Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Torben Becker

Identifizierung und Kartierung dynamischer Umgebungen für autonome Fahrzeuge

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Torben Becker

Identifizierung und Kartierung dynamischer Umgebungen für autonome Fahrzeuge

Masterarbeit eingereicht im Rahmen der Masterprüfung

im Studiengang Master of Science 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

Eingereicht am: 29. Januar 2014

Torben Becker

Thema der Arbeit
Identifizierung und Kartierung dynamischer Umgebungen für autonome Fahrzeuge

Stichworte
SLAM, RatSLAM, Navigation, Lokalisation, Autonom, Fahrzeug, FAUST

Kurzzusammenfassung
Diese Masterarbeit beschäftigt sich mit dem Aufbau einer topologischen Karte auf einer unbekannten Fahrbahn. Diese Aufgabe soll auf Basis des RatSLAM Algorithmus (SLAM: Simultaneous Localization and Mapping) auf einer autonomen Fahrzeugplattform im Maßstab 1:10 bewältigt werden. Darüber hinaus soll eine Aufwertung der erstellten Karte stattfinden. In die Karte sollen Informationen über Kurvenanfang und -ende, Hindernisse sowie Kreuzungen integriert werden. Für diese Aufgabe wird in dieser Masterarbeit eine Erkennung für Hindernisse und Kreuzungen entwickelt, die beide Fahrspuren überprüft.

Title of the paper
Identification and mapping of dynamic environments for autonomous vehicles

Keywords
SLAM, RatSLAM, navigation, localization, autonomous, vehicle, FAUST

Abstract
This master thesis deals with the construction of a topological map on an unknown roadway. This task will be handled on basis of the RatSLAM algorithm on an autonomous vehicle with 1:10 scale. The created map should take place with upgrades. The created map shall be enhanced with map information such as curve start and end point, obstacles and crossings. For this task, a obstacle and crossing recognition system is developed, which checks both lanes.

Inhaltsverzeichnis

1 Einleitung

Ein Themengebiet, in dem sehr viel geforscht wird, sind SLAM Algorithmen (Siciliano und Khatib (2008)). SLAM steht für Simultaneous Localization and Mapping. Dies bedeutet, dass ein Roboter in einer unbekannten Umgebung sich eine Karte erstellt. Gleichzeitig ist der Roboter in der Lage, seine Position innerhalb der Karte zu jeder Zeit zu bestimmen. Solche Algorithmen sind interessant für Search and Rescue Einsätze oder für die Verwaltung von Lagerhallen. Bei einem Search and Rescue Einsatz besitzen die Einsatzkräfte möglicherweise keine Kenntnis über die Umgebung. Beispielsweise bei einem Hurricane, der einen Küstenabschnitt verwüstet hat. Anstatt der Erkundung eines Weges durch die Einsatzkräfte erkundet ein Roboter mit einem SLAM Algorithmus eine mögliche Fahrgasse und teilt diese den Einsatzkräften mit. Bei der Verwaltung von Lagerhallen geht es vor allem um die Positionen der Roboter innerhalb der Lagerhalle. Die Regel kann sein, dass sich pro Gasse nur ein Roboter aufhalten darf. In dem sich die Roboter gegenseitig ihre Position mitteilen, wird diese Regel eingehalten und es kommt zu keinen Blockaden.

Zur Ermittlung der Daten für ein SLAM Algorithmus gibt es verschiedene Sensorarten. Oft benötigt ein SLAM Algorithmus Odometrie-Daten, bestehend aus Translation und Rotation des Roboters. Diese Daten werden entweder direkt von Sensoren zur Verfügung gestellt oder durch andere Sensorarten geschätzt. Zur räumlichen Erfassung werden Stereo-Kameras, Laser-Scanner oder Ultraschall- sowie Radar-Sensoren eingesetzt. Jede Sensorart hat dabei ihre Vor- und Nachteile. Je nach Algorithmus und Einsatzgebiet müssen diese gegeneinander abgewägt werden. Darüber hinaus existieren viele Lösungsansätze zur Lokalisierung innerhalb der erstellten Umgebungskarte. Die zwei bekanntesten Ansätze stellen (Extended-) Kalman-Filter oder Partikelfilter dar. Ein Nachteil dieser Ansätze stellt die enorme Datenmenge dar, die mit der Zeit abgespeichert werden. Durch diesen Umstand muss entweder ein sehr leistungsfähiger Rechner oder eine sehr niedrige Wiederholrate benutzt werden, um ansatzweise eine Echtzeitgarantie zu ermöglichen. In diesem Zusammenhang bedeutet Echtzeitgarantie, dass die Ausführungszeit konstant bleibt, sobald eine vollständige Karte der Umgebung erstellt worden ist. Das bedeutet,

dass nur noch eine Relokalisierung durchgeführt wird, aber keine erneute Erweiterung der bestehenden Karte.

Durch die Erkenntnisse in der Neurowissenschaft mit Nagetieren ist ein biologischer Ansatz entwickelt worden. Neurowissenschaftler haben die Orientierung von Nagetieren im Raum genauer untersucht und dabei bestimmte Zelltypen im Gehirn entdeckt, die ausschlaggebend für die Orientierung sind. Auf Basis dieses Wissens ist der SLAM Algorithmus RatSLAM entwickelt worden. RatSLAM erstellt eine topologische Karte auf Basis der Odometrie (Rotation und Translation des Roboters) sowie Kamerabildern. Die Datenmengen werden möglichst gering gehalten, wodurch auch bei größeren Umgebungen eine Echtzeitgarantie gegeben ist.

1.1 Motivation

Das Forschungsprojekt FAUST beschäftigt sich mit der Entwicklung von autonomen Modellfahrzeugen. Ein Hauptaugenmerk des Forschungsprojektes ist dabei der studentische Wettbewerb Carolo-Cup (Carolo-Cup) der Technischen Universität Braunschweig. Das Ziel der Fahrzeuge ist autonom auf einer Fahrbahn zu fahren, Hindernisse und Kreuzungen zu erkennen sowie einzuparken. Für diesen Zweck ist eine Spurerkennung auf Basis von Polynomen (Jenning (2008)) sowie eine Spurführung nach dem Algorithmus Pure Pursuit (Coulter (1992), Nikolov (2009)) entwickelt worden. Beide Algorithmen arbeiten sehr zuverlässig. Allerdings kommt es vor, dass durch Fehlstellen in der Fahrbahnmarkierung falsche Werte in der Spurerkennung ermittelt werden. Diese falschen Werte besitzen weiterhin Auswirkungen auf die Spurführung. Als Ergebnis fährt das Fahrzeug statt einer Linkskurve eine Rechtskurve oder gerade aus. Dieses Verhalten ist nicht wünschenswert. Für ein solches Fehlverhalten gibt es beim Carolo-Cup (Carolo-Cup-Regelwerk) fünf Strafmeter. Somit gilt es ein Fehlverhalten weitestgehend zu vermeiden.

Um die Aufgabe der Vermeidung von Fehlverhalten zu bewältigen, bietet sich ein SLAM Algorithmus an. Ist die Karte aufgebaut, besitzt das Fahrzeug Kenntnis von der Umgebung und seiner eigenen Position. Allerdings ist die Rechenleistung auf den Fahrzeugen für den Carolo-Cup sehr begrenzt, wodurch SLAM Algorithmen mit Kalman- oder Partikelfilter nicht infrage kommen. Beide Verfahren benötigen viele Daten und eine hohe Rechenleistung. RatSLAM bietet einen Ansatz, der zum einen der Sensorik des Fahrzeuges entspricht. Auf dem Fahrzeug befindet sich eine monochrome Kamera, ein Inkrementalgeber für die Translation und eine Inertial Measurement Unit (IMU)

für die Rotation. Zum anderen ist die benötigte Rechenleistung für RatSLAM nicht besonders hoch. Der verbaute Pico-PC mit einem Intel Atom D2700 berechnet RatSLAM, sobald die Karte vollständig existiert, in einer konstanten Zeit. Diese Zeit ist allerdings nicht so hoch, dass eine Verletzung der Echtzeitgarantie vorliegt. Darüber hinaus kann die erstellte Karte mit weiteren Informationen aufgewertet werden, wie zum Beispiel den approximierten Polynomen, Hindernissen oder Kreuzungen. Dies macht RatSLAM zu einen insgesamt sehr interessanten SLAM Algorithmus für bodengestützte leistungsschwache Roboter.

Darüber hinaus wird ein Algorithmus zur Hindernis- und Kreuzungserkennung benötigt, der sowohl Hindernis auf der linken als auch rechten Fahrspur identifiziert und validiert. In der Vergangenheit ist ein solcher Algorithmus nur auf Basis von Kameradaten implementiert worden. Ein solcher Ansatz kann allerdings auch Reflexionen oder Spiegelungen von anderen Beleuchtungsquellen als Hindernis erkennen und entsprechend markieren. Dies soll allerdings vermieden werden.

1.2 Zielsetzung

In dieser Masterarbeit soll ein Algorithmus zur Hindernis- und Kreuzungserkennung implementiert werden. In der Forschungswelt wird derzeit mit Stereo Kameras (Bertozzi und Broggi (1998), Labayrade u. a. (2002), Seki und Okutomi (2006)) geforscht. Es gibt allerdings auch Ansätze mit omnidirektionalen Kameras (Yamazawa u. a. (1995)). Dieser Algorithmus soll zunächst mithilfe der Kamera Hindernisse und Kreuzungen identifizieren. Als Rahmenbedingung gilt, dass die optische Erkennung nur sehr wenig Ressourcen verbraucht sowie eine geringe Ausführungszeit besitzt. Ist eine solche Identifizierung durch optische Erkennung durchgeführt, erfolgt eine Validierung durch die verbaute Sensorik. Dafür sind in der Front des Fahrzeuges sieben Infrarot Sensoren mit einer maximalen Messdistanz von 150 Zentimetern verbaut. Die Kreuzungserkennung basiert durchweg auf einer optischen Erkennung. Wird entweder eine Kreuzung oder ein Hindernis erkannt, soll dies RatSLAM mit ausreichend Informationen zur aktuellen Lage mitgeteilt werden.

Zunächst soll durch RatSLAM eine Karte der Strecke erstellt werden. Ist diese Karte aufgebaut, erhält RatSLAM durch die Hindernis- und Kreuzungserkennung Nachrichten. Infolgedessen sollen diese entsprechend in die Karte integriert werden. Dabei soll zum einen die Position des Hindernisses oder der Kreuzung in der Karte mit der Realität übereinstimmen. Zum anderen sollen die neu erstellten Pfade, sofern dies möglich ist,

reduziert werden. Damit soll der Speicherverbrauch insgesamt niedrig gehalten werden bei gleichbleibender Informationsfülle. Als weiterer positiver Effekt bleibt auch der Berechnungsaufwand insgesamt niedriger. Weiterhin soll ein Algorithmus implementiert werden, der in einer bestimmten Entfernung die Situation einschätzen kann. Dies bedeutet, dass der Algorithmus vorher sagt, ob sich beispielsweise in einem Meter Entfernung eine Kreuzung, ein Hindernis oder eine Kurve beziehungsweise eine Gerade befindet.

1.3 Gliederung

In dem Kapitel 2 dieser Arbeit wird näher auf den SLAM Algorithmus RatSLAM eingegangen. Dabei wird die grundlegende System-Architektur erläutert sowie die Funktionsweise und Zusammenhänge von RatSLAM beschrieben. Es wird außerdem kurz beschrieben, was RatSLAM von anderen SLAM Algorithmen abgrenzt.

Kapitel 3 erläutert die wichtigsten Fakten aus dem Regelwerk des Carolo-Cups, auf dessen die Testumgebung für diese Arbeit basiert. Es werden die Definitionen für eine Kreuzung sowie statische und fahrende Hindernisse beschrieben. Darüber hinaus erfolgt eine Beschreibung der aktuellen Fahrzeugplattform Carolo-Cup v.2 des Forschungsprojektes FAUST. Es wird erklärt, welche Sensorik für diese Arbeit eingesetzt wird.

Eine Erläuterung des Algorithmus zur Hindernis- und Kreuzungserkennung erfolgt in Kapitel 4. Zunächst wird erläutert, wie die Berechnung des Schwellwertes und der Region of Interest erfolgt. Anschließend wird beschrieben, wie ein Hindernis identifiziert und validiert wird. Darauf folgend wird die Identifizierung einer Kreuzung beschrieben.

In dem Kapitel 5 wird beschrieben, wie in die durch RatSLAM erstellte Karte die Hindernisse und Kreuzungen integriert werden. Es wird kurz erläutert, wie der Aufbau der Karte im entsprechend Teilausschnitt aussieht. Darüber hinaus wird beschrieben, wie die Algorithmen die entsprechenden Informationen über Hindernisse oder Kreuzungen in die Karte integrieren. Ebenfalls wird ein Algorithmus erklärt, der ein Vorausschauen in der Karte ermöglicht.

In den Kapitel 6 und 7 wird die Implementierung sowie die Auswertung der erstellten Algorithmen aus den Kapiteln 4 und 5 erläutert. In diesen Kapiteln werden auf die

Systemarchitekturen sowie die verwendeten Zustandsautomaten eingegangen. Darüber hinaus werden die Testbedingungen erläutert und das Testergebnis präsentiert. Das Kapitel 8 stellt eine Zusammenfassung der Arbeit dar und welche weiteren Aufgaben auf dieser Arbeit aufbauen können.

2 RatSLAM

Die Entwicklung von RatSLAM (Milford u. a. (2004), J.Milford und Wyeth (2008), J.Milford und Wyeth (2010), J.Milford u. a. (2011)) ist durch eine genauere Erforschung des Orientierungssinnes von Nagetieren, im speziellen Ratten, ermöglicht worden. Neurowissenschaftler haben sechs Zelltypen im Gehirn entdeckt, die für die Orientierung in einem Raum zuständig sind. Durch diese Zellen extrahieren Nagetiere verschiedene Merkmale aus der Umgebung und orientieren sich so in einem Raum. Drei dieser Zellen sind dabei von besonderer Relevanz. Das sind die Grid Cells, Head Direction Cells und Place Cells. RatSLAM versucht nicht, jede dieser Zelltypen exakt nachzubilden und zu berechnen. Dies ist aufgrund deren Komplexität nicht möglich. Stattdessen ist eine Modellierung vorgenommen worden, in der versucht wird, die wichtigsten Eigenschaften der drei Zelltypen abzubilden. Daraus sind die Pose Cells und die Local View Cells entstanden. In den Pose Cells sind zwei der drei Zelltypen modelliert worden.

Es wird die Bibliothek OpenRatSLAM ((Ball u. a., 2013)) eingesetzt. OpenRatSLAM ist eine Implementierung von David Ball und Scott Heath aus dem Jahr 2011. Es existieren zwei Versionen der Bibliothek: eine Version für ROS (ROS) und eine C++ Bibliothek, die für einen Roboter namens iRat (Ball u. a. (2010)) entwickelt worden ist. Die größten Unterschiede der beiden Versionen sind die Extraktion von Odometriedaten sowie die Systemarchitektur. Während die ROS Version ein visuelles Odometriesystem besitzt und für Multi-Core Prozessoren optimiert ist, hat die C++ Version keine Weiterentwicklung erfahren. Da die gesamte Software-Architektur des Forschungsprojektes FAUST in der Sprache C++ geschrieben ist, wird ebenfalls die C++ Version von OpenRatSLAM eingesetzt. Der Grund für den Einsatz einer Bibliothek anstelle einer eigenen Entwicklung besitzt mehrere Gründe. Der Umfang der Entwicklung ist enorm groß. Darüber hinaus existieren bereits mehrere Implementierungen von verschiedenen Universitäten. Die Schwerpunkte der verwendeten Bibliothek liegen auf der Echtzeitfähigkeit sowie der Ausführung auf dem Roboter beziehungsweise Fahrzeug.

2.1 Aufbau und Funktionsweise

In Abbildung 2.1 ist die Architektur von RatSLAM dargestellt. Dabei stellt die gestrichelte Linie die RatSLAM Task dar. Vom System wird ein Bild sowie Sensorwerte für die Translation und Rotation an die RatSLAM Task übergeben. Das Bild wird nur von den Local View Cells verarbeitet. Die Translation und Rotation werden sowohl von den Pose Cells als auch von der Experience Map benötigt. Dabei werden die einzelnen Module Local View Cells, Pose Cells und Experience Map in einer festgelegten Reihenfolge berechnet. Dies hat den Grund, dass die einzelnen Module auf die berechneten Ergebnisse des Vorgängermoduls angewiesen sind.

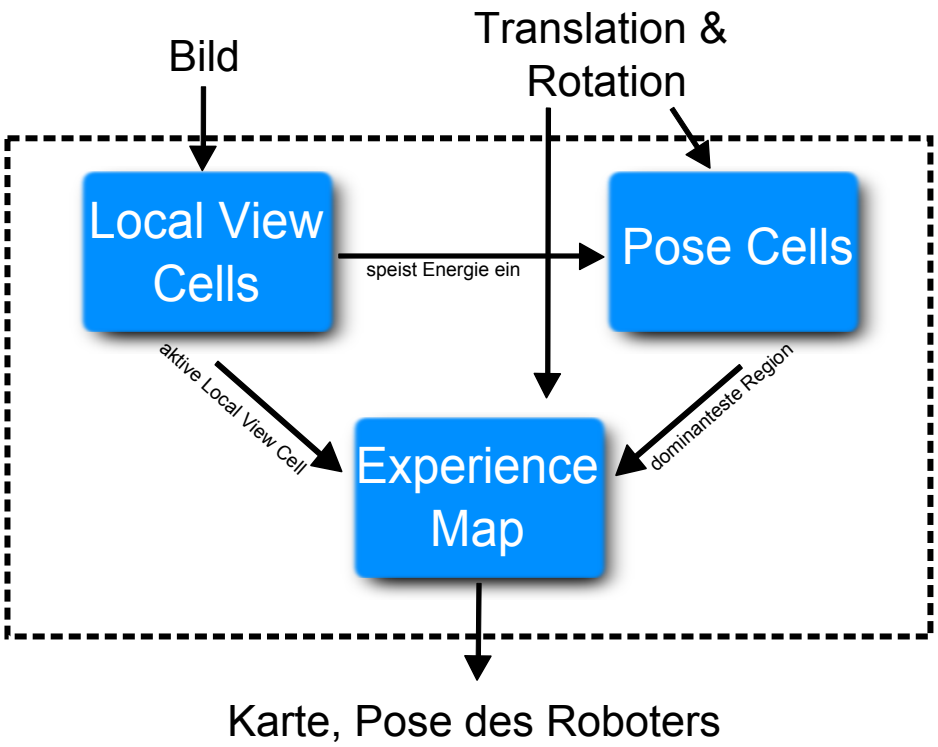

Abbildung 2.1: Architektur von RatSLAM

Zuerst müssen die Local View Cells berechnet werden. Diese injizieren Energie in das Pose Cell Netzwerk, wenn eine vorhandene Local View Cell aktiviert wird. Dabei ist jede Local View Cell mit einer visuellen Szene assoziiert. Ist eine visuelle Szene

neu, wird eine neue Local View Cell erstellt. Als Nächstes wird das Pose Cell Netzwerk berechnet. Hierbei wird eine Pfadintegration der Translation und Rotation vorgenommen, wodurch die Energie im Pose Cell Netzwerk verschoben wird. Somit bilden sich aktive Regionen von zusammenhängenden Pose Cells. Die Experience Map wird als letztes Modul ausgeführt. Dies besitzt den Grund, dass in den erstellten Erfahrungen die zur Zeit aktive Local View Cell sowie die aktivste Region in den Pose Cells hinterlegt werden. Anhand dieser beiden Information kann RatSLAM eine Erfahrung eindeutig identifizieren und sich lokalisieren, wenn die gleiche visuelle Szene mit ähnlichen Aktivitäten in den Pose Cells vorliegt. Für die Erstellung der Verbindungen zwischen den Erfahrungen benötigt die Experience Map die Translation und Rotation. Dabei wird in die Verbindung die Differenz der Translation und Rotation zwischen zwei Berechnungen der Experience Map hineingeschrieben, um so eine Pfadplanung zu realisieren und den kürzesten Weg zu einem Zielpunkt zu berechnen.

Das Ergebnis, welches die Experience Map dem System zur Verfügung stellt, ist eine komplette Karte der gefahrenen Strecke sowie die aktuelle Position des Fahrzeuges innerhalb dieser Karte. Die aktuelle Position wird anhand der zur Zeit aktiven Erfahrung ermittelt. Durch diesen Umstand kann sehr schnell herausgefunden werden, welche Erfahrung als Nächstes aufgerufen werden soll. Somit ist eine gewisse Bahnplanung, sofern die Karte vollständig erstellt ist, mithilfe von RatSLAM möglich.

2.1.1 Local View Cells

Abbildung 2.2: Eingabebild

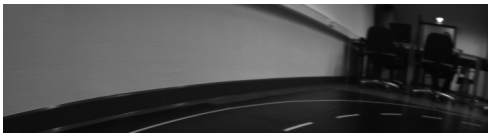

Abbildung 2.3: Genutzter Ausschnitt

Abbildung 2.4: Erstelltes Template

Die Local View Cells bestehen aus einem dynamisch wachsenden Array von Zellen. Dabei stellt jede Local View Cell eine bestimmte visuelle Szene dar (Abbildung 2.2). Mittels vier Parameter kann ein Ausschnitt (Abbildung 2.3) aus dem Originalbild erstellt werden. Es ist dabei der obere Bereich des Bildes gewählt worden, da im unteren Bereich ein Großteil durch die Fahrzeugkarosserie eingenommen wird. Es sollte sichergestellt sein, dass der ausgewählte Bildbereich eine möglichst individuelle visuelle Szene bietet, um eine effektive Wiedererkennung zu ermöglichen. Um den Berechnungsaufwand sowie den Speicherverbrauch noch weiter zu verringern, wird die Pixelanzahl des Bildausschnitts verringert. Das Endergebnis ist in Abbildung 2.4 dargestellt und wird Template genannt. Ein solches Template wird in jeder Local View Cell gespeichert.

Um festzustellen, ob das erstellte Template bereits zu einer Local View Cell gehört oder ob es sich um eine neue visuelle Szene handelt, wird für jede bereits existierende Local View Cell sowie das erstellte Template die Summe der absoluten Differenzen (Sum of Absolute Differences (SAD)) berechnet. Ist der Vergleich des Wertes zwischen einer Local View Cell und dem erstellten Template kleiner als ein einstellbarer Threshold, wird diese Local View Cell aktiviert und injiziert die Energie $\Delta P_{x',y',\theta'}$ in das Pose Cell Netzwerk mittels der folgenden Formel:

$$\Delta P_{x',y',\theta'} = \delta \cdot \sum_i \beta_{i,x',y',\theta'} \cdot V_i \tag{2.1}$$

Dabei bestimmt die Konstante δ den Einfluss der Bewegung auf die Schätzung der Position des Roboters. Darüber hinaus stellt eine Sättigungsfunktion sicher, dass jede visuelle Szene nur eine kurze Zeit Energie in das Pose Cell Netzwerk einbringt. Durch diese Maßnahme werden überflüssige Relokalisationen vermieden, zum Beispiel beim Stillstand des Roboters. Sollte der Threshold beim Vergleich nicht unterschritten werden, wird eine neue Local View Cell erstellt. Diese wird mit dem erstellten Template verknüpft und es wird eine Verbindung β erstellt. Dabei gibt β gibt, welche Region in den Pose Cells zur Zeit der Erstellung am aktivsten gewesen ist. Eine Relokalisation wird ausgelöst, wenn eine ausreichend lange Abfolge von bekannten Szenen in der richtigen Reihenfolge auftritt.

2.1.2 Pose Cells

Das Pose Cell Netzwerk besteht aus einem Continuous Attractor Network (Samsonovich und McNaughton (1997)). Continuous Attractor Networks sind ein spezieller Typ von

neuronalen Netzen. Sie haben ein Array von neuronalen Einheiten mit gewichteten Kanten. Das Pose Cell Netzwerk ist in einem dreidimensionalen Raum angeordnet, welcher aus x- und y-Koordinaten sowie θ als Drehrichtung des Roboters besteht. Somit entspricht das Pose Cell Netzwerk der Umgebung eines Roboters. Die Pose Cells besitzen ähnliche Eigenschaften wie Grid Cells (Hafting u. a. (2005)). Die größte Ähnlichkeit besteht in der Art der Aktivierung von Zellen. Es wird immer eine Region von zusammenhängenden Zellen auf einer Ebene aktiviert. Darüber hinaus existieren an den Kanten des Netzwerkes Verbindungen zu den um- und gegenüberliegenden Kanten, um die eingespeiste Energie optimal zu verteilen. Durch diese Besonderheit wird eine unendliche Umgebung dargestellt, obwohl nur eine endliche Umgebung repräsentiert werden soll. Somit ist es auch möglich, dass Pose Cells mehrere physische Orte darstellen. Zunehmend wird mit fortschreitender Zeit die Repräsentation der Umgebung diskontinuierlich, indem sich Fehler der Odometrie-Sensorik aufsummieren und Schleifen geschlossen werden. Dies bedeutet, dass benachbarte Pose Cells physische Orte darstellen, die sich in der Realität allerdings weit voneinander entfernt befinden.

Das Pose Cell Netzwerk wird in sechs Schritten berechnet. Im ersten Schritt wird die Energie, die durch die Local View Cells eingespeist worden ist (Gleichung 2.1), auf die umliegenden Pose Cells verstreut. Mit diesem Schritt werden aktive Regionen von Pose Cells erzeugt, welche als stabiler Zustand verstanden werden. Im nächsten Schritt wird eine lokale Hemmung der Energie von aktiven Pose Cells vorgenommen. Dies bedeutet, dass den benachbarten Pose Cells einer aktiven Pose Cell Energie abgezogen wird. Diese beiden ersten Schritte dienen der Stabilisierung der Energie im Pose Cell Netzwerk und werden mittels folgender Gleichung durchgeführt:

$$\epsilon_{a,b,c} = e^{-(a^2+b^2)/k_p^{exc}} \cdot e^{-c^2/k_d^{exc}} - e^{-(a^2+b^2)/k_p^{inh}} \cdot e^{-c^2/k_d^{inh}} \tag{2.2}$$

Die beiden Konstanten k_p und k_d beschreiben jeweils die Abweichung für Ort und Richtung. Diese Konstanten sind fest definiert. a, b und c sind jeweils die Distanzen zwischen den Zellen in x', y' und θ' Koordinaten. Im dritten Schritt wird jeder aktiven Pose Cell Energie abgezogen. Dies wird auch als globale Hemmung bezeichnet, wobei die Energie einer Pose Cell nicht negativ wird. Diese Änderung der Energie der Pose Cells wird mit folgender Gleichung berechnet:

$$\Delta P_{x',y',\theta'} = \sum_{i=0}^{S_{xy}-1} \sum_{j=0}^{S_{xy}-1} \sum_{k=0}^{35} P_{i,j,k} \cdot \epsilon_{a,b,c} - \varphi \tag{2.3}$$

S_{xy} ist die Kantenlänge der (x, y)-Ebene des Netzwerkes, wobei die (x, y)-Ebene ein Quadrat darstellt. Die Variable φ gibt die globale Hemmung der Energie der Pose Cells an. Der vierte Schritt besteht in einer Normalisierung des Netzwerkes. In diesem Schritt wird die Energie so normiert, dass die Summe der Energie gleich eins ist. So kann weitere Stabilität in dem Pose Cell Netzwerk erreicht werden. Als Nächstes wird eine Pfadintegration der Translation und Rotation des Roboters vorgenommen. Hierbei wird die Energie im Pose Cell Netzwerk verschoben. Der letzte Schritt besteht darin, das Zentrum der aktivsten Pose Cell Region zu identifizieren, welches für die Experience Map benötigt wird.

2.1.3 Experience Map

Die Experience Map ist eine Karte, die eine Schätzung der Position des Roboters vornimmt. Dafür werden Informationen der Pose Cells und der Local View Cells miteinander kombiniert. Zudem wird die Translation und Rotation des Roboters benötigt, damit die erstellte Karte auch der realen Umgebung des Roboters entspricht. Die Odometriedaten werden durch das System zur Verfügung gestellt und sowohl in den Erfahrungen als auch in den Verbindungen verwendet. Jede Erfahrung in der Experience Map wird als dreier Tupel definiert:

$$e_i = \{P^i, V^i, p^i\} \tag{2.4}$$

Dabei sind P^i und V^i die Zustände in den Pose Cells und Local View Cells jeweils zum Zeitpunkt der Entstehung der Erfahrung. p^i ist der Ort der Erfahrung innerhalb der Experience Map und besteht aus dem akkumulierten Wert für die Translation und Rotation. Es wird eine neue Erfahrung erstellt, sobald die aktuellen Zustände P^i und V^i nicht genau zu einer anderen, bereits existierenden Erfahrung passen. Um dies zu vergleichen, wird der Wert S berechnet:

$$S^i = \mu_p \cdot |P^i - P| + \mu_v \cdot |V^i - V| \tag{2.5}$$

Die beiden Variablen μ_p und μ_v dienen der Gewichtung der Beträge der Zustände für diesen Wert. Wenn $min(S) \geq S_{max}$ zutrifft, wird eine neue Erfahrung erstellt. Die Transition des Roboters wird als Verbindung l_{ij} zwischen der zuletzt aktiven Erfahrungen e_i und der neuen Erfahrung e_j dargestellt:

$$l_{ij} = \{\Delta p^{ij}, \Delta t^{ij}\} \tag{2.6}$$

Δp^{ij} steht für die zurückgelegte Translations- und Rotationsbewegung zwischen den Erfahrungen. Δt^{ij} ist die Zeit, die benötigt wird, um von der zuletzt aktiven Erfahrung zu der neuen Erfahrung zu gelangen. RatSLAM verwendet diese Informationen für die Pfadplanung zwischen der aktuellen Position und dem Zielpunkt. Zur Bestimmung des kürzesten Weges wird der Dijkstra Algorithmus (Knuth (1977)) verwendet. Als Kosten für die Pfadplanung wird die Zeit Δt^{ij} verwendet. Um zu erkennen, ob eine Schleife aufgetreten ist, wird überprüft, ob eine visuelle Szene bereits bekannt ist. Ist dies der Fall, wird überprüft, ob die verknüpften Erfahrungen ähnliche Muster in dem Pose Cell Netzwerk besitzen. Ist dies der Fall, wird eine Verbindung zwischen den beiden Erfahrungen erzeugt und somit die Schleife geschlossen. Ein grafischer Relaxationsalgorithmus verteilt den Fehler der Odometrie. Die Änderung der Position einer Erfahrung ist durch folgende Formel definiert:

$$\Delta p^i = \alpha \cdot \left[\sum_{j=1}^{N_f} (p^j - p^i - \Delta p^{ij}) + \sum_{k=1}^{N_t} (p^k - p^i - \Delta p^{ki}) \right] \tag{2.7}$$

Dabei ist α eine Korrekturgeschwindigkeitskonstante, welche einen Wert von 0.5 besitzt. Der Wert α ist durch viele Tests der Autoren bestimmt worden. Die Variable N_f ist die Anzahl an Links von der aktuellen Erfahrung e_i zu anderen Erfahrungen und N_t ist die Anzahl der Links von anderen Erfahrungen zu der aktuellen Erfahrung e_i.

2.2 Abgrenzung zu anderen SLAM Algorithmen

RatSLAM unterscheidet sich von anderen SLAM Algorithmen in der Datenerfassung und -verarbeitung sowie der Darstellung der Karte. In der Regel erfordern SLAM Algorithmen als Eingabewerte Informationen über die Translation des Roboters und Laser-Scanner-Daten. In manchen Fällen wird auch die Rotation des Roboters beziehungsweise ein Kamerabild benötigt. Diese Daten werden zusammengeführt, um eine exakte Karte der Umgebung zu erzeugen. Dabei handelt es sich in der Regel um geometrische Karten, wie es in Abbildung 2.5 dargestellt ist. Mit jeder Lokalisation besitzt der Roboter genaue Informationen über seine Umgebung, zum Beispiel Abstände zu Wänden und Hindernissen. Hierbei werden sehr große Datenmengen erzeugt. Dabei wird es mit fortschreitender Zeit immer aufwendiger die Karte zu aktualisieren sowie die Position des Roboters zu bestimmen. Aus diesem Grund wird oftmals keine komplette Karte der Umgebung erstellt, sondern nur für einen bestimmten Radius um den Roboter herum.

Die Datengrundlage von RatSLAM ist ein Kamerabild sowie die Translation und Rotation des Roboters. Es werden keine Informationen eines Laser Scanners oder eines ähnlichen Sensors benötigt. Mithilfe dieser Informationen baut RatSLAM eine topologische Karte der zurückgelegten Strecke auf, wie es in Abbildung 2.6 dargestellt ist. Durch diesen Umstand besitzt RatSLAM keine Kenntnis über seine Umgebung. Dies bedeutet, dass RatSLAM nicht weiß, wo Wände oder Hindernisse stehen. RatSLAM kann nur Vorhersagen über die bereits bekannte Streckenführung machen. Der positive Aspekt an dieser Tatsache ist, dass RatSLAM mit einer sehr geringen Datenmenge auskommt. Darüber hinaus besitzt RatSLAM eine Echtzeitfähigkeit. Dies bedeutet, dass der Berechnungsaufwand nahezu konstant bleibt, wenn eine Strecke komplett kartografiert ist. Die Berechnungszeit steigt nur an, wenn unbekannte visuelle Szenen erkannt und eingepflegt werden.

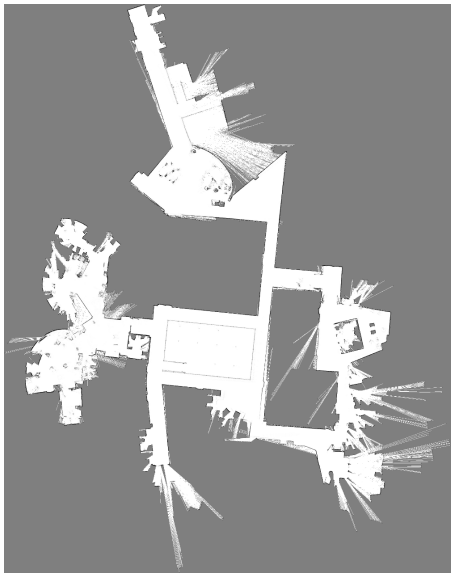

Abbildung 2.5: Karte der meisten anderen SLAM Algorithmen (Quelle: Holz und Behnke (2009))

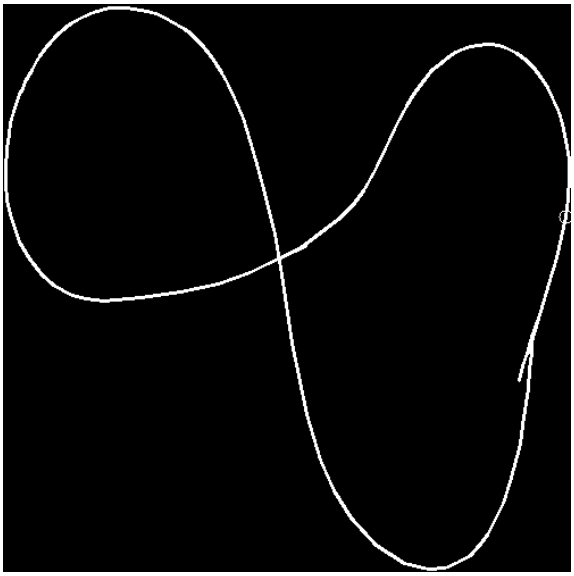

Abbildung 2.6: Mit OpenRatSLAM erstellte Karte

3 Definition und Wahrnehmung dynamischer Umgebungen

In diesem Kapitel wird eine Definition der dynamischen Umgebung vorgenommen. Es werden kurz die betreffenden Ausschnitte des Regelwerks vom Carolo-Cup erklärt. Darüber hinaus wird auch aufgezeigt, welche möglichen Folgen ein Fehlverhalten besitzt. Weiterhin wird beschrieben, welche Sensorik auf der Fahrzeugplattform Carolo-Cup verbaut ist. Es wird kurz auf jeden Sensor eingegangen und sowohl Vor- als auch Nachteile aufgezeigt.

3.1 Definition der dynamischen Umgebung

Der studentische Wettbewerb Carolo-Cup (Carolo-Cup) der Technischen Universität Braunschweig besteht aus vier Disziplinen (Carolo-Cup-Regelwerk). Die Disziplin, auf die spezieller eingegangen wird, ist der *Rundkurs mit Hindernissen*. In dieser Disziplin müssen sowohl feststehende als auch fahrende Hindernisse erkannt und überholt werden. Zusätzlich muss eine Stopp-Linie erkannt werden. Eine Stopp-Linie zeigt eine Kreuzung an. Das Fahrzeug muss an einer Kreuzung mindestens zwei Sekunden warten, bevor es weiterfahren darf. Beim Anfahren muss das Fahrzeug überprüfen, ob sich von rechts ein fahrendes Hindernis nähert. Ist dies der Fall, so muss dem fahrenden Hindernis die Vorfahrt gewährt werden.

Die Definition der dynamischen Umgebung, die für diese Arbeit angewendet wird, orientiert sich weitestgehend an den Regeln des Carolo-Cups. Die Streckenführung der Teststrecke ist in der Abbildung 3.1 dargestellt. Jede Fahrspur besitzt eine Breite von 40 Zentimetern. Alle Fahrbahnmarkierungen sind zwei Zentimeter breit. Der Mittelstreifen ist 20 Zentimeter lang. Zwischen zwei Mittelstreifen befindet sich eine Lücke mit einer Länge von 20 Zentimetern. Eine stellenweise Unterbrechung der Fahrbahnmarkierung wird nicht vorgenommen für diese Arbeit nicht vorgenommen. Sie ist allerdings Bestandteil in dem Carolo-Cup Regelwerk.

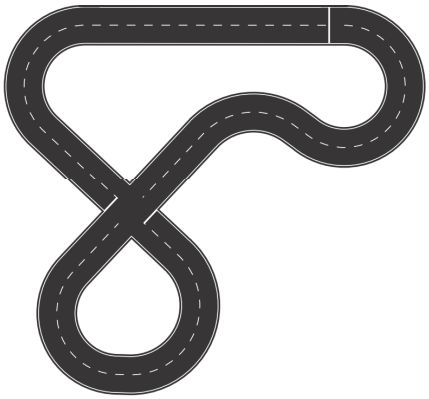

Abbildung 3.1: Streckenaufbau im Raum 13.62, Berliner Tor 7

In der dynamischen Umgebung existieren sowohl feststehende als auch fahrende Hindernisse. Feststehende Hindernisse bewegen sich nicht, besitzen aber keine feste Position. Dies bedeutet, dass sich feststehende Hindernisse überall auf der Strecke befinden können. Darüber hinaus befinden sich die feststehenden Hindernisse mittig auf der Fahrspur. Diese Hindernisse können sich sowohl auf der linken als auch rechten Fahrspur befinden. Fahrende Hindernisse bewegen sich auf der Strecke und besitzen einen festen Start- sowie Endpunkt. Diese Art von Hindernissen kommt nur auf der rechten Seite vor. Sie verhalten sich wie normale Verkehrsteilnehmer. Es existiert kein Gegenverkehr in der dynamischen Umgebung. Sowohl das feststehende als auch das fahrende Hindernis sind weiße Quader mit einer Breite von 10 bis 40 Zentimetern, einer Höhe von 10 bis 24 Zentimetern und einer Länge von mindestens 10 Zentimetern, folglich Abbildung 3.2.

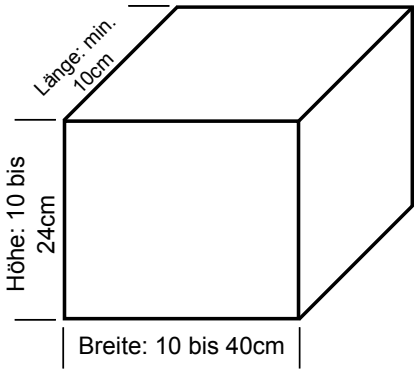

Abbildung 3.2: Definition eines Hindernisses

Neben Hindernissen existieren auch Kreuzungen. Die Aufgabe an einer Kreuzung besteht zum einen in der Erkennung einer Stopp-Linie, an der das Fahrzeug für eine bestimmte Zeit halten muss. Die Stopp-Linie ist vier Zentimeter breit, ebenso wie die Start-Linie. Zum Anderen muss während der Wartezeit überprüft werden, ob sich rechts vom Fahrzeug ein fahrendes Hindernis befindet. Ist dies der Fall, so muss dem fahrenden Hindernis die Vorfahrt gewährt werden. Dabei gibt es einen fest definierten Bereich, in dem sich das fahrende Hindernis befinden muss, um Vorfahrt zu erhalten. Dieser Bereich ist in Abbildung 3.3 dargestellt. Sollte sich das Fahrzeug anstelle des Hindernisses befinden und ein fahrendes Hindernis steht links oder rechts an der Stopp-Linie, so besitzt das Fahrzeug Vorfahrt.

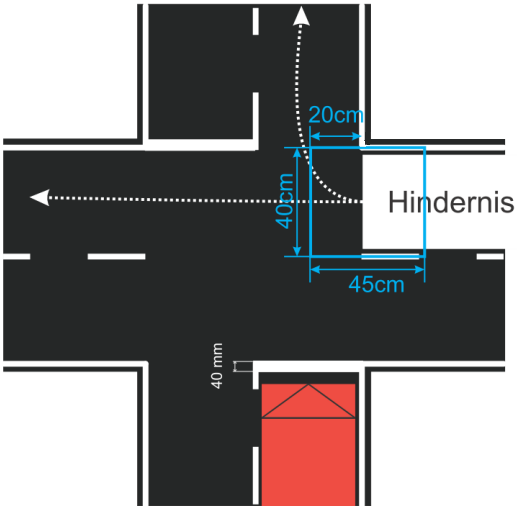

Abbildung 3.3: Definition einer Kreuzung

3.2 Wahrnehmung dynamischer Umgebungen

Zur Wahrnehmung von Umgebungen existieren diverse Sensorarten. Die am meisten verwendeten Arten sind Ultraschall- und Infrarot-Sensoren sowie Laser Scanner beziehungsweise Kameras. Darüber hinaus werden auch Inkremental- oder Hallsensoren zur Translationsbestimmung verwendet. Wenn die Bestimmung einer Rotation nötig ist, kann dies über Gyroskop-Sensoren oder Inertial Measurement Units, kurz IMU, erfolgen. Jede dieser Sensorarten besitzt Vor- und Nachteile.

Ultraschall-Sensoren messen deutlich schneller als Infrarot-Sensoren. Treten allerdings ungewollte Reflexionen des Schalls auf, sind die Messergebnisse nicht mehr verwertbar. Darüber hinaus erfassen Ultraschall-Sensoren keine abgeschrägten Objekte, da diese den Schall umlenken und nicht zum Sensor zurückschicken. Solch ein Verhalten besitzen Infrarot-Sensoren nicht. Infrarot-Sensoren benötigen zwar mehr Zeit pro Messung (circa 35 Millisekunden), allerdings wird das Infrarot-Licht nicht von abgeschrägten Kanten derart stark abgelenkt. Somit ist trotz abgeschrägter Kante eines Hindernisses eine Messung möglich. Laser Scanner messen bis auf wenige Zentimeter über einen großen Bereich sehr genau. Darüber hinaus geben Laser Scanner für jeden Grad in ihrem Messbereich eine Distanz an. Diese zwei Eigenschaften sind der Grund, warum Laser Scanner keine besonders hohe Messfrequenz besitzen. In der Regel misst ein Laser Scanner mit 12 bis 15 Hertz. Zudem ist ein Laser Scanner um ein Vielfaches teurer als die beiden anderen Sensorarten. Neben einer sehr hohen Bildwiederholfrequenz besitzt eine Kamera auch negative Eigenschaften. Da der Untergrund auf der Fahrbahn schwarz und leicht glänzend ist, werden schräg einfallende Lichtquellen sehr stark reflektiert und von der Kamera als weißer Fleck wahrgenommen. Auch eine Binarisierung des Bildes schafft in solch einem Fall keine Abhilfe. In solch einem Fall muss mit einer Sensor-Fusion gearbeitet werden.

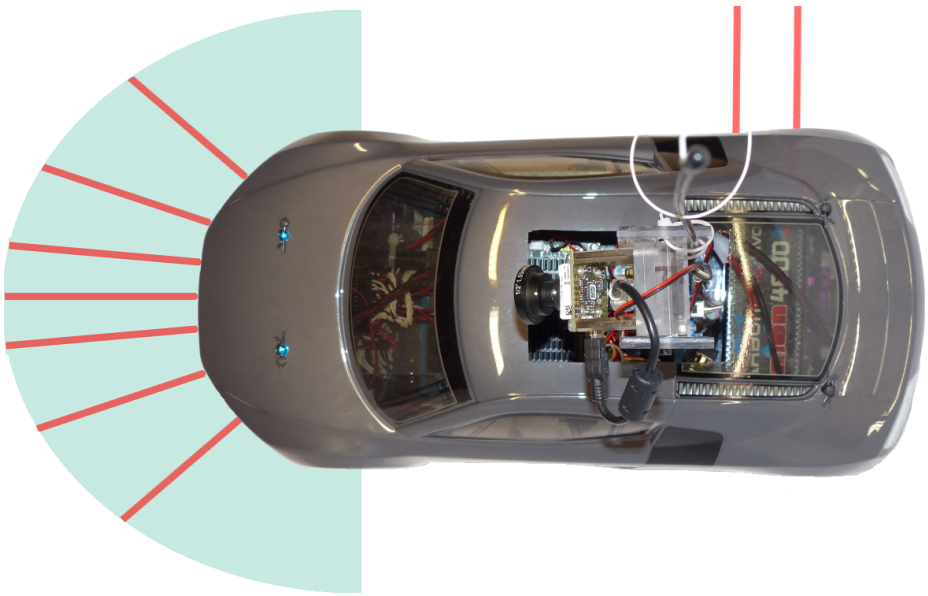

Abbildung 3.4: Sensorkonzept der Fahrzeugplattform Carolo-Cup v.2 zur Wahrnehmung der dynamischen Umgebung

Die Fahrzeugplattform Carolo-Cup v.2 besitzt zur Wahrnehmung dynamischer Umgebungen eine Kamera von IDS mit der Bezeichnung *UI-1221LE* mit einem Weitwinkelobjektiv. Dieses Objektiv hat einen Öffnungswinkel von 150 Grad diagonal. Zusätzlich sind an der Front des Fahrzeuges sieben analoge Infrarot-Sensoren des Typs *GP2Y0A02YK* von Sharp installiert, welche eine Reichweite von 20 bis 150 Zentimetern besitzen. Die Sensoren sind ausgehend von der Mitte in einem Winkel von 10, 30 und 60 Grad angeordnet. Hinten rechts ist ein digitaler Infrarot-Sensor des Typs *GP2Y0D340K* von Sharp befestigt, der bei einer Distanz von weniger als 50 Zentimetern eine eins ausgibt. Ist die Distanz größer als 50 Zentimeter, liefert der Infrarot-Sensor eine null. Zudem befindet sich ebenfalls ein Infrarot-Sensor von Sharp des Typs *GP2Y0A21YK0F* an der rechten Fahrzeugseite. Dieser Sensor misst im Bereich von zehn bis 80 Zentimetern. Der Sensor wird als Rückfallebene verwendet, damit der digitale Sensor nicht durch Gegenstände oder Personen ausgelöst wird, die sich am Fahrbahnrand aufhalten. Die Messung der Translation erfolgt mittels der Hall-Sensoren am Motor. Dieser besitzt 42 Ticks pro Umdrehung, wodurch sich bei einem Radumfang von 20 Zentimetern eine Auflösung von 0.47 Zentimetern ergibt. Zur Erfassung der Rotation sowie von Beschleunigungskräften des Fahrzeuges wird die Inertial Measurement Unit *x-BIMU* der Firma x-io Technologies eingesetzt.

4 Algorithmus zur Hindernis- und Kreuzungserkennung

Damit eine durch RatSLAM erstellte Karte aufgewertet wird, wird eine Erkennung für Hindernisse und Kreuzungen benötigt. Diese Erkennung erfolgt zum einen durch einen optischen Sensor. Zum anderen werden Infrarot Sensoren eingesetzt, um nach der optischen Identifizierung eines Hindernisses diese zu verifizieren. Dabei wird ein Verfahren beschrieben, welches möglichst wenig Berechnungszeit benötigt. Zusätzlich werden erfahrungsgemäß gute Werte für Parameter zur Erkennung von Hindernissen und Kreuzungen dargestellt.

4.1 Region of Interest zur Erkennung von Hindernissen und Kreuzungen

Die Kamera erstellt Bilder mit einer maximalen Auflösung von 752 x 480 Bildpunkten. Somit besteht ein Bild aus 752 Spalten und 480 Reihen, was insgesamt 360.960 Bildpunkten entspricht. Ein Bild B lässt sich als Matrix darstellen:

$$\begin{aligned} n &: \text{Anzahl Bildspalten} \\ m &: \text{Anzahl Bildreihen} \\ B &: \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \end{aligned} \tag{4.1}$$

Der Ursprung eines Bildes ist definiert als a_{11} beziehungsweise $(0, 0)$. Weiter ist definiert, dass der Index y für die Bildreihe und der Index x für die Bildspalte steht.

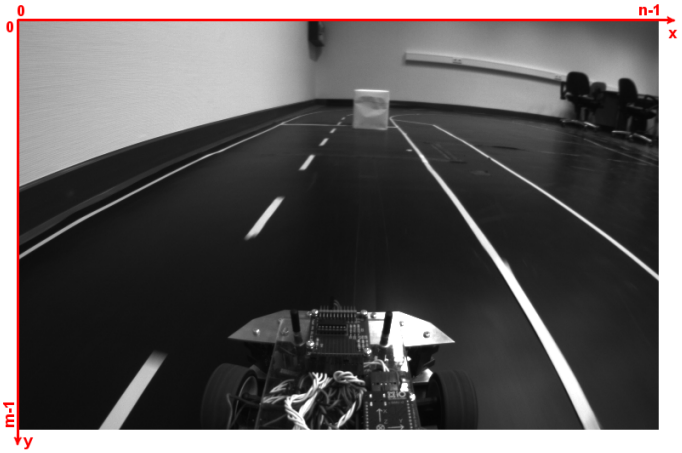

Abbildung 4.1: Beschreibung eines Bildes als Matrix

Der Farbraum eines Bildpunktes beträgt acht Bit, wobei es sich um eine monochrome Kamera handelt. Dies bedeutet, dass der Wert 0 gleichbedeutend mit der Farbe schwarz ist. Die Farbe weiß ist definiert als ein Wert von 255. Somit besteht der Wert eines Bildpunktes aus folgender Menge:

$$G : \{0, \cdots, 255\} \tag{4.2}$$

Um den Wert beziehungsweise Grauwert an einer bestimmten Bildkoordinate (y, x) zu erhalten, wird folgende Funktion definiert:

$$f : B \to G, (y, x) \mapsto g, \text{ wobei } g \in G \tag{4.3}$$

Beispielhaft sind in der Gleichung 4.4 die ersten Grauwerte der Abbildung 4.1 aufgeschrieben. Der Grauwert von $f(0, 0)$ beträgt 160.

$$B : \begin{pmatrix} 160 & \cdots & f(1,n) \\ 162 & \cdots & f(2,n) \\ \vdots & \ddots & \vdots \\ f(m,1) & \cdots & f(m,n) \end{pmatrix} \tag{4.4}$$

Regions of Interest, kurz ROI, bezeichnen Bereiche innerhalb eines Bildes, in denen nach bestimmten Informationen mit besonderem Interesse gesucht wird. Die Verwendung einer Region of Interest besitzt mehrere Vorteile. Durch die Verkleinerung des Bereiches, in dem nach Informationen gesucht wird, erfolgt eine Beschleunigung des Suchvorganges.

Die Beschleunigung erfolgt durch die Verarbeitung von weniger Bildpunkten. Durch die Präzisierung des Suchbereiches findet eine Verringerung von ungewollten Erkennungen in nicht relevanten Bereichen statt.

Je nach Einsatzzweck und Anforderung findet eine Verfeinerung der Region of Interest statt. Sollen Linien oder Objekte erkannt werden, wird mit Suchlinien (Abbildung 4.2), die vertikal verlaufen, nach den Mustern gesucht. Dies ist ein übliches Verfahren. Zur Erkennung von Linien wird in einem Graustufen-Bild mit schwarzem Untergrund und weißen Fahrbahnmarkierungen nach dem Schema schwarz-weiß und drauf folgend weiß-schwarz gesucht. Um Objekte zu erkennen, werden diese Suchlinien dahin gehend überprüft, wie viele der Bildpunkte über einen bestimmten Schwellwert liegen. Sind genug Bildpunkte über dem Schwellwert, wird angenommen, dass sich die Suchlinien auf einem Objekt befinden. Ein alternatives Verfahren, das häufig eingesetzt wird, stellen Punktwolken (Abbildung 4.3) dar. Um mittels Punktwolken Objekte zu detektieren, wird ebenfalls überprüft, wie viele der Punkte über dem definierten Schwellwert liegen. Soll allerdings eine Erkennung von Linien mittels Punktwolken entwickelt werden, so stellt dies eine größere Aufgabe da. In so einem Fall muss eine Mustererkennung durchgeführt werden, die nicht trivial ist.

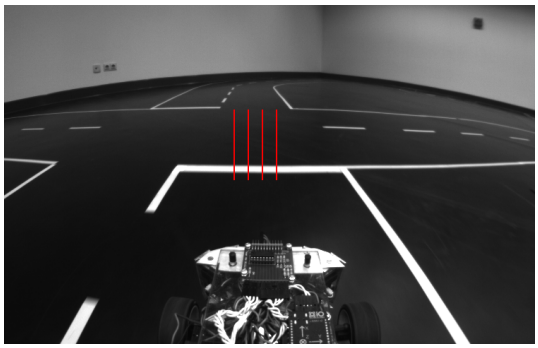

Abbildung 4.2: ROI mit Suchlinien

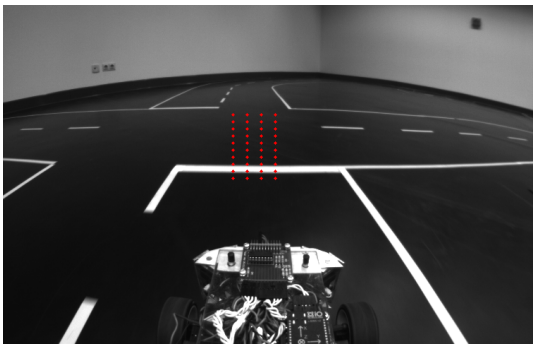

Abbildung 4.3: ROI mit Punktwolke

Jedes der beiden Erkennungsverfahren besitzt sowohl Vor- als auch Nachteile. Die Suchlinien ermöglichen eine leichte Erkennung von Linien als auch Objekten. Der Berechnungsaufwand für die in Abbildung 4.2 dargestellten Suchlinien ist im Vergleich zu den Punktwolken größer. Jede Suchlinie besteht aus 100 Bildpunkten. Insgesamt müssen 400 Bildpunkte überprüft werden. Für die Erkennung von Objekten und Linien werden unterschiedliche Verfahren angewendet. Somit erfolgt eine Überprüfung der

Suchlinien zwei Mal. Die Punktwolke in Abbildung 4.3 besteht aus zehn Bildpunkten pro Reihe mit vier Reihen. Der Berechnungsaufwand ist kleiner, allerdings steigt die Komplexität der Erkennung von Linien und Objekten an.

Um das Verfahren jedoch noch weiter zu beschleunigen, wird in diesem Kapitel eine Alternative zu beiden oben genannten Verfahren entwickelt. Diese Alternative stellt die maximale Minimierung der Region of Interest auf einen Punkt pro Fahrspur dar. Dieser Point of Interest befindet sich sowohl links als auch rechts auf der Fahrbahn. Somit ist eine Erkennung von Hindernissen auf beiden Fahrspuren gegeben. Die beiden Points of Interest werden anhand der approximierten Polynome von Polaris (Jenning (2008)) berechnet. Dies ist in Abbildung 4.4 dargestellt. Polaris approximiert jede Fahrbahnmarkierung mit Polynomen dritten Grades. Zunächst muss allerdings eine Entzerrung und projektive Transformation des Bildes erfolgen (Manske (2008)). Dabei wird das Bild B auf das entzerrte Bild E abgebildet:

$$h : B \to E, (y, x) \mapsto (y', x') \tag{4.5}$$

Das entzerrte Bild E wird wiederum durch eine projektive Transformation auf das Fahrzeugkoordinatensystem F abgebildet:

$$p : E \to F, (y', x') \mapsto (y'', x'') \tag{4.6}$$

Diese Schritte sind notwendig, da die Polynome die Fahrspurmarkierungen im Fahrzeugkoordinatensystem approximieren. Darüber hinaus wird in diesem Koordinatensystem auch der Point of Interest bestimmt.

Zur Berechnung der beiden Points of Interest werden alle drei approximierten Polynome $A(x'')$, $B(x'')$ und $C(x'')$ verwendet:

$$A(x'') = a_A \cdot x''^2 + b_A \cdot x'' + c_A \tag{4.7}$$

$$B(x'') = a_B \cdot x''^2 + b_B \cdot x'' + c_B \tag{4.8}$$

$$C(x'') = a_C \cdot x''^2 + b_C \cdot x'' + c_C \tag{4.9}$$

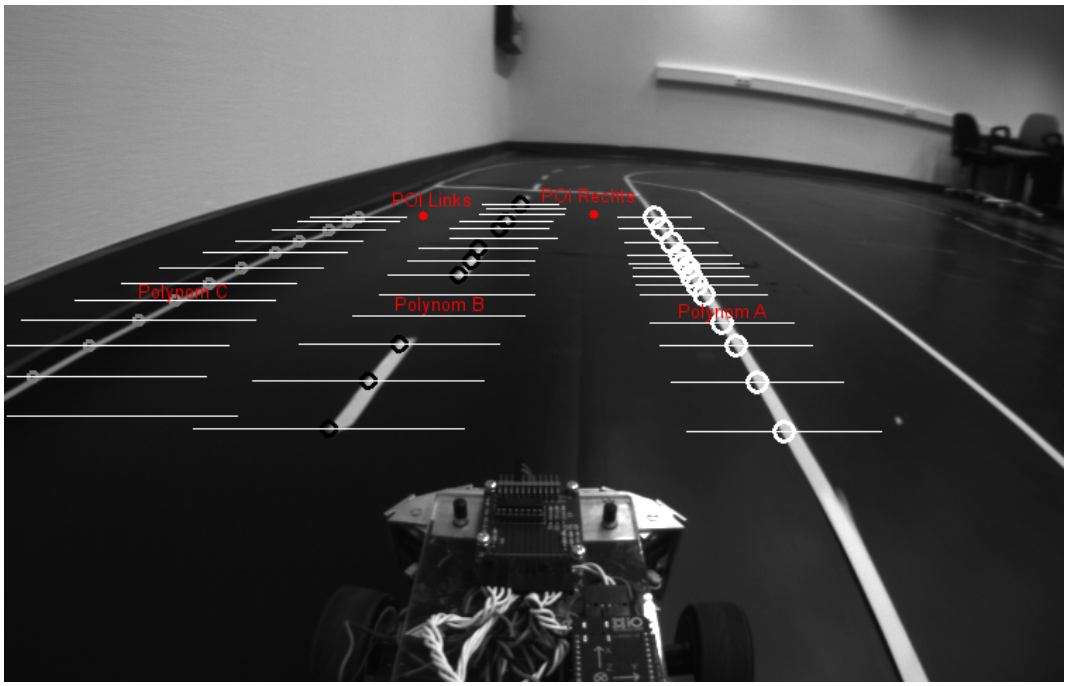

Abbildung 4.4: Berechnte Points of Interest mittels approximierter Polynome

Ein Point of Interest stellt zunächst einen Punkt im Fahrzeugkoordinatensystem dar. Später findet eine Abbildung zurück zum Koordinatensystem des Bildes statt. Somit besteht ein Point of Interest aus einer y'' und x'' Koordinate. Ein Point of Interest ist folgendermaßen definiert:

$$POI_{left} = (y''_l, x'') \tag{4.10}$$

$$POI_{right} = (y''_r, x'') \tag{4.11}$$

Dabei ist der Wert von x'' abhängig von der Steigung des Polynoms an der Stelle x'' der Polynome $A(x'')$, $B(x'')$ und $C(x'')$ (Gleichungen 4.14 bis 4.16).

$$x'' = \begin{cases} \delta_s \text{ falls } isCurve = false \\ \delta_c \text{ falls } isCurve = true \end{cases} \tag{4.12}$$

Dabei sind δ_c und δ_s vom Benutzer einzustellende Werte, die die Entfernung des Points of Interest von der Fahrzeugmitte festlegen. Erfahrungsgemäß sind folgende Wertebereiche sinnvoll:

$$\begin{aligned} \text{Entfernung (cm) Kurve: } 60 \leq \delta_c \leq 70 \\ \text{Entfernung (cm) Gerade: } 110 \leq \delta_s \leq 120 \end{aligned} \tag{4.13}$$

In beiden Gleichungen bestimmt die Variable x den Abstand des Points of Interest vom Fahrzeug. Für x werden zwei Werte definiert: die Entfernung auf geraden Abschnitten δ_s sowie die Entfernung in Kurven δ_c. Die Entfernungen hängen mit den approximierten Polynomen zusammen. In einer Kurve wird das Polynom bis etwa 70 Zentimeter gut approximiert. Wird die Entfernung weiter erhöht, so krümmt sich das Polynom zu stark und der Point of Interest wird nicht mehr zwischen den beiden Fahrbahnmarkierungen positioniert. Das Gleiche gilt bei der Approximation der Polynome auf einem geraden Abschnitt. Um den booleschen Ausdruck $isCurve$ zu bestimmen, wird die Steigung der Polynome an der Stelle x benötigt. Die Steigung der Polynome $A(x'')$, $B(x'')$ und $C(x'')$ wird wie folgt berechnet:

$$\nabla A(x'') = 2 \cdot a_A \cdot x'' + b_A \tag{4.14}$$

$$\nabla B(x'') = 2 \cdot a_B \cdot x'' + b_B \tag{4.15}$$

$$\nabla C(x'') = 2 \cdot a_C \cdot x'' + b_C \tag{4.16}$$

Anhand der Steigung wird bestimmt, ob $isCurve$ wahr ist oder nicht. $isCurve$ ist wie folgt definiert:

$$isCurve = \begin{cases} true \text{ falls } -0.17 \leq \nabla A(x'') \leq 0.17 \\ true \text{ falls } -0.17 \leq \nabla B(x'') \leq 0.17 \\ true \text{ falls } -0.17 \leq \nabla C(x'') \leq 0.17 \\ false \text{ sonst} \end{cases} \tag{4.17}$$

Die Gleichungen für y_l'' und y_r'' bestimmen die Position des Points of Interest auf der vertikalen Achse. Durch die Polynome $A(x'')$, $B(x'')$ und $C(x'')$ werden Punkte auf der Fahrbahnmarkierung approximiert. Dabei kann es vorkommen, dass eines der Polynome

nicht korrekt approximiert wird. Ein Grund für eine fehlerhafte Approximation ist die Erkennung von zu wenig Punkten auf der Fahrbahnmarkierung. Ein solcher Fall kommt in Kurven sowie auf Kreuzungen vor. Eine weitere Möglichkeit, in denen ein Polynom nicht richtig approximiert wird, besteht in fehlenden Fahrbahnmarkierungen. Besonders die mittlere Fahrbahnmarkierung erschwert eine korrekte Approximation durch die immer wieder auftretenden Unterbrechungen. Aus diesem Grund sind sechs Fallunterscheidungen für die Berechnung der y-Koordinaten des Points of Interest definiert worden:

$$y_l'' = \begin{cases} \frac{B(x'') + C(x'')}{2} & \text{falls } \nu_C, \nu_B = true \wedge 35 \leq \delta_{BC} \leq 55 \\ \frac{B(x'') + C(x'')}{2} & \text{falls } \nu_C, \nu_B = true \wedge 45 \leq \delta_{BC} \leq 70 \wedge isCurve \\ C(x'') + \delta_e & \text{falls } \nu_B = false \wedge \nu_C = true \\ B(x'') - \delta_e & \text{falls } \nu_B = true \wedge \nu_C = false \\ A(x'') - (3 \cdot \delta_e) & \text{falls } \nu_A = true \wedge \nu_B, \nu_C = false \\ y_{old}'' & \text{falls } \nu_A, \nu_B, \nu_C = false \end{cases} \tag{4.18}$$

$$y_r'' = \begin{cases} \frac{A(x'') + B(x'')}{2} & \text{falls } \nu_A, \nu_B = true \wedge 35 \leq \delta_{AB} \leq 55 \\ \frac{A(x'') + B(x'')}{2} & \text{falls } \nu_A, \nu_B = true \wedge 45 \leq \delta_{AB} \leq 70 \wedge isCurve \\ A(x'') - \delta_e & \text{falls } \nu_A = true \wedge \nu_B = false \\ B(x'') + \delta_e & \text{falls } \nu_A = false \wedge \nu_B = true \\ C(x'') + (3 \cdot \delta_e) & \text{falls } \nu_C = true \wedge \nu_A, \nu_B = false \\ y_{old}'' & \text{falls } \nu_A, \nu_B, \nu_C = false \end{cases} \tag{4.19}$$

Dabei ist δ_e ein Wert, der vom Benutzer einzustellen ist. δ_e gibt den Abstand von der rechten Fahrbahnmarkierung zur Mitte der Fahrspur an. Die Einheit ist Zentimeter und sollte folgenden Wertebereich besitzen:

$$18 \leq \delta_e \leq 22 \tag{4.20}$$

In den ersten beiden Fälle wird der Mittelpunkt zwischen zwei der drei Polynome berechnet. Dies gilt nur, wenn der Abstand δ_{AB} beziehungsweise δ_{BC} zwischen den

beiden Polynomen einer bestimmten Entfernung entspricht. Befindet sich das Fahrzeug auf einem geraden Abschnitt, sollte dieser Abstand geringer sein, als wenn sich das Fahrzeug in einer Kurve befindet. Der Grund für diese Unterscheidung liegt darin, dass in einer Kurve die Fahrbahnmarkierungen leicht auseinander gleiten und somit ein größerer Abstand berücksichtigt werden muss. Die Abstände δ_{AB} beziehungsweise δ_{BC} zwischen den Polynomen $A(x'')$, $B(x'')$ und $C(x'')$ werden anhand folgender Gleichungen berechnet:

$$\delta_{BC} = \sqrt{(C(x'') - B(x''))^2 + (x'' - x'')^2} = \sqrt{(C(x'') - B(x''))^2} \tag{4.21}$$

$$\delta_{AB} = \sqrt{(A(x'') - B(x''))^2 + (x'' - x'')^2} = \sqrt{(A(x'') - B(x''))^2} \tag{4.22}$$

Darüber hinaus müssen für die beiden ersten Fälle beide Polynome als gültig deklariert sein. Die Gültigkeit der Polynome $A(x'')$, $B(x'')$ und $C(x'')$ werden durch die booleschen Variablen ν_A, ν_B und ν_C definiert:

$$\nu_A = \begin{cases} true \text{ wenn } \dfrac{\varepsilon_{EP,A}}{\varepsilon_{SL,A}} \geq \tau_{valid} \\ false \text{ sonst} \end{cases} \tag{4.23}$$

$$\nu_B = \begin{cases} true \text{ wenn } \dfrac{\varepsilon_{EP,B}}{\varepsilon_{SL,B}} \geq \tau_{valid} \\ false \text{ sonst} \end{cases} \tag{4.24}$$

$$\nu_C = \begin{cases} true \text{ wenn } \dfrac{\varepsilon_{EP,C}}{\varepsilon_{SL,C}} \geq \tau_{valid} \\ false \text{ sonst} \end{cases} \tag{4.25}$$

Die Werte $\varepsilon_{EP,A}$, $\varepsilon_{EP,B}$ und $\varepsilon_{EP,C}$ geben die Anzahl gefundener Punkt auf der Fahrbahnmarkierung an. Die Werte $\varepsilon_{SL,A}$, $\varepsilon_{SL,B}$, $\varepsilon_{SL,C}$ und τ_{valid} werden vom Benutzer eingestellt. $\varepsilon_{SL,A}$, $\varepsilon_{SL,B}$ und $\varepsilon_{SL,C}$ geben die Anzahl Scanlines für Polaris an. τ_{valid} ist ein Schwellwert, ab dem das Polynom als valide und stabil gilt. τ_{valid} wird in Prozent angegeben. Erfahrungsgemäß sollten folgende Wertebereiche eingehalten werden:

$$\begin{aligned} \text{Anzahl Scanlines: } 12 \leq \varepsilon_{SL,A}, \varepsilon_{SL,B}, \varepsilon_{SL,C} \leq 14 \\ \text{Schwellwert: } \tau_{valid} = 0.1 \end{aligned} \tag{4.26}$$

Die Werte $\varepsilon_{SL,A}$, $\varepsilon_{SL,B}$ und $\varepsilon_{SL,C}$ sollten nicht größer als 14 werden, da ansonsten der Berechnungsaufwand zu groß wird.

Der dritte Fall definiert, dass das mittlere Polynom B als nicht gültig deklariert wird und somit eine selbst definierte Distanz δ_e herangezogen wird. Der vierte Fall ist die Umkehrung des dritten Falles. In dieser Situation sind jeweils die äußeren Polynome $A(x'')$ und $C(x'')$ nicht gültig. Sollten beide relevanten Polynome für eine Fahrspur als nicht gültig deklariert werden, wird das gegenüberliegende Polynom verwendet. Bei dem linken Point of Interest wäre dies Polynom $A(x'')$, bei dem rechten Point of Interest ist es das Polynom $C(x'')$. Dazu wird das Dreifache von δ_e subtrahiert beziehungsweise addiert, damit der Point of Interest sich wieder auf der korrekten Fahrspur befindet. Wenn alle drei Polynome als ungültig deklariert werden, wird der zuletzt korrekt berechnete Wert y_{old} erneut verwendet.

Ist der Point of Interest im Fahrzeugkoordinatensystem bestimmt, muss eine Rücktransformation zum Bildkoordinatensystem erfolgen:

$$p' : F \to E, (y'', x'') \mapsto (y', x') \tag{4.27}$$

$$h' : E \to B, (y', x') \mapsto (y, x) \tag{4.28}$$

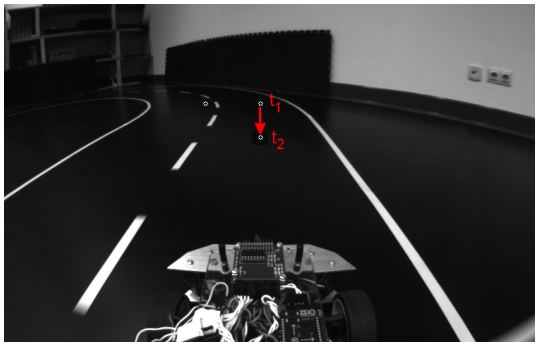

Abbildung 4.5: Verschiebung des Points of Interest vor Kurvenfahrt

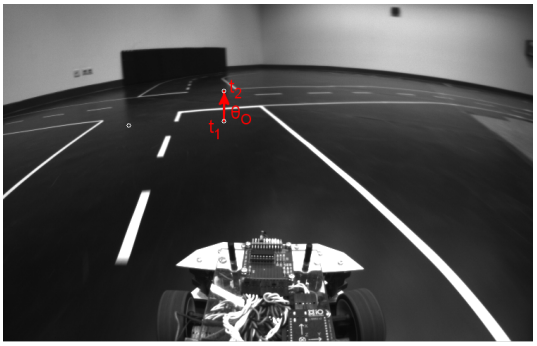

Abbildung 4.6: Verschiebung des Points of Interest nach Kurvenfahrt

Wird der Point of Interest näher an das Fahrzeug herangesetzt, wenn eine Kurve beginnt, ergeben sich daraus keine weiteren Folgen. Dies ist in Abbildung 4.5 dargestellt. Dabei wird der POI von seiner derzeitigen Position zum Zeitpunkt t_1 zu der neuen

Position zum Zeitpunkt t_2 verschoben. Diese Differenz zwischen den beiden Positionen wird in den nächsten Iterationen erneut überprüft. Verlässt das Fahrzeug eine Kurve, dann wird der Point of Interest weiter vom Fahrzeug weg gesetzt. Dies ist in Abbildung 4.6 dargestellt. Der POI zum Zeitpunkt t_1 wird um die Distanz θ_O zum Zeitpunkt t_2 versetzt. Dies bedeutet, dass die Strecke θ_O erneut untersucht werden muss, damit keine Linien oder Hindernisse übersehen werden:

$$POI_n = (y'' - n, x'') \text{ für alle } n \in \{0, \cdots, \theta_O\} \tag{4.29}$$

4.2 Berechnung des Grauwert-Schwellwertes

Die Fahrzeugplattform Carolo-Cup v.2 verfügt über eine Graustufen-Kamera. Jeder Bildpunkt wird durch acht Bit repräsentiert wird. Der Wert 0 eines Bildpunktes bedeutet, dass dieser Pixel schwarz ist. Bei einem Wert von 255 ist der Bildpunkt weiß. Um eine Differenzierung zwischen einem Hindernis beziehungsweise einer Linie und der Straße zu erhalten, wird ein Schwellwert benötigt. Dieser Schwellwert bestimmt, ab welchem Wert ein Pixel als schwarz beziehungsweise weiß gilt. Laut der Definition der Umgebung in Kapitel 3.1 ist der Boden schwarz, Linien und Hindernisse sind dagegen weiß. Dabei ist die Wahl des richtigen Schwellwertes wichtig. Ist der Schwellwert zu niedrig eingestellt, werden zu viele Bildpunkte fälschlicherweise als weiß dargestellt. Dies ist in Abbildung 4.8 dargestellt. Durch diesen Umstand wird die korrekte Ausführung eines Erkennungsalgorithmus erschwert, da auch leichte Reflexionen der Deckenbeleuchtung auf der Straße als weiß deklariert werden. Sollte der Schwellwert zu hoch gewählt werden (Abbildung 4.9), so werden deutlich zu wenig Bildpunkte als weiß markiert, obwohl diese eigentlich als weiß markiert werden müssten. Hierdurch wird eine Erkennung ebenfalls erschwert.

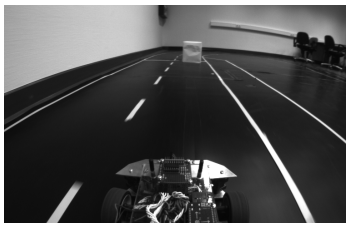

Abbildung 4.7: Originalbild

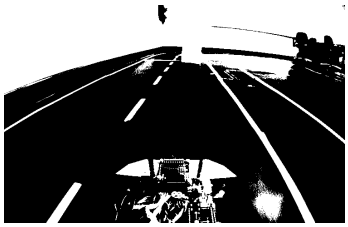

Abbildung 4.8: Schwellwert bei 60

Abbildung 4.9: Schwellwert bei 210

Abbildung 4.10: Hellere Umgebung

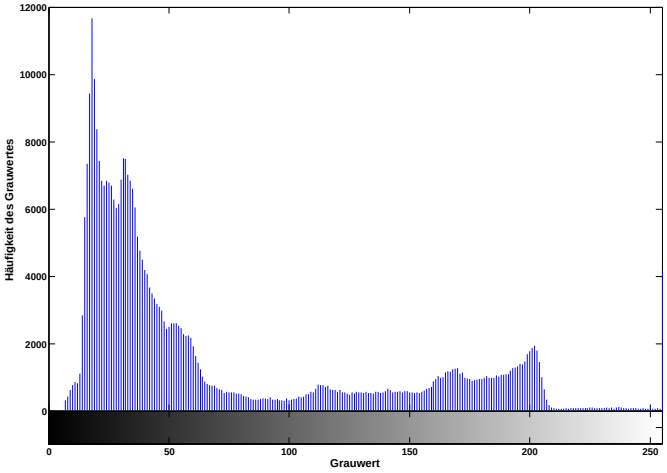

Abbildung 4.11: Histogramm zu Abb. 4.10

Abbildung 4.12: Dunklere Umgebung

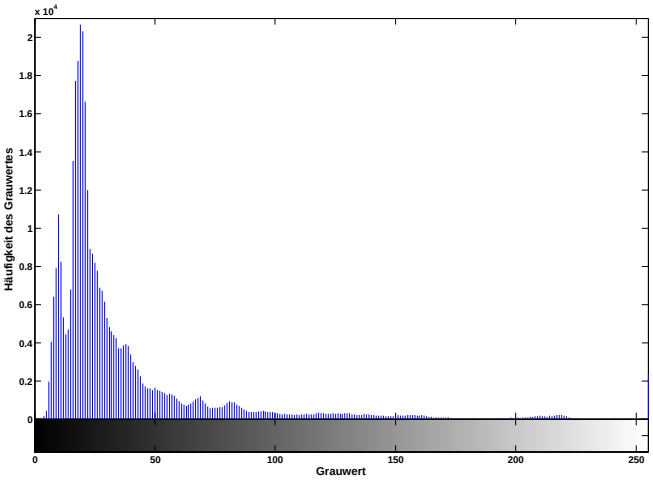

Abbildung 4.13: Histogramm zu Abb. 4.12

Um diesen Schwellwert zu bestimmen, existieren diverse Verfahren. Die trivialste Methode besteht in der Bestimmung eines festen Schwellwertes. Hierbei wird entweder ein Schwellwert nach eigener Einschätzung festgelegt oder einmalig zum Start der Software berechnet. Dazu kann bei der Initialisierung des Fahrzeuges ein Histogramm des Bildes erstellt und die Mitte der beiden Maxima bestimmt werden. Dieser Wert wird dann für die restliche Laufzeit als Schwellwert verwendet. Diese beiden Verfahren vernachlässigen allerdings, dass sich die Lichtverhältnisse während der Fahrt verändern.

Solch eine Situation entsteht, wenn das Fahrzeug weniger gut ausgeleuchtete Abschnitte passiert oder sich von einer insgesamt dunkleren in eine hellere Umgebung begibt. Ist Letzteres der Fall, so wird ein Großteil des Bildes als weiß deklariert, obwohl dies nicht die Realität darstellt.

Während sich in dem ersten Histogramm (Abbildung 4.11) eine große Anzahl an Bildpunkten im unteren Bereich befindet, existieren doch noch größere Häufigkeiten auch in den höheren Grauwerten. Die Häufigkeiten in den höheren Grauwerten sind vor allem auf die Wände zurückzuführen. Zudem ist die Straße auch nicht tiefschwarz, sondern verfügt über ein helleres schwarz. Die Häufigkeit dunklerer Grautöne im zweiten Histogramm (Abbildung 4.13) ist deutlich höher. Zum einen ist auf dem Bild nicht sehr viel von der Umgebung zu sehen. Zum anderen ist das entsprechende Bild (Abbildung 4.12) an einer Stelle aufgenommen worden, an der die Beleuchtung nicht optimal ist. Dies ist an der deutlich dunkleren Umgebung zu erkennen. Darüber hinaus besitzen an dieser Stelle Hindernisse oftmals keine weißen Oberflächen, sondern graue bis dunkelgraue Oberflächen. Werden die größten Ausschläge der Häufigkeiten betrachtet, so liegt dieser bei dem ersten Histogramm (Abb. 4.11) bei circa 12000 Fällen. Im anderen Histogramm (Abb. 4.13) liegt dieser Wert bei circa 21000 Fällen. Es existieren allerdings Verfahren, die solche ungleichmäßigen Lichtverhältnisse besser bewältigen.

Um ungleichmäßige Lichtverhältnisse auszugleichen, muss der Schwellwert τ immer wieder neu berechnet werden. Ob dies bei jedem Bild oder nur jedem zehnten Bild durchgeführt wird, liegt an den Lichtverhältnissen. Sind eher gleichmäßige Lichtverhältnisse vorhanden, so muss τ nicht so häufig berechnet werden, als wenn diese stark schwanken. Für die Berechnung von τ kann ein Histogramm über das gesamte Bild oder nur für den Bereich, in dem nach Linien und Objekten gesucht wird, erstellt werden. Allerdings erfordert die Berechnung eines Histogramms eine gewisse Zeit. Aus diesem Grund wird τ nicht durch ein Histogramm berechnet. An Stelle dessen wird der Grauwert $f(POI)$ des Bildpunktes genommen, auf dem sich der Point of Interest POI befindet und aufsummiert. Um den richtigen Schwellwert τ zu bekommen, werden die aufsummierten Grauwerte $f(POI)$ der Bildpunkte durch die Anzahl der Iterationen n dividiert. Sollte sich der Point of Interest POI auf einem Objekt oder einer Linie befinden, werden diese Werte ignoriert.

$$\tau = \frac{\sum^{n} f(POI)}{n} \tag{4.30}$$

4.3 Erkennung und Behandlung von Hindernissen

Um ein Hindernis zu identifizieren, wird oft überprüft, wie viel Prozent einer Region of Interest oder einer bestimmten Anzahl von Lines of Interest über dem berechneten Schwellwert τ liegen. Werden diese Methoden allein angewendet, um ein Hindernis zu identifizieren, sind nicht gewollte Identifizierungen möglich. Eine falsche Erkennung wird durch Reflexion der Beleuchtung auf der Fahrbahn oder durch eine ungünstig positionierte Region of Interest ausgelöst. Aus diesem Grund wird sowohl die Kamera als auch die Infrarot-Sensoren abgefragt. Zunächst wird überprüft, ob der Grauwert des Points of Interest $f(POI)$ (vgl. Kapitel 4.1) über dem Schwellwert $\tau \cdot \omega_{obs}$ (vgl. Kapitel 4.2) liegt. Sollte dies der Fall sein, wird der boolesche Ausdruck $activePOI$ wahr. Dies ist in folgender Formel beschrieben:

$$activePOI = \begin{cases} true \text{ falls } f(POI) \geq \tau \cdot \omega_{obs} \\ false \text{ sonst} \end{cases} \tag{4.31}$$

$activePOI$ gibt an, ob eine weitere Untersuchung stattfinden soll oder für den aktuellen Point of Interest keine weitere Behandlung nötig ist. Da τ sich an den Grauwert der Straße annähert, wird ein Wert benötigt, der den Schwellwert anhebt, ein sogenannter Offset: ω_{obs}. Dies ist nötig, damit kleinere Helligkeitsunterschiede während der Fahrt ausgeglichen werden. ω_{obs} muss vom Benutzer definiert werden. Erfahrungsgemäß sollte dieser Wert folgenden Bereich besitzen:

$$\text{Offset: } 2.0 \leq \omega_{obs} \leq 2.2 \tag{4.32}$$

Indem der Wert für τ verdoppelt wird, werden Helligkeitsunterschiede in der Fahrbahn nicht berücksichtigt und ignoriert. Hindernisse werden trotzdem einwandfrei erkannt. Besitzt $activePOI$ den Wert $true$, wird eine weitere Untersuchung vorgenommen. Es wird untersucht, wie hoch und breit die weiße Fläche ist. Dies ist in der Abbildung 4.14 dargestellt.

Durch diese Untersuchung werden Linien von Hindernissen unterschieden. Die Mindesthöhe für ein Hindernis ist um ein vielfaches größer, als die Höhe einer Linie, selbst wenn die Linie direkt vor dem Fahrzeug liegt. Die gleiche Bedingung gilt für die Breite eines Hindernisses im Vergleich zu einer Linie. Um die Höhe eines Hindernisses zu ermitteln, wird folgender Algorithmus verwendet:

```
1 while(f(y_up, x) > τ · ω_obs ∧ (y_start - y_up) ≤ β)
```

```
2   y_up = y_up - 1
3 do
```

Es gilt dabei:

$$y_{up} = y_{start} \tag{4.33}$$

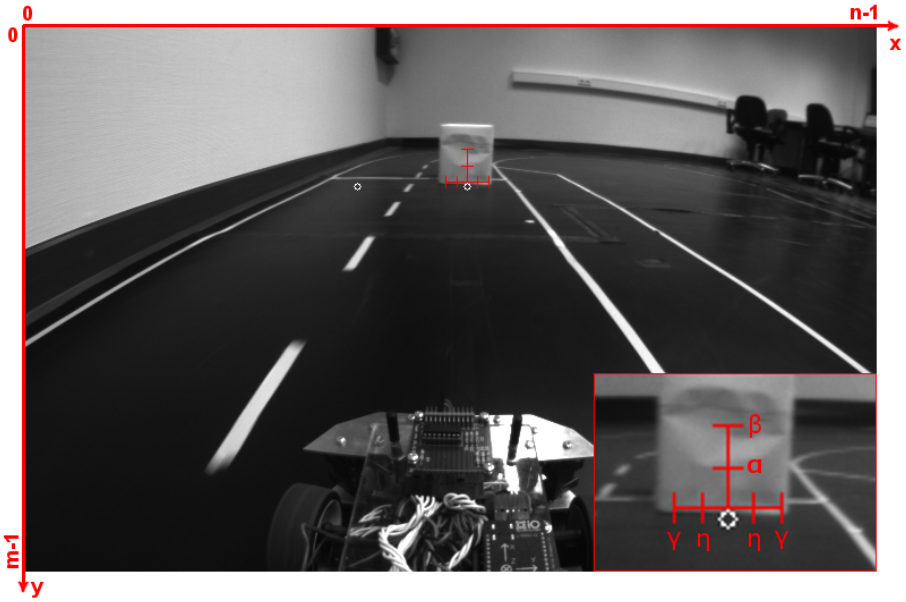

Abbildung 4.14: Größenmessung eines Hindernisses

Dabei sind y_{start} und x die Bildkoordinaten eines Point Of Interest. y_{up} wird zum Zählen verwendet. Wird entweder der Schwellwert $\tau \cdot \omega_{obs}$ unterschritten oder es sind mehr als β Bildpunkte überprüft worden, wird der Algorithmus terminiert. β wird vom Benutzer definiert und sollte folgenden Wert besitzen:

$$\text{max. zu untersuchende Bildpunkte: } \beta = 30 \tag{4.34}$$

Ist der Algorithmus terminiert, wird eine Überprüfung durchgeführt. Dabei wird überprüft, ob mindestens α Bildpunkte über $\tau \cdot \omega_{obs}$ liegen:

$$(y_{start} - y_{up}) \leq \alpha \tag{4.35}$$

α ist ein vom Benutzer einzustellender Wert und ist mit dem kleinsten Hindernis in circa einem Meter Entfernung ermittelt worden:

$$\text{min. zu untersuchende Bildpunkte: } \alpha = 15 \tag{4.36}$$

Um die Breite des Hindernisses zu ermitteln, wird einmal vom Startpunkt (y_{side}, x) nach links und nach rechts gesucht. Diese Suche ist in folgenden Algorithmen definiert:

```
1 while(f(y_side, x_right) > τ·ω_obs ∧ (x_right - x) ≤ γ)
2   x_right = x_right + 1
3 do
```

```
1 while(f(y_side, x_left) > τ·ω_obs ∧ (x - x_left) ≤ γ)
2   x_left = x_left + 1
3 do
```

Wobei gilt:

$$\begin{aligned} y_{side} &= y_{start} - \frac{y_{start} - y_{up}}{2} \\ x_{right} &= x_{left} = x \end{aligned} \tag{4.37}$$

y_{side} und x_{right} beziehungsweise x_{left} legen den Startpunkt des Suchvorganges fest. Darüber hinaus dienen x_{right} beziehungsweise x_{left} zur Ermittlung der Breite des Hindernisses. y_{side} wird dabei aus der ermittelten Höhe des Hindernisses berechnet. Beide Algorithmen suchen so lange, bis entweder der Schwellwert $\tau \cdot \omega_{obs}$ unterschritten oder die maximale Breite des Suchbereichs γ in Bildpunkten überschritten wird. γ wird vom Benutzer definiert und sollte folgenden Wert besitzen:

$$\text{max. zu untersuchende Bildpunkte: } \gamma = 20 \tag{4.38}$$

Ist eine der Bedingungen nicht mehr gegeben, terminiert der Algorithmus. Nach der Terminierung wird nochmals eine Überprüfung durchgeführt. Dabei wird überprüft, ob bei beiden Breitenmessungen mindestens η Bildpunkte über $\tau \cdot \omega_{obs}$ liegen:

$$(x - x_{left}) \leq \eta \tag{4.39}$$

$$(x_{right} - x) \leq \eta \tag{4.40}$$

η ist ein vom Benutzer einzustellender Wert und ist mit dem kleinsten Hindernis in circa einem Meter Entfernung ermittelt worden:

$$\text{min. zu untersuchende Bildpunkte: } \eta = 10 \tag{4.41}$$

Sind alle drei Überprüfungen 4.35, 4.39 und 4.40 negativ, wird mit den Infrarot Sensoren überprüft, ob es sich um ein physisches Hindernis handelt oder nur eine Reflexion erkannt worden ist.

4.3.1 Hindernisse auf rechter Fahrbahnseite

Ist das Hindernis optisch erkannt worden, muss durch die Infrarot Sensoren $S_1, \cdots, S_7$ ebenfalls eine Bestätigung erfolgen. Durch diesen Schritt wird verifiziert, dass es sich um ein echtes Hindernis handelt und nicht nur eine Bodenspiegelung oder eine Linie als Hindernis erkannt worden ist. Um diese Verifizierung auf der rechten Fahrbahnseite durchzuführen, existieren drei Fälle. Der erste und trivialste Fall ist die Verifizierung eines Hindernisses auf einer geraden Strecke. Die Fälle zwei und drei behandeln die Verifizierung von Hindernissen in Links- und Rechtskurven.

In den Abbildungen 4.15 bis 4.17 sind diese Fälle exemplarisch skizziert. Die Anordnung sowie die Reichweite der Infrarot Sensoren sind in Kapitel 3.2 beschrieben. In Abbildung 4.15 ist Fall eins dargestellt. Hierbei ist erkennbar, dass vor allem die Sensoren S_3 bis S_5 geeignet sind, um Hindernisse auf der rechten Fahrbahn in einem geraden Abschnitt zu erkennen. Die Sensoren S_3 bis S_5 werden verwendet, da zum einen das Hindernis nicht immer genau mittig in der Fahrbahn steht und zum anderen das Fahrzeug selbst auch leicht schwanken kann.

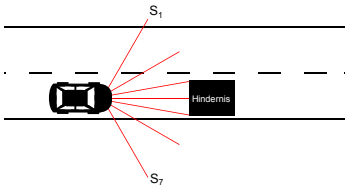

Abbildung 4.15: Detektion auf gerade Strecke

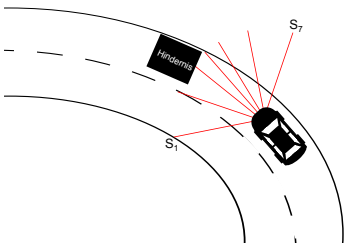

Abbildung 4.16: Detektion in Linkskurve

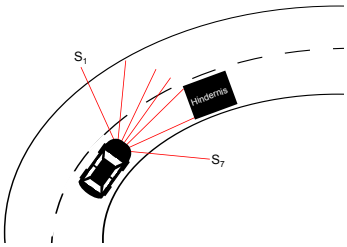

Abbildung 4.17: Detektion in Rechtskurve

Damit das Hindernis als valide erkannt wird, muss der Sensorwert s_i der Sensoren S_3, S_4, S_5 zum Zeitpunkt t unter einem bestimmten Wert θ_s liegen:

$$S_3, S_4, S_5 : s_i \leq \theta_s \ (t) \tag{4.42}$$

Dabei ist θ_s die Entfernung von der Fahrzeugfront bis zur Kante des Hindernisses. Der Benutzer muss θ_s einstellen. θ_s sollte erfahrungsgemäß folgenden Wert haben:

$$\text{Entfernung in cm: } 90 \leq \theta_s \leq 100 \tag{4.43}$$

Im zweiten Fall (Abbildung 4.16) werden die Sensoren S_2 bis S_4 verwendet. Dieser Fall deckt eine Linkskurve ab und ist in der Gleichung 4.44 beschrieben. Die Sensoren S_5 und S_6 werden für den dritten Fall (Abbildung 4.17) verwendet. In diesem Fall werden Hindernisse in einer Rechtskurve erkannt. Der dritte Fall ist in Gleichung 4.45 definiert. Für beide Fälle gelten die gleichen Distanzen vom Fahrzeug zum Hindernis θ_c.

$$S_2, S_3, S_4 : s_i \leq \theta_c \ (t) \tag{4.44}$$

$$S_5, S_6 : s_i \leq \theta_c \ (t) \tag{4.45}$$

θ_c ist wie θ_s die Entfernung von der Fahrzeugfront bis zur Kante des Hindernisses. θ_c muss ebenfalls vom Benutzer eingestellt werden. Ein erfahrungsgemäßer Wertebereich ist:

$$\text{Entfernung in cm: } 70 \leq \theta_c \leq 80 \tag{4.46}$$

In der Abbildung 4.18 ist die Trajektorie des Ausweichvorganges eingezeichnet. Die rot gestrichelte Linie beschreibt die optimale Trajektorie, die das Fahrzeug ausführen muss. Wird der Überholvorgang eingeleitet, wird der linke Blinker aktiviert. Befindet sich das Fahrzeug gleichauf mit dem Hindernis, wird der linke Blinker deaktiviert. Dies erkennt das Fahrzeug durch einen Pegelwechsel an dem hinteren digitalen Infrarot Sensor S_8. Zusätzlich wird der analoge Infrarot Sensor S_9 überprüft, ob dieser einen bestimmten Sensorwert s_i zu einem Zeitpunkt t besitzt:

$$S_8 = true \wedge S_9 : s_i \leq \theta_{si} \ (t) \tag{4.47}$$

Dabei ist θ_{si} ein Wert, der vom Benutzer eingestellt werden muss. Er gibt an, wie groß die Messdistanz zwischen S_9 und dem Hindernis sein muss. Erfahrungsgemäß sollte θ_{si} folgendermaßen definiert sein:

$$\text{Entfernung in cm: } \theta_{si} = 30 \tag{4.48}$$

Durch erneute Änderung des Pegels von S_8 ist bekannt, dass das Hindernis passiert worden ist. Zusätzlich wird wiederholt S_9 verwendet:

$$S_8 = false \wedge S_9 : s_i \geq \theta_{si}\ (t) \tag{4.49}$$

Es wird zurück auf die rechte Fahrbahnseite gefahren. Während dieses Vorganges ist der rechte Blinker aktiviert. Der Grund für die Verwendung von S_9 ist, dass S_8 bei rund 50 Zentimetern einen Pegelwechsel erzeugt. Allerdings kann in dieser Distanz ein Hindernis oder eine Person am Rand stehen. Wird S_9 nicht mit einbezogen, würde das Fahrzeug weiter auf der linken Fahrspur fahren, sofern sich am Fahrbahnrand ein Hindernis befindet.

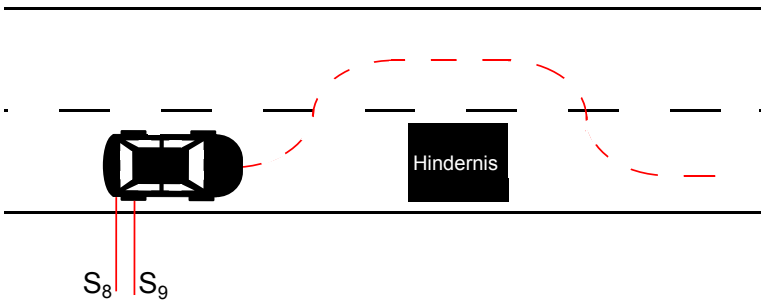

Abbildung 4.18: Trajektorie bei Erkennung eines Hindernisses

4.3.2 Hindernisse auf linker Fahrbahnseite

Die Validierung von Hindernissen auf der linken Fahrbahnseite verhält sich ähnlich wie die Validierung auf der rechten Fahrbahnseite in Kapitel 4.3.1. Der Unterschied liegt in der Auswahl der Infrarot Sensoren sowie der Trajektorie bei Erkennung eines Hindernisses. Die Validierung von optisch erkannten Hindernissen besteht ebenfalls aus drei Fällen, welche in den Abbildungen 4.19 bis 4.21 exemplarisch skizziert sind.

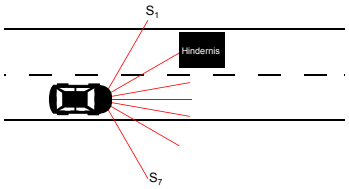

Abbildung 4.19: Detektion auf gerade Strecke

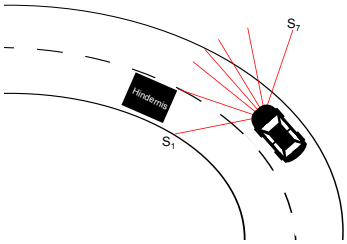

Abbildung 4.20: Detektion in Linkskurve

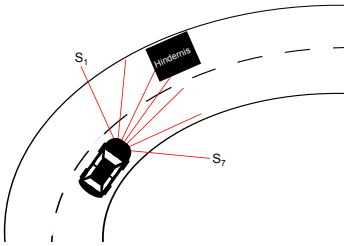

Abbildung 4.21: Detektion in Rechtskurve

In der Abbildung 4.19 ist der erste Fall skizziert. Dabei ist erkennbar, dass die Validierung von Hindernissen auf der linken, geraden Strecke nur mittels des Sensors S_2 erfolgt. Es kann auch der Infrarot Sensor S_3 genutzt werden. Zum Zeitpunkt t gilt zur Validierung eines Hindernisses:

$$S_2, S_3 : s_i \leq \theta_s \ (t) \tag{4.50}$$

In den Abbildungen 4.20 und 4.21 ist erkennbar, dass in Links- als auch Rechtskurven nur die linke Hälfte der Infrarot Sensoren verwendet wird. In einer Linkskurve werden nur die Sensoren S_1 und S_2 verwendet (Gleichung 4.51). Für eine Rechtskurve werden die Sensoren S_3 und S_4 benötigt (Gleichung 4.52). Für beide Fälle gilt zum Zeitpunkt t und dem Sensorwert s_i Folgendes:

$$S_1, S_2 : s_i \leq \theta_c \ (t) \tag{4.51}$$

$$S_3, S_4 : s_i \leq \theta_c \ (t) \tag{4.52}$$

Die Trajektorie, die das Fahrzeug bei Erkennung eines Hindernisses zurücklegt, ist in Abbildung 4.22 dargestellt. Da sich das Hindernis auf der linken Fahrbahnseite befindet,

muss kein spezielles Ausweichmanöver ausgeführt werden. Es genügt, dem Straßenverlauf zu folgen. Um die Sensorik nicht zu erweitern, wird überprüft, ob das Fahrzeug eine Distanz θ_d zum Zeitpunkt t überwunden hat. Um diese Messung durchzuführen, wird zum einen ein Hallsensor H_1 mit den Sensorwert h_1 verwendet. Zum anderen wird ein Anfangswert θ_{ds} des aktuellen Sensorwertes h_1 zum Zeitpunkt $t-1$ benötigt:

$$\theta_{ds} = h_1\ (t-1) \tag{4.53}$$

$$H_1 : h_1 \geq \theta_{ds} + \theta_d\ (t) \tag{4.54}$$

Dabei ist θ_d ein vom Benutzer einzustellender Wert. θ_d gibt die Distanz an, die das Fahrzeug überwinden muss, damit das Hindernis passiert wird. Erfahrungsgemäß sollte θ_d wie folgt definiert sein:

$$\text{Entfernung in cm: } 100 \leq \theta_d \leq 120 \tag{4.55}$$

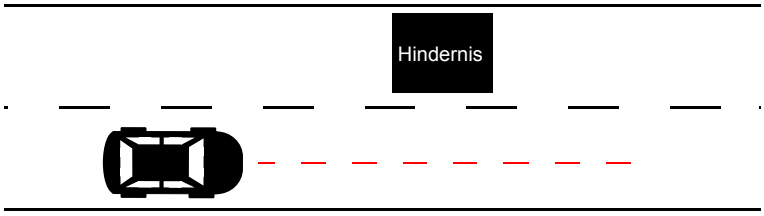

Abbildung 4.22: Trakjektorie bei Erkennung eines Hindernisses

4.4 Erkennung und Behandlung von Kreuzungen

Die Erkennung von Kreuzungen und der Start- und Stopplinie basiert auf dem gleichen Prinzip wie die Erkennung von Hindernissen. Zunächst wird mittels des Points of Interest

POI überprüft, ob der Schwellwert τ einen bestimmten Wert überschritten hat. Bei alleiniger Verwendung von τ kann es vorkommen, dass Beleuchtungsunterschiede oder Reflexionen auf der Fahrbahn zur Auslösung einer Erkennung führen. Aus diesem Grund gibt es wie bei der Erkennung von Hindernissen einen einstellbaren Wert ω_{cros}. Überschreitet der Grauwert des Points of Interest $f(POI)$ (vgl. Kapitel 4.1) den Schwellwert $\tau \cdot \omega_{cros}$ (vgl. Kapitel 4.2), wird der boolesche Ausdruck $activePOI$ wahr. Dies ist in folgender Formel definiert:

$$activePOI = \begin{cases} true \text{ falls } f(POI) \geq \tau \cdot \omega_{cros} \\ false \text{ sonst} \end{cases} \tag{4.56}$$

Wie bei der Hinderniserkennung gibt $activePOI$ an, ob eine genauere Betrachtung des Points of Interest stattfinden soll. τ stellt den berechneten Schwellwert dar, der um den Offset ω_{cros} ergänzt wird. Dabei muss ω_{cros} vom Benutzer eingestellt werden. Erfahrungsgemäß sollte der Wertebereich von ω_{cros} folgender sein:

$$\text{Offset: } 2.0 \leq \omega_{cros} \leq 2.2 \tag{4.57}$$

Besitzt $activePOI$ den Wert $true$, wird der Algorithmus zur Suche nach Linien gestartet. Der Algorithmus überprüft sowohl links als auch rechts von dem Point of Interest POI, ob in einem bestimmten horizontalen Bereich eine Linie erkannt wird. Dabei ist es notwendig, dass der Algorithmus auf einem Bereich mit einem Grauwert $f(y,x) \leq \tau \cdot \omega_{cros}$ startet. Dies ist notwendig, damit das Suchschema eingehalten wird. Das Suchschema besteht aus zwei Schritten. Zunächst wird nach einem Wechsel von Schwarz ($f(y,x) \leq \tau \cdot \omega_{cros}$) auf Weiß ($f(y,x) \geq \tau \cdot \omega_{cros}$) gesucht. Ist dies erkannt worden, wird nach einem Wechsel von Weiß auf Schwarz gesucht.

Der Algorithmus sucht in der horizontalen Achse nach einer Linie. Dies bedeutet, dass alle $y \in \{y_{Start}, \cdots, y_{End}\}$ solange auf folgende Bedingung $f(y,x) \geq \tau \cdot \omega_{cros}$ geprüft werden, bis entweder die Bedingung oder $y \geq y_{End}$ erfüllt ist. Ist die Bedingung erfüllt, werden alle $k \in \{y, \cdots, y_{End}\}$ solange auf die Bedingung $f(k,x) \leq \tau \cdot \omega_{cros}$ geprüft, bis entweder die Bedingung erfüllt ist oder $k \geq y_{End}$ gilt. Ist die Bedingung erfüllt, werden die Koordinaten (k,x) zurückgegeben.

Die Werte für y_{Start}, y_{End} und x werden aus zwei Bestandteilen berechnet. Zum einen ist dies der Point of Interest POI. Zum anderen sind es Werte, die vom Benutzer definiert

werden und somit Einfluss auf die Position nehmen. In den Abbildungen 4.23 und 4.24 sind die Anordnungen für die Suchlinien sowohl für Kreuzungen als auch Start- und Stopplinie eingezeichnet. Der Wert σ beschreibt die seitliche Verschiebung vom POI, κ die vertikale Startposition. Sowohl σ als auch κ dienen der Ausrichtung der Suchlinien für eine Suche auf der rechten Fahrspur. Die Werte μ und ρ werden für die Ausrichtung der Suchlinie auf der linken Fahrspur genutzt. μ beziehungsweise ρ beschreiben die horizontale und vertikale Verschiebung. Diese Werte sollten erfahrungsgemäß folgenden Wertebereich besitzen:

$$\begin{aligned} \sigma &= 30 \text{ Bildpunkte} \\ \kappa &= 10 \text{ Bildpunkte} \\ \mu &= 110 \text{ Bildpunkte} \\ \rho &= 15 \text{ Bildpunkte} \end{aligned} \tag{4.58}$$

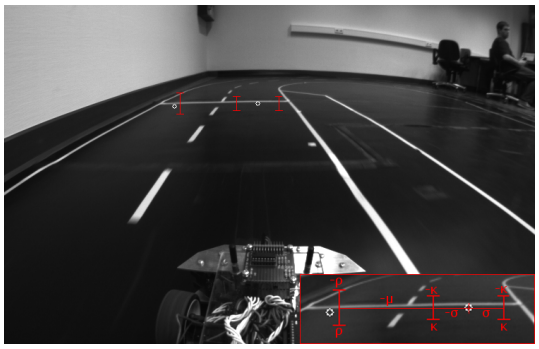

Abbildung 4.23: Erkennung einer Start- und Stopplinie

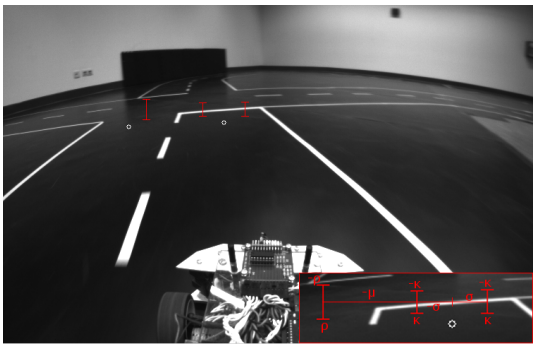

Abbildung 4.24: Erkennung einer Kreuzung

Daraus ergeben sich folgende Startwertberechnungen für die drei Suchlinien:

1. Suchlinie (rechts vom POI):

$$\begin{aligned} y_{Start} &= y_{POI} - \kappa \\ y_{End} &= y_{POI} + \kappa \\ x &= x_{POI} + \sigma \end{aligned} \tag{4.59}$$

2. Suchlinie (links vom POI):

$$\begin{aligned} y_{Start} &= y_{POI} - \kappa \\ y_{End} &= y_{POI} + \kappa \\ x &= x_{POI} - \sigma \end{aligned} \tag{4.60}$$

3. Suchlinie (linke Fahrspur):

$$\begin{aligned} y_{Start} &= y_{POI} - \rho \\ y_{End} &= y_{POI} + \rho \\ x &= x_{POI} - \mu \end{aligned} \tag{4.61}$$

Dabei stehen y_{POI} und x_{POI} für die (y, x)-Bildkoordinaten des POI. Zur Erkennung einer Kreuzung beziehungsweise Start- und Stopplinie wird ausgewertet, ob die dritte Suchlinie Koordinaten zurückliefert. Ist dies der Fall, handelt es sich um eine Start- und Stopplinie. Sollten keine Koordinaten vorliegen, ist eine Kreuzung erkannt worden.

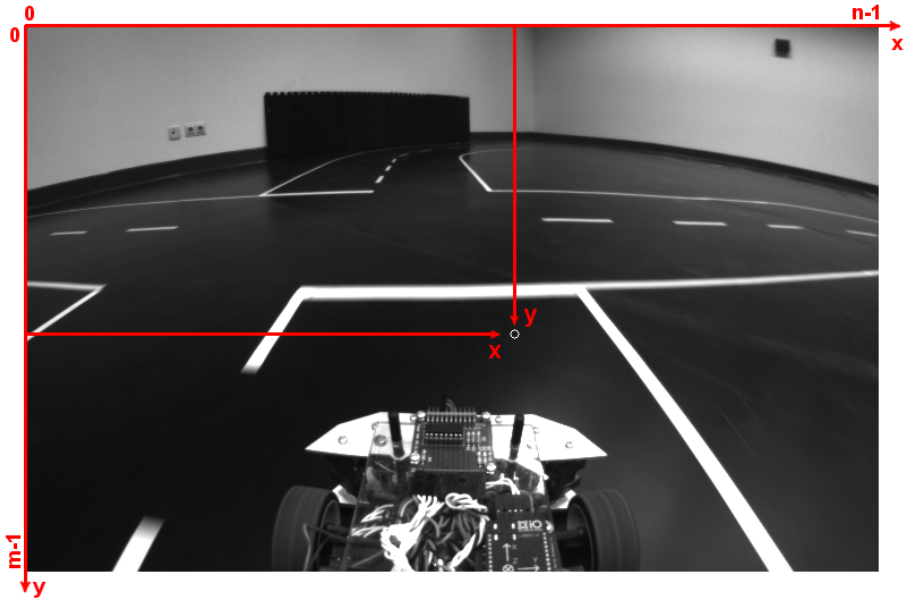

Abbildung 4.25: Nach Erkennung langsames heranfahren an Kreuzung

Ist eine Kreuzung eindeutig identifiziert worden (Abbildung 4.24), muss als nächster Schritt an diese Kreuzung herangefahren werden. Dabei soll das Fahrzeug möglichst dicht an der Haltelinie stoppen und diese nicht überfahren. Würde das Fahrzeug zu früh halten, wird dies als Fehlverhalten angesehen. Da ein Stopp aus hoher Geschwindigkeit zu einem langen Bremsweg führt, wird nach der Erkennung der Kreuzung die Geschwindigkeit auf 60 $\frac{cm}{s}$ verringert. Um zu erkennen, ob das Fahrzeug die Haltelinie erreicht hat, wird vor dem Fahrzeug ein Punkt (y, x) gelegt. In Abbildung 4.25 ist dieser Suchpunkt eingezeichnet. Dabei fährt das Fahrzeug solange weiter, bis gilt:

$$f(y, x) \geq \tau \cdot \omega_{cros} \tag{4.62}$$

Der Punkt (y, x) muss vom Benutzer definiert werden. Es hat sich erfahrungsgemäß bei einer Geschwindigkeit von 60 $\frac{cm}{s}$ folgende Koordinate als erfolgreich erwiesen: $(y, x) = (270, 430)$. Ist diese Bedingung erfüllt, stoppt das Fahrzeug. Anschließend wird überprüft, ob die Kreuzung blockiert ist oder die Möglichkeit besteht, nach einer kurzen Wartezeit weiterzufahren.

Das Regelwerk des Carolo-Cups (vgl. Kapitel 3.1) schreibt vor, dass das Fahrzeug mindestens zwei Sekunden an der Kreuzung warten muss. Befindet sich rechts vom Fahrzeug ein fahrendes Hindernis, muss das Fahrzeug warten. Es muss so lange warten, bis das fahrende Hindernis die Kreuzung überquert hat. Sollte das Fahrzeug vorher losfahren, stellt dies ein Regelverstoß dar. Aus diesem Grund werden die Sensoren S_2 bis S_6 an der Fahrzeugfront eingesetzt. Dies soll sicherzustellen, dass das fahrende Hindernis die Kreuzung definitiv passiert hat. Dieser Vorgang ist in Abbildung 4.26 dargestellt. Es gilt zum Zeitpunkt t Folgendes:

$$S_2, S_3, S_4, S_5, S_6 : s_i \leq \theta_b \ (t) \tag{4.63}$$

Dabei ist θ_b die Distanz, ab der die Kreuzung als blockiert markiert wird. Diese Distanz ergibt sich aus der Straßenbreite von 80 Zentimetern. θ_b ist ein vom Benutzer einzustellender Wert und sollte erfahrungsgemäß folgenden Wertebereich besitzen:

$$\text{Entfernung in cm: } 70 \leq \theta_b \leq 80 \tag{4.64}$$

Würde θ_b großer als der angegebene Wertebereich gewählt werden, kann die Kreuzung trotz vorbeigefahrenem fahrenden Hindernis weiterhin als blockiert gelten. Dies ist darin begründet, dass an der Abbiegung der Kreuzung ein Hindernis positioniert sein

kann. Damit das Fahrzeug die Kreuzung wieder als frei markiert und weiterfährt, muss Folgendes zum Zeitpunkt $t + 1$ gelten:

$$S_2, S_3, S_4, S_5, S_6 : s_i \geq \theta_b \ (t + 1) \tag{4.65}$$

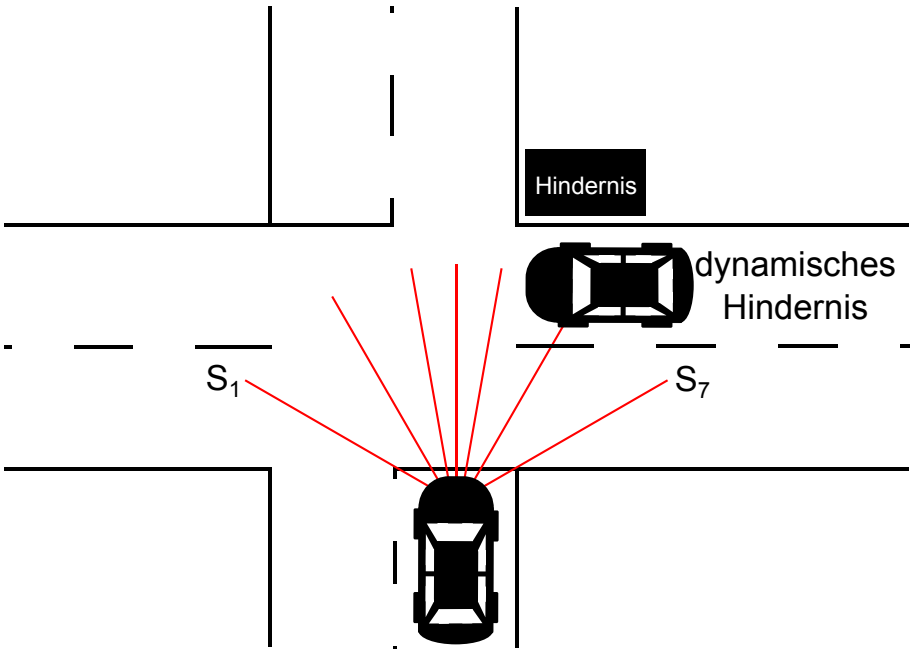

Abbildung 4.26: Blockierde Kreuzung und Erkennung mittels Infrarot Sensoren

5 Algorithmen zur Behandlung dynamischer Umgebungen durch RatSLAM

Die grundsätzliche Funktionsweise von RatSLAM ist in Kapitel 2 beschrieben. RatSLAM bietet standardmäßig nur die Erstellung von Pfaden sowie die Berechnung von Zielen an. Die erstellte topologische Karte in der Abbildung 5.1 passt sehr gut zum Streckenverlauf der Carolo-Cup Strecke (Kapitel 3), allerdings bildet RatSLAM selbst keine zusätzlichen Informationen ab. Interessant ist RatSLAM auch deshalb, weil es möglich ist, sehr schnell zusätzliche Informationen in die Erfahrungen zu implementieren. In der Abbildung 5.1 ist eine solche erste Aufwertung der Karte erfolgt. Die Erfahrungen sind mit den Zusatzinformationen Kurve beziehungsweise gerade Strecke und ob es sich um eine Links- oder Rechtskurve handelt, aufgewertet worden.

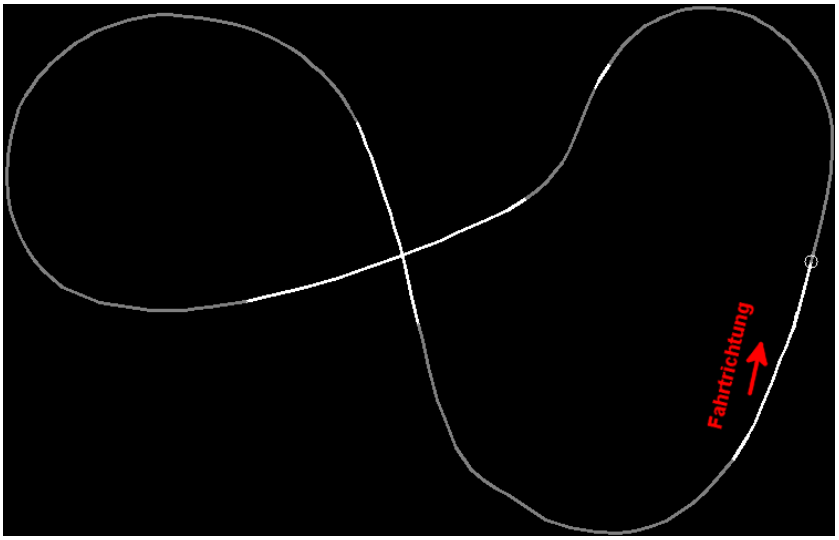

Abbildung 5.1: Erstellte Karte mittels RatSLAM mit Aufwertung: weiß: gerade, grau: Kurve

Solche Zusatzinformationen können dafür genutzt werden, damit die Spurführung in einer Linkskurve nicht schlagartig nach rechts lenkt. Eine andere Möglichkeit der Nutzung dieser Informationen besteht in der Sensibilisierung des Systems, wenn in einer bestimmten Entfernung eine bestimmte Situation auftritt. Dabei wird sich dieses Kapitel weniger mit dem Nutzen dieser Informationen beschäftigen, sondern mit der Aufbereitung und Integration in die topologische Karte.

Wie bereits oben erwähnt, ist RatSLAM nicht mit Algorithmen ausgestattet, die es erlauben, Hindernisse oder Kreuzungen zu erkennen und zu integrieren. Aus diesem Grund werden solche Ereignisse in einer dynamischen Umgebung als neuer beziehungsweise zusätzlicher Pfad dargestellt. Die Bedingung, um die Algorithmen zu entwickeln, ist eine vollständige Karte, in der keine Hindernisse oder Ähnliches existieren. Aus diesem Grund muss zunächst eine Runde komplett auf einer freien Strecke absolviert werden, bevor sich Hindernisse in die Umgebung befinden dürfen.

5.1 Detektion der Situation in bestimmter Entfernung

Die grundlegende Einführung der Nomenklatur ist bereits in Kapitel 2 erfolgt. Zum einfacheren Verständnis wird an dieser Stelle nur eine kurze Einführung gegeben. Im Wesentlichen stellt die Experience Map (Kapitel 2.1.3) ein Graph G mit folgender Definition dar:

$$\begin{aligned} G &= \{E, L\} \text{ , wobei} \\ e_i \in E \wedge e_i &= \{P^i, V^i, p^i\} \text{ sowie} \\ l_{ij} \in L \wedge l_{ij} &= \{\Delta p^{ij}, \Delta t^{ij}\} \text{ mit} \\ i, j &\in \mathbb{N} \end{aligned} \tag{5.1}$$

Dabei ist e_i die aktuelle Erfahrung aus der Menge der Erfahrungen E. Eine Erfahrung e_i besteht aus dem aktuellen Zustand der Local View Cells V^i sowie der Pose Cells P^i und dem aktuellen Ort p^i. Eine Erfahrung e_i wird als aktuell bezeichnet, wenn sich zum Zeitpunkt t eine Erfahrung e_i in E mit der Eigenschaft $\{P^i, V^i, p^i\}$ befindet. Für die Eigenschaft $\{P^i, V^i, p^i\}$ existiert ein Toleranzwert S (vgl. Kapitel 2.1.3). Existieren mehrere ähnliche Erfahrungen, wird die zu erst identifizierte Erfahrung e_i ausgewählt. Die Verbindung l_{ij} stammt aus der Menge der Verbindungen L. Eine Verbindung l_{ij} verfügt über die Distanz Δp^{ij} zwischen zwei Erfahrungen e_i und e_j sowie die benötigte

Zeit Δt^{ij}. Die Mengen E und L sind abhängig vom aktuellen Zeitpunkt t. Diese Mengen wachsen dynamisch und die Indexe i und j stellen lediglich eine Identifikation der Erfahrungen und Verbindungen dar, aber nicht deren reale Position in der Umgebung.

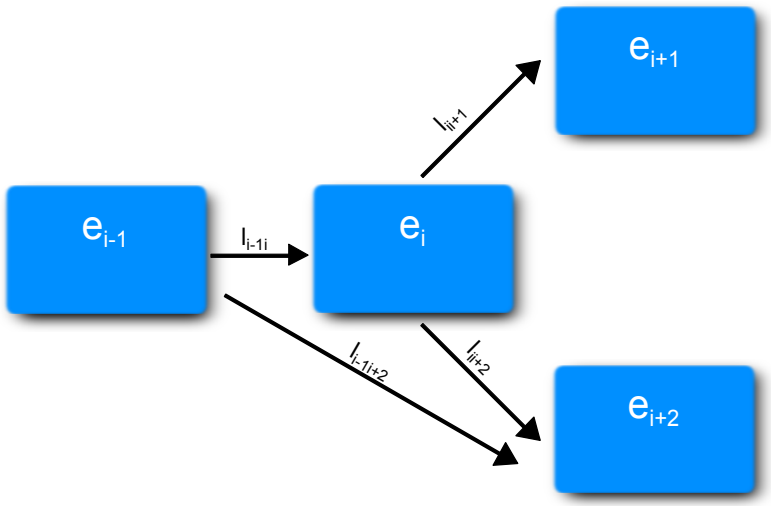

Abbildung 5.2: Aufbau und Verbindung zwischen Erfahrungen

Für die Entwicklung der Algorithmen ist es essenziell, eine bestimmte Distanz θ_{maxdis} ausgehend von der aktuellen Erfahrung e_i zu überwinden. Um dies zu ermöglichen, muss zunächst ein Algorithmus entwickelt werden, der sich von Erfahrung zu Erfahrung bewegt. Wie viele Erfahrungen dabei passiert werden müssen, kann anhand der Verbindungen und der enthaltenen Distanz Δp^{ij} erkannt werden. Der Aufbau der Experience Map ist in der Abbildung 5.2 exemplarisch dargestellt. Dabei besitzt jede Erfahrung eine Verbindung zu einer nachfolgenden Erfahrung. Es kann auch vorkommen, dass eine Verbindung l_{ii-1} erstellt wird. Dies bedeutet, dass eine Verbindung von der Erfahrung e_i zurück zur Erfahrung e_{i-1} aufgebaut wird. Eine Verbindung l_{ii} von der Erfahrung e_i auf sich selbst ist nicht gestattet.

Damit für eine bestimmte Distanz θ_{maxdis} vorwärts durch die Erfahrungen gegangen wird, wird für alle Erfahrungen ausgehend von der aktuellen Erfahrung $e_i \in E\ (t)$ zum Zeitpunkt t folgender Ausdruck definiert:

$$\theta_{dis} = \theta_{dis} + \Delta p^{ii+1} \tag{5.2}$$

Dies wird so lange wiederholt, bis die Bedingung $\theta_{dis} \geq \theta_{maxdis}$ erfüllt ist. Der Wert θ_{dis} wird vom System definiert und beschrieben. Δp^{ii+1} gibt die Distanz in Zentimetern zwischen der aktuellen Erfahrung e_i und der folgenden Erfahrung e_{i+1} an. Dabei ist die Distanz θ_{maxdis} ein Wert, der vom Benutzer einzustellen ist. Dieser Wert muss in Zentimetern angegeben werden und besitzt keinen festen Wertebereich. Es bietet sich allerdings an, θ_{maxdis} folgendermaßen zu definieren:

$$\text{Entfernung in cm: } 150 \leq \theta_{maxdis} \leq 200 \tag{5.3}$$

Mit diesem Wertebereich schaut der Algorithmus in einer Entfernung, die etwas länger ist als die Approximation der Polynome durch Polaris. Somit kann frühzeitig auf eventuell auftretende Ereignisse, wie zum Beispiel Kurven oder Kreuzungen, reagiert werden.

Es kann auch durchaus Sinn machen, dass anstelle einer Vorwärtssuche in den Erfahrungen eine Rückwärtssuche für die Distanz θ_{maxdis} angestrebt wird. Ist dies der Fall, wird folgender Ausdruck für alle Erfahrungen ausgehend von der aktuellen Erfahrung $e_i \in E\ (t)$ zum Zeitpunkt t definiert:

$$\theta_{dis} = \theta_{dis} + \Delta p^{i-1i} \tag{5.4}$$

Auch an dieser Stelle erfolgt so lange eine Wiederholung, bis die Bedingung $\theta_{dis} \geq \theta_{maxdis}$ erfüllt ist. Der einzige Unterschied zur Vorwärtssuche liegt in der Erfassung der Distanz zwischen zwei Erfahrungen. Es wird anstelle der Erfahrungen e_i und e_{i+1} die Distanz Δp^{i-1i} zwischen den Erfahrungen e_{i-1} und e_i ermittelt und aufsummiert.

Die Verfahren durchlaufen immer den zuerst erstellten Pfad in der Karte. Es kommt allerdings vor, dass ein Pfad eine Schleife beinhaltet. Die Schleife kann auch nur aus einer einzelnen Erfahrung bestehen. So ein Fall ist in der Abbildung 5.3 dargestellt. In dieser exemplarischen Situation führt der zuerst erstellte Pfad von der Erfahrung e_i über die Verbindung l_{ii+1} zu der Erfahrung e_{i+1}. Eine weitere Verbindung l_{i+1i} führt zurück auf die Ausgangserfahrung e_i. Bei den oben erklärten Verfahren bewegen sich diese ab der Schleife nur noch im Kreis und summieren die Distanzen auf. Dies würden die Verfahren so lange machen, bis die Abbruchbedingung erfüllt ist. Das ist ein nicht erwünschtes Phänomen, das verhindert werden muss. Aus diesem Grund muss gelten, dass eine Verbindung, die zurück auf die Erfahrung e_i verweist, nur einmal durchlaufen wird.

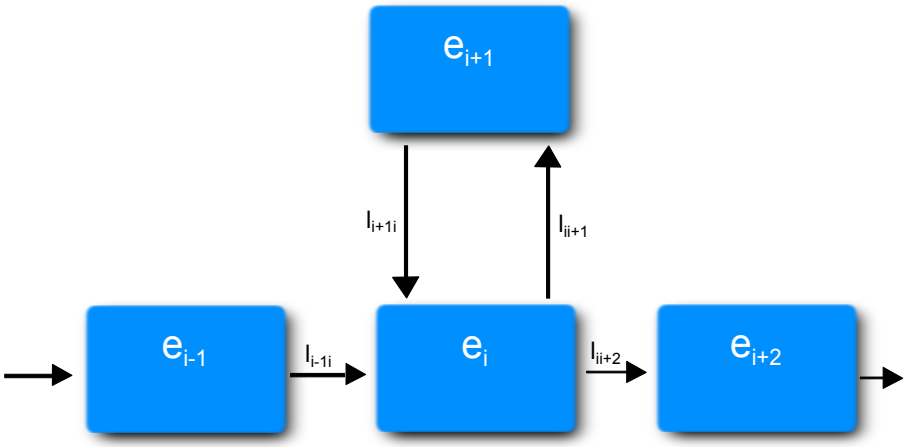

Abbildung 5.3: Auftretende Schleife

Diese Bewältigung der Aufgabe stellt keine perfekte Lösung dar. Es wird fälschlicherweise eine bestimmte Distanz auf θ_{dis} aufsummiert, die eigentlich nicht gewollt ist. Beide Verfahren bieten die Möglichkeit, in einer bestimmten Entfernung die Situation zu erfragen. Somit ist es möglich, eine Aussage über die Beschaffenheit der Fahrbahn in einer bestimmten Entfernung zu treffen oder auf Objekte in der dynamischen Umgebung entsprechend zu reagieren.

5.2 Position des Hindernisses auf der rechten Fahrbahnseite

Befindet sich ein identifiziertes Hindernis auf der rechten Fahrbahnseite (vgl. Kapitel 4.3.1), weicht das Fahrzeug diesem Hindernis aus. Dazu muss das Fahrzeug auf die linke Fahrspur wechseln. Infolgedessen wird durch RatSLAM ein neuer Pfad in der Experience Map eingezeichnet. Dieser neue Pfad ist in der Abbildung 5.4 an der Stelle *B* dargestellt. Der blaue Pfad an der Stelle *A* markiert das erkannte Hindernis.

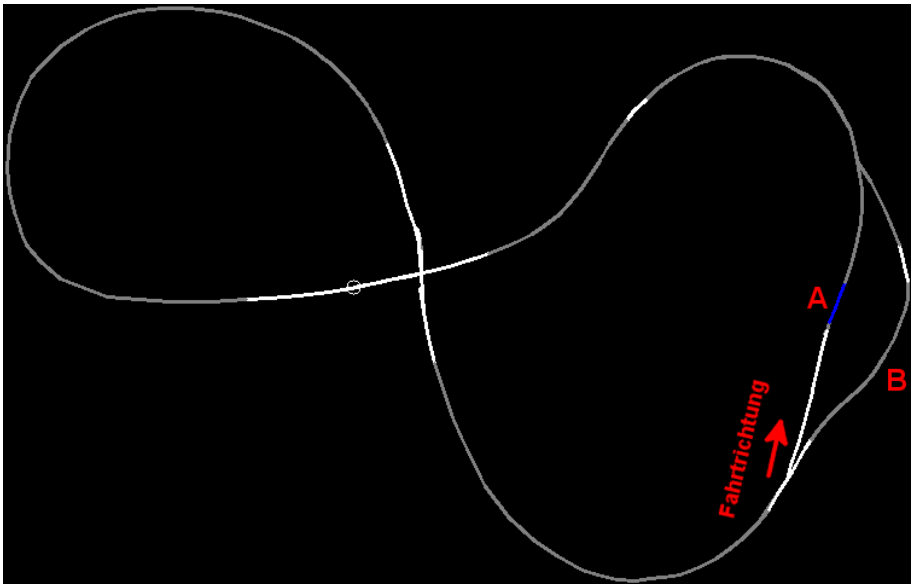

Abbildung 5.4: Experience Map mit Hindernis auf rechter Fahrbahnseite (Blau)

Indem das Fahrzeug dem Hindernis ausweicht, stimmt die Experience Map mit der Realität überein. Es muss folglich keine Anpassung an dem Aufbau der Experience Map vorgenommen werden. Es müssen nur die entsprechenden Erfahrungen mit dem Hinweis versehen werden, dass sich an dieser Stelle ein Hindernis befindet. Damit die Eintragung des Hindernisses auch mit der Umgebung übereinstimmt, wird der Sensorwert s_i des entsprechenden Infrarot Sensors S_i zum Zeitpunkt t genutzt. S_i hängt davon ab, ob sich das Fahrzeug auf einer geraden Strecke, in einer Linkskurve oder in einer Rechtskurve befindet. Auf einer geraden Strecke gilt: $S_3, S_4, S_5 : S_i$. In einer Linkskurve ist S_i folgendermaßen definiert: $S_2, S_3, S_4 : S_i$. Für eine Rechtskurve gilt: $S_5, S_6 : S_i$. Ausgehend von der aktuellen Messdistanz s_i wird das Hindernis in den Erfahrungen eingetragen. In den Abbildungen 5.5 und 5.6 ist beispielhaft die gefahrene Trajektorie sowie der dazugehörige Aufbau in der Experience Map dargestellt.

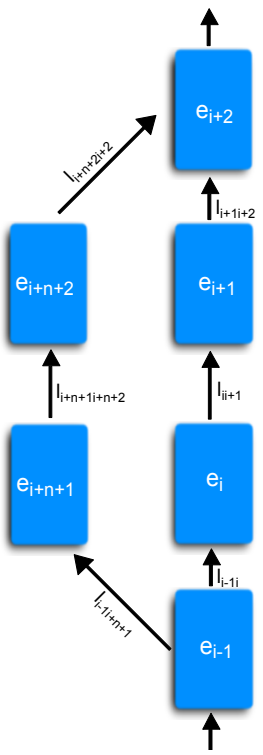

Abbildung 5.5: Teilausschnitt Experience Map - Hindernis rechts

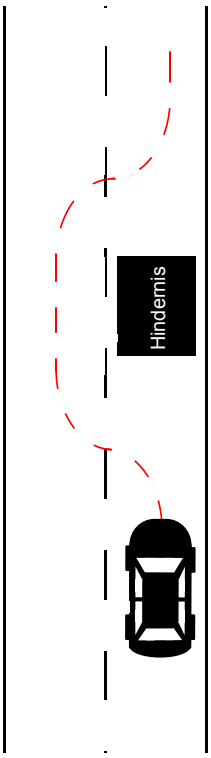

Abbildung 5.6: Trajektorie bei Hindernis auf rechter Fahrbahn

Die Trajektorie des Fahrzeuges stimmt mit dem Aufbau der Experience Map überein. Es wird ein nebenläufiger Pfad beginnend ab der Erfahrung e_{i-1} erstellt. Nach dem Wechsel von der linken zurück auf die rechte Fahrspur findet bei der Erfahrung e_{i+2} eine Relokalisierung auf dem ursprünglichen Pfad statt. Somit wird die Verbindung $l_{i+n+2i+2}$ zurück auf den ursprünglichen Pfad erstellt. Bei den beiden Erfahrungen e_{i+n+1} und e_{i+n+2} steht das n für die Gesamtgröße der Experience Map E zum Zeitpunkt t. Der Algorithmus zur Markierung des Hindernisses startet allerdings nicht, nachdem eine Relokalisierung auf dem ursprünglichen Pfad stattgefunden hat, sondern sobald das Hindernis erkannt worden ist.

Wenn ein Hindernis erkannt wird, existieren zwei Situationen, auf die entsprechend reagiert werden muss. Diese beiden Situationen sind in den Abbildungen 5.7 und 5.8 dargestellt. Die eine Situation stellt den Zustand der Experience Map so dar, dass zum Zeitpunkt der Erkennung bereits mit dem nebenläufigen Pfad begonnen worden ist

(Abbildung 5.7). In der anderen Situation besteht zum Zeitpunkt der Erkennung noch kein nebenläufiger Pfad in der Experience Map (Abbildung 5.8).

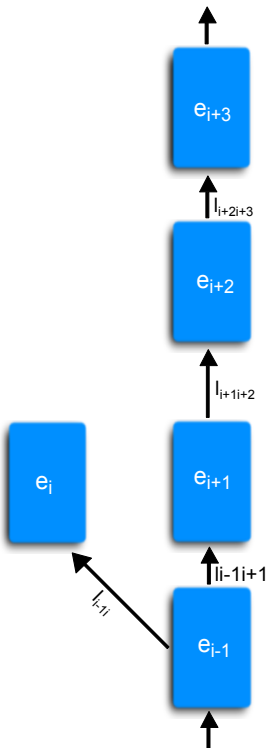

Abbildung 5.7: Erste mögliche Situation

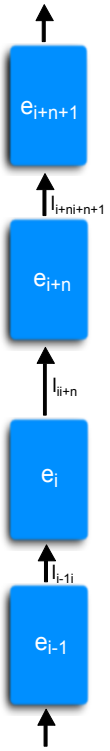

Abbildung 5.8: Zweite mögliche Situation

Tritt die erste Situation (Abbildung 5.7) auf, in der die aktuelle Erfahrung e_i keinen Nachfolger besitzt, muss in den Erfahrungen zurückgegangen werden. Dabei wird für alle Erfahrungen ausgehend von der aktuellen Erfahrung $e_i \in E\ (t)$ zum Zeitpunkt t Folgendes durchgeführt:

$$e_i = e_{i-1} \tag{5.5}$$

Dies wird so lange durchgeführt, bis die aktuelle Erfahrung e_i zwei Nachfolgeerfahrungen e_{i+1} und e_{i+2} über die Verbindungen l_{ii+1} und l_{ii+2} besitzt. Ist diese Bedingung erfüllt, wird mittels der Vorwärtssuche die entsprechende Markierung in der Erfahrung

vorgenommen. Für die Vorwärtssuche wird ausgehend von der aktuellen Erfahrung $e_i \in E\ (t)$ zum Zeitpunkt t für alle Erfahrungen folgender Ausdruck definiert:

$$\theta_{dis} = \theta_{dis} + \Delta p^{ii+1} \tag{5.6}$$

Es wird so lange wiederholt, bis die Bedingung $\theta_{dis} \geq s_i(t)$ gilt. Bei der aktuellen Erfahrung e_i wird dann der boolesche Wert $obsRight$ auf $true$ gesetzt. Dabei wird erneut die Vorwärtssuche genutzt. An dieser Stelle wird ausgehend von der aktuellen Erfahrung $e_i \in E\ (t)$ zum Zeitpunkt t für alle Erfahrungen folgender Ausdruck definiert:

$$\begin{aligned} \theta_{dis} &= \theta_{dis} + \Delta p^{ii+1} \\ obsRight(e_i) &= true \end{aligned} \tag{5.7}$$

Dies wird so lange wiederholt, bis die Bedingung $\theta_{dis} \geq 30$ erfüllt ist. Die Distanz von 30 Zentimetern wird verwendet, da diese mit großer Wahrscheinlichkeit über mehrere Erfahrungen und Verbindungen verläuft. Somit wird sichergestellt, dass das Hindernis durch RatSLAM sicher wieder erkannt wird. Darüber hinaus ist dies auch eine Länge, die ein Hindernis maximal besitzt (vgl. Kapitel 3.1).

Tritt die zweite Situation (Abbildung 5.8) auf, in der die aktuelle Erfahrung e_i einen Nachfolger e_{i+1} besitzt, kann direkt die Vorwärtssuche starten. An dieser Stelle wird erneut das Verfahren aus der Gleichung 5.6 angewendet. Hierbei gilt ebenfalls dieselbe Abbruchbedingung. Ist die Vorwärtssuche abgeschlossen, wird das Verfahren aus Gleichung 5.7 gestartet. Auch bei diesem Verfahren gilt die zuvor definierte Abbruchbedingung.

Damit die Verfahren bei einer erneuten Erkennung an exakt der gleichen Stelle nicht noch einmal starten, muss die Situation gemäß Kapitel 5.1 erfragt werden. Der Grund, warum die Verfahren nur einmal angewendet werden sollen, liegt zum einen darin, dass das Hindernis bereits markiert worden ist. Zum anderen ist die Startbedingung nicht mehr gegeben und es ist nicht sichergestellt, an welcher Stelle das Hindernis markiert wird. Das Verfahren zur Vorwärtssuche wird mit folgender Distanz θ_{maxdis} ausgeführt:

$$\text{Entfernung in cm: } \theta_{maxdis} = 200 \tag{5.8}$$

Sollte die Vorwärtssuche gemäß Kapitel 5.1 auf ein Hindernis auf der rechten Fahrspur treffen, wird ein boolescher Wert *rightObsDetect* auf *true* gesetzt und somit wird die Ausführung der Verfahren verhindert. Ist die Ausführung der Verfahren einmalig verhindert worden, wird der Wert *rightObsDetect* zurück auf $false$ gesetzt.

5.3 Position des Hindernisses auf der linken Fahrbahnseite

Hat die Erkennung für Hindernisse und Kreuzungen ein Hindernis auf der linken Fahrspur identifiziert und validiert (vgl. Kapitel 4.3.2), fährt das Fahrzeug nur an diesem Hindernis vorbei. Dabei nimmt RatSLAM Kenntnis von dem Hindernis. In der Abbildung 5.9 ist die Karte dargestellt, die RatSLAM erstellt. Dabei ist durch die Stelle *A* das Hindernis auf der linken Fahrspur gekennzeichnet. RatSLAM stellt diesen Abschnitt grün dar. Zunächst wird von RatSLAM ein nebenläufiger Pfad erstellt. Durch diesen nebenläufigen Pfad entstehen überflüssige Erfahrungen und Verbindungen, die letztendlich den Berechnungsaufwand vergrößern. Somit muss an dieser Stelle optimiert werden.

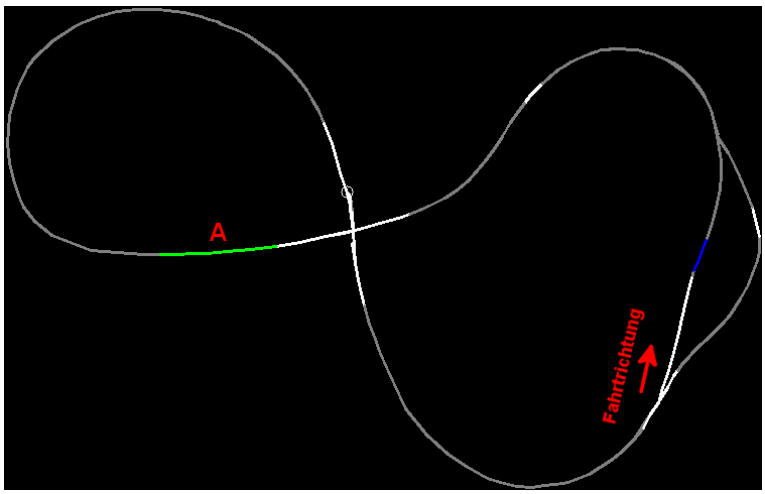

Abbildung 5.9: Experience Map mit Hindernis auf linker Fahrbahnseite (Grün)

Da RatSLAM durch die neue visuelle Szene neue Erfahrungen und Verbindungen erstellt, obwohl sich die Trajektorie nicht verändert hat, muss eine Anpassung der Experience Map vollzogen werden. In den Abbildungen 5.10 und 5.11 sind zum einen ein Teilausschnitt aus der Experience Map sowie die gefahrene Trajektorie dargestellt. Anders als bei der Verarbeitung von Hindernissen auf der rechten Fahrbahnseite (Kapitel

5.2) wird an dieser Stelle nicht nur eine Markierung vorgenommen. Darüber hinaus beginnt das Verfahren nicht sofort nach der Erkennung des Hindernisses. Es werden keine Sensorwerte berücksichtigt. Durch diesen Umstand ist die Position des Hindernisses auf der linken Fahrbahnseite nicht exakt. Außerdem stimmt die eingezeichnete Größe auch nicht mit der realen Größe des Hindernisses überein. Indem allerdings die Sensorwerte nicht berücksichtigt werden, wird das Verfahren zur Integration des neuen Pfades in den ursprünglichen Pfad um ein Vielfaches vereinfacht.

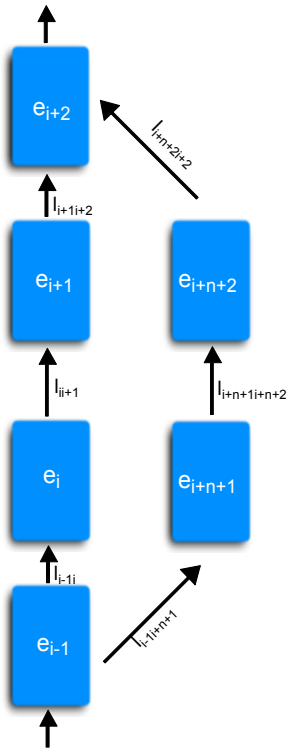

Abbildung 5.10: Teilausschnitt Experience Map - Hindernis links

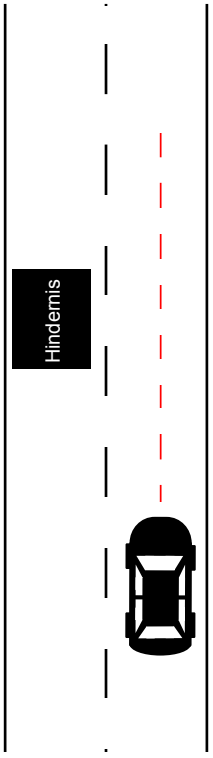

Abbildung 5.11: Trajektorie bei Hindernis auf linker Fahrbahn

Der nebenläufige Pfad beginnt ab der Erfahrung e_{i-1}. Die Relokalisation auf den ursprünglichen Pfad erfolgt bei der Erfahrung e_{i+2}. Der neue Pfad, bestehend aus den Erfahrungen e_{i+n+1} und e_{i+n+2}, verläuft nicht nur in der Abbildung 5.10 parallel und sehr exakt zum ursprünglichen Pfad, sondern auch in der realen Karte. Aus diesem Grund ist eine Integration der neuen Erfahrungen e_{i+n+1} und e_{i+n+2} in die alten Erfahrungen e_i und e_{i+1} möglich. Der Wert n steht für die Gesamtgröße der Experience Map E zum Zeitpunkt t.

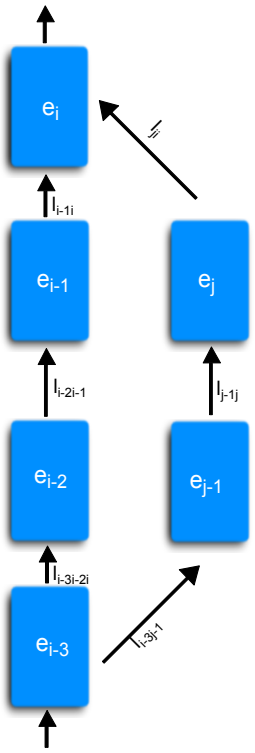

Abbildung 5.12: Ausgangssituation vor Integration

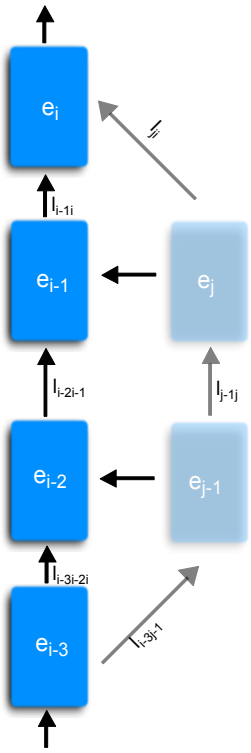

Abbildung 5.13: Situation nach der Integration

Damit das Verfahren zur Integration, wie in den Abbildungen 5.12 und 5.13 dargestellt, beginnen kann, muss die Relokalisierung zurück auf den ursprünglichen Pfad stattgefunden haben. Denn das Verfahren beginnt bei der Erfahrung e_i und fügt dann den neuen Pfad in den ursprünglichen Pfad ein. Sollte die aktuelle Erfahrung keine zwei Vorgängererfahrungen e_{i-1} und e_j besitzen, muss mittels einer leicht veränderten Rückwärtssuche nach Kapitel 5.1 diese Gabelung in den Erfahrungen gefunden werden. Dabei wird für alle Erfahrungen ausgehend von der aktuellen Erfahrung $e_i \in E\ (t)$ zum Zeitpunkt t Folgendes durchgeführt:

$$e_i = e_{i-1} \tag{5.9}$$

Dies wird so lange durchgeführt, bis die aktuelle Erfahrung e_i zwei Vorgängererfahrungen e_{i-1} und e_j über die Verbindungen l_{i-1i} und l_{ji} besitzt. Diese Ausgangssituation ist in der Abbildung 5.12 dargestellt.

Damit die Erfahrungen nun zusammengefasst werden, wird abschnittsweise der ursprüngliche Pfad und darauf aufbauend der neue Pfad durchlaufen. Die Verbindungen l_{i-1i} und l_{ji} werden dabei nicht beachtet. Es wird bei der Erfahrung e_{i-1} sowie e_j gestartet. Es wird für alle Erfahrungen ausgehend von der aktuellen Erfahrung $e_{i-1} \in E\ (t)$ zum Zeitpunkt t Folgendes durchgeführt:

$$\begin{aligned} \theta_{dis} += \Delta p^{jj-1} \\ e_j \mapsto e_{i-1} \\ e_j = e_{j-1} \end{aligned} \tag{5.10}$$

Dabei ist θ_{dis} die aufsummierte Distanz zwischen den Erfahrungen, die auf eine Erfahrung im ursprünglichen Pfad abgebildet werden sollen. Ist folgende Bedingung erfüllt, wird dasselbe Verfahren für die nächste Erfahrung e_{i-2} durchgeführt:

$$\theta_{dis} \geq \Delta p^{i-1i-2} : e_{i-1} = e_{i-2} \tag{5.11}$$

Dieses Verfahren wird insgesamt so lange durchgeführt, bis eine Erfahrung e_{i-1} auftaucht, die zwei Vorgängererfahrungen e_i und e_j besitzt. An dieser Stelle wird das Verfahren abgebrochen.

Damit auch dieses Verfahren nicht erneut startet, wenn dasselbe Hindernis identifiziert und validiert worden ist, muss auch hier eine Überprüfung der Situation erfolgen. Der Grund liegt an der Startbedingung. Die definierte Startbedingung ist nach der Abbildung des neuen auf den ursprünglichen Pfad nicht mehr gegeben. Somit durchläuft der Algorithmus auf der Suche nach der Startbedingung die Erfahrungen. Dabei kommt es vor, dass entweder eine Endlossuche gestartet wird, weil diese Situation nicht noch einmal auftritt oder aber ein falscher auf den ursprünglichen Pfad abgebildet wird. Die Überprüfung ist gleich dem für die Verarbeitung von Hindernissen auf der rechten Fahrbahnseite gemäß Kapitel 5.2 beziehungsweise basiert auf der definierten Vorwärtssuche

aus Kapitel 5.1. Das Verfahren zur Vorwärtssuche wird auch hier mit folgender Distanz θ_{maxdis} ausgeführt:

$$\text{Entfernung in cm: } \theta_{maxdis} = 200 \tag{5.12}$$

Sollte die Vorwärtssuche auf ein Hindernis auf der linken Fahrbahnseite treffen, wird der boolescher Wert $leftObsDetect$ auf $true$ gesetzt. Somit wird die Ausführung des Verfahrens verhindert. Ist die Ausführung des Verfahrens einmalig verhindert worden, wird der Wert $leftObsDetect$ zurück auf $false$ gesetzt.

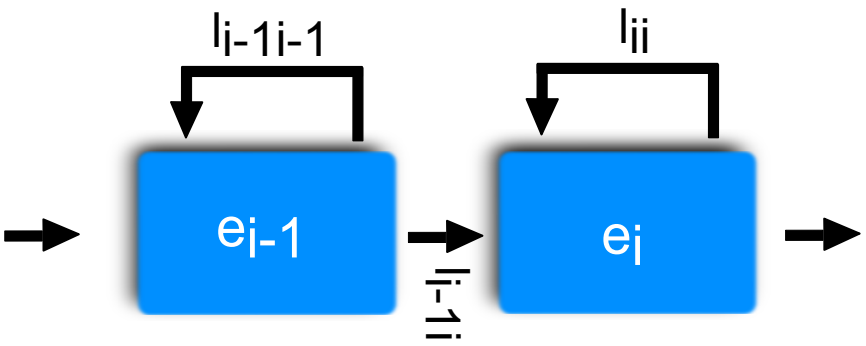

Abbildung 5.14: Erfahrung, die Verbindung auf sich selbst besitzt

Standardmäßig überprüft RatSLAM nicht, zwischen welchen Erfahrungen eine Verbindung erstellt wird. Es wird lediglich überprüft, ob die zu erstellende Verbindung bereits existiert und somit kein zweites Mal erstellt wird. Es kann allerdings der in Abbildung 5.14 dargestellte Aufbau der Experience Map entstehen. Dabei werden Verlinkungen von Erfahrungen auf sich selbst erstellt. Diese Verlinkungen treten im normalen Betrieb von RatSLAM nicht auf. Somit hat es vorher keinen Bedarf gegeben, solch ein Verhalten abzufangen. Durch die Integration des neu erstellten Pfades durch ein Hindernis auf der linken Fahrspur in den ursprünglichen Pfad, ist es vorkommen, dass solche Verbindungen erstellt werden. Verlinkungen, wie zum Beispiel l_{ii} oder l_{i-1i-1}, führen zu einer insgesamt

verzerrten Experience Map. Aus diesem Grund sind solche Verbindungen nicht erlaubt und werden verhindert.

5.4 Kreuzungen

Bei der Erkennung einer Kreuzung fährt das Fahrzeug langsam an diese Kreuzung heran und hält schlussendlich an der Stopplinie. An der Stopplinie wartet das Fahrzeug circa zwei Sekunden. Nach dieser Zeit wird eine Überprüfung vorgenommen, ob es sich um eine blockierte oder freie Kreuzung handelt. Der Gesamtvorgang ist in Kapitel 4.4 beschrieben. Wie RatSLAM eine Kreuzung in die Karte einträgt, ist in der Abbildung 5.15 dargestellt. Die Stelle wird durch die Markierung *A* angezeigt. Eine Kreuzung wird von RatSLAM als roter Pfad dargestellt. Die Verarbeitung von Kreuzungen durch RatSLAM ist nicht trivial. Das Problem besteht in der Inertial Measurement Unit und der Berechnung der Rotation des Fahrzeuges in einem Raum. Diese Berechnungen beruhen auf einer Sensorfusion von einem Kompass sowie einem Gyro Sensor. Da allerdings die Böden im Gebäude Berliner Tor 7 aus Stahlbeton bestehen, schwanken beide Sensoren bei einer langsamen Fahrt beziehungsweise Stillstand.

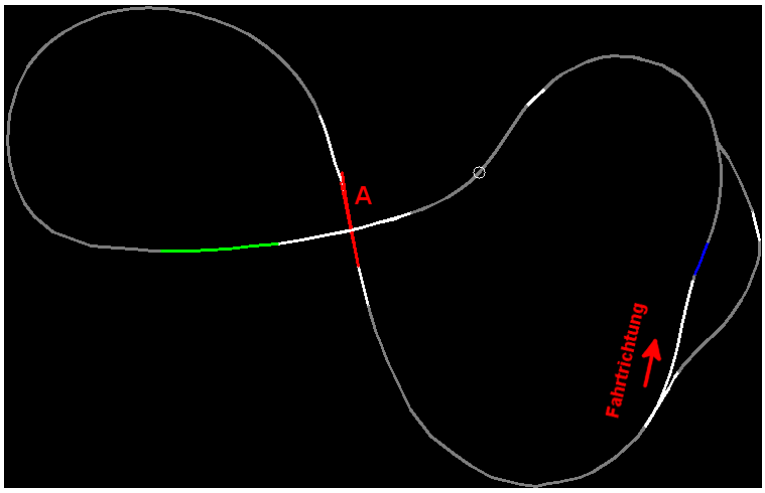

Abbildung 5.15: Experience Map mit Kreuzung (rot)

Durch diesen Umstand erstellt RatSLAM sehr viele Verbindungen und Erfahrung auf der gleichen Position und mit der gleichen visuellen Szene. Das Fahrzeug bewegt sich physisch nicht, allerdings denkt RatSLAM dies durch die Schwankungen des Rotationswertes. In der Abbildung 5.15 ist dies nicht wirklich erkennbar. Es ist nur erkennbar,

wenn eine genaue Aufschlüsselung der Experience Map vorgenommen wird. Ein exemplarischer Teilausschnitt dieser Situation ist in der Abbildung 5.16 dargestellt. Dabei werden willkürlich Erfahrungen und Verbindungen erstellt.

Ein Verfahren, das dies entweder verhindert oder aber an dieser Stelle überflüssige Erfahrungen in eine einzige Erfahrung integriert, ist wie bereits erwähnt nicht trivial. Für diese Aufgabe ist derzeit auch kein adäquater Lösungsansatz entwickelt worden. Es ist der Versuch unternommen worden, die Erstellung von Erfahrungen zu verhindern, wenn die aktuelle Erfahrung e_i als Kreuzung markiert ist. Dafür dient die boolesche Variable *isCross*. Besitzt diese den Wert *true* und existiert eine Nachfolgeerfahrung e_{i+1}, soll keine neue Erfahrung e_{i+2} erstellt werden.

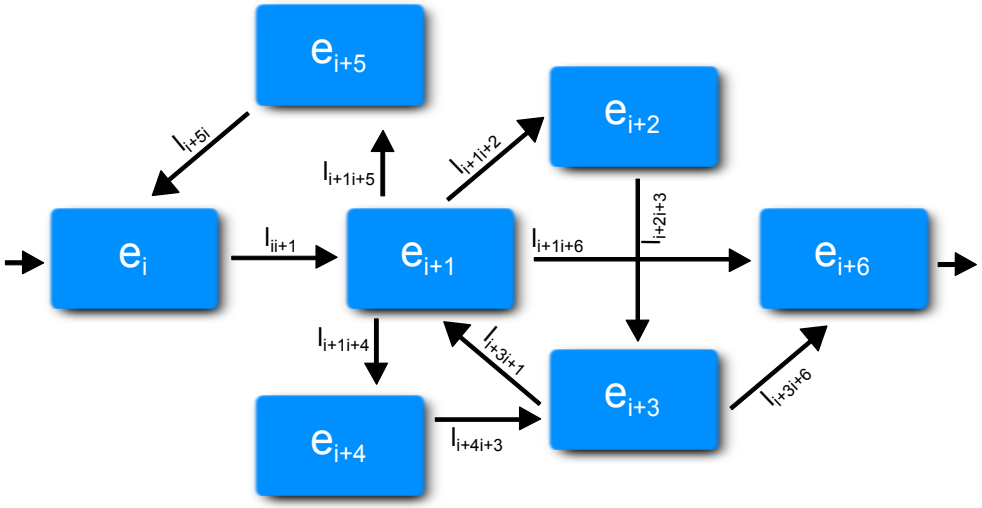

Abbildung 5.16: Teilausschnitt aus Experience Map mit Kreuzung

Anders verhält sich die Situation, wenn es sich um eine blockierte Kreuzung handelt. Dies bedeutet, dass ein fahrendes Hindernis an der Kreuzung steht und das Fahrzeug diesem fahrenden Hindernis die Vorfahrt gewähren muss. Dies ist in der Abbildung 5.17 exemplarisch dargestellt. Dabei wird sehr früh ein nebenläufiger Pfad erstellt, der keinerlei Verbindungen außer der Startverbindung l_{i-1i+7} zum ursprünglichen Pfad besitzt. Zudem werden interessanterweise nicht willkürlich Erfahrungen und Verbindungen auf derselben

Stelle erstellt. Es ist ein durchgehender Pfad, wo jede Erfahrung nur eine Verbindung zum Nachfolger besitzt. Somit ist es möglich, diesen neuen nebenläufigen Pfad zumindest teilweise auf den ursprünglichen Pfad abzubilden.

Damit die Integration des neuen Pfades in den ursprünglichen Pfad beginnen kann, muss auf die Erfahrung zurückgegangen werden, wo zwei Nachfolgeerfahrungen existieren. Dazu wird als aktuelle Erfahrung Folgendes definiert: $e_i = e_{i+9}$. Anschließend wird für alle Erfahrungen ausgehend von der aktuellen Erfahrung $e_i \in E\ (t)$ zum Zeitpunkt t Folgendes durchgeführt:

$$e_i = e_{i-1} \tag{5.13}$$

Dies wird so lange durchgeführt, bis die aktuelle Erfahrung e_i zwei Nachfolgeerfahrungen e_{i+1} und e_{i+2} über die Verbindungen l_{ii+1} und l_{ii+2} besitzt.

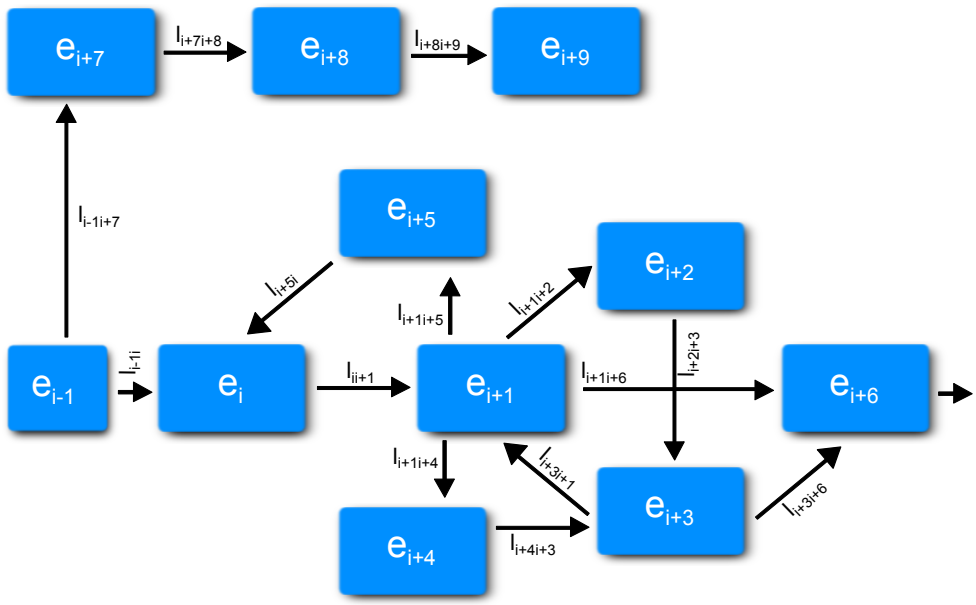

Abbildung 5.17: Teilausschnitt aus Experience Map mit blockierter Kreuzung

Wie bei der Integration von Hindernissen auf der linken Fahrbahnseite in Kapitel 5.3, wird auch in diesem Verfahren abschnittsweise gearbeitet. Darüber hinaus werden die Verbindungen l_{ii+1} und l_{ii+2} nicht beachtet. Fortan wird die Erfahrung e_{i+2} als e_j

bezeichnet. In der Abbildung wäre dies die Erfahrung e_{i+7}. Die aktuelle Erfahrung e_i wird ebenfalls neu zugewiesen: $e_i = e_{i+1}$. Dann wird für alle Erfahrungen ausgehend von der aktuellen Erfahrung $e_i \in E\ (t)$ zum Zeitpunkt t Folgendes durchgeführt:

$$\begin{aligned} \theta_{dis} &= \theta_{dis} + \Delta p^{jj+1} \\ e_j &\mapsto e_i \\ e_j &= e_{j+1} \end{aligned} \tag{5.14}$$

Hierbei ist θ_{dis} wieder die aufsummierte Distanz zwischen den Erfahrungen, die auf eine Erfahrung im ursprünglichen Pfad abgebildet werden sollen. Ist folgende Bedingung erfüllt, wird dasselbe Verfahren für die nächste Erfahrung e_{i+1} durchgeführt:

$$\theta_{dis} \geq \Delta p^{ii+1} : e_i = e_{i+1} \tag{5.15}$$

Es wird dieses Verfahren insgesamt so lange durchgeführt, bis die Erfahrung e_j keine Nachfolgeerfahrung e_{j+1} besitzt. Tritt dies auf, wird das Verfahren unterbrochen.

In der Abbildung 5.18 ist eine mögliche Abbildung des neuen Pfades auf den ursprünglichen Pfad abgebildet. Eine weitere Aufgabe stellt die Fahrt des fahrenden Hindernisses dar. In dieser Situation werden erneut auf derselben Stelle viele neue Erfahrungen und Verbindungen erstellt, da sich die visuelle Szene durchgehend ändert. Für diese Aufgabe ist ebenfalls keine adäquater Lösungsansatz vorhanden. Es wird, solange die aktuelle Erfahrung e_i den booleschen Wert $isCross$ mit $true$ besitzt, keine neue Erfahrung und Verbindung erstellt. Sollte es vorkommen, dass es eine Erfahrung ohne die Markierung $isCross$ existiert, werden infolgedessen viele neue Erfahrungen und Verbindungen erstellt, die nicht effektiv verhindert werden können.

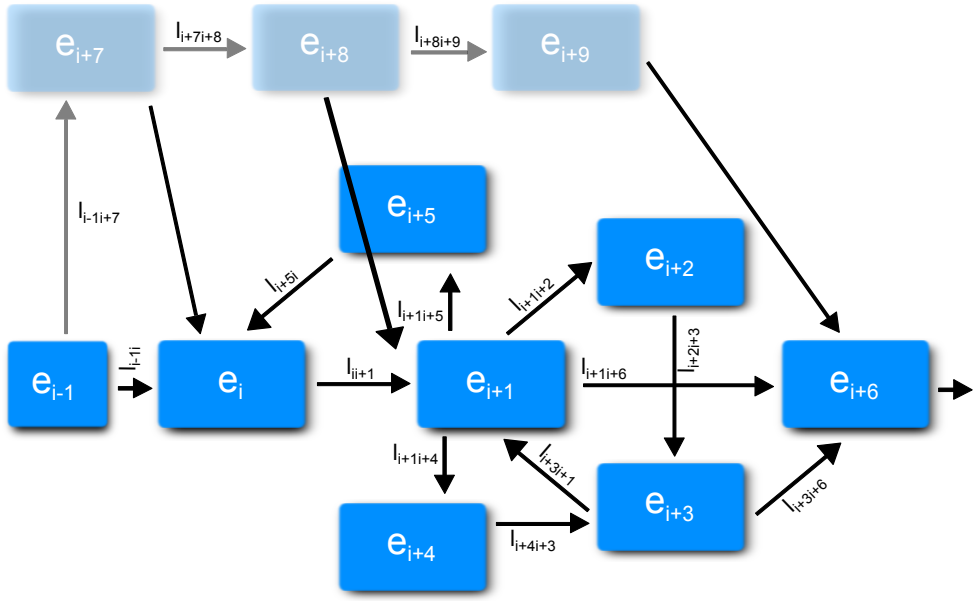

Abbildung 5.18: Teilausschnitt aus Experience Map mit blockierter Kreuzung wird auf ursprünglichen Pfad abgebildet

6 Implementierung der Algorithmen

Dieses Kapitel beschäftigt sich mit der Implementierung der Algorithmen sowie der System-Architektur. Dabei wird vor allem auf die Kommunikation zwischen der FAUST System-Architektur sowie den entwickelten Modulen eingegangen. Weiterhin wird ein Überblick gegeben, welche Werte zur Parametrierung vorhanden sind. Die Werte für die Parameter werden in diesem Kapitel nicht definiert, da sie bereits in den vorherigen Kapiteln definiert worden sind.

6.1 Hindernis- und Kreuzungserkennung

Für die Implementierung der Algorithmen zur Hindernis- und Kreuzungserkennung sind Anpassungen an dem Datencontainer der Spurerkennung *Polaris* durchgeführt worden. Der Datencontainer *PolarisLane* sieht vor, dass nur die aktuell verfolgten Fahrbahnmarkierungen enthalten sind. Dabei werden entweder die mittige und rechte oder die linke und mittige Fahrbahnmarkierung gespeichert. Zur Positionierung der Points of Interest (Kapitel 4.1) werden alle drei Polynome benötigt. Auf Grund dieser Tatsache ist der Datencontainer *PolarisLane* so erweitert worden, dass dieser alle Polynome gespeichert. Zusätzlich ist die Spurführung *SteeringControl* angepasst worden, sodass diese auch mit drei Polynomen arbeitet.

Die Architektur zur Erkennung von Hindernissen und Kreuzungen ist in der Abbildung 6.1 dargestellt. Im Wesentlichen besteht die Architektur aus einem Zustandsautomaten mit drei Zuständen sowie einem zentralen Ort für die Parameter (*OCParameter*). Alle Parameter, die in *OCParameter* deklariert werden, sind über das Webinterface zur Laufzeit einstellbar. Auf alle Parameter kann von jedem Zustand aus zugegriffen werden. Der Zustand *ScanState* sucht nach Kreuzungen und Hindernissen. Sollte eine Kreuzung oder ein Hindernis identifiziert und validiert sein, wird in den entsprechenden Zustand *CrossingState* oder *ObstacleState* gewechselt.

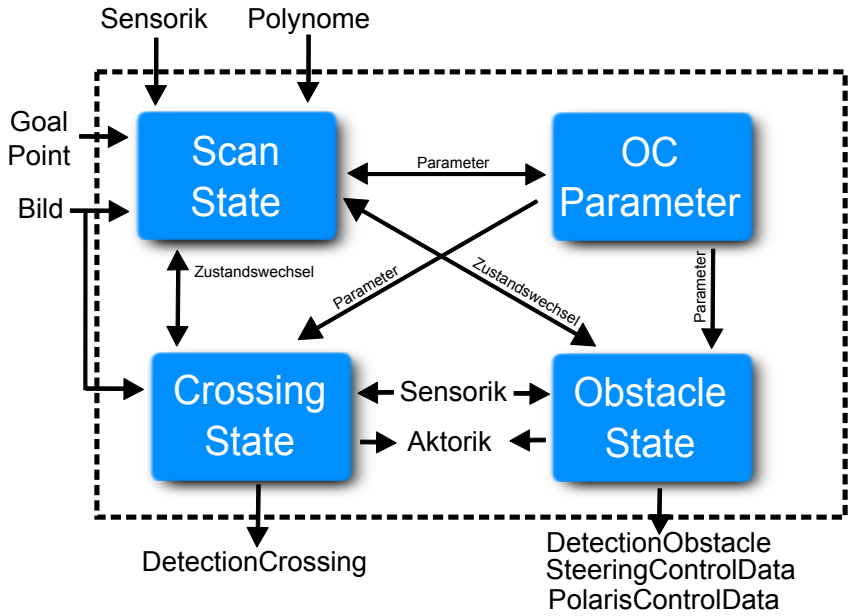

Abbildung 6.1: Architektur des Algorithmus zur Hindernis- und Kreuzungserkennung

An Parametern von *OCParameter* bezieht der Zustand *ScanState* folgende:

- *threshold*, *thresholdObject*, *thresholdLine*: Multiplikator für berechneten Schwellwert (Gleichung 4.31, 4.56)
- *obstacleIRRightStraight*, *obstacleIRRightCurve*: Schwell-Distanz des Infrarot Sensors zur Validierung auf rechter Fahrbahnseite (Kapitel 4.3.1)
- *obstacleIRLeftStraight*, *obstacleIRLeftCurve*: Schwell-Distanz des Infrarot Sensors zur Validierung auf linker Fahrbahnseite (Kapitel 4.3.2)
- *ScanLineRightLanePosX*, *ScanLineRightLaneHeight*, *ScanLineLeftLanePosX*, *ScanLineLeftLaneHeight*: Parameter zum Einstellen der Suchlinien (Kapitel 4.4)

Sofern *ScanState* ein Hindernis identifiziert und validiert hat, wird auch ein Parameter in *OCParameter* verändert. Dieser Wert nennt sich *obstacleRight*. In diesem Parameter wird gespeichert, ob sich das identifizierte Hindernis auf der linken oder rechten Fahrspur befindet. Der Zustand *ScanState* bezieht von den Datencontainern *CameraImage* das aktuelle Bild, *GoalPoint* liefert die Angabe zur Kurvensituation, *SensorValues* enthält die aktuellen Sensorwerte und in *PolarisLane* sind die Polynome zur Berechnung der Points of Interest enthalten.

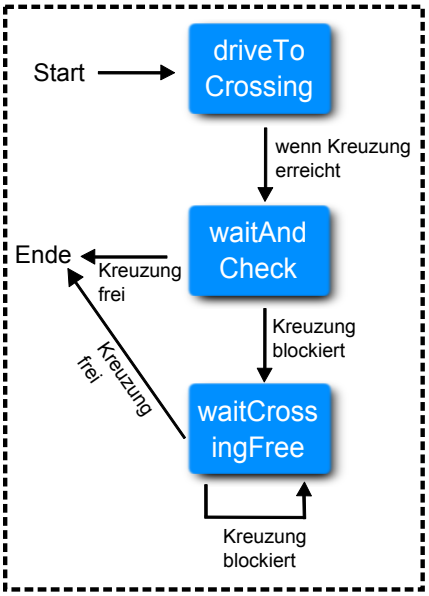

Abbildung 6.2: Zustandsautomat Kreuzungen (CrossingState)

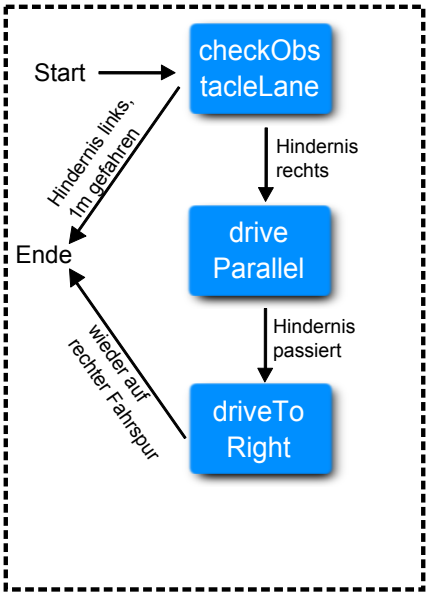

Abbildung 6.3: Zustandsautomat Hindernisse (ObstacleState)

Die beiden Zustände *CrossingState* sowie *ObstacleState* beinhalten einen Subautomaten. Der Subautomat für den Zustand *CrossingState* ist in der Abbildung 6.2 skizziert. Dieser Subautomat besitzt drei Zustände. Im ersten Zustand *driveToCrossing* fährt das Fahrzeug mit der Geschwindigkeit *velocityCrossing* an die Kreuzung heran. Sobald der Point of Interest einen Grauwert erfasst, der über dem Schwellwert liegt, wird der Zustand gewechselt. Dabei ist genau wie die Geschwindigkeit auch der Point of Interest über Parameter einstellbar: *crossingSearchY* und *crossingSearchX* (Kapitel 4.4). Ist das Fahrzeug an die Kreuzung herangefahren, wird in den Zustand *waitAndCheck* gewechselt. In diesem Zustand wird zunächst zwei Sekunden gewartet und nach dieser Zeit geprüft, ob die Kreuzung blockiert ist. Dazu werden die Infrarot Sensoren gemäß Kapitel 4.4 abgefragt. Die zu unterschreitende Messdistanz ist mittels des Parameters *distanceCrossBlocked* einstellbar. Ist die gemessene Distanz größer, so wird die Kreuzung überquert und der Datencontainer *DetectionCrossing* erzeugt, indem angegeben ist, dass eine Kreuzung identifiziert worden ist. Darüber hinaus steht in dem Datencontainer, dass kein Fahrzeug von rechts gekommen ist, wodurch die Kreuzung blockiert wäre. Liegt die Messdistanz unter dem angegebenen Wert, so gilt die Kreuzung als blockiert. Es wird in den Zustand *waitCrossingFree* gewechselt. In diesem Zustand wird gewartet, bis alle

Infrarot Sensoren eine Distanz messen, die größer als *distanceCrossBlocked* ist. Solange die Kreuzung blockiert ist, wird weiterhin der Datencontainer *DetectionCrossing*, mit dem Zusatz, dass die Kreuzung blockiert ist, erstellt. Hat das dynamische Hindernis die Kreuzung passiert, wird der Subautomat beendet.

In Abbildung 6.3 ist der Subautomat für den Zustand *ObstacleState* dargestellt. Wird der Subautomat betreten, wird zunächst der Zustand *checkObstacleLane* aktiviert. In diesem Zustand wird überprüft, ob der durch *OCParameter* übergebene Wert *obstacleRight* wahr oder falsch ist. Ist der Wert falsch, ist ein Hindernis auf der linken Fahrbahnseite identifiziert worden. In diesem Fall fährt das Fahrzeug eine bestimmte Strecke. Diese Strecke ist durch *leftObsDistancePassed* im Webinterface einstellbar. Zusätzlich wird während dieser Zeit der Datencontainer *DetectionObstacle* erstellt. In diesem Datencontainer wird die Entfernung des Hindernisses gespeichert. Darüber hinaus wird auch gespeichert, ob das Hindernis passiert worden ist. Dieser Wert wird auf wahr gesetzt, wenn *leftObsDistancePassed* überschritten worden ist. Befindet sich das Hindernis rechts (*obstacleRight* ist wahr), wird in den Zustand *driveParallel* gewechselt. Außerdem wird einmalig der Datencontainer *DetectionObstacle* erstellt. Es wird der linke Blinker aktiviert und die parametrierbare Geschwindigkeit *velocityObstacle* aus *OCParameter* eingestellt. Im Zustand *driveParallel* wird weiterhin die Geschwindigkeit *velocityObstacle* beibehalten. Es wird zum einen gewartet, bis sich das Fahrzeug parallel zum Hindernis befindet. Dies meldet der digitale Infrarot Sensor, der sich hinten rechts befindet. Ist dies der Fall, wird der Blinker deaktiviert. Befindet sich das Fahrzeug parallel zum Hindernis, wird zum anderen gewartet, bis das Hindernis passiert ist. Wenn dies eingetreten ist, wird der rechte Blinker aktiviert und in den Zustand *driveToRight* gewechselt. In diesem Zustand wird gewartet, bis die Distanz *driveToRightSide* überschritten worden ist. Ist diese Distanz überschritten, wird der rechte Blinker deaktiviert und die normale Geschwindigkeit wieder eingestellt. In Kapitel 4.3 sind die Standardwerte für die Parameter angegeben. Außerdem wird der Subautomat beendet.

Eine weitere Anpassung ist an *Polaris* selbst vorgenommen worden. In der Regel wird die rechte Fahrbahnseite verfolgt, da auf dieser Seite die Erkennung von Linien und somit die Approximation von Polynomen stabil ist. Lediglich zum Ausweichen eines Hindernisses auf der rechten Fahrbahnseite wird kurzfristig die linke Fahrbahnseite verfolgt und auch auf diese Fahrbahnseite die Spur gewechselt. Beim Wechsel zurück auf die rechte Fahrbahnseite kann es allerdings vor kommen, dass die Polynome nicht erfolgreich zurück getauscht werden. Allerdings wird dieser Befehl nur einmal gesendet.

Aus diesem Grund ist Polaris so erweitert worden, so dass die Standardeinstellung die Verfolgung der rechten Fahrbahnseite ist. Dies wird immer wieder ausgeführt, wenn nicht der Datencontainer *PolarisControlData* existiert.

6.2 RatSLAM Erweiterung für dynamische Umgebungen

In der Abbildung 6.4 ist die modifizierte System-Architektur für RatSLAM dargestellt. Die Erklärung der grundlegenden System-Architektur für RatSLAM ist bereits in Kapitel 2.1 beschrieben. Insofern wird an dieser Stelle nur auf die Erweiterung zur Integration einer dynamischen Umgebung in die Experience Map erläutert. In der Experience Map werden die Datencontainer *DetectionCrossing*, *DetectionObstacle* und *GoalPoint* ausgewertet. Anhand des Datencontainers *GoalPoint* wird ermittelt, ob sich das Fahrzeug in einer Kurve befindet und ob es sich um eine Linkskurve oder eine Rechtskurve handelt. Beides wird in den Erfahrungen gespeichert. Die Datencontainer *DetectionCrossing* und *DetectionObstacle* besitzen eine tiefer gehende Integration in der Experience Map.

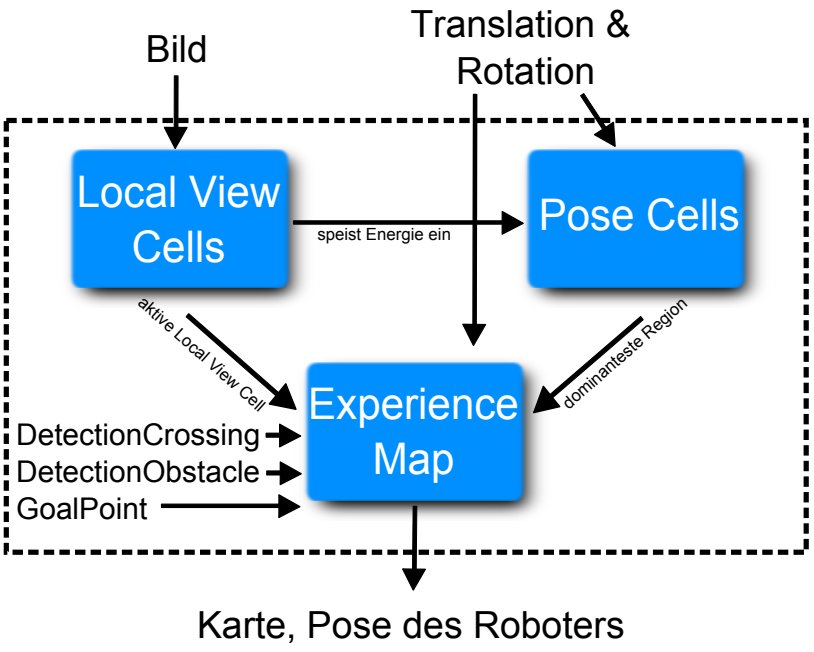

Abbildung 6.4: Modifizierte RatSLAM Architektur

Enthält der Datencontainer *DetectionObstacle* die Information, das ein Hindernis auf der rechten Fahrbahnseite erkannt worden ist, wird der Algorithmus aus Kapitel 5.2 ausgeführt. In diesem wird zunächst geprüft, ob die aktuelle Erfahrung eine Verbindung zu einer nächsten Erfahrung besitzt. Ist dies der Fall, wird in der Entfernung, die im Datencontainer angegeben ist, das Hindernis in die entsprechenden Erfahrungen eingetragen. Besitzt die aktuelle Erfahrung keine Verbindung zu einer nachfolgenden Erfahrung, wird solange in den Erfahrungen zurückgegangen, bis eine Gabelung identifiziert worden ist. Von dieser Gabelung aus, wird dann in der Entfernung, die im Datencontainer angegeben ist, das Hindernis eingetragen. Bei diesem Algorithmus gibt es keine einstellbaren Parameter. Entweder werden entsprechende Parameter durch den Datencontainer *DetectionObstacle* geliefert oder sind fest definiert.

Ist in dem Datencontainer *DetectionObstacle* die Information enthalten, das ein Hindernis auf der linken Fahrbahn identifiziert worden ist, wird diese Information in den neuen Erfahrungen gespeichert. Erst wenn der Datencontainer *DetectionObstacle* die Information übermittelt, dass das linke Hindernis passiert worden ist, startet der Algorithmus aus Kapitel 5.3. Dabei muss beachtet werden, dass der Algorithmus erst beginnen soll, wenn das Fahrzeug komplett an dem Hindernis vorbei gefahren ist. Wird der Algorithmus früher gestartet, kommt es vor, dass die Relokalisierung noch nicht ausgeführt worden ist und die zweite Gabelung nicht existiert. Somit findet der Algorithmus den Startpunkt nicht. Startet der Algorithmus korrekt, wird der neu erstellte Pfad in den ursprünglichen Pfad integriert. Dabei werden die Informationen übernommen, die im alten Pfad gespeichert sind. Auch dieser Algorithmus bietet keine Möglichkeiten der Parametrisierung, da entweder die entsprechenden Daten im Datencontainer enthalten beziehungsweise nicht parametrierbar sind.

Die Informationen über eine Kreuzung sind in dem Datencontainer *DetectionCrossing* enthalten. In diesem Datencontainer wird gespeichert, ob eine Kreuzung blockiert ist und ab wann die Kreuzung passiert worden ist. Wird an eine Kreuzung herangefahren, wird nur bei der ersten Fahrt auf die Kreuzung die Erstellung von Erfahrungen erlaubt. Ansonsten wird die Erstellung nicht erlaubt. Der Grund für diesen Schritt ist in Kapitel 5.4 beschrieben. Die Integration einer blockierten Kreuzung erfolgt rückwirkend, sobald der Datencontainer diese Information beinhaltet. Dabei besitzt die aktuelle Erfahrung keinen Nachfolger, sodass dies als Startbedingung angenommen wird. Die Integration ist nicht besonders leicht, da im Bereich der Kreuzung nicht genau gesteuert werden

kann, auf welche Erfahrungen der neue Pfad abgebildet wird. Darüber hinaus bietet der Algorithmus zur Integration einer blockierten Kreuzung keine Parameter an.

7 Auswertung in einer definierten Testumgebung

7.1 Aufbau der Testumgebung

Die Testumgebung besteht aus fünf Runden, die das Fahrzeug zurücklegen muss. Dabei ist in den Abbildung 7.1 bis 7.5 eingezeichnet, in welcher Runde welches Hindernis an welchem Ort steht. In den ersten beiden Runden befindet sich nichts auf der Fahrbahn. Die Begründung liegt darin, dass zunächst eine Karte erstellt werden soll, die nichts enthält. Dies ist eine Voraussetzung für die implementierten Algorithmen gemäß Kapitel 5. In der dritten Runde befinden sich die Hindernisse H_1 und H_2 auf der Strecke. Das Hindernis H_1 steht kurz vor der Start- und Stopplinie auf der rechten Fahrbahnseite. Es sind keine Hindernisse direkt auf der Start- und Stopplinie erlaubt. Würde ein Hindernis auf der Start- und Stopplinie stehen, ist es möglich, dass dieses nicht als Hindernis, sondern als Start- und Stopplinie erkannt wird. Kurz nach der Kreuzung kommt auf der linken Fahrspur das Hindernis H_2. In der vierten und fünften Runde wird die Blockierung einer Kreuzung durch das fahrende Hindernis H_3 simuliert. Dabei wird eine nicht näher definierte Zeit gewartet, bis das fahrende Hindernis die Kreuzung räumt. Die gewartete Zeit beträgt mehr als zwei Sekunden. Die Aufgabe besteht zum einen in der korrekten Erkennung der Hindernisse sowie der Kreuzung. Dabei ist es egal, ob die Kreuzung blockiert oder frei ist. Zum anderen müssen die Hindernisse und Kreuzungen korrekt in die Experience Map integriert werden. Es soll nach Möglichkeit kein Anstieg in den Erfahrungen und Verbindungen durch Hindernisse erfolgen, nachdem die Ursprungskarte erstellt ist.

Die Geschwindigkeit des Fahrzeuges beträgt in der normalen Fahrt 100 $\frac{cm}{s}$ und wird von einem Regler gehalten. Haben die Algorithmen entweder ein Hindernis auf der rechten Fahrbahnseite oder eine Kreuzung identifiziert, wird die Geschwindigkeit auf 60 $\frac{cm}{s}$ verringert. Der Scheduler arbeitet mit einer Zeit von 30 Millisekunden. Dies bedeutet, dass pro Sekunde circa 33 Bilder aufgenommen und verarbeitet werden müssen. Die

Einstellungen für die Sensorik zur Erkennung von Hindernissen und Kreuzungen sind dem Kapitel 4.3 und 4.4 zu entnehmen. Es wird nur der in Kapitel 4 beschriebene Algorithmus zur Erkennung von Hindernissen und Kreuzungen getestet. Es werden keine verwandte Verfahren implementiert und getestet. Es findet kein Test der einzelnen Komponenten bestehend aus Erkennung von Hindernissen und Kreuzungen sowie RatSLAM statt. Beide Komponenten werden gemeinsam mit dem Aufbau der einzelnen Runden gemäß der Abbildungen 7.1 bis 7.5 getestet.

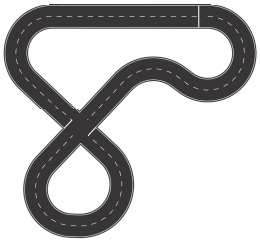

Abbildung 7.1: Erste Runde

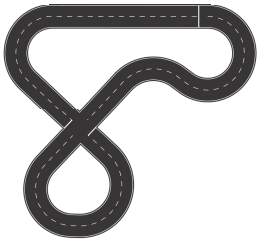

Abbildung 7.2: Zweite Runde

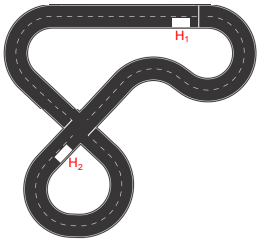

Abbildung 7.3: Dritte Runde

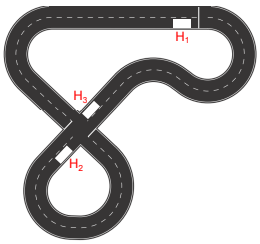

Abbildung 7.4: Vierte Runde

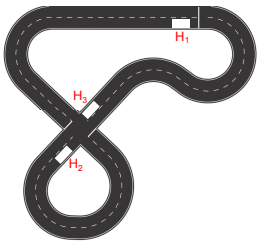

Abbildung 7.5: Fünfte Runde

7.2 Test und Auswertung ohne Modifizierung von RatSLAM

In dieser Auswertung soll die Situation dargestellt werden, wie sich RatSLAM ohne die entsprechenden Algorithmen aus Kapitel 5 verhält und welche Effekte dabei auftreten. Nebenbei lief die Erkennung für Hindernisse und Kreuzungen, um die diese zu testen. In der ersten und zweiten Runde erkennen die Algorithmen weder Hindernisse auf der linken noch auf der rechten Fahrspur. Somit finden in den ersten beiden Runden keine Falscherkennungen von Reflexionen statt. Die freie Kreuzung ist in den ersten

beiden Runden jedes Mal einwandfrei erkannt worden. In der dritten Runde sind die Hindernisse H_1 und H_2 identifiziert und validiert worden. In Runde vier und fünf sind die Erkennungen des fahrenden Hindernisses H_3 an der Kreuzung erfolgreich durchgeführt worden. Zu keiner Zeit ist das Fahrzeug mit einem Hindernis kollidiert. Die durch RatSLAM erstellte topologische Karte ist in der Abbildung 7.6 dargestellt. Auf den ersten Blick sind keine grundlegenden Veränderungen zu erkennen. Diese Karte entspricht weitestgehend der Skizze der Teststrecke (Abbildungen 7.1 bis 7.5).

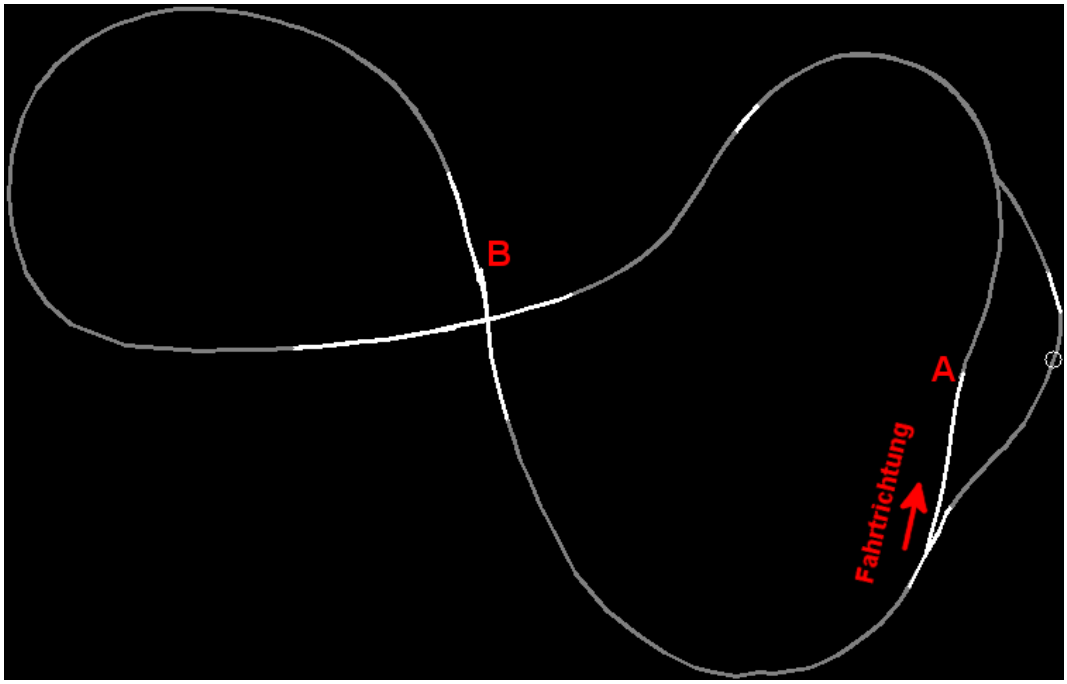

Abbildung 7.6: Erstellte Karte durch RatSLAM ohne Modifizierung

Wird die erstellte Karte genauer betrachtet, so fällt auf, dass auf der geraden Strecke bei der Start- und Stopplinie eine Einbuchtung existiert. Durch diesen Umstand erscheint eine eigentlich gerade Strecke leicht gekrümmt. Diese Stelle ist durch das rote *A* markiert. Diese Einbuchtung hängt mit der Wartezeit an der Kreuzung zusammen. Die berechnete Rotation des Fahrzeuges durch die Inertial Measurement Unit (IMU) dreht bei einem Stopp an der Kreuzung leicht nach. Dies hat zur Folge, dass die nach der Kreuzung erstellten Daten ausgehend vom Startpunkt an der Stelle *A* alle leicht abweichen. Zudem ist mit der Stelle *B* ein weiterer Ort markiert. An dieser Stelle erscheint eine kleine Spitze. Diese Spitze ist ebenfalls ein Resultat des Stopps an der Kreuzung. Durch den Stopp und die Veränderung der Rotation des Fahrzeuges werden neue Erfahrungen und

Verbindungen am selben Ort erstellt. Durch die Änderung der Rotation auf dem gleichen Punkt denkt RatSLAM, dass es sich um eine reale Positionsveränderung handelt.

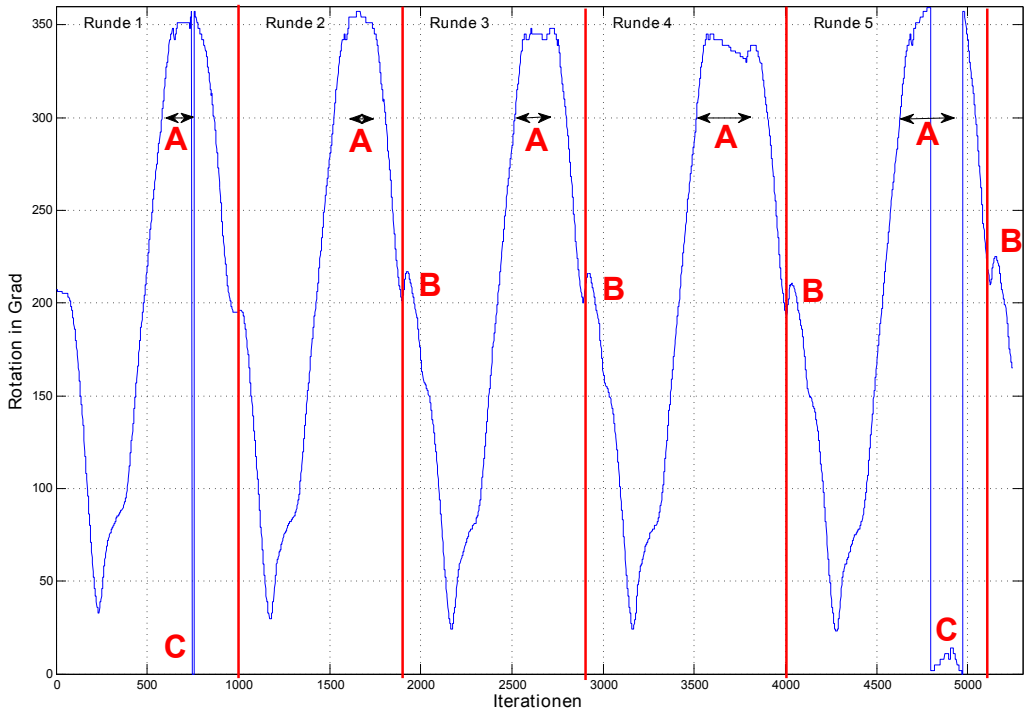

Abbildung 7.7: Rotation des Fahrzeuges während der Testfahrt

In der Abbildung 7.7 ist die Rotation des Fahrzeuges über die gefahrenen fünf Runden dargestellt. Die eingezeichneten Doppelpfeile, die mit *A* markiert sind, zeigen an, wo sich das Fahrzeug an einer Kreuzung im Stillstand befindet. Die extremen Abstürze der Verlaufskurve an den Stellen *C* in der ersten Runde sowie der fünften Runde sind keine Fehler. An diesen Stellen sind die Rotationen über 360 Grad gelaufen und somit wieder bei null angefangen. Der dargestellte Verlauf zeigt, dass sich gerade während eines Stillstandes die Rotation verändert. Zudem ist die Rotation an einer Kreuzung im Stillstand nie gleich. Durch diese Ungleichheit ist es wahrscheinlich, dass bei jedem Stopp an einer Kreuzung neue Erfahrungen und Verbindungen erstellt werden. Ab der zweiten Runde existiert auch nach Überquerung der Start- und Stopplinie ein kleiner Anstieg. Diese Stelle wird durch *B* markiert. An dieser Stelle weicht das Fahrzeug jedes Mal dem Hindernis H_1 aus. Durch diesen Ausweichvorgang werden ebenfalls neue Erfahrungen und Verbindungen erstellt.

Die Abbildung 7.8 zeigt eine Verlaufskurve für die Anzahl der Erfahrungen (Kurve in Blau) und die Anzahl der Verbindungen (Kurve in Grün). Die einzelnen Runden sind mit roten Linien markiert. Die Hindernisse H_1 bis H_3 sind mit ihrer Erscheinungsdauer als Doppelpfeile eingezeichnet. Die erste Runde besteht aus einem schnellen Anstieg der Erfahrungen und Verbindungen. Der Anstieg beider Kurven verläuft parallel. Dies ist gewollt, da in dieser Runde die Karte durch RatSLAM aufgebaut wird. In der zweiten Runde bleibt die Anzahl der Erfahrungen und Verbindungen bis zur Hälfte der Runde konstant.

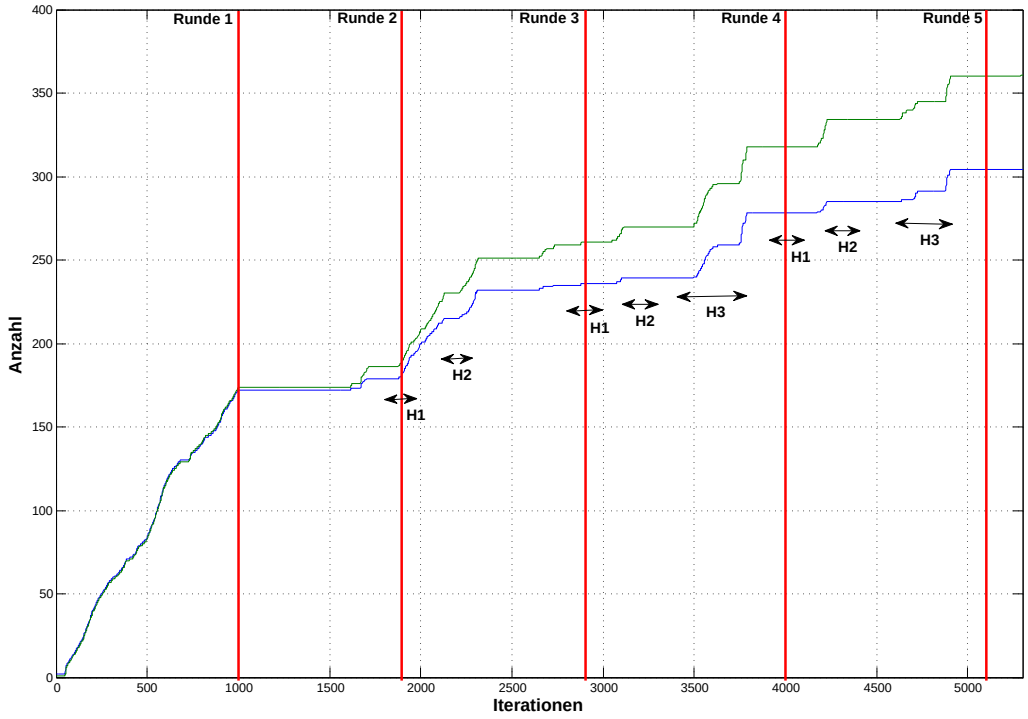

Abbildung 7.8: Anzahl der Erfahrungen und Verbindungen während der Testfahrt

Allgemein ist der Abbildung 7.8 zu entnehmen, dass beim ersten Erscheinen eines Hindernisses ein Anstieg in der Anzahl der Erfahrungen und Verbindungen auftaucht. Darüber hinaus findet zwischen dem ersten Erscheinen der Hindernisse H_1 und H_2 ein konstanter Anstieg statt. Die Begründung für diesen Anstieg der Erfahrungen und Verbindungen ist der Ausreißer, der durch die Markierung B in der Abbildung 7.7 bezeichnet ist. An dieser Stelle verändert sich die Rotation des Fahrzeuges durch den Ausweichvorgang für das Hindernis H_1. Da der weitere Verlauf von der zuvor erstellten

Strecke nicht übereinstimmt, wird ein neuer paralleler Pfad erstellt. In den Runden zwei und drei erfolgt ebenfalls ein moderater Anstieg in den Erfahrungen und Verbindungen. Hierbei erreicht das Fahrzeug die nicht blockierte Kreuzung. Die Erklärung für dieses Phänomen ist bereits im Zusammenhang mit der Abbildung 7.7 sowie der Markierung *A* erläutert worden. Erscheint das fahrende Hindernis H_3, so wird in beiden Fällen ein Anstieg der Erfahrungen und Verbindungen ausgelöst. Wird H_3 zum ersten Mal wahrgenommen, ist der Anstieg um einiges größer als bei der zweiten Wahrnehmung. Die Begründung für den ersten Anstieg liegt in der ersten Wahrnehmung sowie die Überquerung der Kreuzung durch das Hindernis H_3. Der zweite Anstieg besitzt die gleiche Begründung, wobei an dieser Stelle aufgrund einer verschobenen Rotation des Fahrzeuges das Ereignis ausgelöst wird.

7.3 Test und Auswertung mit Modifizierung von RatSLAM

Diese Auswertung analysiert die Ergebnisse, die durch die Modifizierung von RatSLAM erreicht worden sind. In diesem Test sind die in Kapitel 5 beschriebenen Verfahren implementiert und auf die beschriebenen Testdaten aus Kapitel 7 angewendet worden. Die topologische Karte, die RatSLAM mit den Modifizierungen erstellt, ist in der Abbildung 7.9 dargestellt. In dieser Karte ist noch keine blockierte Kreuzung enthalten. Dies bedeutet, dass das Fahrzeug bei dieser Karte nur die ersten drei Runden absolviert hat. Insgesamt sieht die Karte zu diesem Zeitpunkt so aus wie die Skizze (Abbildungen 7.1 bis 7.5).

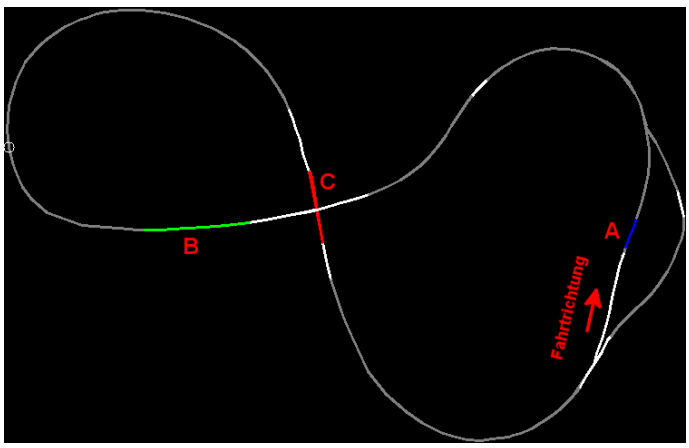

Abbildung 7.9: Erstellte Karte durch RatSLAM mit Modifizierung ohne blockierte Kreuzung

An der Markierung *A* ist das erste Hindernis auf der rechten Fahrbahnseite dargestellt. Hindernisse auf der rechten Fahrspur werden durch einen blauen Pfad in der topologischen Karte dargestellt. Der Ausweichpfad ist gut dargestellt. Die Markierung *B* zeigt den Ort des zweiten Hindernisses auf der linken Fahrbahnseite an. Hindernisse, die sich links auf der Fahrspur befinden, werden von RatSLAM durch einen grünen Pfad dargestellt. Die dritte Markierung *C* stellt die Kreuzung dar. Kreuzungen werden von RatSLAM durch einen roten Pfad visualisiert. Die Kreuzung läuft gerade aus und wird circa in der Hälfte durchquert. Dies ist ausgezeichnet, da die Fahrbahn eine Breite von 80 Zentimetern besitzt. Würde die Kreuzung nicht in der Hälfte durchquert werden, bedeutet dies, dass die Karte an sich nicht stimmt.

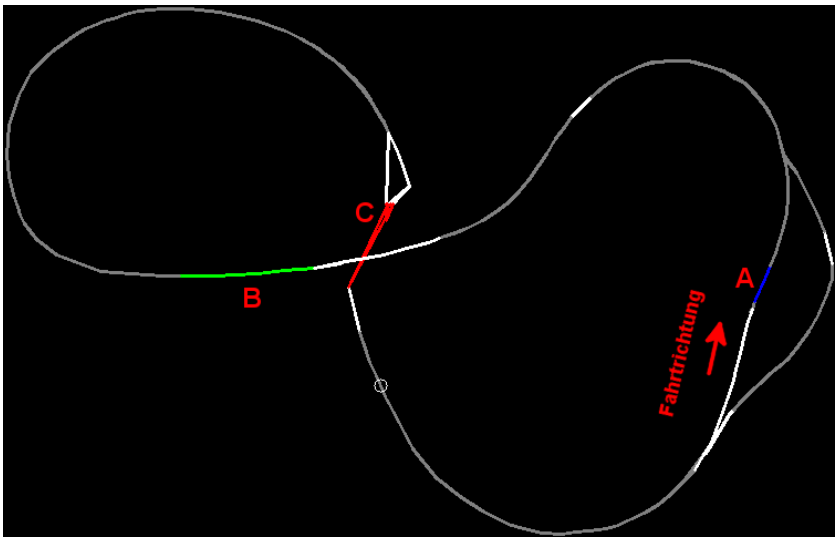

Abbildung 7.10: Erstellte Karte durch RatSLAM mit Modifizierung mit blockierter Kreuzung

Die Karte in der Abbildung 7.10 beinhaltet die vierte und fünfte Runde. Die Markierungen *A* und *B* werden auch in diesen beiden Runden weiterhin gut dargestellt. Nach der Integration des neuen Pfades bei einer blockierten Kreuzung, hat sich die Situation bei der Markierung *C* verschlechtert. Wie bereits in Kapitel 5.4 beschrieben, ist die Integration einer Kreuzung nicht trivial. Durch leichte Schwankungen der Rotation des Fahrzeuges bilden sich viele Erfahrungen und Verbindungen an der gleichen Stelle. Dies ist in der Abbildung 7.9, wo nur drei von fünf Runden absolviert worden sind, nicht sichtbar. Im Gegensatz dazu wird dies in der Abbildung 7.10 sehr deutlich sichtbar. Es existieren zwei Pfade, die nicht als Kreuzung deklariert sind. Dies ist daran erkennbar,

dass diese Pfade weiß sind. Ebenfalls ist erkennbar, dass der rote Pfad, welcher als Kreuzung markiert wird, aus mehreren Pfaden besteht. Zudem ist dieser Teilabschnitt der Kreuzung insgesamt leicht verzerrt und nicht mehr eine gerade Strecke wie in Abbildung 7.9.

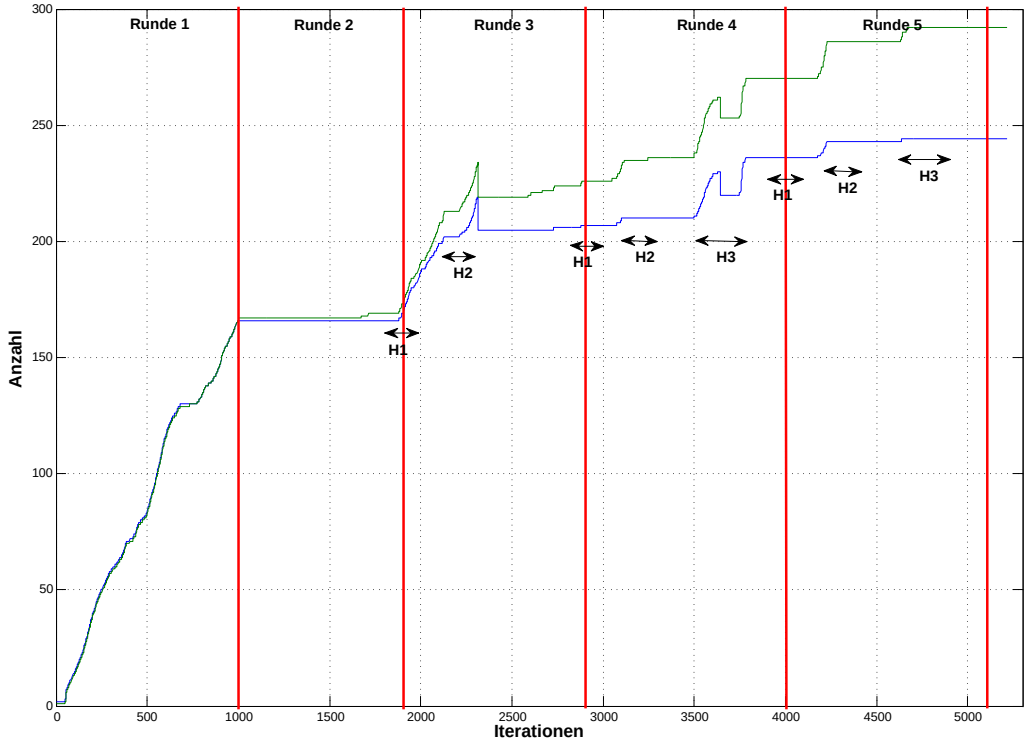

Abbildung 7.11: Anzahl der Erfahrungen und Verbindungen während der Testfahrt

Die Analyse der Entwicklung der Anzahl von Erfahrungen und Verbindungen ist in der Abbildung 7.11 dargestellt. Dabei ist auf der x-Achse die Anzahl der Iterationen aufgetragen. Die y-Achse stellt die Anzahl der Erfahrungen und Verbindungen dar. Die grüne Kurve ist die Anzahl der Verbindungen. Die Anzahl der Erfahrungen ist durch die blaue Kurve dargestellt. Wird die Analyse der Entwicklung vom unmodifizierten RatSLAM aus Abbildung 7.8 mit der Entwicklung in der Abbildung 7.11 verglichen, fällt zunächst die deutlich geringere Grenze der y-Achse auf. Die Anzahl der Verbindungen ist von circa 360 auf rund 290 gesunken. Dies entspricht einer Senkung um 24 Prozent. Es ist ebenfalls die Anzahl der Erfahrungen stark gesunken. Die Anzahl beträgt bei der unmodifizierten Version rund 304 Erfahrungen. Bei der modifizierten Version sind es

dagegen nur noch 244 Erfahrungen. Dies entspricht einer Senkung der Anzahl um rund 25 Prozent.

In der Abbildung 7.11 ist bei der Erscheinung des zweiten Hindernis H_2 ein Anstieg zu erkennen. Nachdem das Verfahren aus Kapitel 5.3 angewendet worden ist, sinkt die Anzahl der Erfahrungen und Verbindungen schlagartig. Bei der zweiten und dritten Erkennung bleibt die Anzahl der Erfahrungen und Verbindungen konstant. Kurz vor der Erkennung steigt die Anzahl allerdings an. Dies liegt daran, dass neue Erfahrungen und Verbindungen erstellt werden, um die Schwankung der Rotation des Fahrzeuges auszugleichen. Das erste Hindernis H_1 erzeugt bei der ersten Erkennung einen massiven Anstieg der Erfahrungen und Verbindungen. Es wird bei der Erkennung des Hindernisses H_1 keine Anpassung des Pfades vorgenommen, wie es in Kapitel 5.2 beschrieben ist. Bei der zweiten und dritten Erkennung ist auch bei dem Hindernis H_1 die Anzahl nahezu konstant. Das fahrende Hindernis H_3, welches die Kreuzung blockiert, erzeugt bei der ersten Erkennung ebenfalls einen starken Anstieg der Erfahrungen und Verbindungen, welche nach Ausführung des Verfahrens gemäß Kapitel 5.4 wieder sinken. Der erneute starke Anstieg nach kurzer Zeit ist in der Überquerung der Kreuzung durch das fahrende Hindernis H_3 begründet. Bei der zweiten Erkennung des fahrenden Hindernisses H_3 erfolgt nur noch ein geringer Anstieg.

Die implementierten Verfahren zeigen, dass eine Integration der Pfade, die durch Hindernisse ausgelöst werden, möglich ist. Die Erkennung und Integration von Hindernissen auf der linken als auch rechten Fahrbahnseite funktionieren stabil und auch die Integration läuft ohne Komplikationen. Bei einer freien Kreuzung wird der Pfad noch korrekt dargestellt. Die Verhinderung der Erstellung von Erfahrungen und Verbindungen funktioniert noch nicht einwandfrei. Gleiches gilt für die Integration des Pfades bei einer blockierten Kreuzung. Darüber hinaus ist in dem Test und der Auswertung aufgefallen, dass häufiger neue Pfade erzeugt werden, um Schwankungen der Sensorik auszugleichen.

8 Zusammenfassung

Das Ziel dieser Arbeit ist die Entwicklung und Implementierung einer Erkennung von Hindernissen und Kreuzungen gewesen, welche die durch RatSLAM erstellte Karte aufwerten sollte. Zwei Nebenziele der Erkennung für Hindernisse und Kreuzungen sind ein möglichst geringer Ressourcenverbrauch sowie die Betrachtung und Detektion auf beiden Fahrspuren. Bei der Aufwertung der Karte sollte durch die Integration von Pfaden sowohl der Speicherverbrauch als auch infolgedessen die Berechnungszeit sinken.

Die Erkennung von Hindernissen und Kreuzungen funktioniert gut. Allerdings existieren noch kritische Stellen. Das Fahrzeug muss nach der Erkennung einer Kreuzung mit 60 $\frac{cm}{s}$ an die Kreuzung heranfahren, damit die weiße Haltelinie nicht überfahren wird. Fährt das Fahrzeug schneller, ist die Wahrscheinlichkeit groß, dass die Haltelinie überfahren wird. Ist die Geschwindigkeit geringer, stoppt das Fahrzeug zu früh. Dies wird als Regelmissachtung geahndet. Darüber hinaus kommt es vor, dass Polaris an einer Kreuzung nicht weit genug die Strecke analysiert und somit kein Polynom approximiert wird. Dies hat zur Folge, dass die Spurführung nach dem Stopp an der Kreuzung nicht weiterfährt. Darüber hinaus springt der Point of Interest zur Detektion auf der linken Fahrspurseite noch recht häufig hin und her. An dieser Stelle muss noch Stabilität eingefügt werden.

Die Integration von Pfaden bei einem Hindernis auf der linken Fahrbahnseite sowie die Markierung von Hindernissen auf der rechten Fahrbahnseite in der Karte funktionieren sehr gut. Bei beiden Verfahren sind bereits einige Besonderheiten berücksichtigt. Die Verarbeitung einer Kreuzung bietet dagegen noch Aufgaben. Eine Aufgabe ist die Verhinderung oder die Beseitigung von Erfahrungen und Verbindungen, die auf der nahezu selben Position generiert werden. Darüber hinaus erzeugt das fahrende Hindernis bei Überquerung einer Kreuzung einen massiven Anstieg in der Anzahl der Erfahrungen und Verbindungen. Auch dies muss unterbunden werden. Insgesamt verkleinern die entwickelten Algorithmen die Anzahl der Erfahrungen und Verbindungen im Schnitt um 25 Prozent.

Die von RatSLAM erstellte Karte deckt sich ziemlich genau mit der Teststrecke. Es werden weder besonders teure Sensoren benötigt, noch besitzt die Karte eine solche Detailgetreue, dass diese nicht von der leistungsschwachen Hardware berechnet werden könnte. Zudem ist die Karte sehr einfach mit zusätzlichen Informationen erweiterbar. Insgesamt ist zu sagen, dass die entwickelten Verfahren einen guten Anfang darstellen. Sie sind allerdings noch ausbaufähig.

8.1 Ausblick

Es sollte an dieser Stelle nicht die Entwicklung von Algorithmen basierend auf RatSLAM oder der Erkennung von Hindernissen und Kreuzungen eingestellt werden. Bei den Verfahren für RatSLAM ist die Hauptaufgabe die bessere Integration von Kreuzungen. Dabei besteht die Aufgabe in der Integration einer blockierten Kreuzung. Zusätzlich sollte ein Verfahren entwickelt werden, welches die Schwankungen der Sensoren zwar nicht ausgleicht, aber eine Anpassung der Erfahrungen und Verbindungen vornimmt. Somit ist es an dieser Stelle abermals möglich, die Anzahl an Erfahrungen und Verbindungen zu senken. Es sollte auch an einem Verfahren gearbeitet werden, welches eine manuelle Fahrt beziehungsweise den RC-Modus erkennt und währenddessen die Anpassung der Karte verhindert.

Die Erkennung von Hindernissen und Kreuzungen besitzt ebenfalls noch Aufgaben. Bei der Anordnung der Infrarot Sensoren werden die äußersten Infrarot Sensoren so gut wie nie benutzt. Diese besitzen einen zu großen Winkel, als das diese in irgendeiner Situation etwas erkennen. Darüber hinaus kommt es auch vor, dass ein Hindernis erst im letzten Moment erkannt wird. Infolgedessen wird das Hindernis angefahren und vom Fahrzeug von der Fahrbahn geräumt. An dieser Stelle ist ein Algorithmus wünschenswert, der bei einer nahezu unvermeidbaren Kollision das Fahrzeug stoppt, ein kleines Stück zurücksetzt und dann das Hindernis umfährt. Darüber hinaus muss der Point of Interest vor allem auf der linken Fahrbahnseite stabilisiert werden. Der Point of Interest springt in bestimmten Situationen sehr heftig hin und her. Diese Situationen finden sich vor allem in sehr stark gekrümmten Kurven.

Literaturverzeichnis

[Ball u. a. 2010] Ball, D. ; Heath, S. ; Milford, M. ; Wyeth, G. ; Wiles, J.: A navigating rat animat / Proceedings of the 12th International Conference on the Synthesis and Simulation of Living Systems, Odense, Denmark. 2010. – Forschungsbericht

[Ball u. a. 2013] Ball, D. ; Heath, S. ; Wiles, J. ; Wyeth, G. ; Corke, P. ; Milford, M.: *OpenRatSLAM: an open source brain-based SLAM system*. Bd. 34. S. 149–176, Autonomous Robots, 2013

[Bertozzi und Broggi 1998] Bertozzi, Massimo ; Broggi, Alberto: *GOLD: A Parallel Real-Time Stereo Vision System for Generic Obstacle and Lane Detection*. Bd. 7. S. 61–81, Transactions on Image Processing (IEEE), 1998

[Carolo-Cup] Carolo-Cup: *Website*. – URL http://www.carolo-cup.de/

[Carolo-Cup-Regelwerk] Carolo-Cup-Regelwerk: *für 2014 / Version vom 03. Juni 2013*. – URL http://www.carolo-cup.de/fileadmin/user_upload/2014/regelwerk/Hauptwettbewerb2014.pdf

[Coulter 1992] Coulter, R. C.: Implementation of the Pure Pursuit Path Tracking Algorithm / Carnegie Mellon University, Pittsburgh. 1992. – Forschungsbericht

[Hafting u. a. 2005] Hafting, T. ; Fyhn, M. ; Molden, S. ; Moser, M.-B. ; Moser, E. I.: *Microstructure of a spatial map in the entorhinal cortex*. Bd. 11. S. 801–806, Nature, 2005

[Holz und Behnke 2009] Holz, D. ; Behnke, S.: Sancta simplicitas - on the efficiency and achievable results of SLAM using ICP-based incremental registration / Universität Bonn. 2009. – Forschungsbericht. http://www.ais.uni-bonn.de/ holz/spmicp/

[Jenning 2008] Jenning, Eike: *Systemidentifikation eines autonomen Fahrzeugs mit einer robusten, kamerabasierten Fahrspurerkennung in Echtzeit*, Masterthesis, 2008

[J.Milford u. a. 2011] J.Milford, M. ; Schill, F. ; Corke, P. ; Mahony, R. ; Wyeth, G. F.: Aerial SLAM with a single camera using visual expectation / International conference on robotics and automation. Shanghai, China. 2011. – Forschungsbericht

[J.Milford und Wyeth 2008] J.Milford, M. ; Wyeth, G. F.: Mapping a Suburb With a Single Camera Using a Biologically Inspired SLAM System / University of Queensland. 2008. – Forschungsbericht

[J.Milford und Wyeth 2010] J.Milford, M. ; Wyeth, G. F.: *Persistent navigation and mapping using a biologically inspired SLAM system*. Bd. 29. S. 1131–1153, International Journal of Robotics Research, 2010

[Knuth 1977] Knuth, Donald: A generalization of Dijkstra's algorithm / Information Processing Letters, Vol. 6. 1977. – Forschungsbericht

[Labayrade u. a. 2002] Labayrade, Raphael ; Aubert, Didier ; Tarel, Jean-Hilippe: *Real time obstacle detection in stereovision on non flat road geometry through v-disparity representation*. Bd. 2. S. 646–651, Intelligent Vehicle Symposium (IEEE), 2002

[Manske 2008] Manske, Nico: Kamerabasierte Praezisionsnavigation mobiler Systeme im Indoor-Bereich / Hochschule fuer Angewandte Wissenschaften (HAW) Hamburg, Hamburg. 2008. – Forschungsbericht

[Milford u. a. 2004] Milford, M. J. ; Wyeth, G. F. ; Prasser, D.: RatSLAM: A Hippocampal Model for Simultaneous Localization / University of Queensland. 2004. – Forschungsbericht

[Nikolov 2009] Nikolov, Ivo: *Verfahren zur Fahrbahnverfolgung eines autonomen Fahrzeugs mittels Pure Pursuit und Follow-the-carrot*, Bachelorthesis, 2009

[ROS] ROS: *Robot Operating System*. – URL http://www.ros.org/wiki/

[Samsonovich und McNaughton 1997] Samsonovich, A. ; McNaughton, B. L.: *Path integration and cognitive mapping in a continuous attractor neural network model*. Bd. 17. S. 5900 – 5920, Journal of Neuroscience, 1997

[Seki und Okutomi 2006] Seki, Akihito ; Okutomi, Masatoshi: *Robust Obstacle Detection in General Road Environment Based on Road Extraction and Pose Estimation*. S. 437–444, Intelligent Vehicles Symposium (IEEE), 2006

[Siciliano und Khatib 2008] Siciliano, Bruno ; Khatib, Oussama: *Springer Handbook of Robotics*. Springer, 2008

[x-io Technologies] Technologies x-io: *Website*. – URL http://www.x-io.co.uk/

[Yamazawa u. a. 1995] Yamazawa, Kazumasa ; Yagi, Yasushi ; Yachida, Masahiko: *Obstacle Detection with Omnidirectional Image Sensor (HyperOmni Vision)*. Bd. 1. S. 1062–1067, International Conference on Robotics and Automation (IEEE), 1995

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 29. Januar 2014 Torben Becker