berechnungsaufwand sehr schnell bei der Kartengenerierung.

3.3 Geometrisches Weltmodell

Das geometrische Weltmodell kommt technisch dem sensorbasiertem Weltmodell ganz nah, da das Modell ebenfalls aus den Sensordaten der Bildverarbeitung bzw. den Sensormesswerte, der entfernungsmessenden Sensoren abgeleitet wird.

Im Unterschied zu dem sensorbasiertem Modell werden hier jedoch nicht alle Messpunkte der Sensoren in ein Weltmodell eingetragen, sondern nur die Daten, welche durch die Merkmalsextraktion zu geometrischen Merkmalen zusammen geführt oder mit Hilfe von Rasterverfahren gebündelt wurden. Anstelle von reinen Sensordaten wird somit die Position ganzer Umgebungsmerkmale bzw. Raster im globalen Koordinatensystem abgespeichert.

Durch die Komprimierung der Sensormesswerte wird das geometrische Modell vorwiegend für Aufgaben in der Navigation, in der lokale Kursplanung und der Positionskorrektur eingesetzt.

3.4 Repräsentation des geometrischen Weltmodells

Je nach Aufgabenstellung kann ein geometrisches Weltmodell komplex oder auch trivial aufgebaut werden. Dient es der Hinderniserkennung und Wegplanung, so muss die Karte zu jedem Zeitpunkt die Hindernissituation möglichst vollständig wiedergeben. Die Weltrepräsentation kann dabei die Hindernisstruktur, die freie Fläche zwischen den Hindernissen oder beide Informationen gleichzeitig enthalten [Knieriemen (1991), S.60].

Die Art der zu verwendeten Merkmalsdarstellung hängt dabei -wie in [Kap. 2] beschrieben- größtenteils von der verwendeten Sensorik ab, welche zur Wiedererkennung der Umgebung eingesetzt wird. Die typischen Beispiele solcher Merkmalsdarstellung sind 2D-Liniensegmente, Polygone oder auch Raster.

3.4.1 Polygonaldarstellung

Bei den 2D-Liniensegmenten und der Polygondarstellung wird das Weltmodell möglichst genau an die Weltstruktur angepasst. Dazu werden überwiegend polygonale Verfahren eingesetzt, welche in [Weber (2003)] genauer erläutert werden. Diese Art der geometrischen Modelle ist nur dann zu gebrauchen, wenn die Natur der Umgebung eine Darstellung der Merkmale durch lineare Elemente unterstützt.

Nichtlineare Umgebungsmerkmale in der Einsatzumgebung (Kurven) verursachen Fehler in der Karte. Auch die Wegplanung auf Basis solch einer Darstellung gestaltet sich laut [Weber (2003), S.24] geometrisch recht aufwendig. In dieser Arbeit wird nicht weiter auf die Polygonale Weltdarstellung eingegangen, da solch eine Repräsentation für eine Fahrtstrecke mit mehreren Kurven ungeeignet ist.

3.4.2 Rasterdarstellung

Enthält die Einsatzumgebung nichtlineare Formen, so ist ein Rastermodell von Vorteil, da hier eine eigene Struktur ohne Rücksicht auf die Einsatzumgebung definiert wird. Bei rasterorientierten Verfahren wird die Umwelt als ein Gittermodell gesehen. Gitterzellen -in dieser Arbeit auch Grids genannt- beschreiben die freie und die belegte Fläche in der Umwelt, indem ihnen ein Wahrscheinlichkeitswert zugewiesen wird.

Durch diese Eigenschaft kann die rasterbasierte Weltmodellierung sowohl in strukturierter als auch in relativ unstrukturierter Umgebung eingesetzt werden, da die Modellierung nicht auf das Erkennen und Zusammenführen von geometrischen Primitiven ausgerichtet ist. Das

primäre Ziel dabei ist, durch sukzessive Aktualisierung der Rasterkarte die tatsächliche Objektkonstellation in der Einsatzumgebung möglichst optimal zu approximieren.

[M. Beckerman (1990)] beschreibt einen rasterbasierten Ansatz der Weltmodellierung. Dabei werden zunächst bei der Initialisierung alle Felder in der globalen Karte als unbekannt markiert. Bei der Weltmodellierung wird dann ein lokales Sensormodell (aus sicht des Fahrzeugs) aufgebaut, welches die Rasterzellen als belegt markiert, die durch Sensoren als Hindernisse identifiziert wurden. Alle anderen Zellen in der lokalen Karte werden als frei markiert. Nun wird die lokale Karte in eine globale Rasterkarte fusioniert, indem die Raster der lokalen Karte mit den dazugehörigen Rastern der globalen Karte abgeglichen werden. Zu der Durchführung stellt Beckermann Kombinationsregeln auf, welche in der [Tab. 3.1] abgebildet sind. Die Zellen können dabei vier unterschiedliche Eigenschaften einnehmen: `"Frei"`, `"Belegt"`, `"Unbekannt"` oder `"Widersprüchlich"`.

	W	U	B	F
W	W	W	W	W
U	W	U	B	F
B	W	B	B	W
F	W	F	W	F

Tabelle 3.1: Kombinationsregeln der Rasterzellen nach Beckerman

[Abb. 3.3] zeigt eine globale Rasterkarte, wo die Gitterzellen drei unterschiedliche Zustände einnehmen können. Weitere rasterorientierte Ansätze werden in [Elfes (1986)] und [Borenstein und Koren (1990)] vorgestellt.

Rasterorientierte Verfahren haben den Vorteil, dass sie sich an der Technik der Bildverarbeitung bedienen, da ein digitales Bild ebenfalls als ein Raster, ein sogenanntes Bitraster (ein Bit pro Pixel), interpretiert werden kann. Diese Ähnlichkeit erlaubt es einer rasterorientierten Repräsentation des Weltmodells verschiedene Bildverarbeitungsalgorithmen im Modell anzuwenden.

Der bedeutendste Vorteil solch einer Repräsentation ist jedoch, dass die geometrische Karte selbst in einer komplexen Einsatzumgebung schnell aufgebaut werden kann. Selbst ein unebenes Gelände kann durch Rasterzellen dargestellt werden, indem die Unebenheit als Parameter in die Zelle eingetragen wird. Die Gridzellen können zudem Informationen -Wahrscheinlichkeitswerte über die Belegung- der Nachbarzellen enthalten, wodurch eine lokale Navigation mit Hilfe globaler Kartenabschnitte durchgeführt werden kann.

[2] Weber (2003)

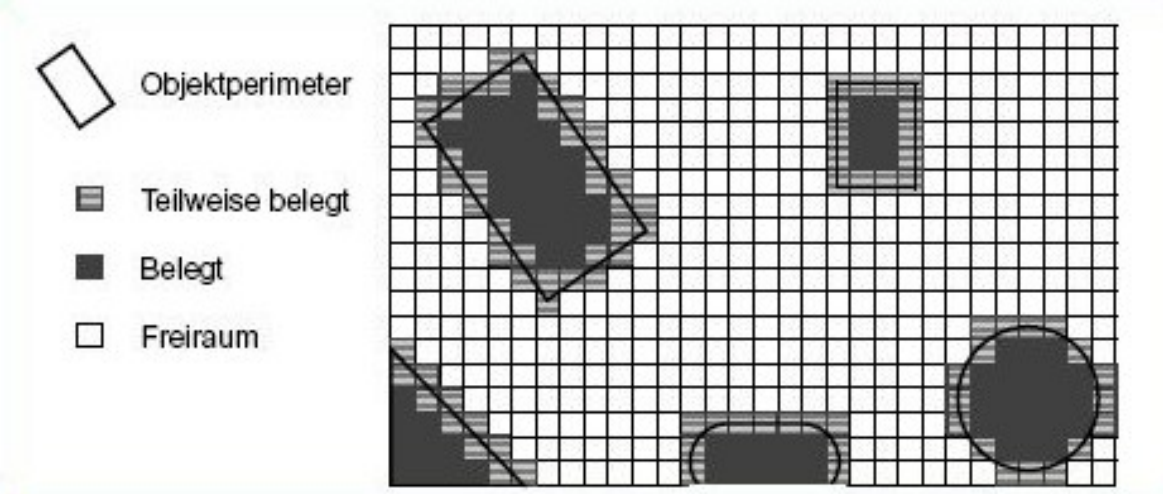

Abbildung 3.3: Wahrscheinlichkeitswerte der Rasterzellen[2]

Auch Wegplanungen werden mit Rastermodellen durchgeführt. So werden in [Gilles (1985), S.935] und [Kai-Tai u. a. (1991), S.423-428] Wegplanungsalgorithmen auf Basis einer Rasterkarte entwickelt. [Abb. 3.4] zeigt den Gilles Algorithmus. Es werden die freien und damit befahrbaren Grids vom Startpunkt aus in senkrechter und waagerechter Richtung nummeriert und hochgezählt. Danach wird geprüft, durch welche Grids in der Rasterkarte gefahren werden muss, um mit der kürzesten Anzahl an freien Zellen zum Ziel zu gelangen. Die gefundenen Zellen repräsentieren den kürzesten Weg zu einem Zielpunkt auf der Karte.

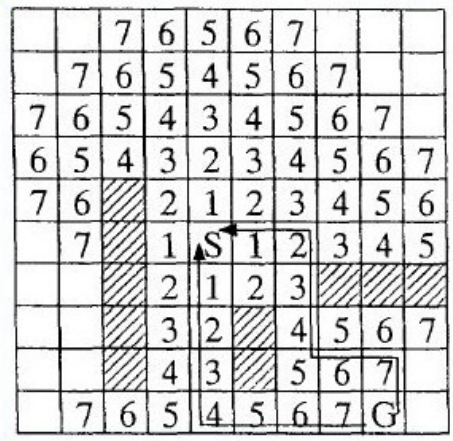

Abbildung 3.4: Gilles Algorithmus[3]

Rastermodelle haben nicht nur Vorteile. Ein gravierender Nachteil bei solchen Modellen

[3] Kai-Tai u. a. (1991), S.426

ist die schlechte Skalierbarkeit. Der Speicherbedarf solcher Karten könnte in weiträumigen Einsatzumgebungen quadratisch ansteigen. Deshalb muss ein Kompromiss zwischen dem Wunsch nach höchstmöglicher Auflösung, Speicher- und Recheneffizienz getroffen werden.

3.5 Topologisches Weltmodel

Wie schon erwähnt sind geometrische Modelle zwar schnell aufgebaut, jedoch in der Auswertung und Skalierbarkeit sehr Speicher- und Rechenintensiv. Um die Wegplanung zu beschleunigen werden zusätzlich zu den sensorbasierten und geometrischen Karten topologische Karten eingesetzt.

Ein topologisches Modell baut sich aus der Lage und den Anordnungen geometrischer Strukturen auf, wie Kanten von Hindernissen, Kreuzungen, Abzweigungen oder auch konvexen Freiräumen, welche laut [Kuipers (2000)]markante Punkte auf einer geometrischen Karte bilden. Diese gefundenen Stellen werden auch als Knoten bezeichnet und sind durch eine topologische Beziehung (Transition / Kanten) miteinander verbunden. Die topologische Karte (Symbolic Level Map) wird als Graph dargestellt und ist in der Modellhierarchie, wie in [Abb. 3.5] zu erkennen, ganz oben angesiedelt.

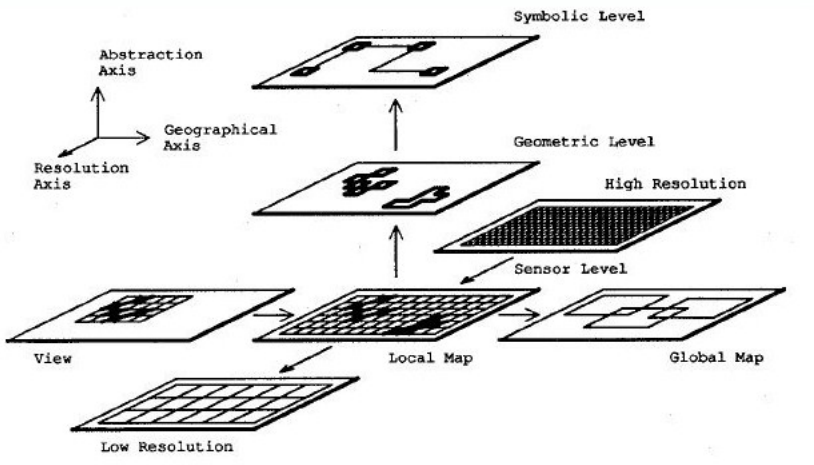

Abbildung 3.5: Modellhierarchie der Karten [4]

Dabei enthalten rein topologische Karten keine metrischen Distanzinformationen. Die Transitionen zeigen lediglich an, ob Knoten A vom Knoten B direkt aus erreichbar

[4] Elfes (1987), S.260

ist und kodieren dabei zusätzlich die Trajektorien Daten. Trajektorien Daten beinhalten eine Wegbeschreibung von einem Knoten zu einem Nachbarknoten. Für das Fahrzeug ist nichts anderes kodiert als: "`fahre 100 Meter geradeaus und biege dann rechts ab`". In [Kuipers (2000), S.199] wird so eine Wegbeschreibung auch als "`trajectory-following`" bezeichnet.

[Abb. 3.6] zeigt eine geometrische Karte und die dazugehörige topologische Darstellung mit den identifizierten Plätzen und Kanten. Die Eckpunkte und die Abzweigungen bilden im Raum markante Punkte, die in der topologischen Karte durch $P1$ - $P5$ dargestellt werden. In den Klammern an den Transitionen werden die Trajektorien Daten dargestellt. So kommt man vom topologischen Punkt $P7$ zum Punkt $P4$ indem man sich an der rechten Wand orientiert.

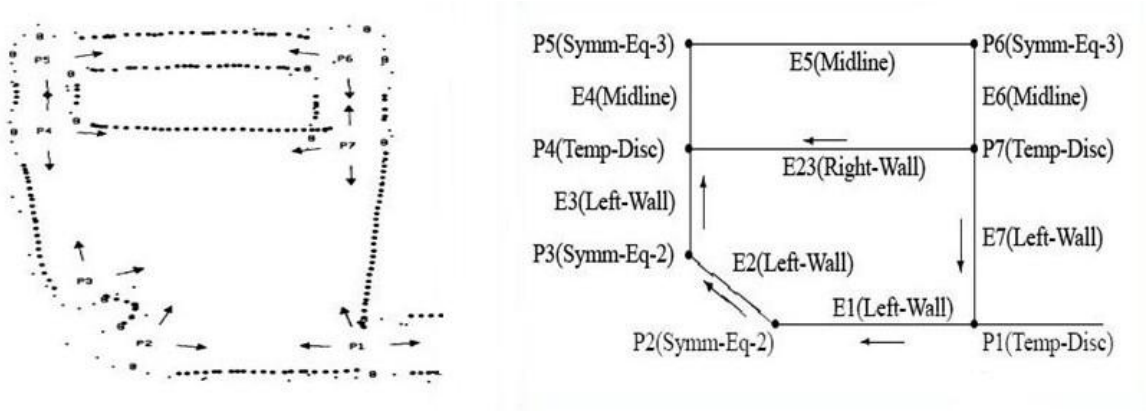

Abbildung 3.6: Geometrische Map (links) und die zugehörige topologische Map (rechts) [5]

3.6 Positionsbestimmung

Der Aufbau eines Weltmodells ist nur dann möglich wenn ein Fahrzeug zu jedem Zeitpunkt seine Position in der Umgebung kennt. Um die, durch die Sensorik detektierte, Umgebungsmerkmale korrekt in eine metrische Karte einzeichnen zu können, muss die Position eigenständig berechnet bzw. bestimmt werden. Man spricht dabei von der Selbstlokalisierung des Fahrzeugs. Eine Lokalisierung ist eine Berechnung der Position und der Orientierung in Bezug auf das Weltkoordinatensystem und ist durch den Vektor:

[5] Kuipers (2000), S.200

$$p\vec{o}s_1 = \begin{pmatrix} x \\ y \\ \varphi \end{pmatrix}$$

gekennzeichnet [6].

Generell wird in der Fachliteratur zwischen zwei verschiedenen Selbstlokalisierungsmaßnamen unterschieden: der absolute und der relative Selbstlokalisierung [Gutmann (1996), S.1-10]. Bei der absoluten Selbstlokalisierung ist die Position am Anfang unbekannt. Das Fahrzeug beinhaltet jedoch eine metrische Karte, welche die Umgebung beschreibt. Das System entscheidet dabei eigenständig über die Position indem die Umwelt mittels der Fahrzeugsensorik analysiert wird und bestimmte Umweltmerkmale in der metrischen Karte wiedererkannt werden. Solch eine Positionsbestimmung hat jedoch zum einen den Nachteil, dass dem Fahrzeug eine Karte zur Verfügung gestellt werden muss an der sich das Fahrzeug orientieren kann und zum anderen den Nachteil des Rechenaufwands, da je nach Größe des Suchraumes entsprechend viel Rechenzeit benötigt wird.

Alternativ wird deshalb die relative Positionsbestimmung verwendet. Dabei wird dem Fahrzeug beim Start eine Position mitgegeben, an der es sich orientiert. Durch die Odometrie (Wegmessung) wird dann der gefahrene Weg bestimmt und die neue Fahrzeugposition errechnet. Werden zu einem späteren Zeitpunkt bestimmte Merkmale wiedererkannt, wird nur noch die Positionskorrektur durch das System berechnet.

Die Problematik der Weltmodellierung liegt darin, dass die Umgebung bekannt sein muss, damit die Position korrekt bestimmt werden kann. Anderseits braucht man die korrekte Position um eine Karte möglichst genau modellieren zu können. Es gilt also das in der Fachliteratur bezeichnete Paradoxon "`Simultaneous Localization and Mapping`" (SLAM) zu lösen [Julier und Uhlmann (2003)].

[6] [Robotics and Process Control Research Group]

4 Analyse

In diesem Kapitel werden grundlegende Anforderungen an die Weltmodellierung im Bezug auf das CaroloCup-Fahrzeug aufgestellt. Es wird auf den Einsatz der Kartographie im Fahrzeug eingegangen, die Algorithmen und Methoden der Weltmodellierung vorgestellt, sowie grundlegende Probleme der Kartenmodellierung, der Kartenskalierung wie auch der Positionsbestimmung gelöst.

4.1 Aufgaben und Anforderungen

Ein großes Problem des CaroloCup-Fahrzeugs ist, dass sich das momentan implementierte System nur anhand von Kameradaten auf der Strecke orientiert. Dabei wird mit einer Kamera mittels TFALDA Algorithmus die Streckenführung erfasst und der Regeleinheit -welche die Geschwindigkeit und die Lenkung steuert- in verpacktem Datenformat zu Verfügung gestellt. Die Fahrspurerfassung durch den TFALDA Algorithmus wird in der Studienarbeit von [Berger (2008)] erläutert und beschrieben. Die Fahrzeugregelung anhand der Daten, wird in der Arbeit von [Reimers (2008)] implementiert.

Die Schwierigkeit des Verfahrens ist jedoch, dass das Fahrzeug immer auf die Daten der Kamera angewiesen ist. Fällt die Kamera aus oder wird die Strecke durch die Kameraeinheit nicht mehr richtig erkannt, so weis die Steuereinheit des Fahrzeugs nicht, wie das Fahrzeug zu steuern ist.

In der [Tab. 4.1] sind einige Fehlerszenarien dargestellt, die zu einem Versagen des Systems führen können. Besonders der letzte Punkt der Tabelle zeigt eine gravierende Schwäche des CaroloCup-Fahrzeugs auf und stellt dadurch eine der Anforderungen an die Weltmodellierung dieser Arbeit.

Fehlerszenarium	Problematik	Konsequenz
schlechte Lichtverhältnisse	inkorrekte Streckendaten	Totalausfall d. autonomen Fahrt
Sonneneinstrahlung	fehlerhafte TFALDA Daten, inkorrekte Streckendaten	Fahrzeugregelung in der Zeit τ nicht möglich
Kameraausfall	keine Kameradaten	Totalausfall d. autonomen Fahrt
Ultraschall-Sensorausfall	Hindernisdaten beschränkt vorhanden	Ausweichen nur mittels Linienlaserdaten möglich
Linienlaserausfall	Hindernisdaten beschränkt vorhanden	Ausweichen nur mittels Ultraschalldaten möglich
Abkommen von der Strecke	TFALDA Daten fehlerhaft	Totalausfall d. autonomen Fahrt

Tabelle 4.1: Fehlerszenarien beim CaroloCup Fahrzeug

Die Idee der Kartierung ist dem Fahrzeug Umgebungsinformationen aus einer globalen geometrischen Karte zur Verfügung zu stellen, damit das Ausfallen bestimmter Sensorik kompensiert werden kann. Da auch die Position des Fahrzeugs im Weltmodell bekannt sein soll, ist es gewünscht, dass das CaroloCup-Fahrzeug nach Abkommen von der Strecke, autonom unter Zunahme der Karteninformation, den Weg zurück zur Strecke findet. Die Aufgaben, welche von der Weltmodellierung dabei erledigt werden müssen um dem Fahrzeug Informationen zur Verfügung zu stellen, sind in der [Tab. 4.2] aufgelistet.

Aufgaben	Umsetzung
Hinderniserfassung	sensorbasiert
lokale Weltmodellierung	sensorbasiert, kartenbasiert (geometrisch)
globale Weltmodellierung	kartenbasiert (geometrisch)
Positionsbestimmung	sensorbasiert
Positionskorrektur	sensorbasiert, kartenbasiert (geometrisch,topologisch)

Tabelle 4.2: Aufgaben der Weltmodellierung

Die Weltmodellierung wird in zwei Phasen eingeteilt, wobei die lokale Weltmodellierung die erste und die Globale die zweite Phase darstellt.

4.2 Lokale Kartierung

Bei der lokalen Weltmodellierung baut sich das Weltmodell aus den Daten der Sensordatenverarbeitung auf. Lokale Karten sin als Momentaufnahme aus der Sicht des Fahrzeugs zu betrachten, in denen zusammengefasste Sensorinformationen bezüglich der Hindernisse und der Streckenführung zu einem Zeitpunkt τ in Form einer geometrischen Karte abgespeichert werden. Anders als bei der reinen Sensorkarte, werden nur die fusionierten Daten in das geometrische Modell aufgenommen. Das lokale Koordinatensystem orientiert sich an der Fahrzeugposition, die den Koordinationsursprung darstellt. Dieser ist für die lokale Karte immer auf den Punkt $P(0, 0)$ festgelegt [Abb. 4.1]. Die Hauptfunktion der lokalen Kartie-

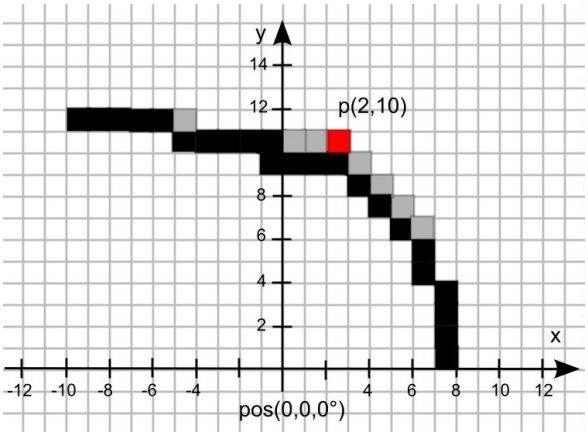

Abbildung 4.1: Streckenführung in einer lokalen Karte

rung ist die fusionierten Sensordaten in ein einheitliches Datenformat zu verpacken. Dieses Datenformat muss ebenfalls von der globalen Weltmodellierung interpretiert werden können, sodass ein einheitliches Kartenformat zwischen den Kartentypen herrscht. Da die Einsatzumgebung auch nicht polygonale Formen besitzt (Kurve in der Strecke) wird für die Umsetzung ein Rastermodell verwendet. Dabei wird der Vorteil der Rasterzellen genutzt, durch die das gewünschte homogene Datenformat, der Kartendaten, schon vorab gegeben ist. Die Rasterzellen werden beim CaroloCup-Fahrzeug von drei Arten der Sensorik initialisiert [Tab. 4.3].

Sensor	Art
Ultraschalsensor	Hinderniserfassung
Linienlaser	Hinderniserfassung
TFALDA Algorithmus	Streckenerfassung

Tabelle 4.3: Sensorik der Weltmodellierung

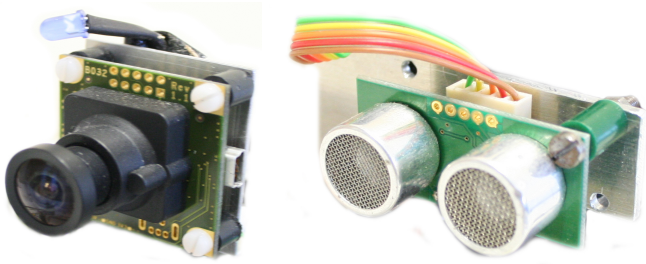

Abbildung 4.2: Eingesetzte Sensorik für das lokale Mapping

Anders als beim Rasterverfahren, welches in [M. Beckerman (1990)] erläutert wird, hat man sich beim CaroloCup-Fahrzeug dazu entschieden nur die Zellen abzuspeichern, welche belegt bzw. mit einer gewissen Wahrscheinlichkeit belegt sind. Freie Zellen werden in der lokalen Karte nicht protokolliert, um die Datenmenge zu reduzieren. Eine Rasterzelle ist in dem System als Trippel (x, y, λ) definiert. Wobei x, y die Koordinaten der Rasterzelle im lokalem Koordinatensystem darstellen und λ den Wahrscheinlichkeitsgrad der Belegung. Die Größe einer Rasterzelle ist variabel und wird in Zentimetern angegeben. Als Standard für das CaroloCup-Fahrzeug wurde die Zelle auf 10x10cm festgelegt.

Das Ziel des Wahrscheinlichkeitsparameters $\lambda)$ ist die dynamische Umstrukturierung des Kartenmodells zu gewährleisten. So können sich Sensordaten, die dieselbe Rasterzelle markieren wiedersprechen, wodurch die Zelleneigenschaft nicht genau definiert ist. Fährt das CaroloCup-Fahrzeug denselben Streckenbereich noch einmal ab und stellt durch die Sensordatenverarbeitung fest, dass die Rasterzelle nun doch als belegt zu markieren ist, wird der Zustand der alten Zelle mit dem Zustand der neuen Zelle aktualisiert. Dazu muss die lokale Kartierung die Möglichkeit bekommen, die zuvor kartographierten Umgebungsinformationen für den gleichen Bereich wie den aktuellen einzulesen und zu aktualisieren. Diese Fähigkeit muss die globale Weltmodellierung der Lokalen bereitstellen.

Folglich wird die lokale Karte L des CaroloCup-Fahrzeugs durch die Menge der aktuellen Sensordaten M_{cur} und der Menge der alten Sensordaten M_{old} des gleichen Umgebungsbereiches, in Form von Rasterzellen R definiert.

$$L = M_{cur} \cup M_{old}$$

Die genauere Analyse der lokalen Kartierung, sowie der Sensordatenfusionierung wird in der Arbeit [Ebert (2008)] beschrieben und implementiert. Die globale Kartierung dieser Arbeit baut dabei auf die dort implementierte lokale auf.

4.3 Globale Kartierung

In der zweiten Phase der Weltmodellierung werden die lokalen Rasterkarten zu einem globalen Gesamtmodell fusioniert. Man spricht auch von einer schrittweisen Konstruktion der Karte. Der Informationsgehalt der Karte hängt von der Anzahl der lokalen Karten ab. Es muss jedoch geklärt werden, wie oft die lokalen Karteninformationen in das globale Weltmodell integriert werden müssen, um die Genauigkeit der Karte zu gewährleisten. Eine globale geometrische Map G, wird als Menge der lokalen Karten definiert.

$$G = \{L_i, L_{i+1}, \ldots, L_n\}$$

Der Koordinatenursprung liegt mitten in der Umgebung. Von diesem baut sich die Karte, je nach Positionierung des Fahrzeugs, auf. Als Orientierungsachse für das Fahrzeug wird die y-Achse des globalen Koordinatensystems [Abb. 4.3] verwendet. Um die Konsistenz einer

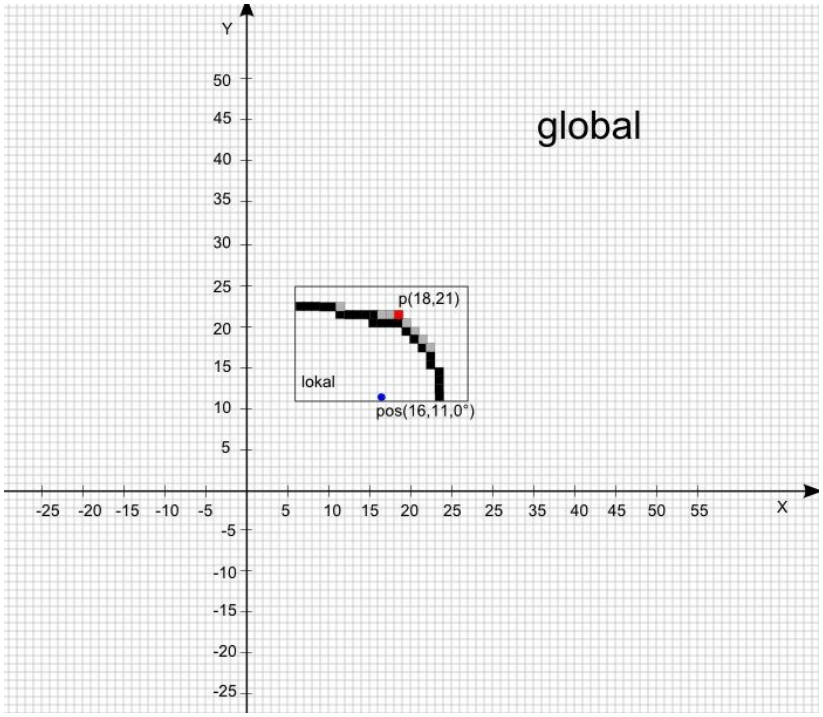

Abbildung 4.3: Globale Karte mit lokaler Integration

globalen Karte zu gewährleisten müssen einige Anforderungen berücksichtigt werden. So darf ein erfasstes Hindernis in der Umgebung auch nur einmal in der globalen Karte vorhanden sein, ungeachtet ob dieses aus einem anderen Winkel oder von einem anderen Sensor

erfasst wurde. Dadurch hängt die Integration der lokalen Kartendaten von der momentanen Fahrzeugposition und Fahrzeugorientierung in der Umgebung ab.

Auch spielt die Skalierbarkeit der Karte eine bedeutende Rolle. Wird eine statische globale Weltmodellierung verwendet, in der die Kartengröße vorab definiert wird, so riskiert man bei einer geringen Kartengröße das Überschreiten des Kartenbereichs. Wird die Größe der Karte jedoch zu hoch gesetzt, so wird die Karte sehr speicherintensiv. Aus diesem Grund wird beim CaroloCup-Fahrzeug eine dynamische globale Weltmodellierung entwickelt, wodurch die Kartengröße mit der Anzahl der lokalen Karten zunimmt. [Abb. 4.4] visualisiert den Unterschied einer statischen und einer dynamischen Weltmodellierung.

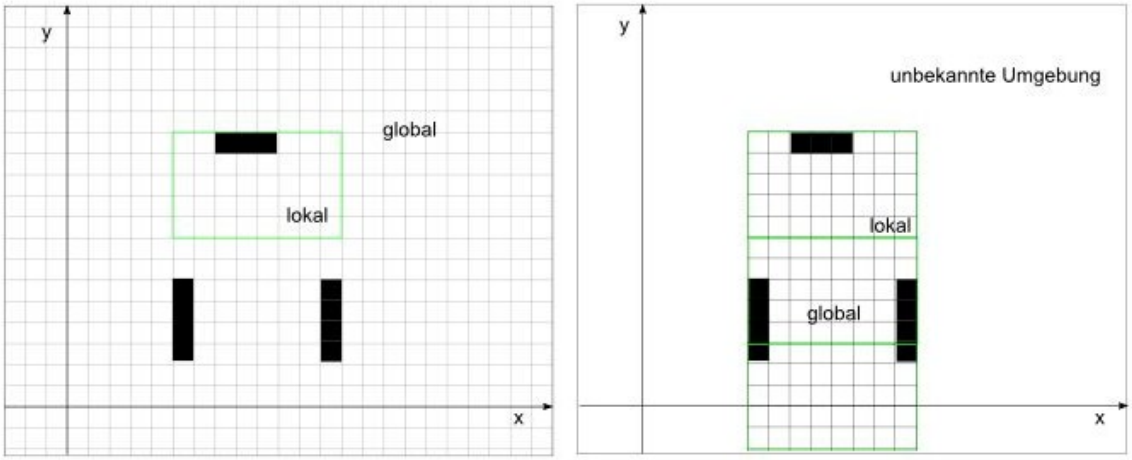

Abbildung 4.4: Statischer Kartenaufbau (links) und dynamischer Kartenaufbau (rechts)

4.3.1 Suksessive Kartenkonstruktion

Die Sukzessive Kartenkonstruktion löst die Anforderung der permanenten Kartenerzeugung, indem die globale Karte erst bei einer Positionsänderung des Fahrzeugs aktualisiert wird. Ändert sich die Positionierung bzw. die Orientierung des Fahrzeugs nicht, so wird keine Integration der lokalen Kartendaten in das globale Weltmodell vorgenommen. Dabei pausiert die globale Weltmodellierung und verschwendet keine Rechenressourcen. Um dieses Verhalten umzusetzen, wird die globale Weltmodellierung als ein Listenermodul betrachtet, welches auf die Positionsdaten wartet. Werden die Positionsdaten vom System neu berechnet, so wird die globale Weltmodellierung darüber informiert. Dabei werden die von der lokalen Weltmodellierung erzeugten Umgebungsinformationen zu der aktuellen Position geladen und in eine globale Karte integriert. Die Position des Fahrzeugs dient als Koordinatenindex für die Positionierung der lokalen Karte im globalen Weltmodell. Durch dieses Verfahren wird die globale Karte schrittweise aufgebaut und wird mit zunehmender Zeit genauer.

4.3.2 Kartenfusionierung

Bei der Fusionierung einer lokalen Karte in ein globales Weltmodell werden die Rasterzellenkoordinaten der aktuellen Karte auf Rasterzellen des globalen Koordinatensystems abgebildet. Dazu gilt es -wie [Abb. 4.5] offenbart- herauszufinden, wo sich die Rasterzellen befinden, die in Relation zu den globalen Zellen stehen. Da die Positionierung des CaroloCup Fahrzeugs immer dem Koordinationsursprung einer lokalen Karte entspricht, können die Quellkoordinaten (x_q, y_q) der Rasterzellen durch die direkte Methode der Transformation in Zielkoordinaten (x_t, y_t) des globalen Weltmodells überführt werden.

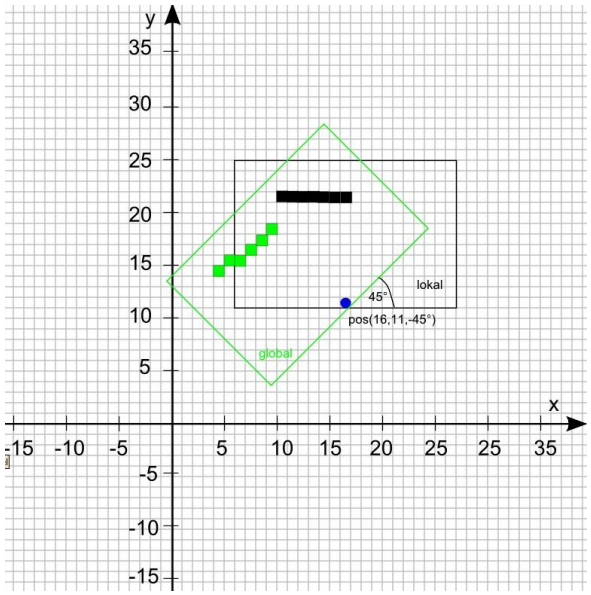

Abbildung 4.5: Kartenfusionierung

Für die Rotation der Koordinaten gilt die Transformationsmatrix:

$$\begin{pmatrix} x_t \\ y_t \end{pmatrix} = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix} * \begin{pmatrix} x_q \\ y_q \end{pmatrix}$$

Dabei stellt der Winkel Alpha die Orientierung des Fahrzeugs dar. Da die Koordinaten zusätzlich um die Positionskoordinaten (x_p, y_p)verschoben werden müssen, gilt die Transformationsmatrix:

$$\begin{pmatrix} x_t \\ y_t \end{pmatrix} = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix} * \begin{pmatrix} x_q \\ y_q \end{pmatrix} + \begin{pmatrix} x_p \\ y_p \end{pmatrix}$$

Zu einer Rasterzelle gibt es nur diskrete Punktpositionen, somit müssen die Koordinaten

(x_t, y_t) zum Schluss nach $(\widetilde{x_t},\widetilde{y_t})$ durch Rundung überführt werden. Es gilt:

$$\begin{pmatrix} \widetilde{x_t} \\ \widetilde{y_t} \end{pmatrix} = \begin{pmatrix} round(x_t) \\ round(y_t) \end{pmatrix}$$

Bei der oben dargestellten Transformationsmatrix wird für die Rotation der Rasterzellen die x-Achse als Orientierungsursprung genommen. Bei dem CaroloCup-Fahrzeug wird jedoch die y-Achse als Ursprung genommen [Abb. 4.6].

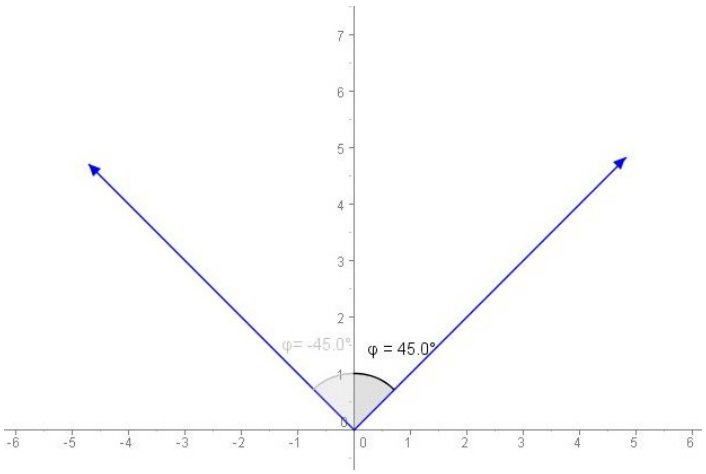

Abbildung 4.6: Fahrzeugorientierung

Dadurch ergibt sich eine Transformationsmatrix von:

$$\begin{pmatrix} x_t \\ y_t \end{pmatrix} = \begin{pmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{pmatrix} * \begin{pmatrix} x_q \\ y_q \end{pmatrix} + \begin{pmatrix} x_p \\ y_p \end{pmatrix}$$

Dieses Verhalten erreicht man auch indem der Winkel φ negiert wird:

$$\begin{pmatrix} x_t \\ y_t \end{pmatrix} = \begin{pmatrix} \cos(-\varphi) & -\sin(-\varphi) \\ \sin(-\varphi) & \cos(-\varphi) \end{pmatrix} * \begin{pmatrix} x_q \\ y_q \end{pmatrix} + \begin{pmatrix} x_p \\ y_p \end{pmatrix}$$

Dabei werden die Rasterzellen der globalen Karte durch die neuen Zellen überschrieben.Für die Kartenfusionierung ergibt sich der [Algorithmus 1].

Algorithm 1 Kartenfusionierung

Require: Rasterzellen d. lokalen Karte: P_i, $Pos(x, y, \varphi)$
Ensure: globale Umrechnung von P_i
  for $i = 1$ to $n - 1$ **do**
    $P_i.x \Leftarrow P_i.x * \cos Pos_\varphi + P_i.y * \sin Pos_\varphi + Pos_x$
    $P_i.y \Leftarrow P_i.x * -\sin Pos_\varphi + P_i.y * \cos Pos_\varphi + Pos_y$

    $P_i.x \Leftarrow round(P_i.x)$
    $P_i.y \Leftarrow round(P_i.y)$

    $insertIntoGlobalMap(P_i)$
  end for
  return

4.3.3 Skalierung der Karte

Die Problematik der globalen Rasterkarte ist, dass das Weltmodell sehr speicherintensiv ist. Werden bei einer 100x100m großen Karte die Rasterzellen auf die Größe von 10x10cm festgesetzt und alle Rasterobjekte -sowohl die Besetztmarkierten, als auch die Freimarkierten- abgespeichert, so müssen bei solch einer Konstellation 1.000.000 Objekte angelegt und verwaltet werden. Stellt man sich nun vor, dass die Karte nur ein Hindernis mit einer Größe von 50x50cm aufweist, wird sofort deutlich, dass solch einer Karte ineffizient ist. Des Weiteren hat man die Schwierigkeit, dass am Anfang der Kartierung keine Umgebungsgröße bekannt ist, wodurch sich die Frage stellt, wie groß ein Weltmodell überhaupt sein darf um bei einer Kartierungsfahrt nicht an die Grenzen des Weltmodells zu stoßen. Bei der Skalierung der Karte gilt folglich herauszufinden, wie Skalierungsgrenzen aufgehoben werden können und wie Rasterobjekte optimal zu verwalten sind.

Bei der hier entwickelten Weltmodellierung werden die Anforderungen durch zwei Ansätze gelöst. Zunächst werden, wie schon bei der lokalen Karte, nicht alle Rasterzellen in das globale Weltmodell aufgenommen. Es werden nur die als belegt bzw. mit einer gewissen Wahrscheinlichkeit belegt markierten Rasterzellen der lokalen Karte in die globale überführt. Dabei werden die globalen Rasterzellen dynamisch erzeugt und nicht bei der Initialisierung der globalen Karte. Wird bspw. festgestellt, dass zu einer umgerechneten lokalen Rasterzelle keine globale vorhanden ist, wird ein globales Rasterzellenobjekt mit den Daten der zugehörigen lokalen Zelle erzeugt. Vorhandene Objekte werden aktualisiert. Für die freien Rasterzellen, werden keine Objekte angelegt, wodurch die Speicherintensität deutlich abnimmt. Eine Karte kann somit auch als eine Liste von belegten Zellen betrachtet werden [Abb. 4.7].

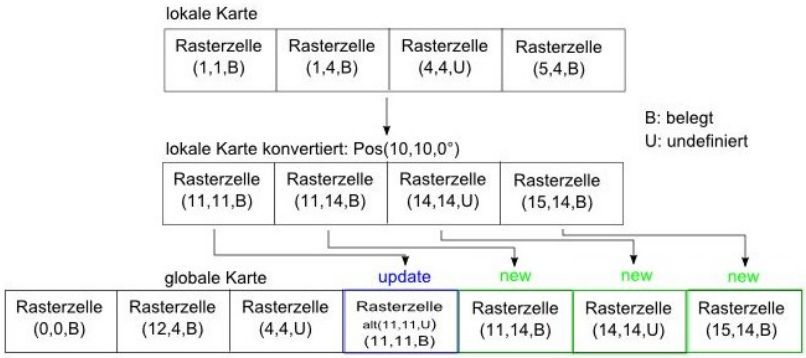

Abbildung 4.7: Verwaltung der globalen Rasterzellen

Beobachtet man das obere Beispiel, so werden bei einem 50x50cm großen Hindernis, nur 25 Zellobjekte angelegt, was eine erhebliche Reduzierung darstellt.

[Abb. 4.7] veranschaulicht ebenfalls eine dynamische Rasterobjektverwaltung. Dabei werden die lokalen Koordinaten zunächst in globale umgerechnet und dann in das globale Weltmodell fusioniert. Während die Rasterobjekte $O_2(11, 14, B)$,$O_3(14, 14, U)$,$O_4(15, 14)$ in der globalen Karte neu erzeugt werden, wird $O_{global}(11, 11, U)$ der globalen Karte durch ein neues umgerechnete lokale Rasterzellenobjekt $O_1(11, 11, B)$ modifiziert.

Durch das Konzept der Zellkoordinaten kann die Weltmodellierung immer die Hindernisse im globalen Weltmodell lokalisieren und diese anderen Modulen zur Verfügung stellen.

Der oben beschriebene Ansatz löst zwar das Problem der Speicherverschwendung, sowie der Skalierung (keine Kartengrenzen), erzeugt jedoch das Problem der Rechenintänsität. Will man einen bestimmten Bereich der globalen Karte nach Hindernissen durchsuchen, so müssen aus allen Rasterzellobjekten die gewünschten Zellen gefiltert werden, wodurch auch das Visualisieren eines globalen Weltmodells verkompliziert wird. Um auch diese Problematik zu lösen, wird im zweiten Ansatz das globale Weltmodell in Boxen unterteilt. Der Koordinatenursprung einer Box entspricht dabei Punkt P im globalen Koordinatensystem, wodurch eine Box einen globalen Bereich der Umgebung repräsentiert. Die Boxgröße wird am Anfang der Kartierung statisch durch den Nutzer festgesetzt. Die Rasterzellen der Boxen, sowie die Boxen selber werden jedoch weiter dynamisch angelegt. Für die Boxgröße gilt:

$$globalMap \geq globalBox \geq localMap$$

Durch die Boxkoordinaten und die Größenparameter der Box, kann das Boxelement, sowie dessen Rasterzellen im gesamten Weltmodell eindeutig identifiziert werden. Für die Identifizierung des gerade befahrenen Bereichs wird die Fahrzeugposition genutzt. Soll bspw. die momentan aktive Umgebung nach bestimmten Hindernissen durchsucht werden, müssen

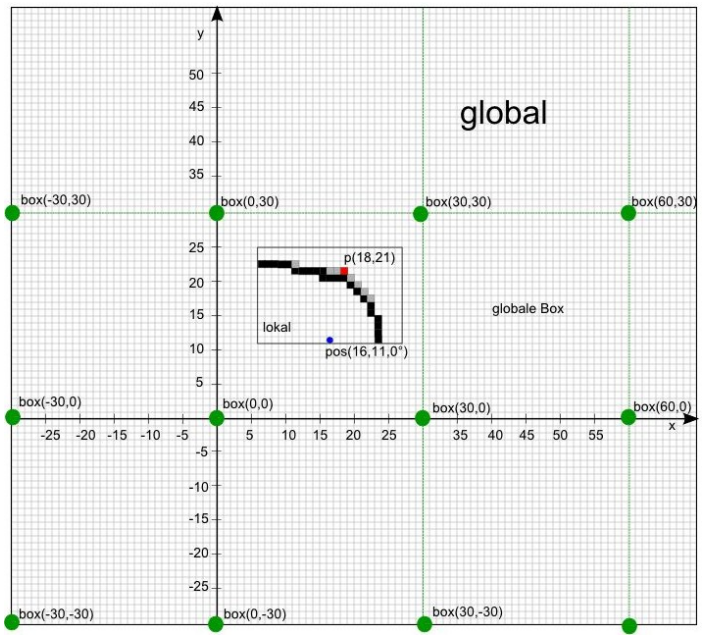

Abbildung 4.8: Globale Boxen in einem globalen Weltmodell

nicht mehr alle Zellen des gesamten Weltmodells analysiert werden, sondern nur die Zellen der globalen Box in der sich das Fahrzeug befindet. Durch Hinzunahme von Nachbarboxen kann auch ein größerer Bereich untersucht werden [Abb. 4.8].

Da eine lokale Karte bei der Kartenfusionierung auch zwischen vier Boxen liegen kann, wird in jeder globalen Box eine Verlinkung zu den Nachbarboxen angelegt, wodurch sich die Rasterzellen der lokalen Karte wie in [Abb. 4.9] auf die Boxen verteilen lassen. Der Pseudocode im [Algorithmus 2] beschreibt die Verteilung der globale Rasterzellen P_i auf die Boxen. Es werden nur die Nachbarboxen der aktuell-selektierten Box analysiert. Die Parameter boxHeight und boxWidth geben die Boxgröße an. Die Boxen sind wie in [Abb. 4.9] angeordnet.

Die Dynamik der globalen Weltmodellierung bleibt bei der boxorientierten Umsetzung erhalten. Ein globales Weltmodell ohne Daten beinhaltet keine Box. Wird beim fusionieren der Daten festgestellt, dass eine benötigte Box fehlt, so wird diese durch die globale Weltmodellierung angelegt. Wird der Wertebereich einer Box verlassen, werden auch Nachbarboxen dynamisch erzeugt und bei anderen Boxen, welche an diese angrenzen, registriert. Der Aufbau der Boxen wird nicht vom Nutzer, sondern vom System gesteuert, indem die globalen Boxen durch die Weltmodellierung verwaltet werden.

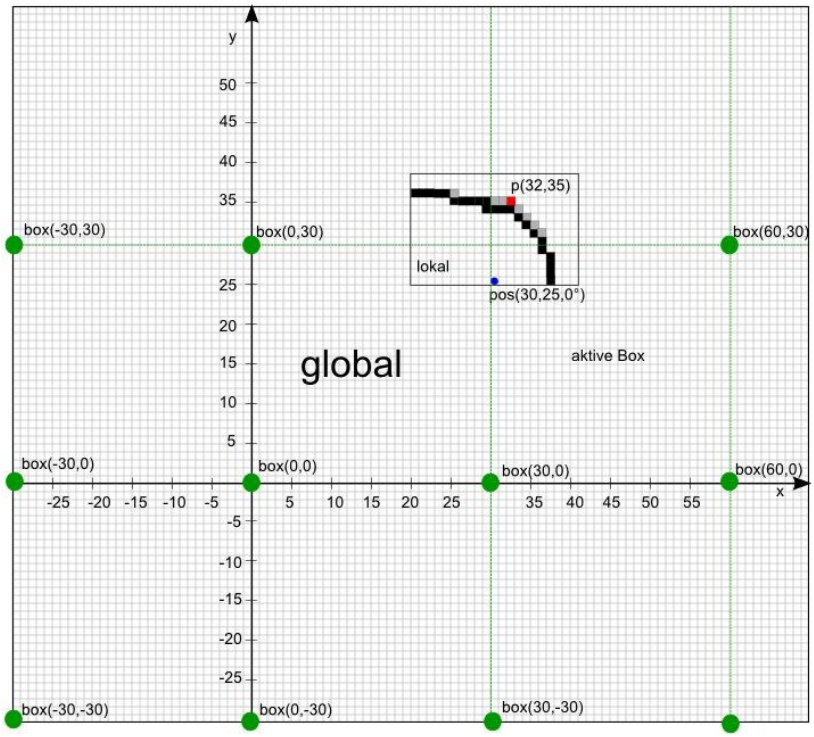

Abbildung 4.9: Rasterzellenverteilung auf vier globale Boxen

Damit die dynamische Umsetzung funktioniert, werden die Koordinaten (x_b, y_b)der globalen Boxen als Schlüsselwerte verwendet. Diese berechnen sich mittels der Positionskoordinaten (x_p, y_p) wie folgt:

$$x_b = round(x_p/boxWight) * boxWidth$$

$$y_b = round(y_p/boxHeight) * boxHeight$$

Diese Phasen der Skalierung erlauben eine dynamische autonome Erweiterung des Weltmodells und zwar auf eine uneingeschränkte Größe. Globale Rasterboxen gestatten zudem ein eingeschränktes Durchsuchen des globalen Weltmodells, indem nur die selektierte Box mit den entsprechenden Nachbarboxen abgefragt wird. Auch das Visualisieren globaler Karten solchen Typus gestattet sich mittels globaler Teilbereiche einfacher. Da durch die positionsorientierte Selektion der Boxen intuitiv, zu dem aktuell befahrenen Bereich, die globale Umgebungsbox visualisiert wird.Die Realisierung dieser Strategie erfolgt in [Kap. 5]

Algorithm 2 Verteilung der Rasterzellen auf Nachbarboxen

Require: umgerechnete lokale Rasterzellen P_i
Ensure:

```
for i = 1 to n − 1 do
    boxCoordinates ⇐ (0, 0)
    tmpX ⇐ 0
    tmpY ⇐ 0
    if p_i.x < currentBox.x then
        tmpX ⇐ (currentBox.x − boxWidth)
        /* Nachbarbox links */
        if p_i.y ≥ (currentBox.y + boxHeight) then
            tmpY ⇐ (currentBox.y + boxHeight)     {Nachbarbox oben}
        else if p_i.y < currentBox.y then
            tmpY ⇐ (currentBox.y − boxHeight)     {Nachbarbox unten}
        else
            tmpY ⇐ currentBox.y                   {Nachbarbox mitte}
        end if

    else if p_i.x ≥ (currentBox.x + boxWidth) then
        tmpX ⇐ (currentBox.x + boxWidth)
        /* Nachbarbox rechts */
        if p_i.y ≥ (currentBox.y + boxHeight) then
            tmpY ⇐ (currentBox.y + boxHeight)     {Nachbarbox oben}
        else if p_i.y < currentBox.y then
            tmpY ⇐ (currentBox.y − boxHeight)     {Nachbarbox unten}
        else
            tmpY ⇐ currentBox.y                   {Nachbarbox mitte}
        end if

    else
        tmpX ⇐ currentBox.x
        /* eigene Box oder Box unten/oben */
        if p_i.y ≥ (currentBox.y + boxHeight) then
            tmpY ⇐ (currentBox.y + boxHeight)     {Nachbarbox oben}
        else if p_i.y < currentBox.y then
            tmpY ⇐ (currentBox.y − boxHeight)     {Nachbarbox unten}
        else
            tmpY ⇐ currentBox.y                   {eigene Box}
        end if

    end if
    box ⇐ (tmpX, tmpY)
    insertIntoBox(box, P_i)
end for
return
```

4.4 Positionsbestimmung und -verifikation

Wie in [Kap. 3] schon erwähnt ist eine korrekte Positionierung im Weltmodell maßgeblich für eine fehlerfreie Kartierung. Dabei ändert das Fahrzeug seine Position (x_p, y_p) wie auch die Orientierung im Bezug zu einem Weltkoordinatensystem. Folgender Vektor beschreibt eine eindeutige Position des Fahrzeugs. Der Vektor wird auch als Pose bezeichnet.

$$\vec{Pose} = \begin{pmatrix} x_p \\ y_p \\ \varphi \end{pmatrix}$$

Da das CaroloCup-Fahrzeug autonom fahren soll, werden von außen keine Positionsdaten in das System eingetragen, dadurch ist eine Selbstlokalisierung des Fahrzeugs erforderlich. Da am Anfang einer Fahrt keinerlei Information bezüglich der Umgebung vorliegt, muss die Position mittels Odometrie bestimmt werden, wodurch in dieser Arbeit eine relative Positionsbestimmung entwickelt wird.

Als Odometrie wird die Berechnung der Position aufgrund des zurückgelegten Weges während einer Zeiteinheit bezeichnet. Das Problem der Odometrie ist, dass sie zwar auf kurzen Strecken eine genaue Positionierung liefert, sich jedoch auf langen Strecken ein Fehler in der Position aufsummiert. So führen kleine Wellen, Hindernisse oder Schlupf zu Fehlern. Ein Fehlerszenarium ist in [Abb. 4.10] zu sehen.

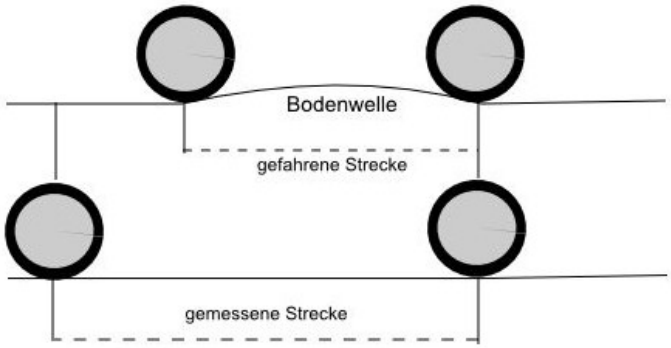

Abbildung 4.10: Odometriefehler durch Bodenwellen

Um die Nachteile zu beheben, wird die Positionsbestimmung in zwei wesentliche Teile gegliedert. Zum einen wird die Pose mittels der interner odometrischer Daten berechnet und zum anderen durch Korrekturmaßnahmen gelegentlich korrigiert und aktualisiert. Durch die Korrekturmaßnahme werden Positionsfehler möglichst klein gehalten. Auch wird durch die Kom-

bination der beiden Module teilweise das Paradoxon "`Simultaneous Localization and Mapping`" gelöst.

Folgende Anforderungen werden an die Positionsbestimmung gestellt:

- Position (x_p, y_p) in der Karte über Odometrie berechnen
- Orientierung (φ_p)in der Karte über Odometrie berechnen
- Position und Orientierung korrigieren
- "`Simultaneous Localization and Mapping`" (SLAM) ermöglichen

4.4.1 Odometriebasierte Positionsbestimmung

Die odometriebasierte Positionsbestimmung kümmert sich um die sukzessive Aktualisierung der Position auf Basis der odometrischen Daten. Problematisch bei dem CaroloCup-Fahrzeug ist, dass für die Orientierung keine interne Sensorik vorhanden ist. Dadurch muss die neue Orientierung entweder über die gefahrene Wegstrecke unter Zunahme des Lenkwinkels oder nur über die Odometrie errechnet werden. Beide Methoden werden in diesem Kapitel analysiert und beschrieben.

4.4.1.1 Positions- und Orientierungsbestimmung unter Zuhilfenahme des Lenkeinschlages

Bei dieser Methode werden außer dem Lenkwinkel, die Fahrzeuglänge und die im Mittel zurückgelegte Strecke benötigt. Wie in [Abb. 4.11] zu sehen, errechnet sich die neue Orientierung aus der Länge *l* des Fahrzeugs und den Positionskoordinaten (x_p, y_p).

$$\varphi = \arcsin(\frac{y_p}{l})$$

$$\varphi = \arccos(\frac{x_p}{l})$$

Der neue Positionswert $\vec{p}$ errechnet sich über die Summe des alten Wertes (x_p, y_p) und des Positionswertes um welchen die Position geändert wurde $(\delta x_p, \delta y_p)$.

$$\vec{p} = \begin{pmatrix} x_p \\ y_p \end{pmatrix} + \begin{pmatrix} \delta x_p \\ \delta y_p \end{pmatrix}$$

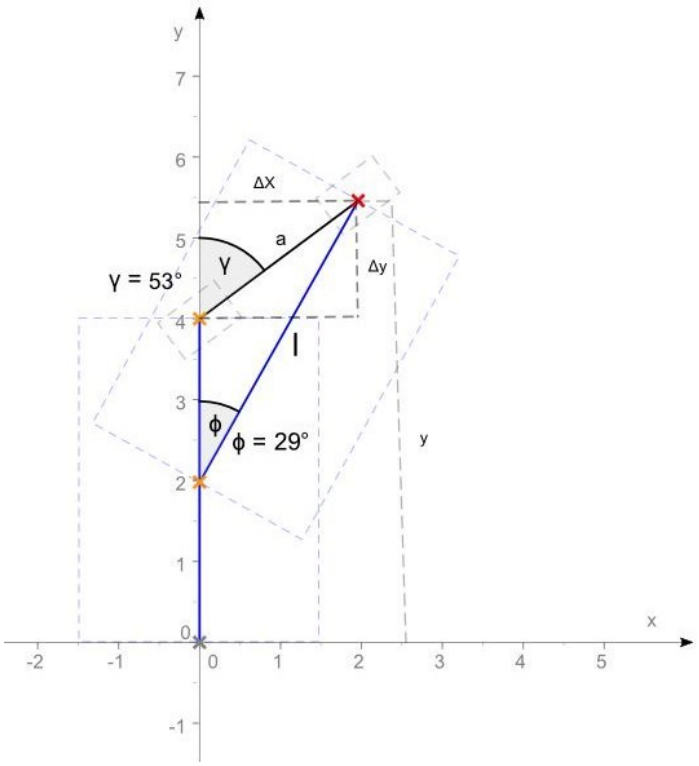

Abbildung 4.11: Positionsberechnung mit dem Lenkeinschlag

Der veränderte Positionswert $\delta P(x, y)$ kann wiederum aus der gefahrenen Strecke a und dem Einlenkwinkel γ berechnet werden. Orientiert man sich in der globalen Karte an der X-Achse so gilt:

$$\begin{pmatrix} \delta x_p \\ \delta y_p \end{pmatrix} = a * \begin{pmatrix} \sin\gamma \\ \cos\gamma \end{pmatrix}$$

In dieser Arbeit wird davon ausgegangen, dass das Fahrzeug eine Orientierung von $0°$ hat, wenn es im globalen Weltmodell senkrecht steht. Dadurch wird wie in [Abb. 4.6] gezeigt die y-Achse als Bezugspunkt für die Drehung genommen. Unter der oberen Annahme, ergibt sich dann:

$$\begin{pmatrix} \delta x_p \\ \delta y_p \end{pmatrix} = a * \begin{pmatrix} \cos\gamma \\ \sin\gamma \end{pmatrix}$$

Für den Orientierungwinkel φ gilt:

$$\varphi = \arcsin(\frac{x_p}{l})$$

$$\varphi = \arccos(\frac{y_p}{l})$$

Für die Methode der Positionsberechnung wird ein genauer Lenkwinkel vorausgesetzt. Da-

durch, dass der Lenkeinschlag beim CaroloCup-Fahrzeug in Prozent angegeben wird, muss zunächst der in Relation zu dem Prozentwert stehende Winkelwert bestimmt werden. In der Bachelorarbeit [Wandiredja (2008)] wurde der Radius für zwölf verschiedene Lenkeinstellungen des CaroloCup-Fahrzeugs bestimmt.

x_i: **Lenkwinkel [%]**	y_i: **Kreisradius [cm]**
100	81
90	84,5
80	92
70	102,5
60	105
50	119
40	150,5
-60	-186
-70	-152,5
-80	-125,5
-90	-113,5
-100	-102,5

Tabelle 4.4: Lenkwinkel des CaroloCup Fahrzeugs[7]

Durch die bekannten 12 Werte und die dazugehörigen Prozentwerte kann durch eine Ausgleichsrechnung eine Ausgleichsgerade

$$y = mx + b$$

errechnet werden. Mit deren Hilfe sich die Radiuswerte für andere Prozentwerte berechnen lassen. Die unbekannten Parameter lassen sich mit Hilfe der Determinanten Methode lösen.

Für die Ausgleichsrechnung gilt nach [Hans Rudolf Schwarz (2006), S.275]:

$$\underline{A}^T \underline{A} * \vec{\xi} = \underline{A}^T * \vec{L}$$

$$\begin{pmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & n \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} y_i * x_i \\ \sum_{i=1}^{n} y_i \end{pmatrix}$$

Zwar lässt sich durch diese Methode die neue Position wie auch die Orientierung einfach errechnen, doch ist diese Methode für das CaroloCup-Fahrzeug nicht zu gebrauchen, da für

[7] Wandiredja (2008)

die Ausgleichgerade nur 12 Werte benutzt werden. Damit ist der Fehler für jeden errechneten Lenkeinschlag enorm hoch und für die Positionsbestimmung kaum zu gebrauchen, da sich dieser pro Berechnung weiter fortpflanzt und aufsummiert wird.

4.4.1.2 Positions- und Orientierungsbestimmung nur anhand der Wegmessungen

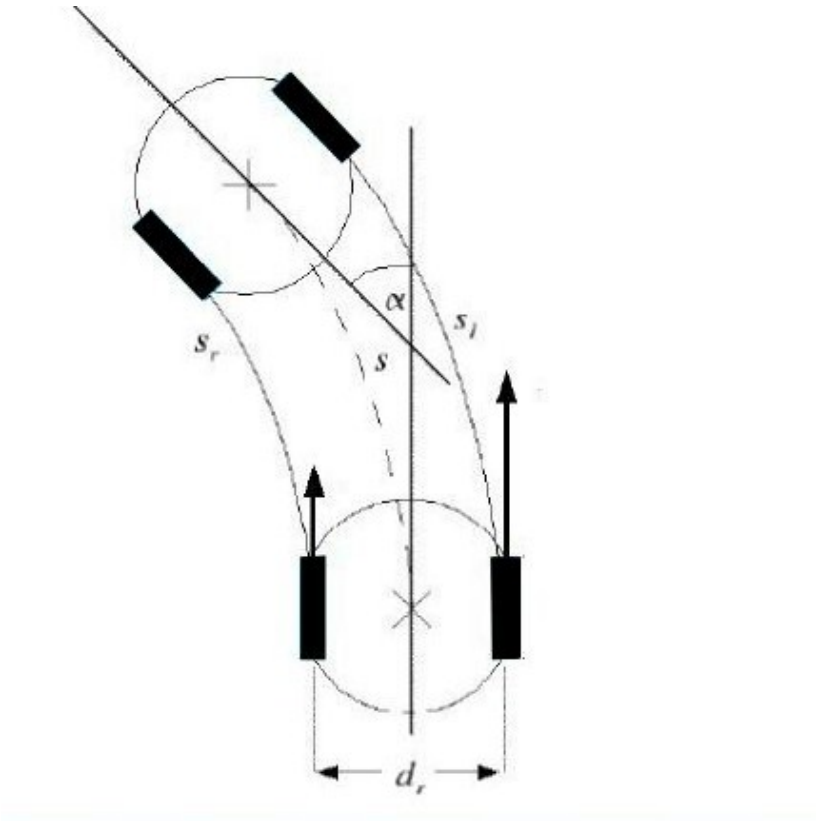

Abbildung 4.12: Vorwärtskinematik

In der Praxis hat sich für kurze Strecken eine Methode etabliert, welche die Position und die Orientierung nur über odemetrische Daten berechnet. Dazu wird vom Fahrzeug eine Wegmessung für ein linkes und ein rechtes Rad benötigt. Bei der Positionsbestimmung, wird die gefahrene Wegstrecke s_l eines Rades pro Zeiteinheit τ der Strecke des zweiten Rades s_r gegenüber gestellt. Man betrachtet die Differenz der Wegstrecken. Der Unterschied ist abhängig vom Radabstand d. Bei einer Kurvenfahrt nach rechts, gilt für beide Räder:

$$2\pi(r_{rechts}) = 2\pi(r_{links} - d)$$

Der zurückgelegte Weg für das linke Rad s_l, das rechte Rad s_r, wie auch die Wegstrecke die im Mittel zurückgelegt wurde s, lässt sich mit Hilfe des Kreisumfangs berechnen.

$$s_l = 2\pi(r + d)$$

$$s_r = 2\pi(r)$$

$$s = \frac{(s_l + s_r)}{2}$$

$$s_l - s_r = 2\pi(d)$$

Durch diese Erkenntnis lässt sich der Winkel $\delta\varphi$, um den sich die Orientierung im Koordinatensystem ändert, mittels der Odometrie bestimmen. Um die neue Orientierung ω zu bestimmen, wird der errechnete Winkel zu der alten Orientierung addiert.

$$\delta\varphi = \frac{s_l - s_r}{2}$$

$$\omega = \omega_{alt} + \delta\varphi$$

Die Positionsveränderung $\delta p\vec{o}s_k = (\delta x, \delta y)^T)$ lässt sich dank dem neuen Orientierungswinkel ω und der im Mittel zurückgelegten Wegstrecke, wie oben beschrieben, errechnen. Um die neue Pose im globalen Koordinatensystem zu bestimmen, werden zu der alten Position und der alten Orientierung die Änderungen dazu addiert. Für die neue Pose gilt:

$$po\vec{s}_{k+1} = \begin{pmatrix} x_k + s_k \sin\omega_k \\ y_k + s_k \cos\omega_k \\ \omega_k + \delta\varphi_k \end{pmatrix}$$

In der Regel ist es sinnvoll den Winkel nach der Berechnung zu normieren, sodass dieser im Intervall von $[0\pi; 2\pi]$ liegt.

Für das CaroloCup-Fahrzeug wurde die Positionsbestimmung ohne den Lenkwinkel gewählt, da der Fehler des Lenkwinkels zu hoch ist und dadurch ein schlechteres Ergebniss der Pose zustande kommt. Die odometrische Positionsbestimmung kann mit Hilfe verschiedener Methoden verfeinert werden, sodass der Fehler, welcher durch die Odometrie zustande kommt, verringert wird. So könnte man für die Orientierungsbestimmung einen Kreiselkompass -in der Fachliteratur auch Gyroskop genannt- auf dem Fahrzeug verbauen, um die Genauigkeit der Orientierungsbestimmung zu erhöhen [Borenstein und Feng (1996)]. Auch könnte man für das Fahrzeug eine Antriebsschlupfregelung entwickeln, welche das durchdrehen der Reifen unterbindet. So ein Verfahren wird in der Bachelorarbeit [Arvidsson (2008)] für das IntelliTruck Fahrzeug entwickelt.

4.4.2 Korrekturmaßnamen

Die Positionsberechnung anhand der Odometriedaten ist nur dann genau, wenn die Fahrbahn perfekt und eben ist und eine unendliche Reibung besitzt, sodass die Räder nicht durchdrehen. Da dies in den meisten Fällen nicht gegeben ist, tritt in jedem Zeitintervall ein Fehler auf, der nach einer Zeit korrigiert werden muss. Dazu wird vom Fahrzeug die Position in einer zuvor schon befahrenen Umgebung eigenständig lokalisieren, indem bestimmte Merkmale wiedererkannt werden.

Die Verfahren zur Selbstlokalisierung lassen sich grob, wie [Abb. 4.13] zeigt, in scanbasierte Verfahren, Verfahren mit geometrischer Merkmalskarte der Umgebung und Verfahren mit künstlichen Landmarken einteilen.

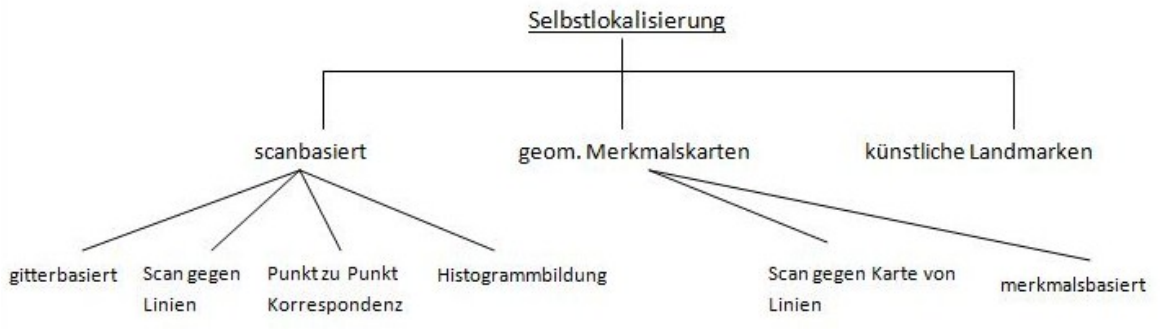

Abbildung 4.13: Möglichkeiten der Selbstlokalisierung

Das einfachste Verfahren ist die Position anhand von künstlichen Landmarken zu korrigieren. Dazu werden in der Umgebung kodierte Marken aufgestellt, die eine eindeutige Position in der Umgebung beschreiben. Kommt das Fahrzeug an einer Marke vorbei, wird die Marke durch die Sensordatenverarbeitung dekodiert und die Position in der Umgebung dem Fahrzeug mitgeteilt. Ein Vorteil solch eines Verfahrens ist die einfache Umsetzung der Positionskorrektur. Ein Nachteil ist jedoch der statische Aufbau, da das Fahrzeug immer auf die Landmarken in der Umgebung angewiesen ist. Werden die Landmarken mal entfern, so wird die Position nicht mehr korrigiert und die Umgebungskarte falsch modelliert.

Auf dem CaroloCup-Fahrzeug ist so ein Verfahren nicht erwünscht, da keine künstlichen Landmarken neben der Strecke vorhanden sind. Um dieses Verfahren zu verfeinern, muss sich die Positionskorrektur an die Umgebung dynamisch anpassen. Die dynamische Anpassung erreicht man mittels scanbasierten Verfahren. Bei der scanbasierten Selbstlokalisierung wird ein aktueller Scan mit einem zuvor aufgenommenen Scan -auch als Referenzscan bezeichnet- verglichen. Gibt es eine Übereinstimmung der beiden Scans durch das Wiedererkennen geometrischer Merkmale oder durch eine ähnliche Anordnung der Koordinatenpunkte im Scan, so muss versucht werden die Scans zu einer Überdeckung zu bringen.

Durch diese lassen sich dann sowohl die Verschiebung, als auch die Verdrehung zwischen den Aufnahmepositionen bestimmen.

Scans werden in diesem Zusammenhang auch als lokale Karten gesehen. Es gibt unterschiedliche Überdeckungsalgorithmen um die paarweise Überdeckung der Scans bestimmen zu können [Abb. 4.13]. In dieser Arbeit wird nur auf die Punkt zu Punkt Überdeckung eingegangen, da der Vorteil darin liegt, dass sich der Algorithmus auch in einer nichtpolygonalen Umgebung einsetzen lässt. Ziel ist, das Konzept so zu gestalten, dass sich auch Kurven auf der Strecke als Wiedererkennungsmerkmale nutzen lassen.

Unter der Kenntnisnahme der Methoden werden folgende Anforderungen an das CaroloCup-Fahrzeug für die Positionskorrektur gestellt:

- Scans mit markanten Umgebungsmerkmalen sollen extrahiert und zwischen gespeichert werden
- Wiedererkannte Umgebungsmerkmale sollen den Referenzscan zum aktuellen Scan liefern
- Überdeckung der beiden Scans muss bestimmt werden.
- Verschiebung und Verdrehung soll errechnet werden.
- Position und Orientierung sollen anhand der Verschiebung und Verdrehung korrigiert werden.

4.4.3 Ablauf der Positionskorrektur

Die Merkmale der Umgebung wurden beim CaroloCup-Fahrzeug auf folgende Punkte eingeschränkt. Solche Punkte werden in dieser Arbeit als "`markante Punkte`" bzw. "`topologische Punkte`" gekennzeichnet.

- Startlinie
- Stopplinie
- Kreuzung

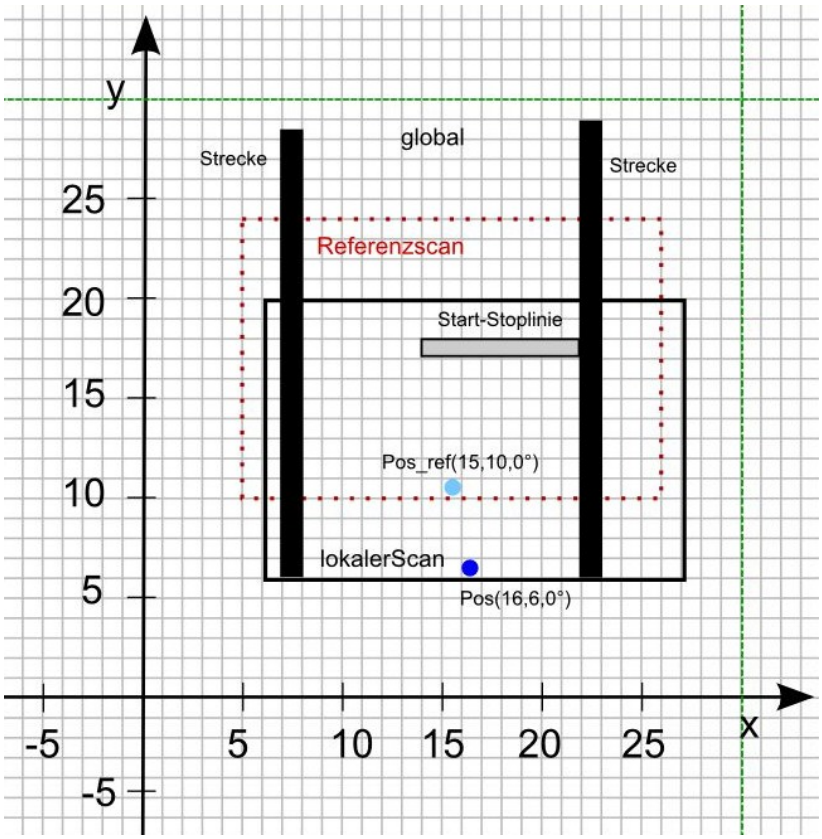

Abbildung 4.14: Referencscan in einer globalen Karte

Werden markante Punkte beim Kartographieren der Strecke zum ersten Mal erkannt, so werden zu diesen Punkten Referenzscans angelegt. Das Erkennen und Speichern solcher Punkte wird durch die Sensordatenverarbeitung und die lokale Weltmodellierung durchgeführt und wird in der Arbeit [Ebert (2008)] für das CaroloCup-Fahrzeug implementiert. Das

Speichern der Referenzscans in topologischen Punkten ist sinnvoll, da sonst der Referenzscan aus der geometrischen Karte rausgesucht werden muss, was je nach Kartengröße einen höheren Zeitaufwand darstellt. Zusätzlich zu den Referenzpunkten wird auch die Fahrzeugposition, aus der der Scan aufgezeichnet wurde, gespeichert. Dadurch ist die Referenzscanposition in der globalen Karte eindeutig kennzeichnet. In der [Abb. 4.14] wird ein Referenzscan in einer globalen Karte dargestellt. Die Fahrzeugorientierung liegt dabei bei 0°.

Wird ein markanter Punkt wiedererkannt, zu dem ein Referenzscan gespeichert ist, so muss eine paarweise Überdeckung zwischen dem aktuellen lokalen Scan, der die lokale Karte darstellt, und dem Referenzscan durchgeführt werden. Formal betrachtet ist die Funktion zur Überdeckung zweier Scans eine Funktion $match$ die zwei Scans auf eine Positionskorrektur abbildet.

$$match(S_{cur}, S_{ref}) = (\delta x, \delta y, \delta \varphi)^T$$

Dabei gibt der Vektor $(\delta x, \delta y, \delta \varphi)^T$ an, um wie viel der aktuelle Scan verschoben und gedreht werden muss, damit eine maximale Überdeckung der Scans zustande kommt.

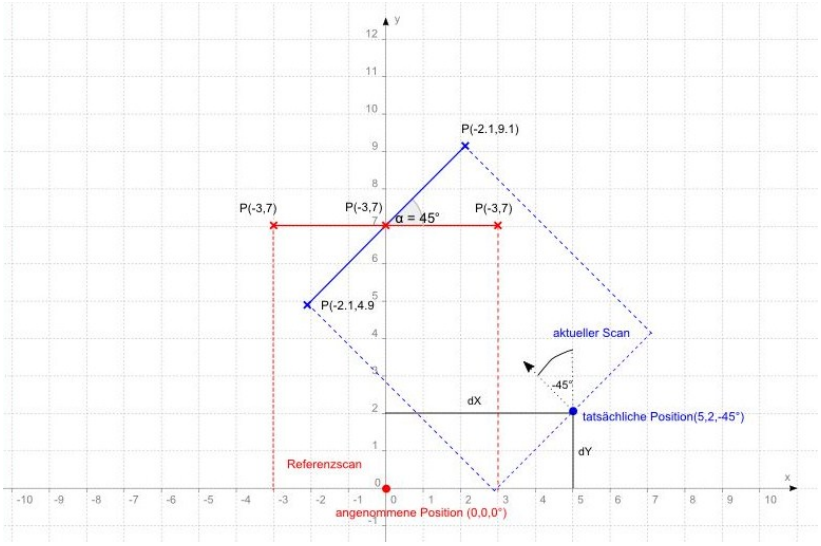

Abbildung 4.15: Lokaler Scan bei einer Fahrzeugorientierung von -45°

[Abb. 4.15] zeigt ein vereinfachtes Beispiel zweier Scans, die um einen Winkel versetzt wurden. Der Referenzscan wurde am Anfang der Fahrt bei der Pose $POS_0(0, 0, 0°)$ aufgenommen. Nachdem das Fahrzeug eine Runde gefahren ist, wurde durch die Odometrie ebenfalls eine Pose $POS_1(0, 0, 0°)$ berechnet. Wie in der Abbildung zu sehen ist diese Pose fehlerhaft, da der aktuelle Scan, der von dieser Position aufgenommen wurde um den Winkel α verdreht ist. Ziel ist es nun Winkel α zu finden, da dadurch die Orientierung des Fahrzeugs auf $-45°$ und die Positionierung auf $P(5, 2)$ korrigiert wird.

Um die paarweise Überdeckung der Scans zu bestimmen wird in dieser Arbeit der in [Lu und Milios (1997)] beschriebene, "`Iterative dual correspondence [IDC]`" Algorithmus benutzt. Da es sich um einen Punkt zu Punkt Algorithmus handelt, hat man den Vorteil, dass keine geometrischen Interpretationen der beiden Scans gebraucht werden, wodurch auch nicht polygonale Strukturen zur Überdeckung gebracht werden können.

4.4.3.1 Idee des IDC Algorithmus

Ziel des Verfahrens ist es jedem Scanpunkt des aktuellen Scans einen Punkt im Referenzscan zuzuordnen. Beide Punkte repräsentieren dann ein Korrespondenzpaar durch das sich die Transformationsabweichung bestimmen lässt. Zwei Algorithmen können für die Bestimmung der Korrespondenzpaare aus den Scans benutzt werden:

- Iterative Closest Point (ICP) (die Regel nächster Punkt)
- Finde Matching Range (FMRP) (die Regel gleiche Entfernung)

Ein Punkt ist auch als eine Rasterzelle zu betrachten.

4.4.3.2 Die Regel "`nächster Punkt`"

Bei der Heuristik "`nächster Punkt`" wird jedem Punkt im aktuellen Scan der Punkt im Referenzscan zugeordnet, welcher dem aktuellen Scanpunkt am nächsten ist. Dabei wird zwischen den Scanpunkten des Referenzscans interpoliert, wodurch alle Punkte durch kurze Liniensegmente verbunden werden. Ein Korrespondenzpartner des aktuellen Punktes muss dadurch nicht unbedingt einem Referenzpunkt entsprechen, sondern kann auch auf dem Liniensegment zwischen den Punkten (P_i, P_{i+1}) liegen, wie in [Abb. 4.16] dargestellt.

Der Algorithmus zur Bestimmung des Zuordnungspartners P_k im Referenzscan zu einem aktuellen Punkt P_a ist trivial. Dazu geht man pro Punkt P_a durch die Punkte im Referenzscan und berechnet den Abstand zum Liniensegment zwischen zwei aufeinanderfolgenden Referenzpunkten (P_i, P_{i+1}). Das Liniensegment mit dem kleinsten Abstand zum aktuellen Punkt beinhaltet den Korrespondenzpunkt P_k, der sich wie folgt berechnen lässt.

Zuerst muss die Geradengleichung $y = mx + b$ der Referenzpunkte P_i und P_{i+1} aufgestellt werden. Aus dieser lässt sich die Steigung m_{G1} wie auch der y-Achsenabschnitt b_{G1} der Gerade berechnen.

$$m_{G1} = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$$

$$b_{G1} = y_i - m_{G1} * x_i$$

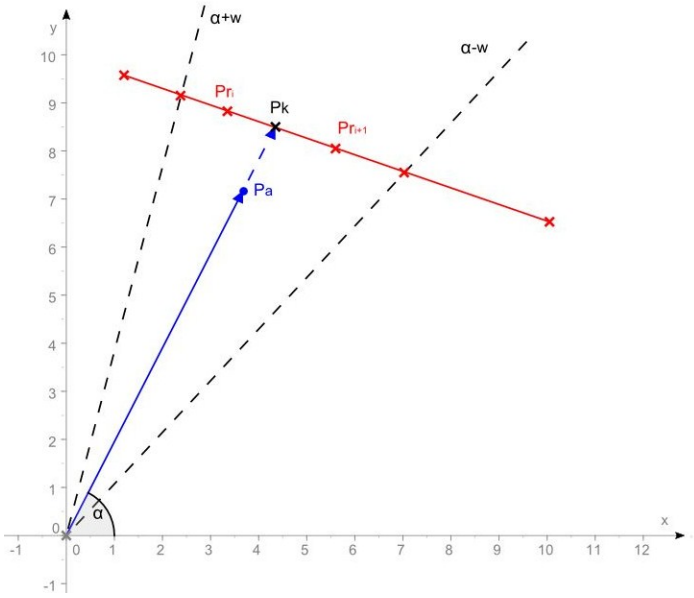

Abbildung 4.16: Winkelintervall für die Suche nach dem Korrespondenzpunkte

Um den Punkt P_k bestimmen zu können wird nach einer Geraden zwischen dem aktuellen Punkten P_a und dem Korrespondenzpartner P_k gesucht. Da diese Gerade senkrecht zu der ersten Gerade steht [Abb. 4.17], gilt für die Steigung m_{G2} der zweiten Gerade:

$$m_{G2} = \frac{-1}{m_{G1}}$$

dadurch lässt sich auch der y-Achsenabschnitt mittels

$$b_{G2} = x_a * m_{G2} + y_a$$

berechnen. Durch das Gleichsetzen der beiden Geraden kann die x-Koordinate des Punktes P_k berechnet werden. Der y-Koordinatenwert errechnet sich dann durch das Einsetzen des x-Wertes in die Geradengleichung G_2.

$$x_k * m_{G1} + b_{G1} = x_k * (-\frac{1}{m_{G1}}) + b_{G2}$$

$$x_k = \frac{b_{G2} - b_{G1}}{m_{G1} + \frac{1}{m_{G1}}}$$

$$y_k = x_k * m_{G2} + b_{G2}$$

Unter der Annahme, dass die Scans maximal um den Winkel α verdreht sind, kann der

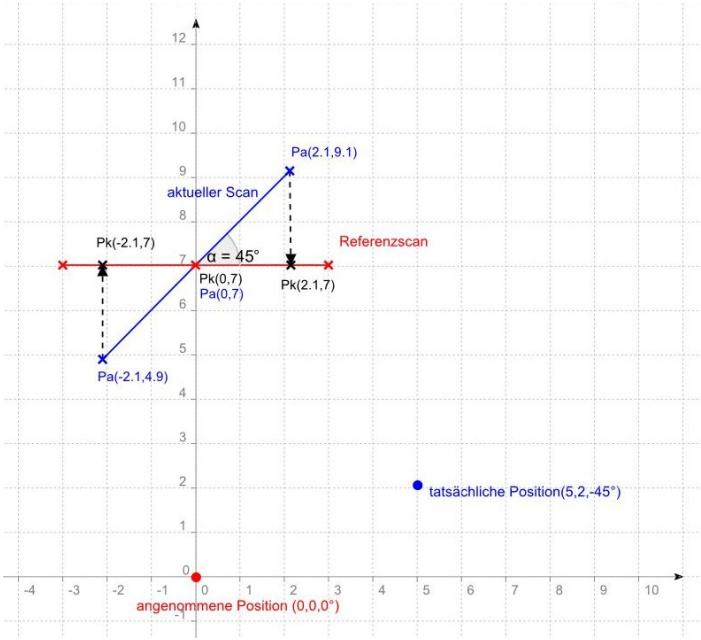

Abbildung 4.17: Korrespondenzpunkte im Referenzscan

Algorithm 3 ICP Algorithmus

Require: Referenuscanpunkte: P_i, aktueller Scanpunkt P_a
Ensure: Korrespondenzpunkt P'

```
d_New ⇐ 0
d_Old ⇐ −1
for i = 1 to n − 1 do
    d_New ⇐ getDistanceToReferenceLine(P_i, P_{i+1})
    if |d_New| < |d_Old| or d_Old = −1 then
        d_Old ⇐ d_New
        P_1 ⇐ P_i
        P_2 ⇐ P_{i+1}
    end if
    P' ⇐ getPointOfLine(P_1, P_2)
end for
return P'
```

Suchbereich im Referenzscan eingeschränkt werden. Dazu werden nur die Scanpunkte P_i des Referenzscans S_{ref} betrachtet, deren Aufnahmewinkel ϕ in einem Bereich um den Aufnahmewinkel von P_a liegt [Abb. 4.16]. Das Einschränken des Suchbereichs verringert die Suchzeit und wird deshalb auch auf dem CaroloCup-Fahrzeug umgesetzt. Es gilt der [Algorithmus 3].

4.4.3.3 Die Regel "gleiche Entfernung"

Wie der ICP Algorithmus sucht auch der Find Matching Range (FMR) Algorithmus für einen Punkt P_a im aktuellem Scan nach einem Korrespondenzpartner P_k im Referenzscan. Anders als beim ICP Algorithmus wird hier jedoch jedem aktuellem Scanpunkt P_a ein Punkt im Referenzscan zugeordnet, welcher exakt die gleiche Entfernung d zur aktuellen Aufnahmeposition Pos_{cur} besitzt. Es werden nur die Punkte des Referenzscans analysiert, welche im Winkelintervall $[\phi - \omega; \phi + \omega]$ liegen.

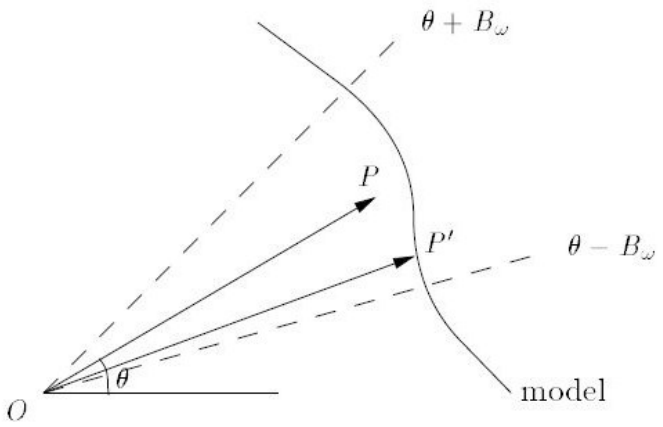

Abbildung 4.18: FMR Korrespondenzpunkt im Winkelintervall

Der Algorithmus untersucht pro aktuellen Punkt P_a, die aufeinanderfolgenden Scanpunkte $[P_i, P_{i+1}]$. Für jedes Punktepaar wird die Entfernung zur aktuellen Aufnahmeposition Pos_{cur} berechnet. Liegen beide Entfernungen d_i, d_{i+1} über oder unter der Entfernung d_a des aktuellen Punktes zur Aufnameposition, so kann mit Hilfe dieser Paare kein Punkt mit der gleichen Entfernung gefunden werden. In diesem Fall wird der Punkt des Referenzscans genommen, der näher am gesuchten Abstand d liegt. Sind die Entfernungen d_i, d_{i+1} der beiden Scans jedoch so, dass ein Wert größer und der andere kleiner ist als die Entfernung d_a im aktuellen Scan, so kann ein Korrespondenzpunkt P_k im Referenzscan zwischen den Punkten P_i und P_{i+1} linear interpoliert werden, dessen Abstand d' zur Aufnahmeposition genau dem Ab-

stand d_a von P_a zur Aufnahmeposition entspricht [Abb. 4.18]. Dazu wird laut [Lu und Milios (1997), S.16] $1/d$ von ϕ zwischen zwei Punkten linear interpoliert.

Algorithm 4 FMR Algorithmus

Require: Referenuscanpunkte: P_i, aktueller Scanpunkt P_a, Position: P_{cur}
Ensure: Korrespondenzpunkt P'

```
d ⇐ distance(P, P_cur)
d' ⇐ ∞
d'' ⇐ ∞
for i = 1 to n − 1 do
    d_1 ⇐ distance(P_i, P_cur)
    d_2 ⇐ distance(P_{i+1}, P_cur)
    if (d_1 < d AND d_2 < d) OR (d_1 > d AND d_2 > d) then
        if |d_1 − d| < |d_2 − d| then
            d_h ⇐ |d_1 − d|
            P_h ⇐ P_i
        else
            d_h ⇐ |d_2 − d|
            P_h ⇐ P_{i+1}
        end if
        if d_h < d' then
            d' ⇐ d_h
            P' ⇐ P_h
        end if
    else
        P_h ⇐ interpolate(P_i, P_{i+1}, d)
        if (d' > 0) or (distance(P_a, P_h) < d) then
            d' ⇐ 0
            d'' ⇐ distance(P_a, P_h)
            P' ⇐ P_h
        end if
    end if
end for
return P'
```

Die lineare Interpolationsfunktion $d'(\phi')$ für den Punkt $P_k(r', \phi')$ zwischen dem Punkt $P_i(d_i, \phi_i)$ und $P_{i+1}(d_{i+1}, \phi_{i+1})$ ist nach [Hans Rudolf Schwarz (2006), S. 99] definiert als:

$$d' = f(\phi') = 1/\left(\frac{1}{d_i} + \frac{\phi' - \phi_i}{\phi_{i+1} - \phi_i}\left(\frac{1}{d_{i+1}} - \frac{1}{d_i}\right)\right)$$

$$f(\phi') = \frac{d_i d_{i+1}(\phi_{i+1} - \phi_i)}{d_i(\phi' - \phi_i) + d_{i+1}(\phi_{i+1} - \phi')}$$

Da die Entfernung d' bekannt ist $[d' = d_a]$ muss der Winkel ϕ', des Korrespondenzpunktes P_k bestimmt werden. Dazu wird die Umkehrfunktion $f(\phi')^{-1}$ der oben gegebenen Interpolationsfuktion genomen.

$$\phi' = f(\phi')^{-1} = \frac{d_i d_{i+1}(\phi_{i+1} - \phi_i) + d'(d_i\phi_i - d_{i+1}\phi_{i+1})}{d'(d_i - d_{i+1})}$$

Durch den Winkel ϕ' und den Abstand d' werden die Punktkoordinaten $[x_k, y_k]$ berechnet. [Abb. 4.18] zeigt ein Beispiel für ein Zuordnungspaar (P_a, P'). P' stellt den Korrespondenzpartner für den aktuellen Punkt P_a dar, welcher denselben Abstand d_a zur Aufnahmeposition Pos_{cur} besitzt. Dieser befindet sich im Winkelintervall $[\phi - \omega; \phi + \omega]$ und ist um den Winkel $\delta\varphi$ gegenüber dem aktuellen Scanpunkt verdreht.

Gibt es im Winkelintervall des Referenzscans mehrere Punktepaare P_i,P_{i+1} mit der unterschiedlichen Entfernungsverteilung, so gibt es keinen eindeutigen Korrespondenzpunkt. In diesem Fall wird der Punkt als Korrespondenzpunkt genommen, welcher dem aktuellen Punkt am nächsten liegt. Der Pseudocode für den Algorithmus ist unter [Algorithmus 4] zu sehen.

4.4.3.4 Berechnungen der Verschiebung und Verdrehung

Nachdem die Korrespondenzpaare $P_i(x_i, y_i)$, $P_i'(x_i', y_i')$ nach einer der oberen Methoden bestimmt wurden, kann daraus durch das Minimieren einer Summe von Abstandsquadraten die Verdrehung und Verschiebung der Punkte berechnet werden. Dabei werden die Punkte des aktuellen Scans um den Winkel $\Delta\alpha$ gedreht und mit $\Delta T = (\Delta x, \Delta y)^T$ verschoben. Danach wird die Summe der quadrierten Abstände zwischen den gedrehten und verschobenen Punkten P_i und P_i' aufgestellt. Ist die Summe 0, so gibt es eine 100% Überdeckung der beiden Scans.

$$E_{fit}(\Delta\alpha, \Delta t) = \sum_{i=1}^{n} |R(\Delta alpha)P_i + \Delta T - P_i'|^2$$

$$= \sum_{i=1}^{n} \left((x_i \cos(\Delta\alpha) - y_i \sin(\Delta\alpha) + \Delta T_x - x_i')^2 + (x_i \sin(\Delta\alpha) + y_i \cos(\Delta\alpha) + \Delta T_y - y_i')^2 \right)$$

Es gilt die Rotationsmatrix:

$$R(\Delta\alpha) = \begin{pmatrix} \cos(\Delta\alpha) & -\sin\Delta\alpha \\ \sin(\Delta\alpha) & \cos\Delta\alpha \end{pmatrix}$$

Die obere Funktion kann nach [Lu und Milios (1997), S.38] minimiert werden, wodurch sich der Rotationswinkel $\Delta\alpha$ und die Verschiebung ΔT mit:

$$\Delta\alpha = \arctan \frac{S_{xy'} - S_{yx'}}{S_{xx'} + Syy'}$$

$$\Delta T = \begin{pmatrix} \tilde{x'} \\ \tilde{y'} \end{pmatrix} - \begin{pmatrix} \cos(\Delta\alpha) & -\sin(\Delta\alpha) \\ \sin(\Delta\alpha) & \cos(\Delta\alpha) \end{pmatrix} \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix}$$

berechnen lässt. Es gilt:

$$\tilde{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \qquad \tilde{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

$$\tilde{x'} = \frac{1}{n}\sum_{i=1}^{n} x'_i, \qquad \tilde{y'} = \frac{1}{n}\sum_{i=1}^{n} y'_i$$

$$S_{xx'} = \sum_{i=1}^{n} (x_i - \tilde{x})(x'_i - \tilde{x'}), \qquad S_{yy'} = \sum_{i=1}^{n} (y_i - \tilde{y})(y'_i - \tilde{y'})$$

$$S_{xy'} = \sum_{i=1}^{n} (x_i - \tilde{x})(y'_i - \tilde{y'}), \qquad S_{yx'} = \sum_{i=1}^{n} (y_i - \tilde{y})(x'_i - \tilde{x'})$$

Soll die korriegierte Position berechnet werden, wird die Position $Pos_{ref} = (x_{ref}, y_{ref}, \alpha_{ref})$, die dem Referenzscan zugeordnet ist, um $(\Delta T_x, \Delta T_y)$ verschoben. Die Orientierung wird mittels der Verdrehung korriegiert.

$$Pos_{Tkor} = \begin{pmatrix} x_{ref} \\ y_{ref} \end{pmatrix} + \begin{pmatrix} \Delta T_x \\ \Delta T_y \end{pmatrix}$$

$$Pos_{\alpha kor} = \alpha_{ref} - \Delta\alpha$$

4.4.3.5 Der IDC Algorithmus

Sowohl durch den "`iterative closest point`" Algorithmus, als auch den "`find matching range`" Algorithmus lässt sich die Verschiebung und Verdrehung der Scans berechnen. In [Lu und Milios (1997)] wurden beide Algorithmen für die Korrespondenzpaarsuche der Scans untersucht. Dabei wurde festgestellt, dass der ICP Algorithmus besser für die Bestimmung der Verschiebung und der FMR Algorithmus besser für die Bestimmung der Verdrehung geeignet ist. Der IDC Algorithmus vereint dadurch beide Algorithmen:

1. Setze Vektor für die Überdeckung $(\Delta T_x, \Delta T_y, \Delta\alpha)^T = (0, 0, 0)^T$
2. Verschiebe und drehe den aktuellen Scan um den Vektor $(\Delta T_x, \Delta T_y, \Delta\alpha)^T$. Dadurch wird ein neuer Scan angelegt auf welchem in den folgenden Schritten gearbeitet wird.
3. Bestimme für jeden aktuellen Punkt P_a den Korrespondenzpunkt P' anhand der nächsten Punkt-Regel und P'' anhand der nächsten Entfernungs-Regel
4. Bestimme durch die aktuellen Punkte P_i und die Referenzpunkte P'_i die Verschiebung und Verdrehung $(\delta T_{x_1}, \delta T_{y1}, \delta\alpha_1)^T$
5. Bestimme durch die aktuellen Punkte P_i und die Referenzpunkte P''_i die Verschiebung und Verdrehung $(\delta T x_2, \delta T_{y_2}, \delta\alpha_2)^T$
6. Berechne Überdeckungsvektor neu $(\Delta T_x, \Delta T_y, \Delta\alpha)^T = (\Delta T_x, \Delta T_y, \Delta\alpha)^T + (\delta T_{x_1}, \delta T_{y1}, \delta\alpha_2)^T$
7. Springe zurück zu Schritt 2, wenn Verschiebung größer als die minimal zugelassene Verschiebung oder, wenn Verdrehung größer als die minimal zugelassene Verdrehung

Somit terminiert der Algorithmus, wenn:

$$\sqrt{\Delta T_x{}^2 + \Delta T_y{}^2} \leq E_{dist}$$

$$|\Delta\alpha| \leq E_\alpha$$

Ein Nachteil des IDC Algorithmus ist, dass dieser in der Bestimmung der Korrespondenzpaare eine bestimmte Zeit beansprucht. Da pro Punkt des aktuellen Scans ein entsprechender Punkt im Referenzscan bestimmt werden muss, ergibt sich ein Aufwand von $O(n^2)$. Durch ein Winkelintervall wird zwar der Suchbereich im Referenzscan pro aktuellen Punkt reduziert, der Aufwand bleibt dennoch von der Anzahl n der Punkte im Winkelintervall abhängig. Dementsprechend bleibt der Aufwand bei $O(n^2)$. Wird der IDC Algorithmus nun k mal durchlaufen um eine optimale Überdeckung zu gewährleisten, so ergibt sich ein Aufwand von $O(kn^2)$.

Da beim CaroloCup-Fahrzeug die Position mehrmals pro Runde korrigiert wird, hat man sich bei der Entwicklung dafür entschieden den IDC Algorithmus nicht k-mal durchlaufen zu lassen sondern nur einmal. Zwar nimmt man dadurch bei jeder Positionskorrektur eine Fehlertoleranz in Kauf, die jedoch nicht weiter relevant ist, da die Position beim nächsten markanten Punkt wieder korrigiert wird, wodurch eine Fehlerfortpflanzung nicht zu Stande kommt.

5 Implementierung

In diesem Kapitel wird die entwickelte Softwarearchitektur der Weltmodellierung auf dem CaroloCup Fahrzeug beschrieben. Dabei wird der Aufbau, der Kartierungsablauf, die einzelnen Softwaremodule, sowie die Beziehung zwischen diesen, erläutert.

5.1 Aufbau der Software Architektur

Bei der Architektur hat man sich an [Kuipers (2000)] orientiert. So wurde die Software in mehrere Module unterteilt und hierarchisch aufgebaut. Dabei müssen einzelne Module die Anforderungen erfüllen, welche im [Kap.4] an die Weltmodellierung gestellt wurden, sowie die dort erläuterten Algorithmen implementieren. [Abb. 5.1] zeigt die Softwarearchitektur der implementierten Weltmodellierung. Diese besteht aus sechs wichtigen Bestandteilen: SensorControl, Scanner, TopologicControl, Lokal Mapping, PositionCorrector und dem Cartographer. Die Module sind dabei hierarchisch gegliedert. Den sequentiellen Ablauf erkennt man an der Nummerierung im Diagramm.

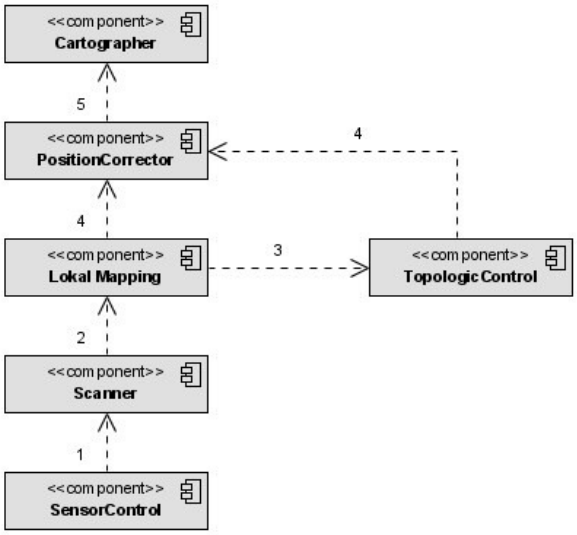

Abbildung 5.1: Module der Weltmodellierung

SensorControl: Das Modul SensorControl kümmert sich um die Sensordatenverarbeitung. Dabei werden Sensordaten der Kamera, des Linienlasers sowie der Ultraschall Sensoren zu korrekten Werten zusammengeführt und die internen Wegsensorwerte dem Scannermodul bereitgestellt.

Scanner: Die Komponente Scanner übernimmt bei der Weltmodellierung die Positionsbestimmung anhand der odometrischen Daten.

Lokal Mapping: Im Lokal Mapping Modul wird eine lokale Weltmodellierung an Hand der SensorControl Daten durchgeführt. Die Karte wird als ein MapData Objekt abgespeichert, welches eine Liste mit Rasterzellobjekten (GridObject) beinhaltet [Abb. 5.2].

Topological Control: Die Komponente Topological Control kümmert sich um das Wiedererkennen markanter Punkte auf der Strecke und das Verwalten der Referenzscans. Die Referenzscans werden dabei als ReferenceMapData Objekte abgespeichert und beinhalten eine Liste mit GridObjects einer markanten Stelle in der Umgebung [Abb. 5.2].

PositionCorrector: Wie der Name angibt, wird von dieser Komponente die Positionsabweichung unter Zunahme der aktuellen Karte und des dazugehörigen Referenzscans ermittelt.

Cartographer: Das Cartographer Modul ist in der Hierarchie ganz oben angesiedelt. Durch diesen wird eine globale Welt mit Hilfe der lokalen Karten und der Positionsdaten modelliert.

Für die Kommunikation der Module wurden spezielle Datenobjekte entwickelt, welche als Eventdaten zu Verfügung gestellt werden. Die Module werden bei den gewünschten Daten als Listener registriert. Die Datenobjekte werden in der CaroloCup Software Architektur im State Reflektor verwaltet, wodurch die Registrierung bei den Objekten ebenfalls beim StateReflektor erfolgt. [Abb. 5.2] zeigt die Datenobjekte die bei der Weltmodellierung als Kommunikationsdaten im StateReflektor abgespeichert werden.

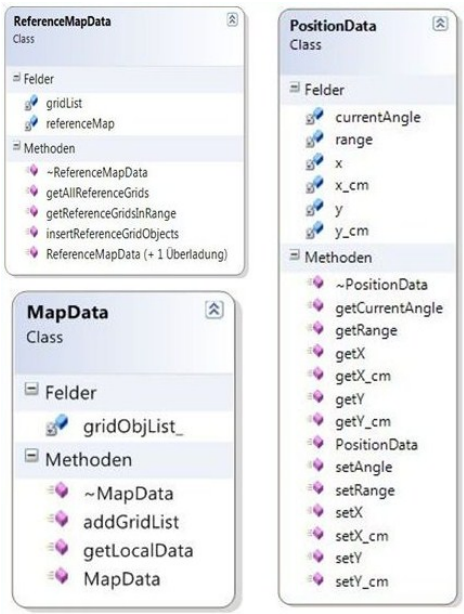

Abbildung 5.2: Kommunikations Datenobjekte bei der Weltmodellierung

Die Rasterzellen, die als gemeinsames Datenformat für die lokale, globale Karte, sowie den Referenzscan eingesetzt werden, sind -wie in [Abb. 5.3] dargestellt- aufgebaut.

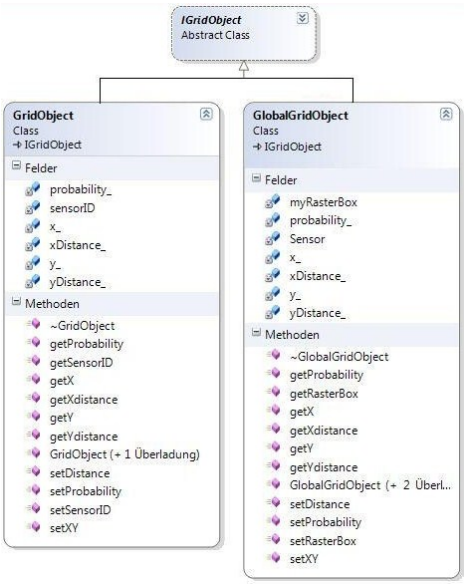

Abbildung 5.3: Aufbau der Rasterzellen

In dieser Arbeit werden die Komponenten für die globale Weltmodellierung entwickelt. Dazu gehören die Komponenten: Scanner, PositionCorrector sowie der Cartographer. Die Module für die lokale Kartierung werden in der Arbeit [Ebert (2008)] implementiert.

5.2 Arbeitsweise der Software

Der Ablauf der globalen Weltmodellierung in der Software erfolgt in drei Schritten. Zunächst wird vom Scanner die relative Fahrzeugposition mit Hilfe der Odometriedaten berechnet. Diese wird als PositionData Objekt abgespeichert und bei der Aktualisierung der Referenzscandaten (markanter Punkt in der Umgebung erkannt) durch den PositionCorrector korrigiert. Durch die neu berechnete Position wird wiederum der Cartographer aktiviert, der mit Hilfe der GridObjects der lokalen Karte (MapData) die Welt modelliert. Das Verhalten der Aktivierung wird dadurch erreicht, indem sich die Module Scanner, PositionCorrector, sowie der Cartographer bei den jeweiligen Daten als Listener registrieren.

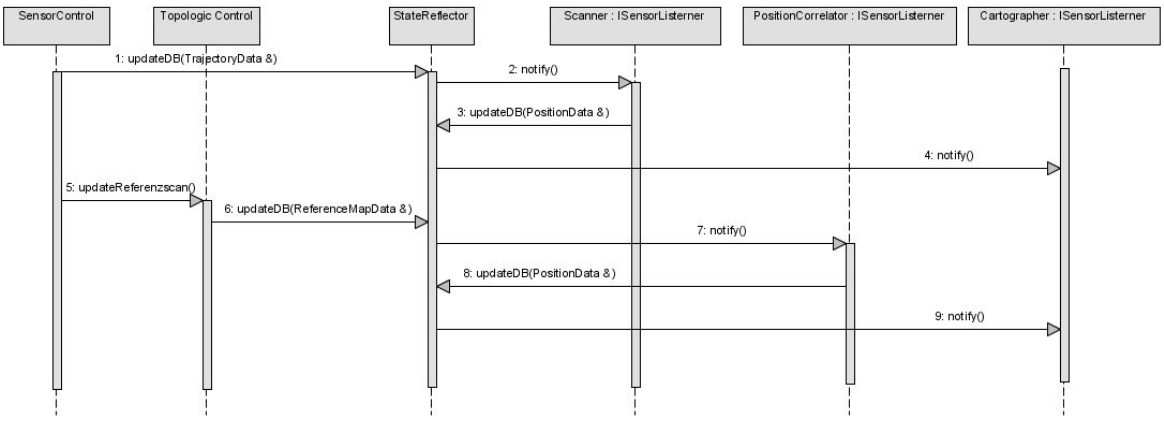

Abbildung 5.4: Sequenzdiagramm: Ablauf der globalen Weltmodellierung

5.3 Global Mapping Modul

Das Global Mapping Modul, beinhaltet die Logik für die globale Kartenerstellung. Der Aufbau der Komponenten ist dabei modular aufgebaut, wodurch die Aufgaben, wie das Umrechnen der lokalen Rasterzellkoordinaten in globale Rasterzellkoordinaten, wie auch das Verteilen der globalen Rasterzellen auf globale Boxen, durch mehrere Klassen gelöst werden. In [Abb. 5.5] wird der Aufbau des Moduls durch ein Klassendiagramm veranschaulicht.

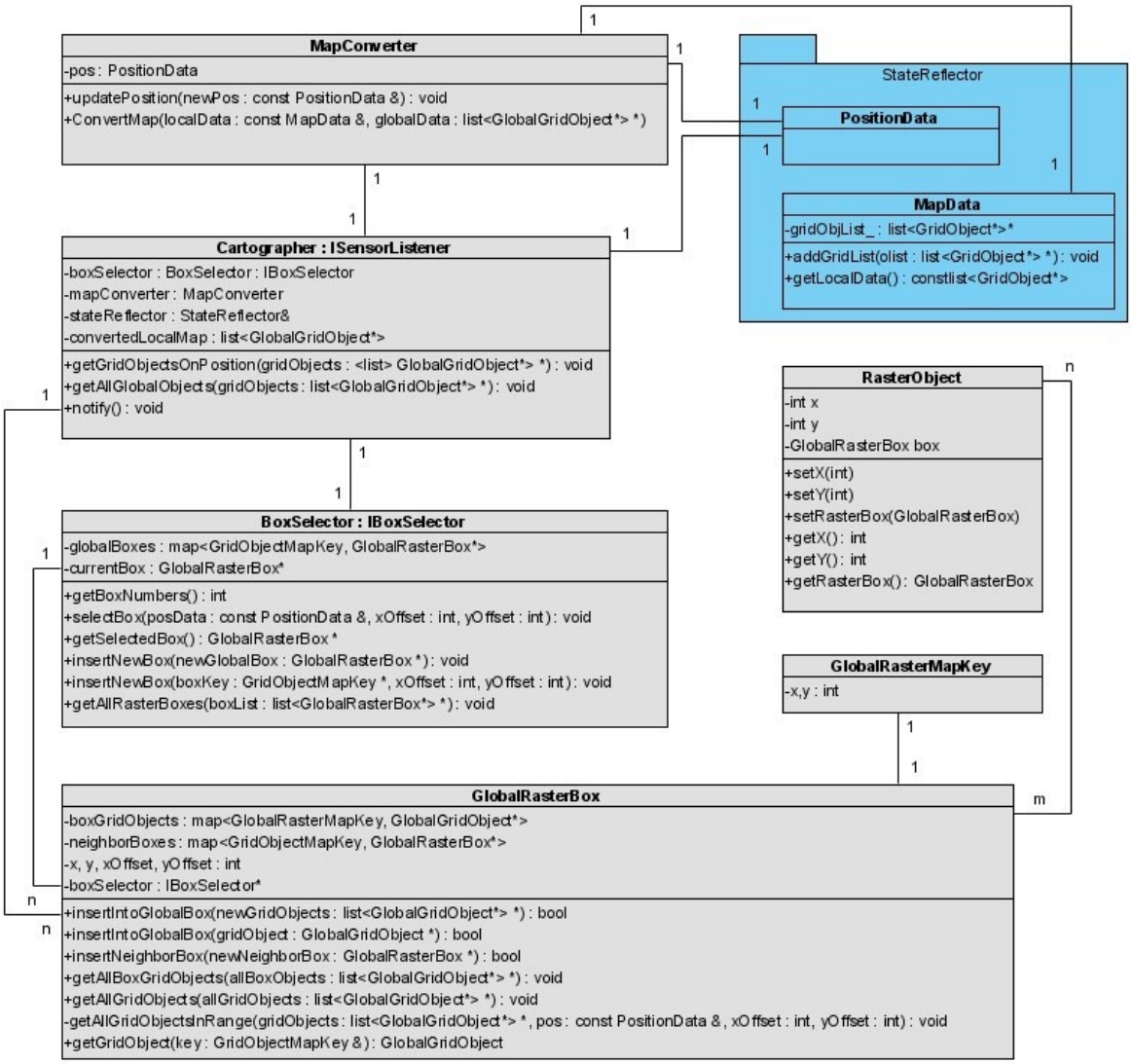

Abbildung 5.5: Klassendiagramm: Cartographer Modul

5.3.1 Ablauf der globalen Kartierung

Der Cartographer stellt die Basis der globalen Mapping Architektur dar. Dieser wird einmal in der CaroloCup Architektur als Objekt angelegt und bei dem PositionData Objekt im StateReflektor registriert. Da der Cartographer als Listener agiert, wird er bei einer Aktualisierung der Positionsdaten über die Funktion notify() informiert. Zunächst werden vom Cartographer zu der Position die lokalen Kartendaten geholt, dessen lokale Rasterzellen durch den MapConverter in globale konvertiert werden. Dabei wird die Umrechnung nach dem [Algorithmus 1] durchgeführt. Die konvertierten Rasterzellen werden wiederum an die vom BoxSelector ausgewählte globale Box geschickt, welche sich um das Speichern der Grind Objekte kümmert. Das Sequenzdiagramm in der [Abb. 5.6] erläutert den Ablauf der globalen Kartierung.

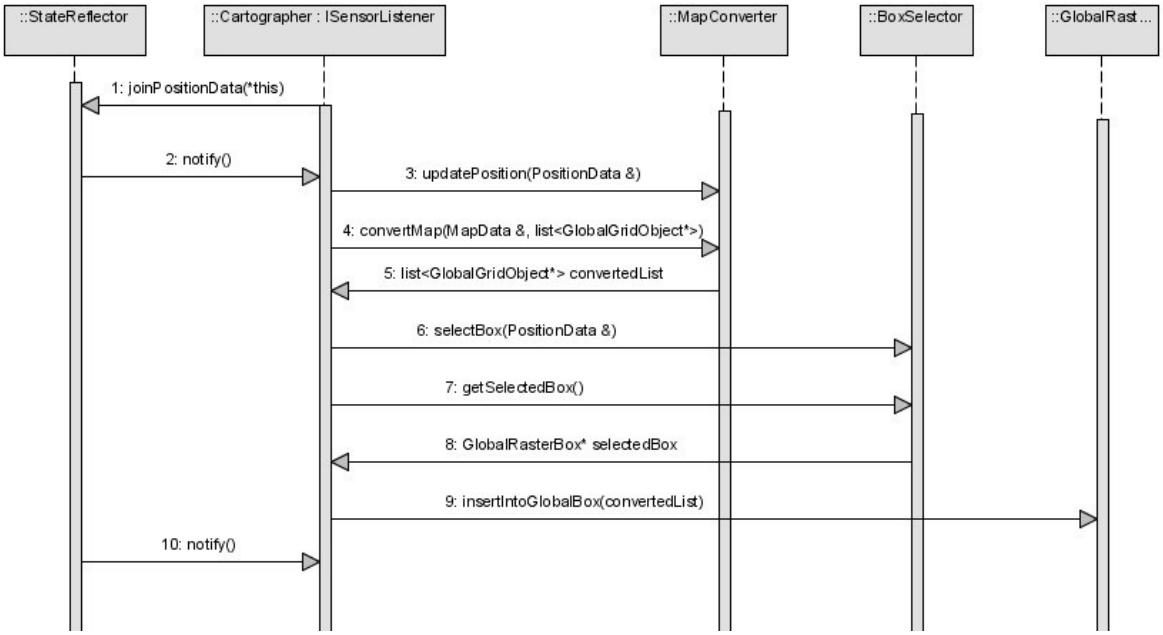

Abbildung 5.6: Ablauf der globalen Kartierung

5.3.2 Rasterbox

Wie bereits erläutert beinhalten Rasterboxen die Logik für die Verteilung der Rasterzellen auf die Boxen. Dabei wird der [Algorithmus 2] umgesetzt. Rasterzellen werden in einer Map gespeichert, da mit Hilfe eines Grid Keys ein direkter Zugriff bei Anfrage auf die Rasterzellen möglich ist. Durch den Grid Key wird eine Rasterzelle im gesamten Weltmodell eindeutig beschrieben, da sich in diesem die Koordinaten einer Rasterzelle im Weltkoordinatensystem

befinden. Der Map Key verhindert auch, dass zu einer Rasterzelle, mehrere Objekte im Weltmodell gespeichert werden. Beinhaltet die Map eine Rasterzelle mit dem gleichem Key wie eine neu übergebene Zelle, so wird die alte Zelle mit dem gleichem Key aus der Map entfernt und die neue abgespeichert.

Eine weitere Map beinhaltet Zeiger auf die Boxobjekte, welche an die Box angrenzen. Auf diese Weise ist das Boxobjekt in der Lage, die Rasterzellen, deren Koordinaten nicht im Wertebereich der Box liegen, direkt auf die Nachbarn zu verteilen. Wird beim Verteilen festgestellt, dass zu einem Rasterzellobjekt keine Nachbarbox im Weltmodell vorhanden ist, so wird der BoxSelector über das Fehlen informiert. Dieser erzeugt die fehlende Rasterbox und teilt allen Boxen im Weltmodell die neu angelegte über die Funktion insertNeighborBox(. . .) mit [Abb. 5.7, Punkt 5].

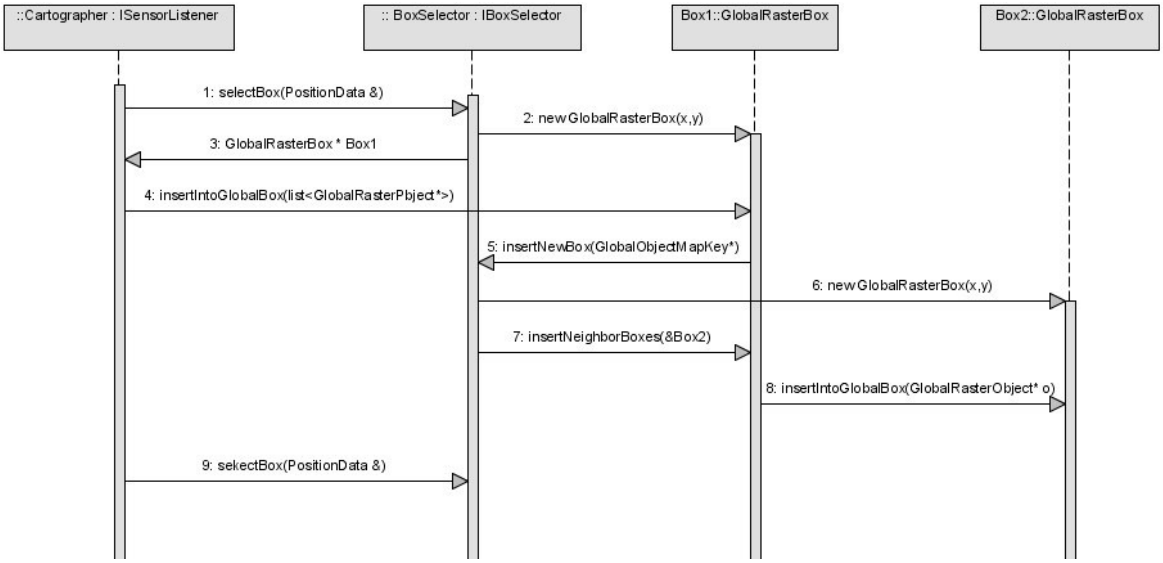

Abbildung 5.7: Ablauf der Boxselektierung und Boxgenerierung

5.3.3 BoxSelector

Der BoxSelector garantiert die in [Kap. 4.3.3] analysierte Skalierung der globalen Karte. Dazu werden globale Rasterboxen mit einem Box Key in einer Map verwaltet. Der Box Key, besteht aus den Boxkoordinaten und dient als eindeutiger Index in der Map. Der Selector wählt immer die Box aus, in der sich das Fahrzeug befindet. Dabei wird ein Zeiger auf das Objekt in der Map gelinkt. Wird vom BoxSelector keine RasterBox zu dem PositionData Objekt gefunden, so wird eine neue Box angelegt und selektiert. Da die Verteilung der Rasterzellen durch die Rasterboxen geschieht, kann es vorkommen, dass die Boxen eigenständig

feststellen, dass eine Nachbarbox fehlt. In diesem Fall bietet der BoxSelector die Funktion insertNewBox() an. Wird diese aufgerufen, wird eine neue Box mit dem gewünschten Schlüssel angelegt und in die Map gespeichert. Daraufhin werden alle anderen Boxen über die Aktualisierung informiert. [Abb. 5.7] beschreibt den Ablauf der Boxselektierung, sowie die dynamische Boxgenerierung.

5.4 Scannermodul

Die odometrische Positionsbestimmung erfolgt im Scannermodul anhand des TrajectoryData Objektes, das durch den RS232 Driver im StateReflector abgespeichert und aktualisiert wird. Das TrajectoryData Objekt beinhaltet die Daten über die zurückgelegten Wegstrecken der beiden Räder, sowie den Einlenkwinkel in Prozent, der jedoch bei der Berechnung verworfen wird. In der Initialisierungsphase der CaroloCup Software registriert sich das Scannerobjekt bei den Daten und wird nach jeder Aktualisierung über Änderungen der Daten benachrichtigt. Aus den Daten wird die relative Position wie in [Kap. 4.4.1.2] beschrieben berechnet und die PositionsDaten im StateReflector aktualisiert. Da die Distanz im TrajectoryData Objekt in cm angegeben wird, werden die Positionskoordinaten durch die Rasterzellgröße dividiert um die Rasterzelle im Weltmodell bestimmen zu können in welcher sich das Fahrzeug gerade befindet. Vorsichtshalber werden die Koordinaten auch in cm gespeichert.

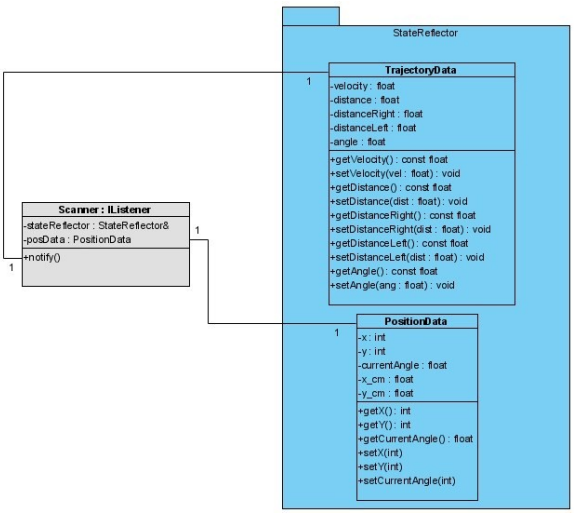

Abbildung 5.8: Aufbau Scannermodul

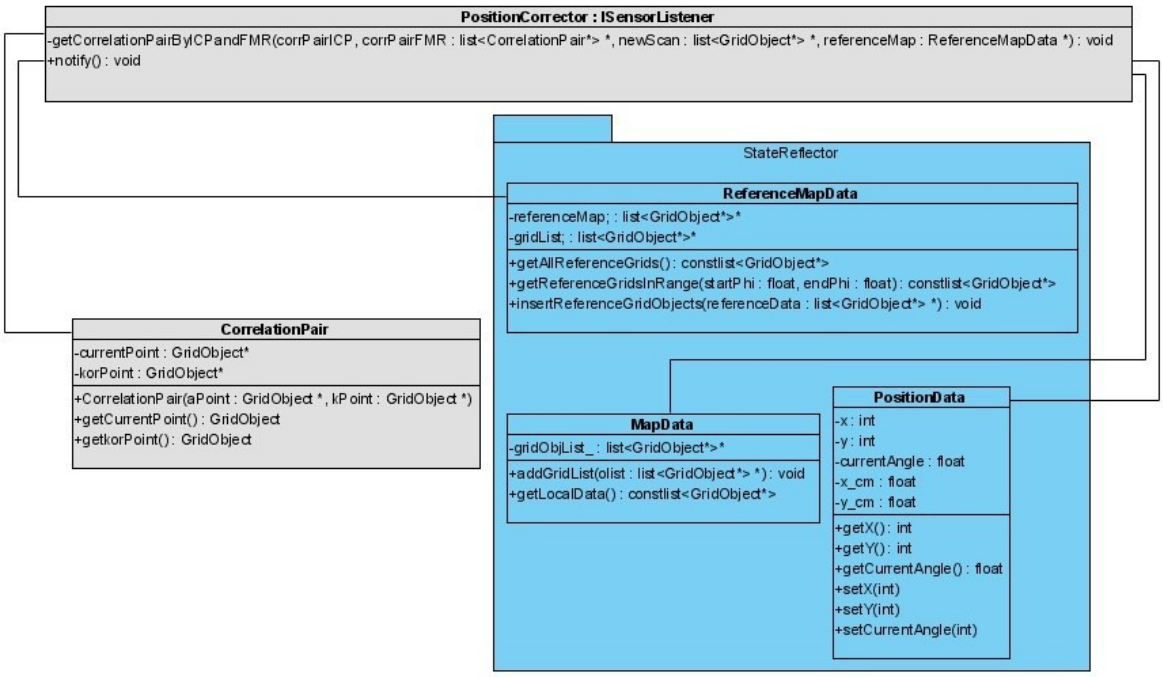

Abbildung 5.9: Aufbau des PositionCorrector Moduls

5.5 PositionCorrector

Der IDC Algorithmus für die Korrektur der Position wird direkt in der notify() Funktion des PositionCorrektor Objekts durchgeführt. Der Aufruf der Funktion erfolgt durch den StateReflector nachdem das ReferenceMapData Objekt aktualisiert wurde. Die Aktualisierung vollzieht das TopologicMap Modul. Dabei wird eine Liste von Rasterzelle einer markanten Stelle in der Umgebung im ReferenceScan gespeichert. Die Funktion getReferenceGridsInRange(. . .) des ReferenceMapData Objekts bietet dem PositionCorrector die Möglichkeit, Referenzpunkte nur innerhalb eines Winkelbereichs im Referenzscan zu laden. Dadurch müssen pro aktuellen Punkt im MapData Objekt nur ein Teil der Punkte im ReferenceScanData Objekt analysiert werden, um das Korrespondenzpaar zu bestimmen.

Die Suche nach den Korrespondenzpunkten erfolgt durch die "`Iterative Closest Point`" und "`Finde Matching Range`" Algorithmen in der Funktion getCorrelationPairByICPandFMR(..). Die gefundenen Korrespondenzpaare werden dementsprechend als CorrelationPair Objekte in der Liste corrPairICP bzw. corrPairFMR abgelegt. Nach der Abarbeitung des IDC Algorithmus, wird das Positionsdatenobjekt geladen und die Orientierung α_p, wie auch die Positionskoordinaten (x_p, y_p) aktualisiert. Die neue Pose (x_p, y_p, α_p) im globalen Weltmodell wird auf:

$$(x_p, y_p, \alpha_p)^T = \begin{pmatrix} x_p + \delta x_p \\ y_p + \delta y_p \\ \alpha_p - \delta\alpha \end{pmatrix}$$

geändert.

6 Weltmodellierung im Einsatz

In diesem Kapitel wird die entwickelte Architektur der globalen Weltmodellierung im realen Einsatz getestet. Dazu wird zunächst die Kartierung der Umgebung bei verschiedenen Fahrten geprüft und auf Genauigkeiten analysiert. Im zweiten Testabschnitt wird dann die Korrektheit, der auf dem Fahrzeug eingesetzten Positionskorrektur mittels Testdaten ermittelt.

6.1 Kartierung während der Fahrt

Zur Prüfung der Korrektheit, wird die entwickelte Weltmodellierung direkt auf dem CaroloCup-Fahrzeug getestet. Dabei werden verschiedene Fahrten auf einer Testrecke durchgeführt und die Umgebung rasterbasiert kartographiert. Die Hinderniserfassung erfolgt mittels der Ultraschallsensoren. Die Rasterzellgröße der lokalen, sowie der globalen Karte ist auf 10x10cm festgesetzt, wodurch die entwickelte Weltmodellierung im optimal Fall auf 10cm genau kartographieren soll.

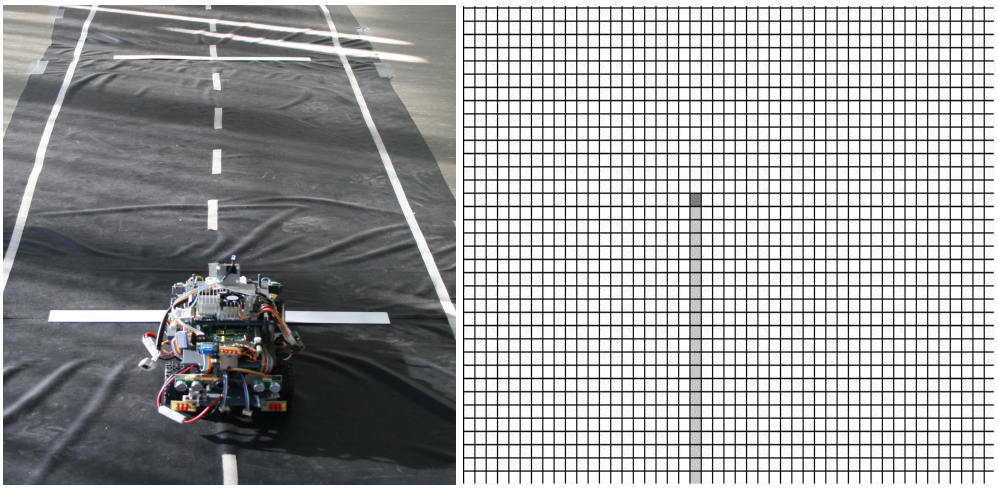

Abbildung 6.1: Geradeausfahrt

[Abb. 6.1] visualisiert die erste Testfahrt. Dabei wird eine 210cm lange Geradeausfahrt ohne Hindernisse kartographiert. Das Fahrzeug startet bei der Position $Pos_{start}(50, 0, 0°)$ und erreicht nach der Fahrt die Position $Pos(50, 21, 0°)$ Das dunkelgraue Feld beschreibt die aktuelle Position des Fahrzeugs in der Umgebung, die Hellen die befahrenen Rasterzellen. Wie in der Abbildung zu sehen ist, wird die Position richtig berechnet. Das Fahrzeug bleibt bei der Rasterzelle mit den Koordinaten $(50, 21)$ stehen und ist mit einem Orientierungswinkel von $0°$ ausgerichtet.

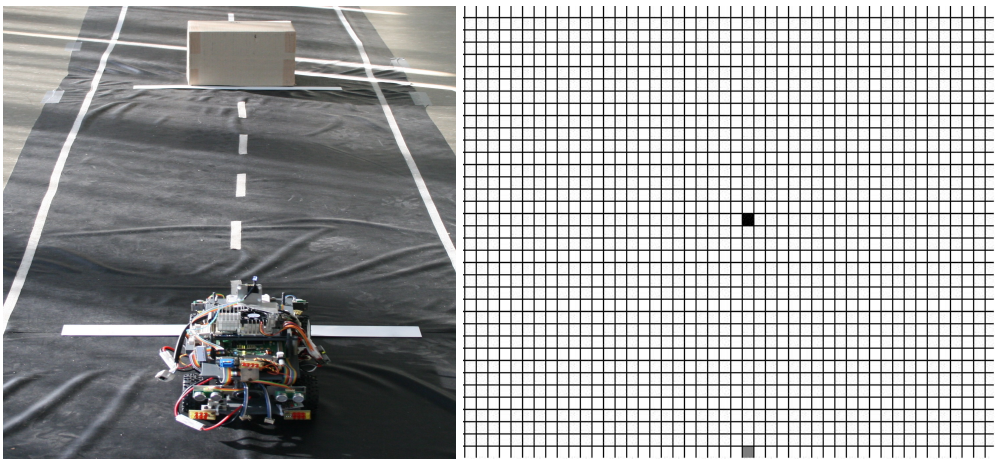

Abbildung 6.2: Kartographiertes Hindernis in 2m Entfernung

Bei der zweiten Testfahrt wird eine 135m lange Geradeausfahrt mit einem Hindernis, das sich 2m vom Fahrzeug entfernt befindet, durchgeführt. Da nur Ultraschallsensoren zur Hinderniserfassung verwendet werden, wird beim Start der Fahrt nur eine Rasterzelle $P_h(50, 20)$ als `"Belegt"` markiert [Abb. 6.2].

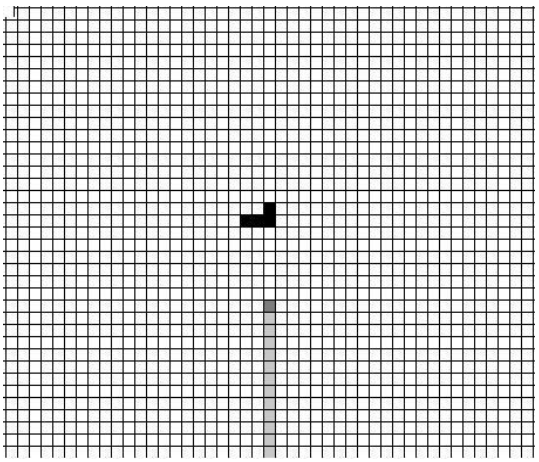

Abbildung 6.3: Kartographiertes Hindernis während einer Fahrt

Während der Fahrt beträgt die Genauigkeit der Entfernungsmessung mittels der Ultraschalsensoren $> 10cm$, was zur Aktualisierung weiterer Rasterzellen auf der y-Ebenen $P_1(50, 21)$ in der globalen Karte führt [Abb. 6.3]. Die Rasterzellen auf der x-Ebene $P_2(51, 20)$, $P_3(52, 20)$ werden durch die Erfassung der Hindernisbreite aktualisiert.

Bei der dritten Testfahrt wird eine Kurvenfahrt um 180° kartographiert. Die Startposition des Fahrzeugt beträgt $Pos_{start}(50, 0, 0°)$, die gewünschte Zielposition soll $Pos_{Ziel} \approx (79, 0, 180°)$ entsprechen. Mittig der Kurve befindet sich ein Hindernis, das um einen Winkel ϕ gedreht ist. Bei dem Test soll explizit die Positionsbestimmung, sowie die Fusionierung der lokalen Kartendaten in das globale Weltmodell geprüft werden. Wie in [Abb. 6.4] zu sehen wird das Hindernis mittig der Kurve im globale Weltmodell gemappt. Auch die Verdrehung des Hindernisses ist im Rastermodell zu erkennen. Die rechts neben der Kurve als "`Belegt`" markierten Rasterzellen stellen eine Wand in der Umgebung dar und sind ebenfalls korrekt. Doch auch fehlerhaftmarkierte Rasterzellen sind im Rastermodell fusioniert. Diese werden zum Teil durch die fehlerbehafteten Ultraschallsensormesswerte auf dem CaroloCup-Fahrzeug verursacht.

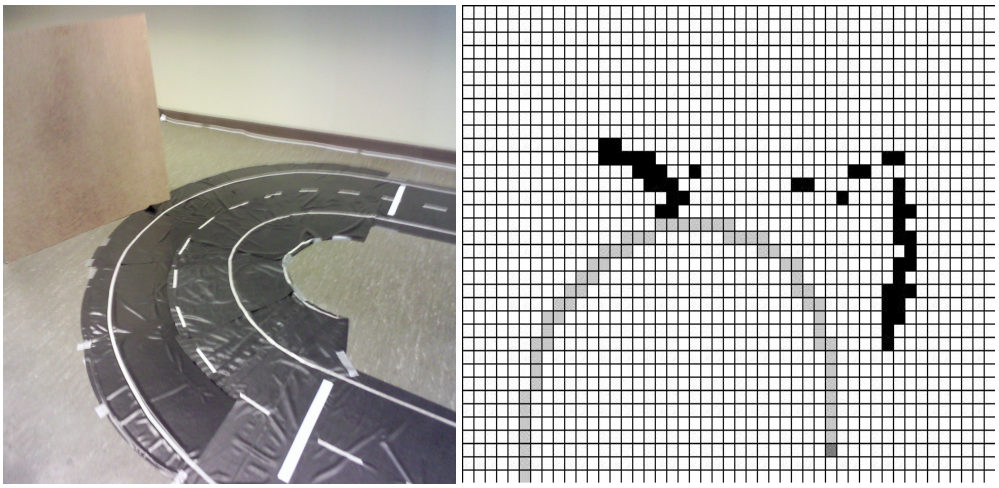

Abbildung 6.4: Kartographierte Rechtsfahrt

Nach der Kurvenfahrt beträgt der Orientierungswinkel des Fahrzeugs 185°, was einer Abweichung von 5° der gewünschten 180° entspricht. Diese Abweichung macht sich in der durch die Odometrie berechnete Endposition bemerkbar. So weicht die berechnete Positionierung der Fahrzeugs $Pos_{odom} = (77, 2, 185°)$ von der realen $Pos_{real} \approx (79, 0, 180°)$ um $(\delta 1, \delta 2, \delta 5°)$ ab.

Eine weitere Testfahrt [Abb. 6.5] repräsentiert eine Linksfahrt der selben Kurve. Diesmal legt das Carolo-Cup Fahrzeug einen größeren Weg zurück und fährt einen ausgedehnten Bogen. Das Hindernis in der Umgebung wurde zum Fahrzeug gedreht, sodass das Fahrzeug beim Einlenken darauf zufährt. Der protokollierte Fahrweg zeigt die Lenkschwankungen des Fahrzeugs bei einer autonomen Fahrt. Dabei erschweren die Schwankungen die odometriebasierte Positionsbestimmung, da die Wegmessung beider Vorderräder bei dauernder Hin- und Herlenkung ungenauer wird, sodass sich der Fehler in die Positionierung überträgt. Die errechnete Position des Fahrzeugs ist, wie in der Abbildung zu erkennen, $Pos_{odom} = (16, 21, 172°)$. Die tatsächlich erreicht Position war während des Tests $Pos_{real} \approx (19, 19, 180°)$. Die belegtmarkierten Rasterzellen links neben der Strecke, stellen einen kartographierten Pfeiler neben der Strecke dar.

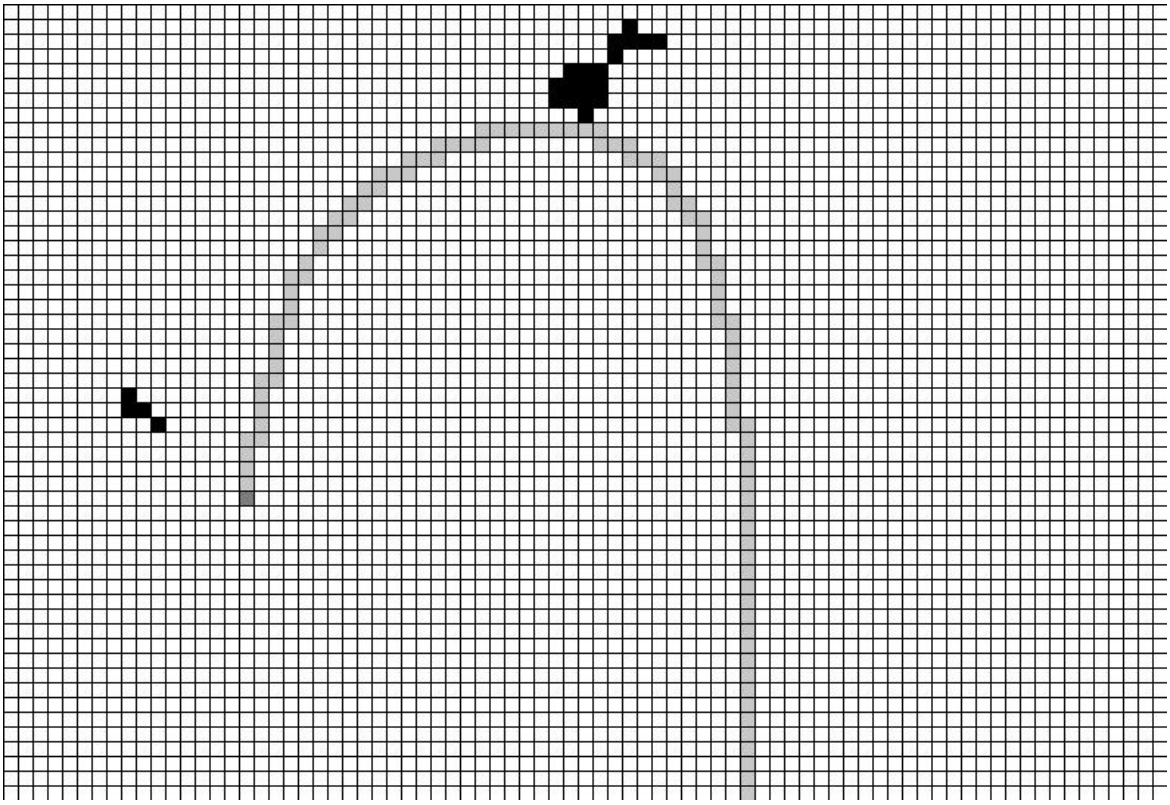

Abbildung 6.5: Kartographierte Linksfahrt

6.2 Positionskorrektur im Test

Die Positionskorrektur wird unter drei Szenarien getestet. Bei allen Szenarien wird eine lokale Karte mit einer Stopplinie als Referenzscan verwendet. Die Linie ist anhand von vier Referenzpunkten P_{ref} signiert. Das Fahrzeug fährt in den Testszenarien jeweils auf die Linie zu, wobei ein aktuelle Scan aufgenommen und mit dem Referenzscan abgeglichen wird. Die Linie im neu aufgenommenen Scan wird ebenfalls durch vier Punkte P_a signiert. Die Suche nach der Überdeckung der beiden Scans geschieht mittels dem IDC Algorithmus, der einmal durchlaufen wird. Dabei liegt der Suchbereich der Korrespondenzpunkte P'_{idc} und P''_{fmr} pro Punkt P_a im Winkelintervall $[\alpha_{Pa} - 13°, \alpha_{Pa} + 13°]$. Die Aufnahmeposition des Referenzscans Pos_{ref} beträgt in allen drei Testszenarien $(0, 0, 0°)$. Die Korrespondenzpaare werden mittels ICP, sowie dem FMR Algorithmus bestimmt.

6.2.1 Testszenarium 1.

Im ersten Testfall fährt das Fahrzeug auf die Linie mit einer Orientierung von 0° zu. Der aktuelle Scan wird gegenüber dem Referenzscan mit einer Verschiebung von $(\Delta T_x, \Delta T_y)$ aufgenommen. Die Punkte des Referenzscans P_{ref}, des aktuellen Scans P_a, sowie die berechneten Korrespondenzpunkte P'_{idc} und P''_{fmr} sind in der [Tab. 6.1] aufgeführt.

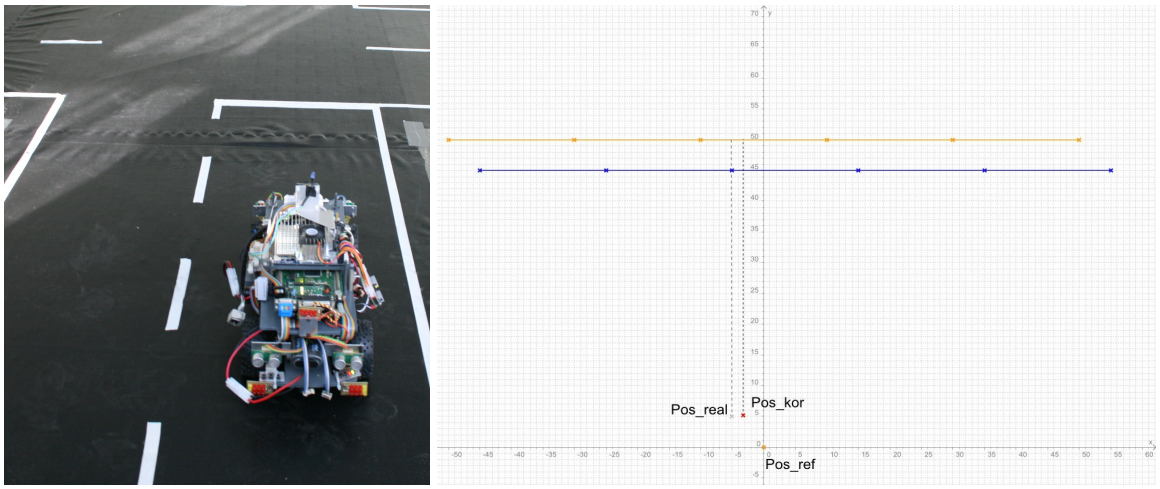

Abbildung 6.6: Testszenarium bei einer Fahrzeugorientierung von 0°

P_{ref}(x,y)	P_a(x,y)	P'_{icd}(x,y)	P''_{fmr}(x,y)
(-50,50)	(-45,45)	(-50,50)	(-50,50)
(-30,50)	(-25,45)	(-30,50)	(-30,50)
(-10,50)	(-05,45)	(-10,50)	(-10,50)
(10,50)	(15,45)	(15,50)	(10,50)
(30,50)	(35,45)	(35,50)	(30,50)
(50,50)	(55,45)	(50,50)	(50,50)

Tabelle 6.1: Korrespondenzpaare bei einer Fahrzeugorientierung von 0°

Die berechnete Verschiebung ΔT und Verdrehung $\Delta\alpha$ ist in der [Tab. 6.2] aufgelistet. Um die Genauigkeit zu analysieren, werden die daraus resultierende Positionswerte $Pos_{kor}(T_{kor}, \alpha_{kor})$ mit den realen Positionswerten $Pos_{real}(T_{real}, \alpha_{real})$ gegenüber gestellt. Anhand der Tabelle erkennt man, dass der ICP Algorithmus ein schlechteres Ergebnis als der FMR Algorithmus liefert. Das liegt daran, dass die Referenzlinie nur durch vier Punkte definiert ist, wodurch für die Korrespondenzpunktsuche wenige Referenzpunkte P_{ref} berücksichtigt werden.

Algorithmus	$(\Delta T_x, \Delta T_y, \Delta\alpha)$	$Pos_{real}(T_{real}, \alpha_{real})$	$Pos_{kor}(T_{kor}, \alpha_{kor})$	Abweichung
FMR	(-5,5,0°)	(-5,5,0°)	(-5,5,0°)	(0,0,0°)
ICP	(-3,5,0°)	(-5,5-0°)	(-3,5,0°)	(-2,0,0°)
IDC	(-3,5,0°)	(-5,5-0°)	(-3,5-0°)	(-2,0,0°)

Tabelle 6.2: Positionskorrektur bei einer Fahrzeugorientierung von 0°

6.2.2 Testszenarium 2.

Im zweiten Testfall wird die Linie in der Umgebung unter einem negativem Orientierungswinkel, sowie einer Verschiebung $(\Delta T_x, \Delta T_y)$ aufgenommen. Wie im Testszenarium 1 wird auch hier anhand der Korrespondenzpaare des FMR Algorithmus das beste Ergebnis erzielt.

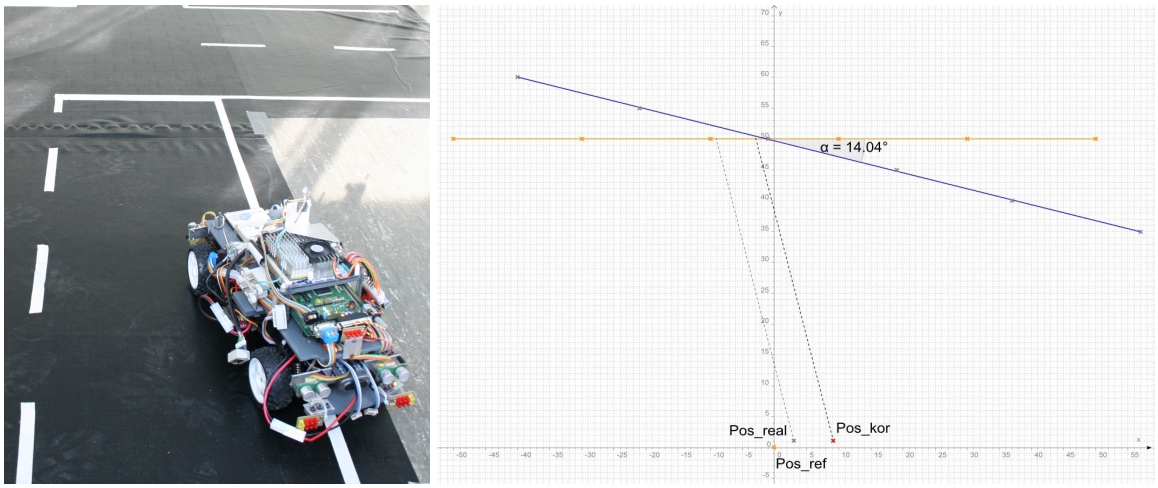

Abbildung 6.7: Testszenarium bei einer Fahrzeugorientierung von -14°

P_{ref}(x,y)	P_a(x,y)	P'_{icd}(x,y)	P''_{fmr}(x,y)
(-50,50)	(-40,60)	(-40,50)	(-50,50)
(-30,50)	(-21,55)	(-21,50)	(-30,50)
(-10,50)	(-01,50)	(-01,50)	(10,50)
(10,50)	(19,45)	(19,50)	(10,50)
(30,50)	(37,40)	(37,50)	(30,50)
(50,50)	(57,35)	(50,50)	(50,50)

Tabelle 6.3: Korrespondenzpaare bei einer Orientierung von -14°

Algorithmus	$(\Delta T_x, \Delta T_y, \Delta\alpha)$	$Pos_{real}(T_{real}, \alpha_{real})$	$Pos_{kor}(T_{kor}, \alpha_{kor})$	Abweichung
FMR	(5,1,13.7°)	(3,1,-14°)	(5,1,-13.7°)	(2,0,-0.3°)
ICP	(9,1,13.2°)	(3,1,-14°)	(9,1,-13.2°)	(6,0,-0.8°)
IDC	(9,1,13.7°)	(3,1,-14°)	(9,1,-13.7°)	(6,0,-0.3°)

Tabelle 6.4: Positionskorrektur bei einer Fahrzeugorientierung von -14°

6.2.3 Testszenarium 3.

Im letzen Testfall ist der Orientierungswinkel, unter welchem die Linie aufgenommen wird, positiv. Auch hier ist die aktuelle Aufnahmeposition um den Betrag $(\Delta T_x, \Delta T_y)$ gegenüber der Referenzposition verschoben.

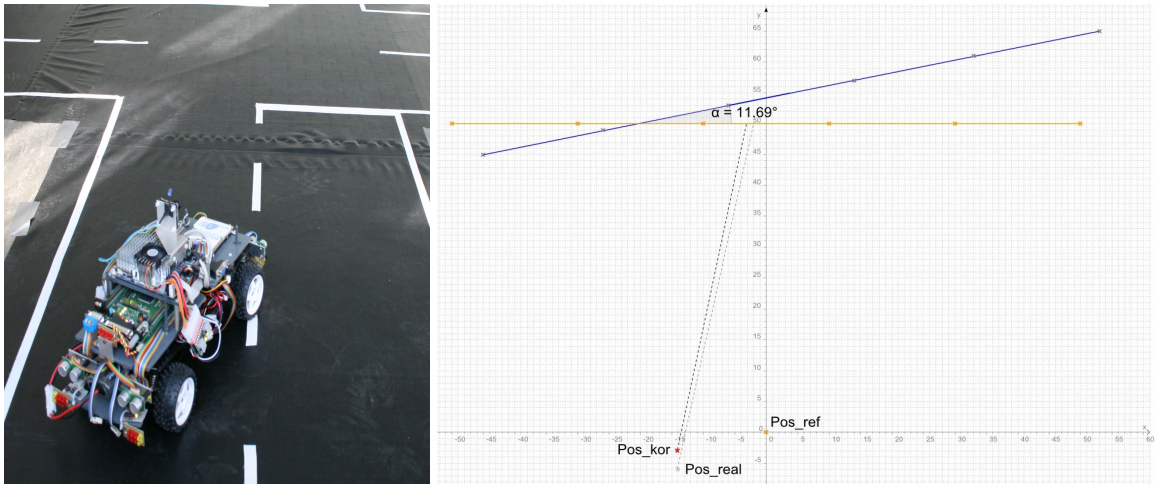

Abbildung 6.8: Testszenarium bei einer Fahrzeugorientierung von +11°

P_{ref}(x,y)	P_a(x,y)	P'_{icd}(x,y)	P''_{fmr}(x,y)
(-50,50)	(-45,45)	(-50,50)	(-50,50)
(-30,50)	(-26,49)	(-30,50)	(-30,50)
(-10,50)	(-06,53)	(-10,50)	(-10,50)
(10,50)	(14,57)	(10,50)	(10,50)
(30,50)	(33,61)	(30,50)	(30,50)
(50,50)	(53,65)	(53,50)	(50,50)

Tabelle 6.5: Korrespondenzpaare bei einer Orientierung von +11.7°

Algorithmus	$(\Delta T_x, \Delta T_y, \Delta\alpha)$	$Pos_{real}(T_{real}, \alpha_{real})$	$Pos_{kor}(T_{kor}, \alpha_{kor})$	Abweichung
FMR	(-14,-3,-11.5°)	(-14,-6,11.7°)	(-14,-3,11.5°)	(0,-3,0.2°)
ICP	(-14,-3,-11.5°)	(-14,-6,11.7°)	(-14,-3,11.5°)	(0,-3,0.2°)
IDC	(-14,-3,-11.5°)	(-14,-6,11.7°)	(-14,-3,11.5°)	(0,-3,0.2°)

Tabelle 6.6: Positionskorrektur bei einer Fahrzeugorientierung von +11.7°

Eine 100% Überdeckung der beiden Scans wird durch das einmalige Durchlaufen des IDC Algorithmus nicht erreicht.

7 Resümee und Perspektiven

Wie die Tests auf dem CaroloCup-Fahrzeug ergeben haben, erreicht man mit dem entwickelten Konzept der Kartenmodellierung ein gutes Ergebnis. So wird die globale Karte aus den lokalen Daten dynamisch modelliert und dabei dynamisch skaliert. Durch die Variation der Zellgröße lässt sich zudem die Genauigkeit der Karte erhöhen.

Problematisch ist die odometrische Positionsbestimmung. Die Schwankungen bei der Wegmessung des linken Rades, sowie des rechten Rades verursachen auf dem CaroloCup-Fahrzeug eine ungenaue Orientierungsberechnung, sodass sich die Abweichung auch auf die Fahrzeugkoordinaten und die Zellenkoordinaten im globalen Weltmodell überträgt. Diese wird zwar durch die Positionskorrektur gering gehalten, die fehlerhaften Rasterzellen der globalen Karten bleiben jedoch im weiteren Verlauf der Weltmodellierung erhalten. Um das Konzept zu verbessern kann die odometrieberechnete Position während der Fahrt mehrfach korrigiert werden. Die Sensordatenverarbeitung muss dafür zusätzliche topologische Punkte in der Umgebung bestimmen, die vom "`Topologic Control`" Modul verwaltet werden. Da in dieser Arbeit ein Punkt-zu-Punkt Überdeckungsalgorithmus bei der Positionskorrektur implementiert ist, können Referenzscans mit Kurven für die Korrektur eingesetzt werden, wodurch sich die Position an jeder Kurve korrigieren lässt. Eine Verbesserung der Korrektur könnte durch zusätzliche Durchläufen des IDC Algorithmus und durch die Zunahme zusätzlicher Referenzpunkte erreicht werden. Ein Gyroskop würde ebenfalls zu einer Verbesserung der Genauigkeit führen. Dieser wäre insbesondere hilfreich für die erste Kartierungsfahrt.

Durch die markanten Punkte und die dazugehörigen Referenzscans bietet das Konzept auch die Möglichkeit der topologischen Weiterentwicklung des Weltmodells. So kann in den markanten Punkten nicht nur ein Referenzscan abgelegt werden, sondern ganze Fahranweisungen an die Fahrzeugsteuerung. Die Fahranweisungen würden dabei eine Art Kante vom Knoten zum Knoten (markanter Punkt) bilden, wodurch sich eine Wegplanung und eine Navigation trivial gestalten lassen.

Die in dieser Arbeit entwickelte globale Weltmodellierung ist so konzipiert, dass es plattformenunabhängig arbeitet. Damit kann auch ein Roboter mit der Architektur Umgebungskarten modellieren. Sensordaten anderer Sensorquellen müssen dabei lediglich in Rasterzellenobjekte vom Typ "`GridObject`" verpackt und als Liste im "`MapData`" Objekt abspeichert werden.

Literaturverzeichnis

[Arvidsson 2008] ARVIDSSON, Martin: *ARM - gesteuerte Antriebsschlupfregelung eines autonomen Modellfahrzeuges: Analyse, Design, Implementierung*, Hochschule für Angewandte Wissenschaften Hamburg, Bachelorarbeit, 2008

[Berger 2008] BERGER, Denis: *Fahrspurerkennung mit Three Feature Based Lane Detection Algorithm(TFALDA)*, Hochschule für Angewandte Wissenschaften Hamburg, Studienarbeit, 2008

[Borenstein und Feng 1996] BORENSTEIN, J. ; FENG, L.: Gyrodometry: a new method for combining data from gyros and odometry in mobile robots. In: *IEEE International Conference on Robotics and Automation* (1996), S. 423–428

[Borenstein und Koren 1990] BORENSTEIN, J. ; KOREN, Y.: Teleautonomous guidance for mobile robots. In: *IEEE Transactions on Systems, Man and Cybernetics* 20 (1990), Nr. 6, S. 1437–1443

[Ebert 2008] EBERT, Michael: *Aktives Mapping und Positionsbestimmung eines autonomen Modellfahrzeugs*, Hochschule für Angewandte Wissenschaften Hamburg, Bachelorarbeit, 2008

[Elfes 1986] ELFES, Alberto: A Sonar-Based Mapping and Navigation System. In: *IEEE International Conference on Robotics and Automation* 3 (1986), April, S. 1151–1156

[Elfes 1987] ELFES, Alberto: Sonar-Based Real-World Mapping and Navigation. In: *IEEE Journal of Robotics and Automation* 3 (1987), Juni, Nr. 3, S. 249–265

[Gilles 1985] GILLES, W.: Universal Computer Program for Seeking a Corrision Free Path for an Industrial Robot. In: *15th. International Industrial Robot Exhibition* (1985), September, S. 935

[Gutmann 1996] GUTMANN, Jens-Steffan: *Vergleich von Algorithmen zur Selbstlokalisierung eines mobilen Roboters*, Universität Ulm, Diplomarbeit, 1996

[Hall und Llinas 1997] HALL, D.L. ; LLINAS, J.: An introduction to multisensor data fusion. In: *Proceedings of the IEEE* 85 (1997), Januar, Nr. 1, S. 6–23

[Hans Rudolf Schwarz 2006] HANS RUDOLF SCHWARZ, Norbert K.: *Numerische Mathematik*. Teubner, 2006. – ISBN 3835101145

[Isler und Bajcsy 2006] ISLER, V. ; BAJCSY, R.: The Sensor Selection Problem for Bounded Uncertainty Sensing Models. In: *IEEE International Conference on Automation Science and Engineering* 3 (2006), Oktober, Nr. 4, S. 372–381

[Julier und Uhlmann 2003] JULIER, S.J. ; UHLMANN, J.K.: Using multiple SLAM algorithms. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2003), S. 200–205

[Kai-Tai u. a. 1991] KAI-TAI, Song ; CHAO-YUAN, Chen ; CHARLES, C.: On-line motion planning of an autonomous mobile robot based onsensory information. In: *IEEE/RSJ International Workshop on Intelligent Robots and Systems* 2 (1991), November, S. 423–428

[Knieriemen 1991] KNIERIEMEN, Thomas: *Autonome Mobile Roboter*. BI-Wiss.-Verlag, 1991. – ISBN 3411150319

[Kuipers 2000] KUIPERS, Benjamin: The spatial semantic hierarchy. In: *Artificial Intelligence* 119 (2000), Februar, S. 191–233

[Lu und Milios 1997] LU, Feng ; MILIOS, Evangelos: Robot Pose Estimation in Unknown Environments by Matching 2D Range Scans. In: *Journal of Intelligent and Robotic Systems* 18 (1997), S. 249–275

[M. Beckerman 1990] M. BECKERMAN, E.M. O.: Treatment of systematic errors in the processing of wide-anglesonar sensor data for robotic navigation. In: *IEEE Transactions on Robotics and Automation* 6 (1990), April, Nr. 2, S. 137–145

[Reimers 2008] REIMERS, Hannes: *Fahrbahnverfolgung und Geschwindigkeitsregelung mittels eines Fuzzy-Logik Reglers für ein autonomes Modellfahrzeug*, Hochschule für Angewandte Wissenschaften Hamburg, Bachelorarbeit, 2008

[Wandiredja 2008] WANDIREDJA, Jonathan: *Hinderniserkennung und Ausweichmanöver für autonome Fahrzeuge: Analyse, Modelierung und Implementierung*, Hochschule für Angewandte Wissenschaften Hamburg, Bachelorarbeit, 2008

[Weber 2003] WEBER, Joachim: *Globale Selbstlokalisation für mobile Service Roboter*. Universität Kaiserslautern, 2003. – ISBN 9783925178917

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §22(4) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 20. August 2008	
Ort, Datum	Unterschrift