RFID-basierte Navigation für autonome Fahrzeuge

Vom Promotionsausschuss der
Technischen Universität Hamburg-Harburg
zur Erlangung des akademischen Grades
Doktor-Ingenieur
genehmigte Dissertation

von
Arne Bosien

aus
Henstedt-Ulzburg

2012

Gutachter:
Prof. Dr. Volker Turau
Prof. Dr. Wolfgang Meyer

Vorsitzender des Prüfungsausschusses:
Prof. Dr. Hermann Rohling

Tag der mündlichen Prüfung:
23. September 2011

Inhaltsverzeichnis

1 Einleitung

In unserer heutigen Welt werden vielfach Systeme benötigt, die Gegenstände – vorzugsweise automatisch – von einem Ort zu einem anderen bewegen. Aus der Industrie sind solche Transportsysteme bekannt, die beispielsweise in der Autoproduktion erforderliche oder gefertigte Teile innerhalb des Betriebsgeländes autonom transportieren. Diese sogenannten Fahrerlosen Transportsysteme (FTS) werden im Allgemeinen dann eingesetzt, wenn Güter flexibel transportiert werden müssen, aber der Einsatz von Fahrern zu monoton, unwirtschaftlich oder gefährlich ist [Bau09]. Aber auch in anderen Umgebungen, wie in Krankenhäusern oder Pflegeheimen kann Bedarf für den einfachen Transport von Medikamenten oder anderen Dingen herrschen.

Eine der größten Herausforderungen für ein automatisches Transportsystem besteht in der Bestimmung der momentanen Position und der Ermittlung des gewünschten Weges. Während es dem Menschen sehr leicht fällt, Positionen oder Gegenstände zu identifizieren, sind automatisierte Systeme dafür auf Hilfsmittel angewiesen. Identifikationssysteme versuchen eine Brücke zu bauen, um physikalisch vorhandene Gegenstände auf virtuelle Objekte abzubilden, mit denen ein Informationssystem umgehen kann.

Die wohl am weitesten verbreitete Technik eines Identifikationssystemes ist der Barcode. Ein solches System unterteilt sich in zwei Komponenten: das Lesegerät und eine Vielzahl von Etiketten, die von dem Lesegerät erkannt werden können und einem Gegenstand eine ID zuordnen. Beispielsweise in Kassensystemen ermöglicht diese Technologie, den Preis zu einem Produkt abzurufen. Während Barcodes jedoch auf eine Sichtverbindung angewiesen sind, existiert mit Transpondern eine weitere Technologie, die ohne Sichtverbindung die Identifizierung von Objekten, die sich sogar im Pulk befinden können, ermöglicht.

Eine Ausprägung für ein Transpondersystem wird durch Radio Frequency Identification (RFID) beschrieben. Obwohl die technischen Grundlagen für RFID bereits Mitte des letzten Jahrhunderts gelegt wurden [Sto48, Lan05a], ist es dennoch eine sehr aktuelle Technologie, die zunehmend den Weg in die Wahrnehmung der Öffentlichkeit findet. Insbesondere durch die Miniaturisierung der elektronischen Etiketten (im Folgenden als Tags bezeichnet) können die Potenziale von RFID erst jetzt praktisch genutzt oder gar erforscht werden. Aufgrund der zunehmenden Verwendung von RFID sind die Preise für die Tags gefallen, so dass sich durch die Möglichkeiten zum intensiven Gebrauch neue Einsatzszenarien ergeben. RFID bietet sich als eine grundlegende Technologie für alle Anwendungen an, die auf der automatischen Identifikation von Alltagsgegenständen beruhen und deren Forschungsrichtung unter anderem durch den Begriff *Allgegenwärtiges Rechnen* (Ubiquitous Computing) beschrieben wird. Hier werden Szenarien untersucht, in denen mit Mikroelektronik ausgestattete Alltagsgegenstände viele Bereiche des Lebens beeinflussen und Computer ihre Aufgaben unsichtbar im Hintergrund ausführen.

In der Vision des Allgegenwärtigen Rechnens werden miniaturisierte Sensoren millionenfach in die Umwelt ausgebracht oder in physische Strukturen, wie Brücken, Straßen oder Wasserleitungen integriert. Angeschlossene Mikroprozessoren gewährleisten die Kommunikation über Funk [Mat03].

Die Logistik ist heutzutage eines der großen Einsatzgebiete für RFID. Dort wird es genutzt um Pakete, Container und sonstige Gegenstände aller Art zu identifizieren und automatisch handzuhaben [MTS08]. Eine aktuelle Entwicklung des Allgegenwärtigen Rechnens ist jedoch dahingehend, dass Transponder nicht nur genutzt werden, um mobile Objekte automatisch identifizierbar zu machen, sondern dass Tags stationär eingesetzt werden, um unbewegliche Gegenstände oder Positionen – ähnliche wie Landmarken – zu kennzeichnen. Somit kann RFID auch für Navigationszwecke eingesetzt werden und kommt in einem Bereich zum Einsatz, der für RFID bislang wenig erschlossen und erforscht ist. Nicht nur im industriellen Umfeld können transponderbasierte Navigationssysteme Verwendung finden. Im Zugverkehr werden ebenfalls Transponder benutzt, die in das Gleisbett eingelassen sind und vorbeifahrenden Zügen Streckeninformationen zur Verfügung stellen. Auch für die Fahrerunterstützung im normalen Straßenverkehr ist der Einsatz denkbar, wobei sicherlich nicht in Betracht gezogen werden kann, ein Fahrzeug automatisch zu steuern, sondern auch hier im Vordergrund steht, den Fahrer über seine Umgebung zu informieren.

Durch die weiterhin fortschreitende Entwicklung sind Tag-Generationen entstanden, die über neue Fähigkeiten verfügen. Während einfache Tags nur eine ID zurückliefern, ist es seit einiger Zeit zusätzlich möglich, Informationen auf den Tags abzulegen und diese zu einem späteren Zeitpunkt wieder abzurufen. Folgt man der Vision des Allgegenwärtigen Rechnens einer Welt, in der alle (oder zumindest sehr viele) Gegenstände automatisch identifizierbar sind, so entsteht eine Umgebung, die mit Informationen beschrieben werden kann. Diese Fähigkeiten können für verschiedene Anwendungen genutzt werden, unter anderem auch für die bereits angesprochenen Navigationssysteme. Hiermit ist es beispielsweise möglich, Informationen für nachfolgende Fahrzeuge zu hinterlassen, so dass ein Fahrzeug nicht nur in die Lage versetzt wird, seine Position zu bestimmen, sondern dass beispielsweise ein Pfad markiert wird, dem ein anderes Fahrzeug folgen kann. Ein derartiges Verfahren orientiert sich an dem Verhalten von Ameisen, die mit Pheromonen einen Weg kennzeichnen.

Diese neuen Einsatzmöglichkeiten führen allerdings auch zu neuen Herausforderungen. Es zeigt sich, dass die existierenden Algorithmen des Inventorys nicht für Szenarien geeignet sind, in denen sich das Lesegerät bewegt und die Tags einen festen Platz einnehmen. Das Inventory ist notwendig, um die in Reichweite des Lesegerätes befindlichen Tags zu erkennen und die Kommunikation mit ihnen aufzubauen. Es hat einen direkten Einfluss auf die Möglichkeiten der Navigationsverfahren.

Ziel dieser Arbeit ist die Entwicklung eines pheromonbasierten Ansatzes, bei dem die Schreibmöglichkeiten neuerer Tag-Generationen für Navigationsaufgaben genutzt werden. Ein wesentlicher Vorteil gegenüber bestehenden RFID-Systemen soll durch eine einfache Einrichtung der benötigten Umgebung geschaffen werden. Dies kann erreicht werden, wenn die Tags zwar einen Weg markieren, aber ihre Position nicht relevant ist.

Die Pfade können durch eine manuelle Steuerung festgelegt werden und sollen flexibel geändert oder gelöscht werden können. Andere Fahrzeuge sollen in die Lage versetzt werden, einem Pfad unmittelbar oder zu einem späteren Zeitpunkt zu folgen. Durch die Verwendung beschreibbarer RFID-Tags sollen zudem Voraussetzungen geschaffen werden, um zusätzliche Aufgaben zu bewältigen.

Zwei Bereiche sollen näher betrachtet werden. Die erste Aufgabe besteht im Entwurf von Methoden, die die eigentliche Navigation gewährleisten. Hierfür können zwei Modi unterschieden werden. Im Teach-Modus müssen relevante Informationen in die Umgebung geschrieben werden, die es im anschließenden Follow-Modus einem weiteren Fahrzeug ermöglichen, den Pfad zu reproduzieren. Zum zweiten ist für die schnelle Navigation eine schnelle Detektion der in Reichweite befindlichen Tags eine zwingende Voraussetzung. Die allgemein verbreiteten Algorithmen können zwar verwendet werden, jedoch führt dies nicht zu den besten Ergebnissen.

Die Arbeit ist wie folgt gegliedert. Im zweiten Kapitel wird eine Übersicht über Allgegenwärtiges Rechnen, RFID sowie Spurführung gegeben. Es werden verschiedene Anwendungsmöglichkeiten skizziert und insbesondere die bestehenden Navigationssysteme beleuchtet, die bereits Transponder benutzen.

Das dritte Kapitel gibt eine Einführung in die RFID-Grundlagen. Die Funktionsweise der Tag-Detektion wird erklärt und es findet sich eine Zusammenstellung der wichtigsten Algorithmen hierfür, sowie die Einordnung bezüglich ihrer Eignung für pheromonbasierte Navigation.

In Kapitel vier wird das angestrebte Navigationssystem vorgestellt, worin seine Vorteile begründet liegen und welchen Herausforderungen es begegnen muss. Des Weiteren werden Ansätze entwickelt, wie sich das Navigationssystem realisieren lässt.

Das fünfte Kapitel zeigt Verbesserungen für das Inventory auf, um es für Navigationsaufgaben zu beschleunigen. Es wird ein neuer Algorithmus vorgestellt, der insbesondere für diese Aufgaben geeignet ist.

Im sechsten Kapitel wird auf den Einfluss der Tag-ID-Verteilung eingegangen. Die besprochenen Algorithmen werden in verschiedenen Simulationen miteinander verglichen.

Am Ende dieser Arbeit steht die Zusammenfassung in Kapitel 7. Neben dem Fazit der ermittelten Ergebnisse wird dort ein Ausblick für die weitere Entwicklung gegeben.

2 Stand der Technik

2.1 Allgegenwärtiges Rechnen

Pervasive Computing (Rechnerdurchdringung), Ubiquitous Computing (Allgegenwärtiges Rechnen) oder auch Ambient Intelligence (Umgebungsintelligenz) sind Begriffe, die sich relativ schwer voneinander abgrenzen lassen und auch nur recht unscharf eine aktuelle Forschungsrichtung zur rechnergestützten Informationsverarbeitung beschreiben[1]. Es lässt sich jedoch formulieren, dass angestrebt wird, durch die Vernetzung intelligenter Alltagsgegenstände ein „Internet der Dinge“ zu schaffen [FM05]. Ziel ist es, die bestehende Lücke zu schließen, die sich dort findet, wo versucht wird physikalisch vorhandene Gegenstände auf die virtuelle Welt der Informationssysteme abzubilden. Dies soll die Entwicklung neuer und intelligenterer Anwendungen ermöglichen. Die Szenarien für die Nutzung sind jedoch sehr unterschiedlich.

In einem Bereich der Forschung geht es beim Allgegenwärtigen Rechnen vielfach darum, eine Kontextsensitivität zu schaffen, um darüber dem Menschen den Alltag zu erleichtern. Die Vision des intelligentes Hauses, in dem ein intelligenter Kühlschrank steht, der selbstständig bemerkt, wenn zum Beispiel die Milch zur Neige geht und dieses dem Bewohner mitteilt oder neue bestellt, die Verfallsdaten überwacht und anhand der vorhandenen Lebensmittel sogar Kochrezepte vorschlägt [Fac04], ist nur eine mögliche.

Von industrieller Seite aus gibt es klare wirtschaftliche Interessen, die darauf abzielen, Prozesse zu automatisieren oder zu vereinfachen. Dies kann beispielsweise im Supply-Chain-Management sein, wo über eine Form ständiger Inventur die Kosten für die Lagerhaltung verringert werden, weil aufgrund der genauen Kenntnis des Lagerbestandes eine geringere Reserve vonnöten ist und zur Neige gehende Materialien just-in-time nachbestellt werden können [CeB09, Wis07]. Auch um eine genaue Nachvollziehbar der Herkunft von Produkten erreichen zu können und um auf diese Weise eine bessere Fälschungssicherheit zu gewährleisten, kann das Allgegenwärtige Rechnen eingesetzt werden.

Mit RFID existiert eine Basistechnologie für ein kontaktloses ID-System, welches die meisten der Anforderungen des Allgegenwärtigen Rechnens an ID-Systeme erfüllt. RFID erlaubt zum einen, Gegenstände automatisch zu identifizieren, bietet aber auch die Möglichkeit, weitere Informationen abzulegen. Je nachdem, welche konkrete Technologie genutzt wird, verfügen RFID-Tags über weitergehende Funktionalitäten. Dies können beispielsweise auch kryptographische Fähigkeiten sein [Hes09].

[1] In dieser Arbeit wird durchgehend der Begriff „Allgegenwärtiges Rechnen“ benutzt und keine Abgrenzung der einzelnen Begriffe vorgenommen.

2.2 Einsatz von RFID

Obwohl die Technologie bereits seit mehreren Jahrzehnten entwickelt wird, sind RFID-Systeme vor allem in den letzten Jahren der Allgemeinheit bekannter geworden. Neben den bereits benannten Logistikprozessen gibt es weitere Anwendungsgebiete, wie bei Sicherungssystemen, der Zugangskontrolle und ähnlichem. Auch in offiziellen Ausweisdokumenten wird RFID inzwischen verwendet.

Die erfolgreichsten RFID-Anwendungen, die heute eingesetzt werden sind [RK09]:

ICAO e-Passports maschinenlesbare Reisedokumente (Ausweise, Pässe, Visa), die persönliche und biometrische Informationen enthalten.

Fahrkarten für den öffentlichen Nahverkehr, die beispielsweise ein Guthaben verwalten oder über eine zeitlich beschränkte Gültigkeit verfügen.

Supply-Chain-Management verzeichnet einen der größten Zuwächse bei der Verwendung von RFID, da es den präzisen Austausch von Informationen erlaubt, welche Produkte geliefert, verbraucht oder verkauft wurden. Mit dem EPC (Electronic Product Code) existiert ein Standard, um die Interoperabilität zwischen verschiedenen Unternehmen zu gewährleisten.

Wie beim Barcode besteht ein RFID-System aus einem Lesegerät (Reader) und einem Etikett (Tag). Während die Übertragung der Informationen beim Strichcode auf einem optischen Verfahren beruhte, wird dies nun über Funk realisiert. Dies hat zum Vorteil, dass keine Sichtverbindung erforderlich ist, um eine ID abzurufen und ermöglicht so auch eine Gruppe von Gegenständen gewissermaßen gleichzeitig zu erfassen[2].

Im Gegensatz zu vielen anderen elektrischen Systemen, benötigen RFID-Tags nicht notwendigerweise eine eigene Stromversorgung, wie beispielsweise ein Batterie. Passive Tags sind in der Lage, die notwendige Energie aus dem Feld des Readers zu gewinnen. Somit ähneln sie auch in dieser Hinsicht den Strichcodes und machen sie vielfältig einsetzbar.

Neben den von Barcodes bekannten Möglichkeiten bietet RFID weitere Potenziale. Neuere RFID-Generationen erlauben es beispielsweise zusätzlich zu der ID andere Informationen (bei einem ISO-14443-Tag bis zu mehreren Kilobytes) auf einem Tag abzulegen und davon abzurufen.

Auch wenn Tags heutzutage sehr klein geworden sind, flexibel gebogen, massenhaft eingesetzt und ähnlich wie Etiketten einfach auf Gegenstände aufgeklebt werden können (siehe auch Abbildung 2.1), so bestehen sie doch aus einem Mikrocontroller und einer Antenne, so dass die Kommunikationsverfahren erheblich komplexer sind. Des Weiteren können Probleme bei der Datensicherheit entstehen, da es weniger offensichtlich ist, wer

[2] Prinzipielle Nachteile der Nutzung von RFID, die sich auf Fragen wie Datensicherheit beziehen, werden in dieser Arbeit nicht betrachtet. Zielsetzung ist die Entwicklung eines System für den Einsatz in einer Umgebung, in der diese Fragestellung vernachlässigt werden kann. Im Allgemeinen, insbesondere wenn RFID-Tags zur Überwachung von Personen oder Tätigkeiten verwendet werden (siehe auch die folgenden Kapitel 2.3.3.2 und 2.3.3.6), sollte diesem Themengebiet mehr Aufmerksamkeit gewidmet werden. Zu Datenschutzaspekten der RFID-Technologie siehe auch [Lan05b].

Abbildung 2.1: Verschiedene 13,56 MHz-RFID-Tags

wann welche Daten abruft und ob dieser Zugriff autorisiert ist. Aufgrund der Charakteristik von Funk ist es weitaus komplizierter die Position eines Tags zu bestimmen. Je höher die Reichweite wird, desto mehr steigt auch der Schwierigkeitsgrad.

Sehr viele Anwendungen begreifen RFID als einfaches und überschaubares Produkt, das sie, wie früher den Barcode, einfach nur nutzen wollen. Wie [RK09] jedoch begründet einwendet, erfordert der erfolgreiche Einsatz von RFID eine genauere Betrachtung sowie eine detaillierte Kenntnis der Anforderungen. Die verschiedenen Merkmale, die ein RFID-System auszeichnen, beziehen sich unter anderem auf die Funkfrequenz, die Reichweite, die Art der Energieversorgung, Schreibkapazitäten und Speicherplatz. Es existiert ein sehr großer Variantenreichtum, der sich aus den zahlreichen Kombinationsmöglichkeiten dieser Merkmale ergibt. Eine große Anzahl von Standardisierungen definiert Systeme für unterschiedliche Einsatzszenarien und legt die Kommunikationsprotokolle fest, die für die Datenübertragung sowie die Detektion der Tags benötigt werden. (In Kapitel 3 wird ein detaillierter Einblick in die technischen Grundlagen der RFID-Technologie gegeben.)

Für das Allgegenwärtige Rechnen sind vor allem passive Tags von Interesse. Sie gewährleisten den massenhaften und wartungsfreien Einsatz, welcher für das Allgegenwärtige Rechnen notwendig ist.

Tags können in verschiedenen Baumformen realisiert werden. Das reicht von Klebeetiketten, über Chipkarten bis zu Tags, die Tieren injiziert werden. Die Eignung von Tags für transponderbasierte Navigation für FTS bezüglich der mechanischen und klimatischen Widerstandsfähigkeit wurde in [Bau09] untersucht.

2.3 Leitsysteme

Leitsysteme für autonome Fahrzeuge können im Groben in zwei Bereiche eingeteilt werden. Auf der einen Seite steht die Spurführung mittels einer kontinuierlichen Linie, bei der ein Fahrzeug einer hinterlegten Spur folgt, die beispielsweise visuell auf dem Boden

markiert ist. Die Spur bestimmt zu jedem Zeitpunkt die neue Richtung und der Pfadverlauf ist bei Antritt nicht bekannt. Das Fahrzeug ermittelt permanent seine relative Position zur Spur und korrigiert diese.

Auf der anderen Seite stehen Systeme, die Stützpunkte benutzen. Hier werden Referenzpunkte genutzt – wie sie beispielsweise entstehen können, wenn die Knotenpunkte eines virtuellen Gitters durch Punkte farblich auf den Boden gekennzeichnet sind – um einen vorher festgelegten und dem Fahrzeug bekannten Pfad abzufahren. Zwischen den Referenzpunkten navigiert das Fahrzeug über relative Methoden (Betrachtung der Fahrbewegung), um seine geschätzte Position beim Erreichen des nächsten Referenzpunktes wieder auf einen absoluten Wert festlegen zu können.

2.3.1 Spurführung

Traditionelle Navigationssysteme, wie sie in der industriellen Umgebung genutzt werden oder wurden, um Transporte zu automatisieren, können in die folgenden Bereiche unterteilt werden [Hol87, Bau09]:

Mechanische Spurführung ist eine Form der Führung, bei der der einzunehmende Pfad durch mechanische Einrichtungen vorgeben ist und vom Fahrzeug nicht geändert werden kann. Dies kann beispielsweise in Form von Gleisen geschehen. Sie ist heute für FTS allerdings nicht mehr von großer Relevanz, da ihre Installation aufwendig ist und Änderungen schwierig sind. Schienen sind u.U. für lange Strecken sinnvoll, bei denen selten Kreuzungen auftreten.

Induktive Spurführung gehört zu den bekanntesten Systemen, um Fahrzeuge zu steuern. Dazu wird im Boden ein Draht eingelassen, der Wechselstrom leitet. Die FTS können mittels Induktionssensoren das Feld detektieren und Positionsabweichungen quer zur Fahrbahn korrigieren. Die Stromversorgung des Fahrzeuges kann ebenfalls induktiv erfolgen. Auch Abzweigungen sind möglich, wenn die unterschiedlichen Bahnen durch verschiedene Frequenzen bei der Stromversorgung gekennzeichnet werden.

Optische/chemische/magnetische Spurführung benutzt eine passive Spur auf der Oberfläche des Bodens, die von Sensoren detektierbar ist. Um die Strecke zu ändern, sind somit keine großen baulichen Maßnahmen erforderlich.

Koppelnavigation ist ein sehr einfaches System, das ein Fahrzeug ab einem Referenzpunkt durch das Ausführen zuvor festgelegter Anweisungen steuert. Wird ein neuer Referenzpunkt erreicht, kann die Position korrigiert werden.

Inertiales Navigationssystem erfasst (über Trägheitssensoren) die Dreh- und Fahrtbewegungen des Fahrzeugs ohne auf Messungen der Umwelt zurückzugreifen, um so die aktuelle Position feststellen zu können. Referenzpunkte können genutzt werden, um die Position regelmäßig zu korrigieren. Im Gegensatz zur Koppelnavigation kann hier eine sehr viel höhere Genauigkeit erreicht werden, weil u.a. Schlupf berücksichtigt wird.

Die Linienbasierte Navigation ist unflexibel bei kurzfristigen Änderungen. Soll ein Pfad geändert oder ein neuer ermöglicht werden, muss die Spur neu eingerichtet und die alte entfernt werden. Während dies bei rein optischen Spuren mit weniger Aufwand durchführbar ist, ist es bei einem eingelassenen Draht ein komplexerer Vorgang. Auch Abzweigungen, Kreuzungen oder mehrere parallele Wege sind je nach verwandter Technologie mehr oder weniger aufwendig.

Die stützpunktbasierten Systeme sind flexibler, stellen aber auch höhere Anforderungen an die Navigation. Die Abweichung zur gewünschten Spur darf nicht zu groß werden, ansonsten wird möglicherweise die Position relativ zum falschen Stützpunkt korrigiert. Um diesen Fehler gering zu halten, kann die Odometrie oder ein inertiales Messsystem genutzt werden. Neben dem gewünschten Pfad muss das Fahrzeug auch Kenntnisse über die Umgebung besitzen, wie etwas über das Muster, nach dem die Stützpunkte vorhanden sind.

2.3.2 Navigationssysteme, freie Navigation

Im Gegensatz zu den vorherig beschriebenen Möglichkeiten, werden nun noch die infrastrukturlosen Systeme vorgestellt. Infrastrukturlose Systeme sind solche, bei denen der Betreiber keine weiteren Einrichtungen vorsehen muss.

GPS kann im Freien genutzt werden, um Fahrzeuge zu steuern und erfordert den Datenempfang von mindestens vier Satelliten. Da GPS empfindlich für Störungen ist, die durch bauliche Gegebenheiten hervorgerufen werden können, wenn die direkte Verbindungslinie von den Satelliten nicht vorhanden ist, ist es dort von höherem Interesse, wo diese Störungen nicht auftreten können. Dies kann zum Beispiel bei der Steuerung von Landmaschinen sein [Dem07]. Obwohl bei GPS immer ein Positionierungsfehler auftritt, ist dieser jedoch über kleine Zeiträume lokal konstant und kann so durch differentielle GPS-Lösungen gut reduziert werden.

Weitere Möglichkeiten für freie Navigation sind die Benutzung von Radar, Laserscanner, Kameras oder anderen Systemen, die die Umgebung erfassen und in einer bekannten Umgebung die Positionierung ermöglichen [Bau09, GKT10]. Diese Systeme sind jedoch technisch sehr aufwendig.

2.3.3 Navigationssysteme mit RFID/Transpondern

Meist werden RFID-Tags als Etiketten aufgefasst, die eingesetzt werden, um mobile Objekte zu identifizieren. Darüber hinaus können RFID-Tags aber auch stationär genutzt werden, um die Lokalisierung zu ermöglichen. Da jedes Tag eindeutig identifiziert werden kann und auch nur innerhalb einer bestimmten Reichweite detektierbar ist, kann mit Hilfe einer Karte, die die genauen Standorte aller Tags und ihrer IDs verzeichnet, ein System für die Positionsbestimmung (Lokalisierung) geschaffen werden [PH09, KNS+08].

2.3.3.1 Transpondernavigation in Container Terminals

Diese Technologie wurde u.a. als *Free Ranging on Grid* (FROG) bei den *Europe Combined Terminals* (ECT) in Rotterdam eingeführt [Tex00] und wird ebenso beim *Container Terminal Altenwerder* (CTA) in Hamburg genutzt (Abbildung 2.2). Zu diesem Zweck wurden über 12.000 Transponder in den Boden des CTAs eingelassen [Met06]. Die Transponder werden zur zentimetergenauen Lokalisierung genutzt und ermöglichen die vollkommen autonome Steuerung von Fahrzeugen, die parallel auf dem gleichen Gelände arbeiten und sich sogar überholen können. Für die Kommunikation wurde ein separates System installiert.

Abbildung 2.2: Fahrerlose Transportsysteme im Container Terminal Altenwerder (Foto: Marcus Venzke)

2.3.3.2 RFID-Tags in Teppichen

Ein vergleichbarer Ansatz für dem Einsatz im Innenbereich besteht darin, RFID-Tags direkt in den Teppich zu integrieren [KKPS08]. Kommerzielle Varianten wurden von dem Staubsauger- und Teppichhersteller Vorwerk und dem Sensorsystemhersteller Future-Shape vorgestellt und werden auch vertrieben [Vor06, Fut08].

Hintergrund dieser Idee ist die Vorstellung Reinigungsgeräte autonom agieren zu lassen. Die Tags können unter anderem dazu genutzt werden, um zu dokumentieren, ob oder wann die Reinigung stattgefunden hat, sowie besondere Anweisungen zu Art und Umfang der Reinigung enthalten. Für die Navigation wird wie im vorherigen Beispiel eine

zentrale Karte verwendet. Die Roboter orientieren sich anhand der Identifikationsnummer der Tags. Außerdem wird von Vorwerk als weitere Möglichkeit in Betracht gezogen, Personen nachzuverfolgen. Dies ist unter Umständen im Altenheim von Vorteil, um Bewohner wiederzufinden, die ihre Orientierung verloren haben. Bei der Entwicklung von Future-Shape nimmt der automatisierte Transport von Gegenständen (beispielsweise Medikamente oder Wäsche) eine wichtige Position ein, wie er in Krankenhäusern und Pflegeheimen anfallen kann.

2.3.3.3 Spur in Tags hinterlassen, Virtuelle Tags

Forscher der ETH Zürich haben einen ähnlichen Ansatz entwickelt [BM04]. Sie schlagen den hoch redundanten Einsatz von Tags vor, bei dem sehr viele Transponder über große Areale oder auf den Oberflächen von Gegenständen verteilt werden. Sie ziehen es in Betracht, eine virtuelle Spur auf einem RFID-Fußboden zu hinterlassen. Dabei werden vor allem *virtuelle Tags* betrachtet. Ein virtuelles Tag wurde eingeführt, um das Problem des beschränkten Speicherplatzes eines Tags oder dass es unter Umständen nicht schreibbar ist, zu lösen. Eine Datenbank im Hintergrund wird genutzt, um die ID eines realen Tags auf ein virtuelles Tags abzubilden. Wie auch bei den vorherigen Ansätzen, wird hier eine verlässliche Infrastruktur im Hintergrund vorausgesetzt. Es werden allerdings keine konkreten Angaben gemacht, wie ein solches System aussehen kann. Auch wird keine Strategie beschrieben, wie ein Fahrzeug einem solchen Pfad folgen kann, ohne dass es den Großteil der Zeit für die Suche nach dem Pfad verbraucht.

2.3.3.4 Positionen in der Spur speichern

Einen anderen, aber sehr viel konkreteren Ansatz, verfolgt die Universität Hannover in dem Projekt *TagDrive* [NBOF06, OBNF06]. Wie beim CTA wird hier der Untergrund mit RFID-Tags ausgestattet. Diese Tags sind in diesem Fall jedoch wiederbeschreibbar und nehmen als Information ihre absolute Position auf. Somit entfällt die Notwendigkeit eine Karte mit den Standorten der Tags zu pflegen, da sich jedes Tag seiner Position gewahr ist. Neben der eigenen Position ist vorgesehen, dass auf einem Tag auch die absolute Position des folgenden Tags abgelegt werden kann. Auf diese Art und Weise kann ein Pfad auf den Tags gespeichert werden.

Die einzelnen Tags liegen soweit auseinander, dass sich zu jedem Zeitpunkt nur ein Tag in Reichweite des Lesegerätes befindet. Dies erlaubt die einfache Detektion eines Tags und dessen Lokalisierung, die mit Hilfe eines Antennenarrays vorgenommen wird. Zwischen den Tags wird mittels Odometrie navigiert, so dass der Tag-Abstand sehr groß werden kann.

Es wird in Betracht gezogen, die Tags zu nutzen, um weitere Informationen zu speichern. Zu den pfadrelevanten Daten können die erlaubte Geschwindigkeit, Steigungen, das maximal zulässige Gewicht für einen Abschnitt oder Vorfahrtsregeln gehören. Auch könnten Lagerplätze markiert oder eine Strecke als gesperrt gekennzeichnet werden.

2.3.3.5 Navigation für Blinde

In [Pec08] wird eine Möglichkeit vorgestellt, um Blinden mittels RFID-Tags die Orientierung in unbekanntem Gebiet zu erleichtern. Dazu wird in den Blindenstock ein RFID-Reader integriert. Die Umgebung wird mit Tags versehen, die über ihre ID eine feste Position bezeichnen. Der RFID-Reader ist mit einem PDA verbunden, auf dem sich ein Verzeichnis der Tags und ihrer Positionen befindet. Über einen Kopfhörer werden die gewonnenen Positionsinformationen an die blinde Person weitergegeben.

Der Test eines solchen Ansatzes wurde in Laveno Mombello in Italien durchgeführt. 1260 Tags wurden dazu in Gehwege eingebettet.

Die Idee, Tag-IDs zu benutzen, um Personen die Orientierung an unbekannten Orten zu erleichtern, ist selbstverständlich nicht auf Blinde beschränkt. Auch Nichtbehinderte können von dieser Art von *Location Based Services* profitieren, wenn sie beispielsweise vor einer Sehenswürdigkeit stehen und mit Hilfe eines RFID-Tags und einem PDAs alle wichtigen Daten aus einer Datenbank abrufen können. Die Unterstüztung beim Auffinden von Waren in einem Supermarkt ist ein anderes Beispiel für ein solches System [RGD+09].

2.3.3.6 Pheromonbasierter Ansatz

Eine etwas andere Herangehensweise verfolgen Ansätze, die sich an den pheromonbasierten Verhaltensmustern von Ameisen orientieren. Wenn eine einzelne Ameise eine Futterquelle erkundet hat, dann markiert sie den Pfad von der Futterquelle zurück zum Ameisenhaufen mit einer Spur aus Duftstoffen. Andere Ameisen sind nun in der Lage, dieser Spur nachzugehen und so ebenfalls zum Futter zu gelangen. So lange die Futterquelle nicht versiegt ist, wird die Spur von den passierenden Ameisen permanent erneuert. Sie verschwindet, sobald die Spur nicht mehr genutzt wird.

Diskretisiert man diese Spur, so kann man sich vorstellen, einen Pheromon-Pfad mit beschreibbaren RFID-Tags nachzubilden. Zambonelli und Mamei entwickelten ein System, das sich an dieser Idee orientiert und bei dem eine Spur in einer Umgebung von RFID-Tags hinterlassen wird [MZ07]. Im Gegensatz zu den vorherig vorgestellten Ansätzen wird hier nicht versucht eine absolute bzw. überhaupt eine Form von Positionierung zu erreichen. Stattdessen steht hier im Vordergrund ein Kontextbewusstsein für Anwendungen zu schaffen. In dem konkreten Fall ging es darum eine Möglichkeit aufzutun, vergessene Gegenstände wiederaufzufinden. Dazu wurde eine Person mit einem RFID-Reader ausgestattet. Der Reader erkennt die umliegend Tags, die in Positionstags und Objekttags unterschieden werden können. Wenn sich die Person bewegt und einen Gegenstand bei sich trägt, wird die Identifikation des Gegenstandes in die umliegenden Positionstags geschrieben. Auf diese Weise wird der komplette Pfad eines Objektes gespeichert. Zu einem späteren Zeitpunkt kann man feststellen, welchen Orte man mit dem Objekt passiert hat. Ein automatisches Folgen wurde nur im Simulator realisiert, bei dem in die Positionstags zusätzlich noch die Bewegungsrichtung des Objektes abgelegt wurde.

2.3.4 Einordnung geeigneter Systeme für FTS

Im Folgenden werden von den zuvor vorgestellten Navigationsverfahren, die wichtigsten schematisch gegenübergestellt. Der Ansatz der ETH Zürich und die Navigation für Blinde werden hier nicht berücksichtigt. Ersterer beschreibt kein Prinzip, wie die Navigation erfolgen kann und beim zweiten steht die Anwendung für Menschen im Vordergrund.

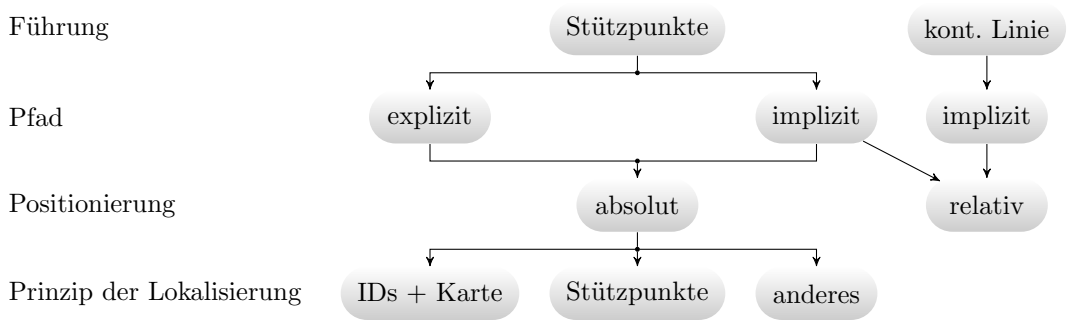


Abbildung 2.3: Merkmale von Navigationssystemen

Abbildung 2.3 zeigt, wie die unterschiedlichen Merkmale, die ein Navigationssystem aufweisen kann, zusammenhängen. Bei dem Führungsprinzip kann in Führung mittels Stützpunkten und Führung mit einer kontinuierlichen Linien unterschieden werden. Im ersten Fall kann der Pfad dem Fahrzeug vor Antritt der Fahrt explizit gegeben sein, wie das bei den meisten stützpunktbasierten Ansätzen der Fall ist. Weisen die Stützpunkte den Weg selbst, ist der Pfad implizit durch Auswertung der Fahrzeugposition und die Richtung des Pfades gegeben. Bei der Führung mit einer kontinuierlichen Linie ist der Pfad immer durch die Linie implizit bekannt. Die Positionierung kann in der Folge dann auch nur relativ zum Pfad geschehen.

Die absolute Positionierung bedeutet, dass die Positionierung bezüglich eines Stützpunktes und somit eindeutiger Koordinaten innerhalb des Aktionsradius vorgenommen wird. Geben die Stützpunkte den Pfad implizit vor, kann die Positionierung wie bei den pheromonbasierten Verfahren relativ erfolgen oder wie beim Beispiel TagDrive absolut.

Das Prinzip der Lokalisierung kann auf den Tag-IDs und einer zugehörigen Karte beruhen, die Stützpunkte können ihre absolute Position kennen oder es werden über GPS globale Koordinaten erlangt oder ähnliches.

Tabelle 2.1 fasst die Merkmale der vorgestellten Navigationsprinzipien zusammen. Zu den Navigationssystemen mit RFID zählen CTA, TagDrive und pheroromonbasierte Verfahren, welche alle RFID-Tags als Stützpunkte benutzen. Beim CTA existiert ein explizit vorgegebener Weg, der außerhalb des Navigationssystems bestimmt wurde und den das Fahrzeug einnehmen soll. Anhand einer Karte wird die absolute Position des Vehikels ermittelt. Auch bei TagDrive ist die Positionierung absolut. Allerdings wird keine Karte verwendet, sondern die Tags selbst enthalten ihre Position. Der Pfad kann dadurch vorgegeben werden, dass die nächste anzusteuernde Position in den Tags gespeichert wird (implizit), oder der Pfad wurde extern vorgegeben. Die pheromonbasierten

	CTA	TagDrive	Pheromon	induktiv/ optisch	Koppel-Navi.	inertiale Führung	freie Navi.
Führung	SP	SP	SP	kont. Linie	SP	SP	SP
Pfad	explizit	expl./impl.	implizit	implizit	explizit	explizit	explizit
Positio-nierung	absolut	absolut	relativ	relativ	absolut	absolut	absolut
Prinzip	Karte	SP			Karte und SP	Beschleuni-gungssensoren	z.B. GPS

Tabelle 2.1: Merkmale der Navigationssysteme für FTS, SP = Stützpunkt

Systeme bestimmen ihre Position relativ zum Pfad, und die Informationen über die Route liegen im Pfad selbst.

Systeme, die ein Fahrzeug induktiv oder optisch mittels einer kontinuierlichen Linie navigieren, bestimmen wie pheromonbasierte Systeme ihre Position relativ zum markierten Pfad und folgen diesem, ohne dass dieser explizit bestimmt wird. Die Koppelnavigation nutzt Stützpunkte, anhand derer sie sich mit Hilfe einer Karte absolut positionieren kann. Die einzunehmende Route muss extern vorgegeben sein. Inertiale Navigationssysteme verwenden Beschleunigungssensoren, um ihre absolute Position zu ermitteln. Mittels Stützpunkten wird die Position kalibriert, der Pfad wird extern vorgegeben. Freie Navigation z.B. über GPS benutzt Stützpunkte, bei der GPS-Module jede Sekunde die eindeutige globale Position liefern. Auch hier muss die Route wieder extern bestimmt werden.

2.4 Vorteile von Transpondernavigation

Die transponderbasierten Navigationsverfahren verfügen über eine Reihe von Vorteilen. Zum einen verfügen sie über die gleichen Fähigkeiten wie herkömmliche Verfahren, die Stützpunkte benutzen (Koppelnavigation, inertiale Führung, freie Navigation). Des Weiteren erlauben sie es aber auch, an den Stützpunkten dynamisch Informationen abzulegen, die eine Anwendung zusätzlich zur Navigationsaufgabe benötigt.

Einen weiteren Vorteil der Transpondersysteme kann man darin sehen, dass sie prinzipiell miteinander kompatibel sind. Die gleichen Tags, die genutzt werden, um mittels ihrer ID und einer Referenzkarte die momentane Position zu bestimmen (Beispiel CTA), können zusätzlich Daten enthalten, die die Position des nächsten Zieles festlegen (Beispiel TagDrive).

2.5 RFID im Hochgeschwindigkeitsbereich

Es existieren Ansätze und Anwendungen, bei denen RFID für den Einsatz bei hoher Geschwindigkeit (des Lesegerätes oder der Tags) vorgesehen wird. In dem Projekt TagDrive (vergleiche Abschnitt 2.3.3.4) wurde experimentell ermittelt, unter welcher Maximialgeschwindigkeit es einem bewegten Lesegerät noch möglich ist, ein stationäres Tag zu detektieren. Dabei konnte für 13,56 MHz-Tags eine Geschwindigkeit von 42,9 m/s erreicht werden [Bau09]. Die maximale Lesereichweite lag in dem Test bei 436 mm und es wurde kein Inventory verwendet, da sich nur ein Tag in Reichweite befand [OBNF06].

In [LYP+09] wurden die Geschwindigkeiten untersucht, bei denen ein mit einem RFID-Lesegerät ausgestattetes Fahrzeug stationäre Tags erkennen kann. Die Reichweite des Lesegerätes betrug bis zu 5 m. Es zeigte sich, dass Tags bei dieser Reichweite bis zu einer Geschwindigkeit von 100 km/h erkannt werden können. Eine Lokalisierung der Tags wurde nicht betrachtet.

Im Zugverkehr werden RFID-Systeme genutzt, um die Position eines Zuges zu bestimmen. Dabei werden sowohl mobile als auch stationäre Lesegeräte eingesetzt. Im ersten Fall ist das Lesegerät am Zug montiert und dient dazu, dem Zug seine Position und Informationen über die Strecke mitzuteilen [Fin06]. Es ist aber auch möglich, die Züge mit Tags zu versehen und mittels einem stationären Lesegerät zu ermittelt, welcher Zug sich auf welchem Streckenabschnitt befindet [ZLTT10]. Da die Züge hohe Geschwindigkeiten aufweisen können, ist ein schnelles Lesen der Daten notwendig. Mit entsprechend hohen Lesereichweiten, der Verwendung aktiver (also batteriegestützter) Transponder und hohen Frequenzen können Geschwindigkeiten über 200 km/h erreicht werden [AC09, ZT11]. Des Weiteren verzichten derartige Systeme meist auf ein Inventory, da sich zu jedem Zeitpunkt nur ein einzelner Transponder in Reichweite befindet.

2.6 Verwandte RFID-Projekte

Vielfach werden RFID-Tags herangezogen, wenn es um Aufgaben der Orientierung und Navigation geht. Die Ziele sind jedoch teilweise sehr verschieden.

Verschiedene Ansätze verwenden RFID in Kombination mit anderen Sensoren [RABB07, DNRK10]. RFID-Tags werden hier als verlässliche Referenzpunkte eingesetzt, um auftretende Messfehler anderer Systeme ausgleichen zu können. In den meisten dieser Ansätze steht die Lokalisierung im Vordergrund, während die Navigation nachrangig ist.

Ebenfalls kann das Ziel sein, bei einer bekannten Position des Lesegeräts, die Positionen von Tags zu berechnen. In [HBF+04] wird eine Karte mit den Positionen von RFID-Tags erstellt. Dazu wird ein mobiler Roboter genutzt, der sich mittels Laserscanner orientiert und die Position der Tags ermittelt. Ein ähnliches Szenario ist bei *Activities of daily living*-Anwendungen gegeben. Sie haben das Ziel Menschen zu helfen, die beispielsweise in ihrer Bewegungsfreiheit eingeschränkt sind [PFP+04]. Hier soll sich ein Roboter in einer Umgebung aus markierten Gegenständen bewegen, diese identifizieren und mit diesen interagieren.

In [MQZ06] wird die Verwendung von Tags vorgeschlagen, um die Synchronisation zwischen verschiedenen Agenten zu ermöglichen. Die Agenten können dabei sowohl Roboter als auch Menschen sein. Als Beispiel für den praktischen Nutzen wurde der Einsatz von Rettungsteams nach einem Erdbeben angegeben, bei dem die verschiedenen Teams Bereiche als medizinisch versorgt kennzeichnen können.

3 Grundlagen der RFID-Technologie

Im Bereich von RFID existieren sehr viele Standards und Spezifikationen, die die physikalischen Eigenschaften der Komponenten, sowie Kommunikationsprotokolle und alle weiteren wichtigen Parameter festlegen. Gemein ist allen Ausprägungen, dass ein RFID-System aus zwei wesentlichen Komponenten aufgebaut ist. Auf der einen Seite steht die Schreib-/Leseeinheit (auch als Reader oder Lesegerät bezeichnet) und auf der anderen Seite stehen die Tags. Die Tags verhalten sich passiv, d.h. sie initiieren keine Kommunikation, sondern antworten nur auf Anfragen des Lesegeräts. Des Weiteren kommunizieren die Tags nicht untereinander, sondern immer direkt mit dem Lesegerät.

Der wesentliche Unterschied von RFID zu herkömmlicher Funktechnologie besteht in dem Prinzip der Informationsvermittlung. Bei klassischen Funkgeräten wird von jedem Kommunikationsteilnehmer zur Informationsübertragung ein elektromagnetisches Feld ausgesendet, welches von allen empfangen wird. Bei RFID dagegen baut nur das Lesegerät ein magnetisches oder elektromagnetisches Feld auf, das sowohl vom Lesegerät selbst, als auch von den RFID-Tags zur Datenübertragung genutzt wird [Fin06].

Es sind verschiedenen Arten der Kopplung möglich, dazu gehören die kapazitive Kopplung, die induktive Kopplung und das Backscatter-Verfahren. Eine weite Verbreitung hat die induktive Kopplung erfahren, die neben der Informationsübertragung auch die Energieversorgung der Transponder gewährleisten kann und somit den Bau von Tags ermöglicht, die als passive Tags bezeichnet werden.

Innerhalb des Lesebereiches eines Readers können sich mehrere Tags befinden. Damit ein Reader gezielt mit einem Tag kommunizieren kann, muss ihm dessen ID bekannt sein. Ist diese nicht bekannt, ist es bestenfalls möglich, mit allen Tags in Reichweite gleichzeitig zu kommunizieren. In diesem Fall kann nicht zwischen den Tags unterschieden werden und es ist nicht möglich deren Antworten zu verstehen, da die gleichzeitigen Antworten verschiedener Tags zu Kollisionen führen. Eine Möglichkeit, die vorhandenen Tags in Reichweite zu identifizieren, ist daher eine wichtige Fähigkeit, die ein Reader besitzen muss. Die Detektion der IDs wird als Inventory bezeichnet. Sind sehr viele Tags in Reichweite, kann die Durchführung eines Inventorys viel Zeit in Anspruch in nehmen. Ein schnelles Inventory ist somit für die meisten Anwendungen essentiell.

Entgegen diesen grundlegenden Gemeinsamkeiten steht die technische Realisierung. Abbildung 3.1 gibt einen Überblick der prinzipiellen Merkmale, in denen sich ein RFID-System unterscheiden kann und wie sie miteinander kombiniert werden können.

Bei der **Betriebsart** kann zwischen Sequentiellen Systemen (SEQ) und Full- (FDX) oder Half-Duplex-Systemen (HDX) unterschieden werden. Bei den sequentiellen Systemen finden die Energieübertragung und die Datenübertragung zu verschiedenen Zeitpunkten statt. Bei den Duplex-Systemen geschieht dies gleichzeitig. Mittels des Merk-

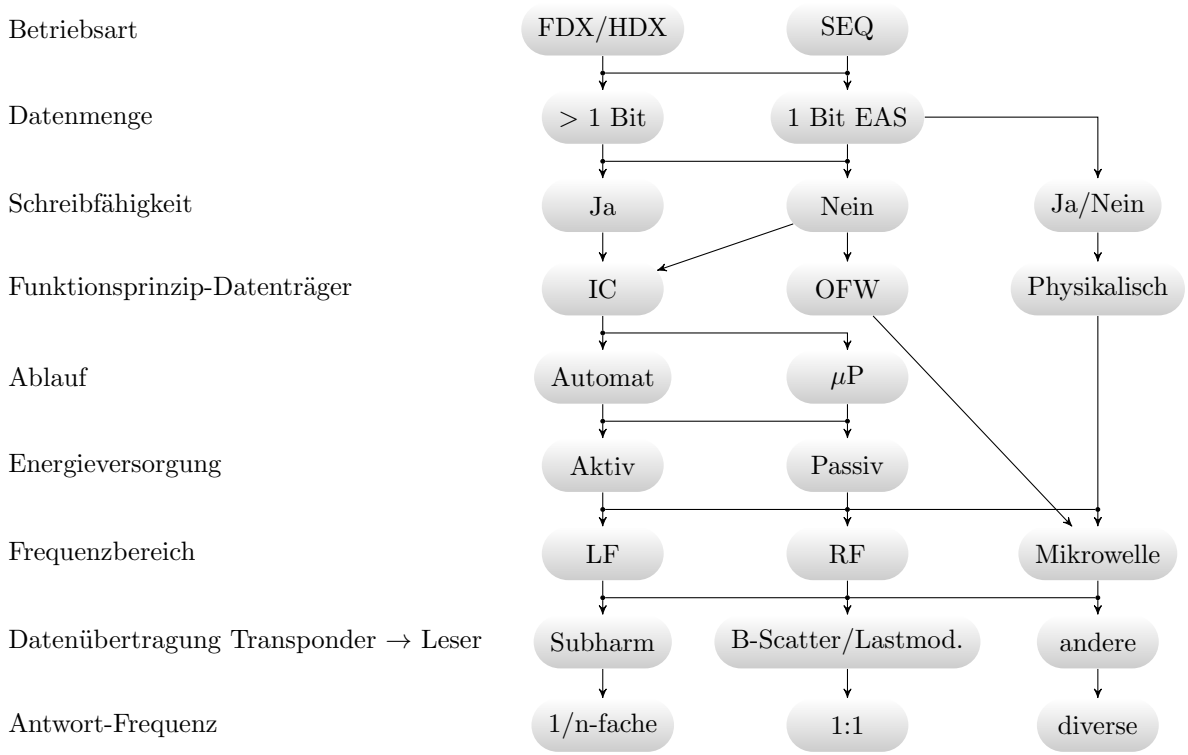


Abbildung 3.1: Verschiedene Unterscheidungsmerkmale von RFID-Systemen [Fin06]

mals **Datenmenge** kann man Tags voneinander unterscheiden, die tatsächliche Nutzdaten aufnehmen können und solche, die nur die Information übertragen, dass sie sich im Readerfeld befinden. Unter **Schreibfähigkeit** wird verstanden, ob die Tags auch nach der Produktion noch mit neuen Daten beschrieben werden können. Wie diese Daten gespeichert werden, wird über das **Funktionsprinzip der Datenträger** festgelegt. Der **Ablauf** unterscheidet, wie die Logik für den Schreib- und Lesezugriff umgesetzt wird. Die **Energieversorgung** eines Transponders kann über das elektromagnetische Feld oder mittels einer Stützbatterie erfolgen. Das elektromagnetische Feld kann in unterschiedlichen **Frequenzbereichen** aufgebaut werden. Dabei sind verschiedene Verfahren der **Datenübertragung** möglich, welche in unterschiedlichen **Antwort-Frequenzen** resultieren.

In den folgenden Abschnitten werden die wichtigsten Merkmale besprochen werden. Für weitergehende Informationen sei auf [Fin06] und [Lah06] verwiesen.

3.1 Funktionsprinzip

3.1.1 Datenübertragung vom Transponder zum Lesegerät

Je nach Entfernung zwischen Lesegerät und Tag findet die Energie- und Informationsübertragung im Nah- oder Fernfeld statt. Wo der Übergang stattfindet, hängt von der

verwendeten Frequenz ab, welche somit in Abhängigkeit von der Entfernung die mögliche Kopplung bestimmt.

Im Nahfeld ist der Mechanismus der Energieübertragung hauptsächlich induktiv. Die Kopplung geschieht über das gemeinsame Magnetfeld von Lesegerät und Tag (siehe Abbildung 3.2). Durch An- und Ausschalten eines Lastwiderstandes ist der Transponder in der Lage, die Impedanz seiner Antenne zu verändern. Dies führt in der Folge zu einer Spannungsänderung an der Antenne des Lesegerätes, die das Lesegerät detektieren kann.

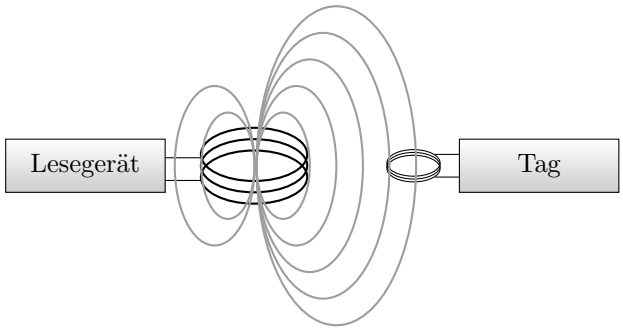


Abbildung 3.2: Induktive Kopplung im Nahfeld

Beim Fernfeld hat dagegen die elektromagnetische Energie die größere Bedeutung (siehe Abbildung 3.3). Ein Teil der bei der Transponderantenne ankommenden Leistung wird wie beim RADAR reflektiert und zurückgesendet. Für die Kopplung mittels Backscatter-Verfahren wird hier jedoch die Belastung der Empfangsantenne gezielt variiert, was in einer Beeinflussung der Reflexionseigenschaften der Antenne resultiert. Auf diese Weise kann die reflektierte Leistung in ihrer Amplitude moduliert werden. Das Signal kann dann vom Lesegerät empfangen und ausgewertet werden kann.

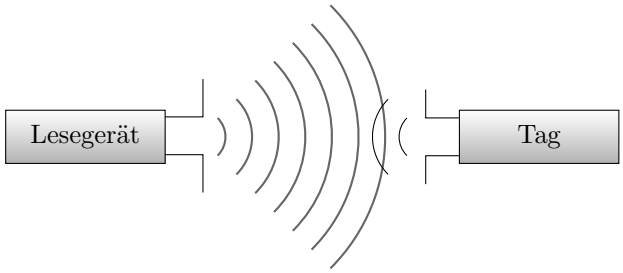


Abbildung 3.3: Elektromagnetische Kopplung im Fernfeld

Das Backscatter-Verfahren ist insbesondere für hohe Frequenzen geeignet, da die kurzen Wellenlängen erst die Konstruktion kleiner Antennen mit gutem Wirkungsgrad ermöglichen [Fin06].

Neben der induktiven und der elektromagnetischen Kopplung existiert die kapazitive Kopplung. Hier erfolgt die Energieübertragung durch parallele, in geringem Abstand zueinander angeordnete Metallflächen. Da es hierfür meist erforderlich ist, das Tag zur genauen Positionierung in ein Lesegerät zu stecken, hat die kapazitive Kopplung kaum

Vorteile gegenüber herkömmlichen kontaktierten Chipkarten und wird daher selten verwendet [Fin06].

3.1.2 Datenübertragung vom Lesegerät zum Transponder

Für die Datenübertragung vom Lesegerät zum Transponder werden die folgenden grundsätzlichen Verfahren verwendet [Fin06]:

- ASK: Amplitude Shift Keying
- FSK: Frequenz Shift Keying
- PSK: Phase Shift Keying

Sie sind unabhängig von der Arbeitsfrequenz oder dem Kopplungsverfahren einsetzbar. Aufgrund der einfachen Demodulationsmöglichkeiten wird jedoch ASK mehrheitlich angewendet.

3.2 Klassifikation von RFID-Systemen

Zu Beginn dieses Kapitels wurde bereits ein Überblick der verschiedenen technischen Merkmale gegeben. Zu den wichtigen Auswahlkriterien beim RFID-Einsatz zählen jedoch der Frequenzbereich, die Reichweite und die Art der Energieversorgung (und diesbezüglich die Kopplung) der Transponder [Bun04]. Hinzu kommen optionale Schreibfähigkeiten und die Kapazität des Speichers. Bezüglich dieser Eigenschaften werden RFID-Systeme nun klassifiziert.

3.2.1 Reichweite, Frequenz und Kopplung

Die Reichweite, innerhalb derer die Kommunikation zwischen Reader und Tags möglich ist, stellt eine der offensichtlichsten Größen bei der Wahl eines RFID-Systems dar. Je nach Anwendung können unterschiedliche Reichweiten von Interesse sein. Es können die folgenden Bereiche unterschieden werden [Fin06]:

Close-Coupling-Systeme sind Systeme mit sehr kleiner Reichweite, die typischerweise bei bis zu einem Zentimeter liegt. Zur Kommunikation mit den Transpondern müssen diese entweder in ein Lesegerät gesteckt oder auf einer vorgesehenen Fläche platziert werden. Die Kopplung kann sowohl induktiv als auch kapazitiv erfolgen.

Remote-Coupling-Systeme besitzen Reichweiten bis etwa einen Meter. Diese stellen die Mehrzahl der heute verwendeten Systeme dar, was in einer nahezu unüberschaubaren Anzahl verschiedener Ausprägungen resultiert. Für verschiedene Standardanwendungen wurden unterschiedliche Normen verabschiedet. Die Mehrheit der Systeme nutzt die induktive Kopplung.

Long-Range-Systeme erreichen Kommunikationsdistanzen von deutlich über einem Meter. Mit passiven (batterielosen) Transpondern können Reichweiten von ca. 3 m erreicht werden, mit aktiven (batteriegestützten) Transpondern sind dagegen Reichweiten von bis zu 15 m möglich. Für die Datenübertragung wird mehrheitlich das auf der RADAR-Technik basierende Backscatter-Verfahren genutzt.

Für Navigationssysteme sind vor allem Remote-Coupling-Systeme von Bedeutung. Je größer die Reichweite wird, desto schwieriger wird es auch, die Position von Tags zu bestimmen und desto mehr Tags müssen – eine gleichbleibende Tag-Dichte vorausgesetzt – beim Inventory erkannt werden.

Eng mit der möglichen Reichweite steht die Betriebsfrequenz in Verbindung. Bei Close-Coupling-Systemen sind Frequenzen bis zu 30 MHz möglich. Für Remote-Coupling-Systeme werden Frequenzen unter 135 kHz verwendet. Des Weiteren ist 13,56 MHz verbreitet. Hierfür sind die beiden folgenden Standards von besonderer Bedeutung, auf die im weiteren Verlauf noch genauer eingegangen wird:

- Proximity-Coupling-Cards (ISO-14443)
- Vincinity-Coupling-Cards (ISO-15693)

Long-range-Systeme nutzen Frequenzen im UHF- oder Mikrowellenbereich. Tabelle 3.1 gibt eine Übersicht der Frequenzen und einiger ihrer charakteristischen Eigenschaften.

Mit steigender Frequenz nimmt die Geschwindigkeit zu, in der Daten übertragen werden und auch der mögliche Leseabstand steigt. Dies wird unter anderem dadurch erreicht, dass eine Stützbatterie verwendet wird, um dem IC des Transponders beim Einsatz von Backscatter genügend Energie zur Verfügung zu stellen. Diese kann bei der elektromagnetischen Kopplung in den meisten Fällen nicht aus dem Feld des Lesegerätes gewonnen werden.

Systeme mit niedrigeren Frequenzen bis in den Megahertzbereich zeichnen sich durch ihre Unempfindlichkeit gegenüber Feuchtigkeit aus, da die Absorptionsrate von Wasser in diesem Frequenzbereich gering ist. Bei 1 GHz ist die Absorptionsrate verglichen mit 100 KHz bereits rund 100.000fach größer, so dass bei Ultrahochfrequenzen oder Mikrowelle ein negativer Einfluss von Feuchtigkeit zu verzeichnen ist. Generell weisen niederfrequente Systeme ein besseres Verhalten bei Fragen der Durchdringung von Gegenständen auf [Fin06, Lah06].

Werden RFID-Tags in einer Umgebung benutzt, in der viel Metall vorhanden ist (beispielsweise auch dann, wenn Tags auf Dinge wie Konserven aufgeklebt werden), wird die Kommunikation mit den Lesegerät bei den niedrigeren Frequenzen unter anderem durch Multipath gestört. Die Kommunikation mittels Ultrahochfrequenzen oder Mikrowelle ist hingegen wenig anfällig für solche Störungen.

3.2.2 Energieversorgung

Ein weiteres wichtiges Merkmal von Transpondern ist die Art ihrer Energieversorgung. Sie ermöglicht die Unterscheidung in aktive und passive Transponder [Fin06].

	Niedrigfrequenz	Hochfrequenz	Ultrahochfrequenz	Mikrowelle
Frequenz	125 - 134 kHz	13,56 MHz	868 oder 915 MHz	2,45 oder 5,8 GHz
Leseabstand	bis 1,2 m	bis 1,2 m	bis 4 m	bis 15 m
Lesegeschwindigkeit	langsam	je nach ISO-Standard	schnell	sehr schnell (aktive Transponder)
Feuchtigkeit	kein Einfluss	kein Einfluss	negativer Einfluss	negativer Einfluss
Metall	negativer Einfluss	negativer Einfluss	kein Einfluss	kein Einfluss

Tabelle 3.1: Kenngrößen von RFID-Technologien [Bun04]

Aktive Transponder verfügen über eine separate Stromversorgung, wie beispielsweise eine Batterie.

Passive Transponder beziehen ihre gesamte benötigte Energie aus dem Feld, das der Reader aufgebaut hat. Somit können solche Transponder wartungsfrei betrieben werden.

Unabhängig davon, ob eine zusätzliche Energiequelle für die Versorgung der Schaltung benutzt wird, erfolgt die Informationsübertragung immer über das Feld, das vom Lesegerät aufgebaut wurde. Aufgrund der großen Anzahl von Transpondern die beim Allgegenwärtigen Rechnen [RK09] und den angestrebten Navigationsverfahren benötigt werden, sind die passiven Ausführungen hier von besonderem Interesse.

3.3 Tag-Zustände

Ein RFID-Tag kann verschiedene Zustände einnehmen (Abbildung 3.4). Im Folgenden werden die einzelnen Zustände für ein ISO-15693-Tag behandelt.

Gelangt ein Tag in die Reichweite eines Lesegerätes, wechselt der Zustand von *Power Off* in den initialen Status *Ready*. Ein Wechsel in einen anderen Status muss durch einen Befehl des Readers veranlasst werden. Im Zustand *Quiet* reagiert das Tag nicht mehr auf allgemeine/unadressierte Anfragen, bis das Lesegerät das Tag in einen anderen Zustand versetzt oder es das Reader-Feld verlässt. Wird es dagegen in *Selected* versetzt, werden Anfragen beantwortet, die ohne Tag-ID gesendet wurden. Wie aus der Abbildung erkennbar ist, kann aus jedem aktiven Zustand ein beliebig anderer angenommen werden. Des Weiteren kann ein Tag in den Zuständen *Ready*, *Quiet* und *Selected* verweilen. Verlässt ein Tag den Bereich des Lesegerätes, wird es das nächste Mal erneut anfangs in den Zustand *Ready* wechseln.

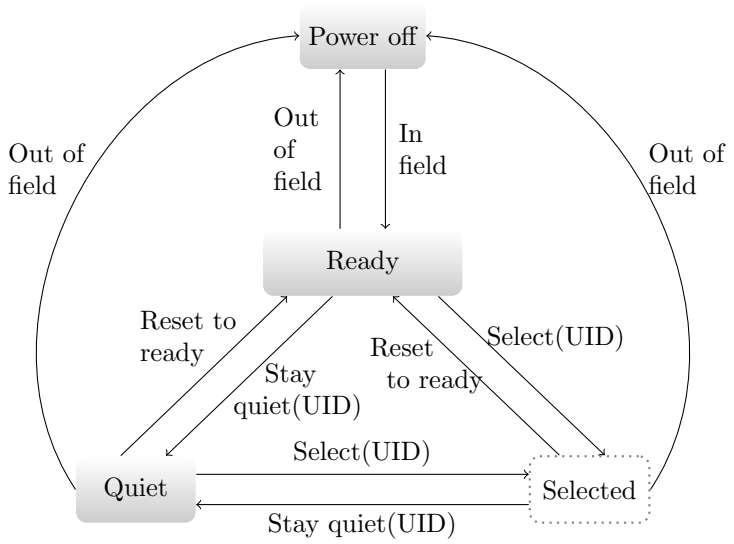


Abbildung 3.4: Mögliche Zustände eines ISO-15693-Tags [ISO00]

3.4 Tag-Identifikation

Jedes Tag verfügt über eine Tag-ID, die es eindeutig identifiziert. Die ID kann darüber hinaus aber auch über andere Merkmale des Tags Auskunft geben. Eine ISO-15693-Tag-ID besteht beispielsweise aus 64 Bit. Die ersten acht Bit sind vorgegeben, die folgenden acht Bit sind für die ID des Chip-Herstellers vorgesehen und die übrigen 48 Bit können von diesem frei für eine Seriennummer genutzt werden [ISO00].

Mit dem dem Electronic Product Code (EPC) existiert ein internationaler Standard [Fin06], um ein Objekt innerhalb einer Lieferkette weltweit eindeutig zu kennzeichnen. Anstatt alle zur Verfügung stehenden Bits für eine fortlaufende Seriennummer zu verwenden, enthält die ID Informationen über Herkunft und Eigenschaften des Objektes.

Unabhängig von der ID kann ein Tag über einen optionalen Application Family Identifier (AFI) verfügen, welcher über dessen Verwendungszweck informiert. Zu den möglichen Verwendungszwecken gehören Transport, Financial, Identification, Telecommunication, Medical, Airline bags, Postal services, Express parcels, Item management, Data storage, Gaming und Multimedia.

3.5 Inventory

Das Inventory ist nötig, um die erreichbaren Tags zu ermitteln. Sendet der Reader eine Nachricht, die alle Tags auffordert zu antworten und ihre ID zu übertragen, kommt es zu Kollisionen, die verhindern, dass der Reader die Antworten (also die IDs) verstehen kann. Es ist ein Antikollisionsalgorithmus nötig, der diese Kollisionen vermeidet oder auswertet, um die Anfrage des Readers anzupassen und eine kollisionsfreie Übertragung der ID zu gewährleisten.

Anzahl Transponder	durchschnittlich	99%ige Sicherheit	99,9%ige Sicherheit
2	150 ms	350 ms	500 ms
3	250 ms	550 ms	800 ms
4	300 ms	750 ms	1,00 s
5	400 ms	900 ms	1,25 s
6	500 ms	1,2 s	1,60 s
7	650 ms	1,5 s	2,00 s
8	800 ms	1,8 s	2,70 s

Tabelle 3.2: Zeitbedarf von ALOHA zum Auslesen von Transpondern eines Beispielsystems [Fin06]

Im RFID-Bereich werden zwei unterschiedliche Techniken verwendet. Diese können in zustandsbehaftete und zustandslose Algorithmen aufgeteilt werden. Zustandsbehaftete Algorithmen erfordern, dass die beteiligten Tags während des Vorgangs Informationen speichern. Die zustandslosen Algorithmen stellen dagegen geringere Anforderungen an die Tags.

3.5.1 ALOHA (zustandslos)

ALOHA-basierte Antikollisions-Protokolle sind sehr einfach strukturiert und werden meist nur für Read-Only-Transponder eingesetzt. Sie erlauben den Tags, ihre Daten (bzw. ihre ID) jederzeit unsynchronisiert zu senden [Fin06]. Die Kollisionsvermeidung basiert auf dem Prinzip der Stochastik.

Der Ablauf ist wie folgt. Sobald der Antikollisionsvorgang startet, fangen die Tags unkoordiniert an, ihre ID zu senden. Nach einer Ruhezeit, die bei jedem Tag leicht unterschiedlich ist, wiederholen sie ihre Übertragung und legen erneut eine Ruhezeit ein. Somit besteht eine gewisse Wahrscheinlichkeit, dass die Daten nicht zum selben Zeitpunkt gesendet werden und nicht kollidieren. Mit jeder Wiederholung steigt die Wahrscheinlichkeit, dass alle Tags erkannt werden. ALOHA bietet somit ein einfaches Verfahren für die Detektion von Tags, welches aber in einer schlechten Ausnutzung des Kanals resultiert. Typischerweise bleiben mehr als 80% der Kapazität ungenutzt [Fin06]. Tabelle 3.2 zeigt den durchschnittlichen Zeitbedarf für ein Beispielsystem mit verschiedenen Tranponderzahlen. Wie man deutlich erkennt, steigt der Zeitbedarf erheblich, möchte man Tags mit einer höheren Wahrscheinlichkeit detektieren.

Slottet-ALOHA verbessert dieses Verfahren, indem es eine bestimmte Anzahl von Zeitabschnitten (Zeitschlitzen) einführt. Die Tags übertragen ihre ID nun nicht mehr zu einem willkürlichen Zeitpunkt, sondern entscheiden sich zufällig für einen Zeitschlitz, in dem sie antworten. Mit dieser Methode kann der erreichbare Durchsatz auf ca. 37% erhöht werden [Fin06]. Zu Beginn des Antikollisionsverfahrens sendet das Lesegerät einen Inventory-Befehl, der zu einem synchronen Start des Verfahrens bei allen Tags führt. Die Anzahl der Slots sowie Dauer der Zeitschlitze ist festgelegt. Nach dem Empfang des

Inventory-Befehls wählen die Tags einen zufälligen Zeitschlitz aus und übertragen ihre ID. Erkennt das Lesegerät Kollisionen, sendet es eine erneute Anfrage für einen synchronen Start und die Tags entscheiden sich wieder für einen zufälligen Zeitschlitz [Hua06]. Optional können erkannte Tags durch Verwendung eines weiteres Befehls von der Teilnahme am restlichen Antikollisionsprozess ausgeschlossen werden, um die Anzahl der auftretenden Kollisionen weiter zu verringern [Fin06].

Die Anzahl der Runden R bis alle Tags mit einer statistischen Sicherheit von α erkannt werden, ist gegeben durch

$$R \geq \frac{log(1-\alpha)}{log(1-\frac{Np_1}{n})},$$

wobei N die Anzahl der Slots, n die Anzahl der Tags und $p_1 = (1-\frac{1}{N})^{n-1}$ die Wahrscheinlichkeit einer erfolgreichen Übertragung angibt [KCR10].

Dynamisches Slotted-ALOHA ist eine nochmalige Erweiterung. Im Gegensatz zum vorherigen Ansatz ist hier die Anzahl der Zeitschlitze nicht unveränderlich, sondern wird an die Anzahl der beteiligten Tags angepasst [LL06, Fin06]. Die Inventory-Anfrage enthält nun als Parameter die Anzahl der möglichen Zeitschlitze, die ein Tag verwenden kann. Treten viele Kollisionen auf, kann das Lesegerät die Zeitschlitze-Zahl erhöhen. Auf diese Weise wird erreicht, dass bei sehr vielen Tags die Anzahl der zu Verfügung stehenden Zeitschlitze ausreicht, aber auch nicht von vornherein mit einer zu großen Anzahl gearbeitet wird, was wiederum in einer schlechten Kanalausnutzung resultieren würde. Es kann gemäß [KCR10] ein Durchsatz von 42,6 % erreicht werden.

3.5.2 Binärbaum-Suche (zustandsbehaftet)

Ein Binärbaum-Antikollisionsalgorithmus erkennt die Tags durch die Traversierung des binären Baums der möglichen Tags-IDs [BAQZY06]. Zu diesem Zweck überträgt das Lesegerät eine Folge von Bits und startet bei jedem Durchlauf von der Baumwurzel. Nach jedem übertragenen Bit vergleichen die Tags, ob das korrespondierende Bit ihrer eigenen ID übereinstimmt. Die Tags benutzen dafür einen Zeiger, der die aktuelle Bitposition in der ID markiert. Stimmen die Bits überein, wird der Zeiger inkrementiert und das dortige Bit gesendet [CJK05]. Sind die Bits dagegen unterschiedlich, nimmt das Tag in dem aktuellen Durchlauf nicht weiter teil. Das Lesegerät empfängt demzufolge immer den Wert für das folgende Bit. Tritt hier eine Kollision auf, muss sich das Lesegerät für eine Möglichkeit entscheiden und gleichzeitig die Position speichern, um in einem weiteren Durchlauf den anderen Zweig abzulaufen. Tritt hingegen keine Kollision auf, sendet das Lesegerät das empfangene Bit. Ein Durchlauf ist dann beendet, wenn das letzte ID-Bit empfangen und somit ein Tag identifiziert wurde. In den nächsten Durchläufen muss das Lesegerät dann die verbliebenen und gespeicherten Kollision auf die gleiche Weise auflösen. Bei jedem Neustart setzten die Tags ihren Zeiger zurück.

In Tabelle 3.3 ist ein Beispiel für eine solche Suche für den Fall von vier Tags und einer jeweiligen ID-Länge von vier Bits angegeben. Zu Beginn sendet das Lesegerät eine `0`. Dieser Wert stimmt mit den niederwertigsten Bits (die Bits welche am weitesten rechts stehen) von Tag 1 und Tag 2 überein, daher antworten beide jeweils mit ihrem nächsten

Schritt	Anfrage	Tag 1 1000	Tag 2 0100	Tag 3 0101	Tag 4 1101	Ergebnis
1	0	0	0			weiter mit 0
2	0	0	1			Kollision, weiter mit 0
3	0	1				Tag 1 erkannt, Neustart
4	0	0	0			weiter mit 0
5	0	0	1			Kollision, diesmal weiter mit 1
6	1		0			Tag 2 erkannt, Neustart
7	1			0	0	weiter mit 0
8	0			1	1	weiter mit 1
9	1			0	1	Kollision im letzten Bit, Tag 3 und 4 erkannt

Tabelle 3.3: Binärbaumsuche

Bit. Diese beiden Bits sind identisch und stellen damit die Anfrage dar, die im nächsten Schritt gesendet wird.

Die Antworten im zweiten Schritt stimmen nun nicht mehr überein, so dass eine Kollision die Folge ist. Der Algorithmus entscheidet im nächsten Schritt mit 0 fortzufahren. Dieser Wert trifft nur noch für das erste Tag zu, das mit dem letzten Bit (1) antwortet und somit identifiziert ist. Die ID setzt sich somit aus der Anfrage 000 und dem zuletzt empfangenen Bit 1 zusammen.

Der Antikollisionsdurchlauf beginnt von vorne und startet erneut mit der Anfrage 0. Der fünfte Schritt entspricht dem zweiten Schritt, jedoch wird nun im sechsten Schritt auf eine 1 geprüft, um diesen Zweig zu überprüfen. Im sechsten Schritt antworten nun nur noch das zweite Tag, welches damit ebenfalls identifiziert ist.

Im letzten Durchlauf startet der Algorithmus nun mit einer 1. Die nächsten beiden Antworten stimmen überein und erst beim letzten Bit tritt eine Kollision auf. Dies ermöglicht es, die Tags 3 und 4 gleichzeitig zu identifizieren.

Um nicht nach jedem identifizierten Tag den Baum erneut von der Wurzel abzuschreiten, ist es möglich, dass sich die Tags merken, an welcher Bitposition sie bei dem vorherigen Durchlauf ausgestiegen sind. Anstatt nach einem Neustart erneut mit dem ersten Bit zu beginnen, informiert das Lesegerät die Tags, an welcher Position das Antikollsions-verfahren fortgeführt wird [GH07].

3.5.3 Query-Tree (zustandslos)

Bei Anwendung eines Query-Tree-Algorithmus [MLSS07] sendet der Reader eine Inventory-Anfrage, die einen Teil der Tag-ID (die sogenannte Maske) beinhaltet. Dies veranlasst alle Tags mit korrespondierender ID mit ihrer vollständigen ID zu antworten. Da dies zu

Kollisionen führen kann, wird die Maske im nächsten Schritt erweitert, so dass diese nun zu weniger Tags passt. Diese Prozedur wird so lange wiederholt, bis nur noch ein Tag antwortet und somit identifiziert werden kann. Gemäß [LLS00] kann als obere Grenze für die Komplexität der Kommunikation $2{,}21l \log(n) + 4{,}19l$ angenommen werden, wobei n die Anzahl der Tags und l die Länge der Tag-ID angibt.

In jedem Schritt liegt die komplette Antikollisions-Logik auf der Seite des Readers. Die Tags müssen nur den empfangenen ID-Teil mit ihrer ID vergleichen und entscheiden, ob sie eine Antwort senden oder nicht. Der Vorteil von Query-Tree-Algorithmen liegt daher in der Einfachheit auf Seiten der Tags und der Möglichkeit ein besseres Inventory dadurch zu entwickeln, dass nur der zentrale Algorithmus des Lesegeräts geändert wird, ohne dass das Verhalten der Tags angepasst werden muss.

Eine ausführliche Beschreibung des Query-Tree-Verfahrens wird in Abschnitt 3.7 vorgenommen.

3.6 Wichtige Standards: ISO-14443 und ISO-15693

Eine Vielzahl der existierenden Standards und Spezifizierungen wurden zum Teil von Organisationen wie der ISO festgelegt, zum anderen haben aber auch die großen Hersteller von RFID-Hardware eigene Entwürfe entwickelt, die ebenfalls als Industriestandard gelten können. Zwei der weitverbreitesten Spezifikationen sind die ISO-Standards 14443 und 15693. Beide arbeiten mit 13,56 MHz und bieten sich für RFID-Navigationssysteme an [BNA+07], da sie einen Kompromiss aus Kommunikationsgeschwindigkeit und Widerstandsfähigkeit gegenüber Einflüsse durch Feuchtigkeit und Metall bilden (siehe auch Tabelle 3.1).

ISO-14443 (Proximity-Coupling-Cards) wurde für Identifikationssysteme und Zugangskontrollen entworfen, bei denen eine Reichweite von bis zu 15 cm erreicht werden soll [Fin06]. Verwendung findet er beispielsweise bei den ICAO-ePässen, wie dem deutsche Reisepass oder dem deutschen elektronischen Personalausweis.

Es wurden zwei verschiedene Tag-Typen mit unterschiedlichen Eigenschaften spezifiziert, die aber beide von einem Lesegerät angesprochen werden können müssen. Die Tag-Arten unterscheiden sich sowohl im Datenübertragungsverfahren als auch im Inventory-Prinzip. Während Typ-A-Tags die Binärbaumsuche benutzen, wird bei Typ-B-Tags Slotted-ALOHA verwendet. Es sind Tags mit einer Speicherkapazität von bis zu 8 kByte erhältlich. Die Lesegeschwindigkeit liegt zwischen 106 kbit/s und 847 kbits/s [ISO99].

ISO-15693 (Vicinity-Coupling-Cards) wurde für RFID-Tags entworfen, die aus einer größeren Entfernung als Proximity-Coupling-Cards gelesen werden sollen. Die Reichweite kann je nach Lesegerät bis über 1 m betragen. Die Tags können ebenfalls für die Zugangskontrolle benutzt werden, jedoch werden sie im Allgemeinen häufig dort eingesetzt, wo kleine kostengünstige Datenträger benötigt werden.

Die Tags können je nach Hersteller auch hier bis zu 8 kByte Daten speichern. Dabei werden die Daten in Blöcke aufgeteilt. Möglich sind bis zu 256 Blöcke mit einer jeweiligen Größe bis zu 32 Byte (256 Bit). Die Lesegeschwindigkeit beträgt in Abhängigkeit von

der Entfernung und je nach Parameter bis zu 26,48 kbits/s. Als Antikollisionsverfahren kommt Query-Tree zum Einsatz.

3.7 Antikollisionsalgorithmen mit Query-Tree

Im Folgenden werden Algorithmen vorgestellt, die die Aufgabe der Antikollision mittels Query-Tree-Verfahren lösen. Aufgrund der beschriebenen Vorteile bei der Entwicklung neuer Algorithmen, wird das Query-Tree-Verfahren genauer betrachtet. Jeder Ansatz wird detailliert beschrieben und auf seine Eignung bezüglich der Navigation mittels RFID untersucht.

3.7.1 Basistechnologie

Als Grundlage für die Untersuchung von Antikollisionsalgorithmen werden die Kommunikationsprinzipien genutzt, die durch die ISO-15693 festgeschrieben wurden. Dazu gehört zum einen das Query-Tree-Verfahren selbst, welches auf einer synchronen Übertragung der Daten von den Tags zum Lesegerät und der Manchester-Kodierung beruht. Des Weiteren werden die identifizierenden Masken betrachtet, die lokal begrenzt der Identifizierung von Tags dienen.

3.7.1.1 Datenübertragung/Manchester-Kodierung

Für die Übertragung der Daten vom Tag zum Reader wird die Manchester-Kodierung verwendet. Dies bedeutet, dass ein Bit in zwei Signale geteilt wird (Abbildung 3.5). Eine fallende Flanke drückt eine logische 0 aus und eine steigende Flanke eine logische 1. Unterscheiden sich zwei simultan übertragende Signale, enthält der empfangene Datenstrom ungültige Abschnitte, welche ausgewertet werden können. Der Vorteil gegenüber anderen Verfahren liegt darin, dass bei einer synchronen Übertragung der Daten die Position der Kollision bitgenau erkannt werden kann.

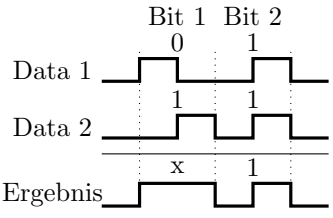


Abbildung 3.5: Kollisionserkennung mit Manchester-Kodierung, der ungültige Teil der resultierenden Antwort wurde mit einem `x` gekennzeichnet

3.7.1.2 Anwendung beim Inventory

Eine einzelne Kollision kann von einer beliebigen Menge von Tags verursacht werden und es ist nicht möglich, anhand der Anzahl der Kollisionen die genaue Anzahl der Tags zu bestimmen (Tabelle 3.4). Dennoch gibt dieser Wert Aufschluss über die obere Grenze beteiligter Tags. Da jede Kollision von mindestens zwei Tags verursacht wird und sich zwei IDs in mindestens einem Bit unterscheiden müssen, damit sie eindeutig sind, können c Kollisionen maximal von 2^c Tags verursacht worden sein. Die vier Kollisionen in Tabelle 3.4 wurden zwar nur durch zwei Tags verursacht, jedoch können sich hinter vier Kollisionen maximal $2^4 = 16$ Tags verbergen.

Tag 1	`1001`				
Tag 2	`0101`	Tag 1	`1001`	Tag 2	`0101`
Tag 3	`1101`	Tag 2	`0101`	Tag 4	`1010`
Ergebnis	`xx01`	Ergebnis	`xx01`	Ergebnis	`xxxx`

Tabelle 3.4: Kollisionen verschiedener Tag-Zusammenstellungen

Manche RFID-Reader geben nur die Position des ersten Kollisionsbits zurück, obwohl sie auch die restlichen Kollisionen erkennen. Für einige der folgenden Algorithmen ist es erforderlich, dass alle Kollisionsbits bekannt sind.

3.7.1.3 Identifizierende Masken

Bei der Query-Tree-Methode ermittelt der Algorithmus diejenigen Masken, die ein einzelnes Tag eindeutig innerhalb der Reichweite des Readers identifizieren. Im Allgemeinen wird durch die Übertragung einer Maske während des Inventorys eine Menge von Tags angesprochen. Passt die Maske nur für ein einzelnes Tag, kann ein Tag mit dieser Maske (anstatt mit der vollständigen ID) identifiziert werden. Eine solche Maske wird Identifizierende Maske genannt. Die IDs mehrerer Tags lassen sich als binärer Baum darstellen. Jeder Knoten entspricht einem Bit der ID. Die identifizierende Maske stellt dann den Pfad zu einem Baumabschnitt dar, der keine Abzweigungen mehr enthält. Abbildung 3.6 zeigt ein Beispiel für vier Tag-IDs (`0111`, `1011`, `1001` und `1000`) mit ihren entsprechenden identifizierenden Knoten[1]. Die Masken `111`, `011`, `01` und `0` identifizieren die angegebenen Tags in dem Fall, dass genau diese vier Tags in Reichweite sind. Ändert sich die Anzahl der Tags oder die ID eines Tags, müssen auch die identifizierenden Masken neu ermittelt werden.

Um ein Tag anhand seiner identifizierenden Maske zu identifizieren, muss der Bereich gewählt werden, in dem die identifizierende Maske gültig ist. Dieser Bereich hängt von der Reichweite des Lesegerätes ab.

[1] Die identifizierenden Knoten entsprechen den identifizierenden Masken und markieren deren höchstwertiges Bit im ID-Baum.

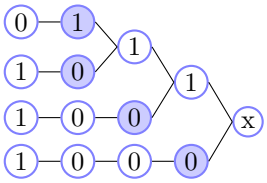


Abbildung 3.6: Unbalancierter Baum von vier Tag-IDs; die identifizierenden Knoten wurden hervorgehoben

Tag	ID (Hex)	Binärwert
1	BA6	1010 1010 0110
2	A45	1010 0100 0101
3	F6B	1111 0110 1011
4	AAA	1011 1010 1010

Tabelle 3.5: IDs der verwandten Tags

3.7.2 Rekursiver Standard-Algorithmus (Algorithmus 1: stdRec)

ISO-15693 spezifiziert nicht, wie der Antikollisionsalgorithmus verfahren muss. Dennoch beinhaltet der Standard einen einfachen rekursiven Algorithmus (Algorithmus 1) als Beispiel dafür, wie für ein komplettes Inventory durchgeführt werden kann [ISO00]. Der Algorithmus implementiert eine Tiefensuche [Sed02], um alle erreichbaren Tags zu erkennen. Er startet mit einer leeren Maske, die für alle Tags gültig ist. Wenn es zu einer Antwort kommt, die Kollisionen beinhaltet, ruft sich der Algorithmus rekursiv auf. Einmal geschieht der Aufruf mit einer Erweiterung der Maske um `0` und einmal um `1`. Somit werden beide möglichen Unterzweige des ID-Baums separat behandelt. Trat hingegen keine Kollision auf, hat entweder kein Tag oder genau eines geantwortet. Im letzteren Fall kann das Tag identifiziert werden.

Der angegebene Algorithmus wird nun anhand eines Beispiels von vier Tags erläutert. Tabelle 3.5 gibt die involvierten Tags für einen kompletten Durchlauf an, der in Tabelle 3.6 dargestellt ist. Abbildung 3.7 zeigt die letzten Bits der Tag-IDs in Baumform. In einem ersten Schritt wird eine leere Maske gesendet, auf die alle Tags mit einer Antwort reagieren. Als nächstes wird die Maske um `0` erweitert, welche nun zu Tag 1 und zu Tag 4 passt. Die erneute Erweiterung zu `00` führt zu keiner Antwort, da die ID keines der Tags mit dieser Bitfolge endet. Daher wird dieser Zweig nun verlassen und im vierten Schritt wird die Maske zu `10` geändert, die erneut zu Tag 1 und Tag 4 passt. Die Übertragung von `010` führt schließlich zu der Identifizierung eines ersten Tags. Als nächstes liefert `110` ein weiteres Tag. Nun ist der `0`-Zweig abgeschritten und Schritt 7 startet mit `1`. Die verbleibenden Schritte entsprechen den vorherig beschriebenen und detektieren die übrig geblieben Tags 2 und 3. Nach neun Abfragen werden so die vier Tags erkannt.

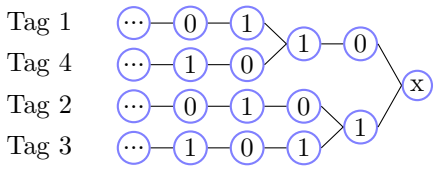


Abbildung 3.7: Struktur der letzten vier Bits der Tag-IDs

Schritt	Maske	Antwort	Bedeutung
1	ø	`1x1x xxx0 xxxx`	Kollisionen, Erweiterung der Maske
2	`0`	`101x 1010 xx10`	Kollisionen, Erweiterung der Maske
3	`00`	ø	keine Antwort
4	`10`	`101x 1010 xx10`	Kollisionen, Erweiterung der Maske
5	`010`	`1011 1010 1010`	**Tag 4 erkannt**
6	`110`	`1010 1010 0110`	**Tag 1 erkannt**
7	`1`	`1x1x 01x0 xxx1`	Kollisionen, Erweiterung der Maske
8	`01`	`1010 0100 0101`	**Tag 2 erkannt**
9	`11`	`1111 0110 1011`	**Tag 3 erkannt**

Tabelle 3.6: Vollständiges Inventory

Algorithm 1: stdRec

Data: mask
response = sendRequest(mask)
if *response hat Kollisionen* **then**
    stdRec(0 + mask)
    stdRec(1 + mask)
else
    if *response enthält ID* **then**
        return ID
    end
end

3.7.3 Identifizierung von zwei Tags bei einer Kollision (Algorithmus 2: twoAtOnce)

Der in ISO-15693 angegebene Algorithmus lässt Raum für Verbesserungen. Tritt nur eine einzige Kollision auf, ist es ohne weitere Voraussetzungen möglich, zwei Tags zur gleichen Zeit zu identifizieren [LXXL05]. Da jede ID eindeutig ist, müssen sich zwei Tags in mindestens einem Bit unterscheiden. Ist nur eine Kollision vorhanden, hat eine ID an dieser Position eine 0 und die andere eine 1. Im Beispiel von Tabelle 3.7 existieren nur die IDs von Tag 1 und Tag 2, um die Kollision zu erklären.

Tag 1	`1001`
Tag 2	`1101`
Ergebnis	`1x01`

Tabelle 3.7: Eine Kollision bei zwei Tags

Algorithm 2: twoAtOnce

```
Data: mask
response = sendRequest(mask)
if response hat 1 Kollision then
    ID1 = ersetze Kollisionsbit mit '0'
    ID2 = ersetze Kollisionsbit mit '1'
    return ID1 and ID2
else if response hat Kollisionen then
    twoAtOnce(0 + mask)
    twoAtOnce(1 + mask)
else
    if response enthält ID then
        return ID
    end
end
```

Algorithmus 2 zeigt eine erweiterte Tiefensuche, die sich diese Beobachtung zu Nutze macht. Der Algorithmus gleicht dem Algorithmus 1 zu großen Teilen, jedoch wird in einer zusätzlichen Abfrage geprüft, ob die erhaltene Antwort genau eine Kollision beinhaltet. In diesem Fall können zwei Tags gleichzeitig identifiziert werden. In allen anderen Fällen ändert sich das Verhalten nicht. Bezogen auf das Beispiel aus dem vorherigen Abschnitt ergibt sich kein Geschwindigkeitsvorteil, da dort alle Antworten (siehe Tabelle 3.6) mindestens zwei Kollisionen aufweisen.

3.7.4 Tag-ID-Design mit Paritätsbit

Dieser Ansatz stellt eine Kombination aus einem Antikollisionsalgorithmus und der Auswahl von bestimmten Tag-IDs dar [KKLA08]. Er macht die Annahme, dass das niederwertigste Bit einer Tag-ID als Paritätsbit betrachtet werden kann. Wenn dieses

gewährleistet ist, können in einem Schritt zwei Tags auf einmal erkannt werden, sofern nur zwei Kollisionen aufgetreten sind. Die gewöhnlichen RFID-Tags, die kein Paritätsbit benutzen, können von diesem Ansatz nicht genutzt werden.

Im folgenden Beispiel (Tabelle 3.8) gibt das letzte Bit die Parität an, wobei eine 1 eine gerade Anzahl von Einsen in einer Tag-ID beschreibt. Wie in Abschnitt 3.7.3 können in einem ersten Schritt zwei mögliche Tag-IDs ermittelt werden: `x001` und `x011`. Da das letzte Bit nun jedoch auf die Anzahl der Einsen hinweist, folgt, dass die erste Tag-ID `0001` und die zweite `1011` sein muss.

Antwort	`x0x1`
	`x001`
	`x011`
Tag 1	`0001`
Tag 2	`1011`

Tabelle 3.8: Zwei Kollisionen bei zwei Tags

Während dieses Verfahren auf der einen Seite die Geschwindigkeit des Inventorys erhöht, reduziert sich auf der anderen Seite die Anzahl der möglichen Tag-IDs, die vergeben werden können, um den Faktor 2. Für Anwendungen, die nicht auf der Eindeutigkeit von Tag-IDs fußen, kann es ausreichend sein, dass eine Tag-ID nur in einem gewissen Bereich eindeutig ist. Dies würde in dem Fall, dass die Verfügbarkeit ungenutzter IDs gering ist, die mehrfache Verwendung von IDs erlauben. In diesem Ansatz ändert sich der Verwendungszweck der Tag-ID dahingehend, dass sie nicht mehr die eindeutige Identifizierung eines Gegenstandes erlaubt, sondern dass sie dem Aufbau der Kommunikation dient.

3.7.5 Wiederholtes Inventory, Masken erinnern (Algorithmus 3: remM)

Myung und Lee haben einen Vorschlag gemacht, um die Geschwindigkeit wiederholter Inventorys zu erhöhen [MLS06]. Ihr Algorithmus speichert die Knoten im Baum, die Tags eindeutig identifizieren (Identifizierender Knoten) und diejenigen, die zu keinen Antworten führen (Keine-Antwort-Knoten). Im erneuten Durchlauf werden die gespeicherten Knoten überprüft.

Neu hinzugekommene Tags verursachen entweder Kollisionen mit einer bereits gefundenen Tag-ID oder zu führen dazu, dass Keine-Antwort-Knoten zu identifizierenden Knoten werden. Im ersten Fall wird ein erneuter Antikollisions-Durchlauf für den letzten Teil der ID nötig. Verlässt ein Tag stattdessen den Lesebereich, wird aus einem identifizierenden Knoten ein Nicht-Antwort-Knoten und kann unter Umständen mit einem weiteren Nicht-Antwort-Knoten vereinigt werden.

Dieser Ansatz reduziert die Dauer eines wiederholten Inventorys, wenn sich die involvierten Tags nicht oder zu einem kleinen Teil ändern. Ein Nachteil kann erwartet werden, wenn sich eine große Anzahl der Tags ändert, da in diesem Fall die Speicherung der Masken keinen Mehrwert liefert und im Gegenteil zu vielen ergebnislosen Anfragen führt. Daher hängt die Eignung dieses Ansatzes von der Geschwindigkeit ab, mit der beispielsweise ein Fahrzeug folgt, da diese den Anteil der sich ändernden Tags bestimmt.

Algorithm 3: rememberMasks (remM)

Data: identifyingMasks, noResponseMasks
foreach *mask* ∈ *noResponseMasks* **do**
    anticollision(mask)
end
foreach *mask* ∈ *identifyingMasks* **do**
    anticollision(mask)
end
storeIdentifyingMasks()
storeNoResponseMasks()

4 Navigation mittels wiederbeschreibbarer RFID-Tags

In diesem Kapitel werden die Idee und die Herausforderungen für ein pheromonbasiertes Navigationsverfahren für autonome Fahrzeuge skizziert. Des Weiteren wird betrachtet, welche Parameter beim Aufbau eines solchen Systems verändert werden können. Anschließend werden ein allgemeiner und ein konkreter Ansatz für Algorithmen entwickelt, um einen Pfad zu schreiben und ihn zu einem späteren Zeitpunkt zu verfolgen.

4.1 Idee und Möglichkeiten

Im Institut für Telematik wurde die nachfolgend beschriebene Idee entwickelt, bei der die Navigation vollständig auf Basis von RFID-Tags und ohne Zuhilfenahme anderer Infrastruktur realisiert werden soll [BVT08].

4.1.1 Beschreibung des Systems

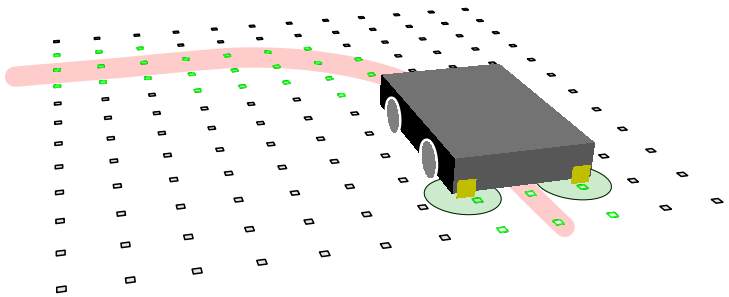


Abbildung 4.1: Navigation in einer Umgebung aus RFID-Tags, die Tags müssen nicht notwendigerweise symmetrisch verteilt sein, wie in dieser Abbildung dargestellt

Es wird eine Umgebung vorausgesetzt, in der viele RFID-Tags vorhanden sind (siehe Abbildung 4.1). Diese Tags sollen wie bei Zambonelli [MZ07] wiederbeschreibbar sein. Entlang eines vorher festgelegten Pfades können so Informationen gespeichert werden, die es ermöglichen, den Pfad zu einem späteren Zeitpunkt nachzufahren. Auf eine zentrale Kommunikationsinfrastruktur (wie beispielsweise Funk) wird verzichtet.

In einer Produktionsumgebung kann ein von einem Fahrer gesteuertes Fahrzeug genutzt werden, um einen Pfad für andere autonome Fahrzeuge zu markieren. Dies wird

als Teach-Phase oder Teaching bezeichnet. Während sich das Fahrzeug auf dem gewünschten Pfad befindet, werden die Markierungen geschrieben, die später für das Folgen notwendig sind (beim sogenannten Following oder in der Follow-Phase). In der gleichen Umgebung sind verschiedene Wege durch unterschiedliche Markierungen möglich. Dies ermöglicht alternative Routen und unterschiedliche Wege für unterschiedliche Fahrzeugklassen.

Der Ansatz bietet eine Kombination aus den Eigenschaften einer Spurführung mit Stützpunkten und aus einer Führung mit kontinuierlicher Linie (vgl. Abschnitt 2.3.4). Zum einen orientiert sich das Fahrzeug an Stützpunkten, aber zum anderen gibt jede Position auch die Richtung vor, in die es sich fortbewegen soll. Da der Pfadverlauf dem autonom agierenden Fahrzeug nicht bekannt ist, ist der Pfad implizit gegeben und da die absolute Position nicht ermittelt werden muss, handelt es sich um eine relative Positionierung zum Pfad.

Dieser Ansatz bietet viel Flexibilität. Wird die Produktionslinie geändert, können neue Routen mit geringem Aufwand und ohne bauliche Maßnahmen eingerichtet werden. Überlappende Pfade sind ebenfalls möglich. Es besteht keine Notwendigkeit eine Karte mit den Standorten aller Tags zu pflegen. Funkverbindungen für die Kommunikation mit einem zentralen Server sind nicht notwendig und ermöglichen den Einsatz in Gebieten in denen Funk nicht verlässlich funktioniert oder nicht erlaubt ist. Durch die verteilte Architektur, die lediglich eine verlässliche Kommunikation mit den RFID-Tags voraussetzt, ist der Ansatz sehr robust. Aufgrund der fallenden Tag-Preise und der einfachen Installation, bei der die Tags zufällig verteilt werden können, ohne dass ihre Positionen bestimmt und aufgenommen werden müssen, ist er kostengünstig. Des Weiteren können beschädigte oder fehlerhafte Tags einfach ersetzt werden. Da die nötigen Informationen redundant hinterlegt werden, ist der Ausfall einzelner Tags tolerabel, sofern der Follow-Algorithmus in der Lage bleibt, den Pfad zu erkennen.

Es besteht die Möglichkeit, neben den Pfadinformationen weitere Daten auf den Tags abzulegen, um innerhalb des Produktionsprozesses zusätzliche Einsatzszenarien zu erlauben. FTS können einem gesteuerten Fahrzeug unmittelbar folgen und so eine Art virtuellen Zug formen. Dies beinhaltet die Kommunikation über die aktuelle Fahrgeschwindigkeit, um Kollisionen mit dem vorherfahrenden Fahrzeug zu verhindern. Entdeckt ein FTS eine starke Verschmutzung, eine beschädigte Strecke oder ein anderes, defektes Fahrzeug, kann es einen Bereich als gesperrt markieren. Das gleiche kann geschehen, wenn ein Fahrzeug einen Fehler in seinem eigenen System entdeckt, bevor es anhält oder seinen Dienst einstellen muss. Andere Fahrzeuge können so ausweichen und alternative Routen benutzen. Des Weiteren können die bereits in Abschnitt 2.3.3.4 genannten Erweiterungen des Projektes TagDrive ermöglicht werden (Speichern von Geschwindigkeit, Steigung, zulässiges Gewicht, Vorfahrtsregeln usw.).

Der Ansatz ermöglicht außerdem die Steuerung der Position für Parkplätze oder Service-Stellen. Bevor der Pfad verlassen wird, markiert das Fahrzeug eine Position als besetzt. Andere Fahrzeuge erkennen daran, dass die Position nicht mehr verfügbar ist und fahren zur nächst freien weiter.

4.1.2 Anforderungen an das Navigations-System

Die Hauptherausforderung besteht darin, Algorithmen zu entwickeln, die es einem FTS ermöglichen, einem Pfad zuverlässig zu folgen, ohne ihn zu verlieren. Dazu muss während der Teach-Phase ein Pfad hinterlassen werden, der das Folgen möglichst einfach gestaltet. Der Pfad sollte neben den unumgänglichen physikalischen Lücken zwischen den Tags, keine Informationslücken aufweisen, die beispielsweise entstehen könnten, wenn nicht alle erreichbaren Tags erkannt wurden oder der Schreibvorgang nicht erfolgreich zu Ende geführt wurde. Hierzu kann es erforderlich sein, die maximale Teach-Geschwindigkeit auf einen Wert zu begrenzen, der unterhalb der erreichbaren Geschwindigkeit in der Follow-Phase liegt.

Da der Pfad nur aus Punkten in Form von Tags besteht, die markiert wurden und sich in einem gewissen Abstand zueinander befinden, muss kontinuierlich die physikalische Lücke zwischen den Tags überwunden werden. Es muss verhindert werden, dass das Fahrzeug in diesem Bereich von dem gewünschten Pfad so sehr abweicht, dass es die nächsten markierten Tags nicht mehr findet.

Werden zu einem Zeitpunkt nur Tags detektiert, die keine Pfadinformationen enthalten, kann das Fahrzeug schlussfolgern, dass es den Pfad verloren hat. Ist es nicht in der Lage abzuschätzen, in welche Richtung die Fahrt korrigiert werden muss, um wieder auf den Pfad zu treffen, stehen zwei Möglichkeiten zur Auswahl. Die Folge-Aufgabe kann abgebrochen werden, oder das Fahrzeug fährt zur letzten bekannten Position zurück, um nach dem korrekten Pfadverlauf zu suchen.

Es muss bedacht werden, dass Schleifen gefahren werden oder sich zwei unterschiedliche Pfade kreuzen könnten. Der Ausfall von Tags muss ebenso in Betracht gezogen werden, wie Tags mit fehlerhaften Informationen oder Tags, die überhaupt nicht beschrieben wurden. Eine weitere Aufgabe besteht darin, den Bedarf für den beschränkten Speicher eines RFID-Tags weitestgehend zu reduzieren. Dies ist nötig, um andere Anwendungen zu ermöglichen und den Zeitbedarf für die Kommunikation zu beschränken (und die Fahrtgeschwindigkeit zu erhöhen). In diesem Zusammenhang stellt sich auch die Frage, welche Daten auf den Tags abgelegt und wie diese ermittelt werden. In [MZ07] wird beispielsweise die Fahrtrichtung gespeichert. Diese könnte unter Zuhilfenahme eines Kompasses bestimmt werden.

In Produktionsumgebungen kann es vorteilhaft sein, eine Form von Alterung für Pfade zu ermöglichen (wie dies auch in der Natur bei den Ameisen der Fall ist). Somit könnten alte Pfade vergessen und Speicherplatz für neue Pfade geschaffen werden. Alternativ werden Algorithmen benötigt, die explizit Pfade löschen. Natürlich darf die Alterung permanent wichtige Pfade nicht betreffen. Des Weiteren sollte die Anzahl der Schreib- und Leseoperationen minimiert werden, um eine höhere Fahrgeschwindigkeit zu erreichen.

4.2 Identifikation relevanter Parameter

Es existieren verschiedene Faktoren, die bei der Anwendung von RFID für Navigationsverfahren einen Einfluss auf den Inventory-Prozess oder das Design von Antikollisionsalgorithmen sowie auf die Teach- und Following-Verfahren haben.

Es können die folgenden Gebiete unterschieden werden, die anschließend genauer besprochen werden:

- Einrichtung des Fahrzeugs und des RFID-Lesesystems
- Einrichtung der Umgebung (des Tag-Feldes)
- Anforderungen an die Anwendung
- zusätzliche Informationen, die dem Algorithmus zur Verfügung stehen

4.2.1 Einrichtung des Fahrzeugs und des RFID-Lesesystems

Das RFID-Lesesystem ist mit dem Fahrzeug verbunden. Wie viele Lesegeräte benutzt werden und wo diese angebracht werden, hat große Auswirkungen auf die Möglichkeiten des Pfad-Folgens.

Anzahl der Lesegeräte oder Antennen

Die Anzahl der Lesegeräte ist prinzipiell variabel. Auch kann ein Lesegerät mit mehreren Antennen ausgestattet werden, die gleichzeitig oder abwechselnd benutzt werden. Je mehr Lesegeräte oder Antennen verwandt werden, desto genauer ist Bestimmung der Tag-Position möglich.

In Abhängigkeit von der Reichweite der einzelnen Antennen muss sichergestellt werden, dass es nicht zu Reader-Kollisionen kommt. Das Problem von Reader-Kollisionen entsteht, wenn sich die Felder zweier gleichzeitig aktiver Reader überschneiden [Eng01, LNC05]. In diesem Falle ist keinem der beteiligten Reader die Kommunikation mit den Tags möglich.

Position der Lesegeräte

Für die Lesegeräte bietet sich die Montage an der Vorderseite des Fahrzeuges oder aber auch an dessen Rückseite an. Dieses beeinflusst nicht nur wo die Tags in Bezug auf das Fahrzeug erkannt werden, sondern auch zu welchem Zeitpunkt. Für das Folgen ist es sinnvoll, den Pfad möglichst früh zu erkennen und den/die Reader demzufolge vorne zu befestigen. Im Gegensatz dazu ist es beim Teaching eher von Vorteil, den Pfad spät zu schreiben, um so möglichst viele Informationen über den folgenden Pfadverlauf verwenden zu können. Auch die Kombination von Lesegeräten an verschiedenen Positionen kann sinnvoll sein. Bei der Montage der Lesegeräte muss auch der Abstand zum Boden betrachtet werden und wie sich der Lesebereich des Readers verhält.

Reichweite des Lesegerätes

Je höher die Reichweite und je mehr Tags detektiert werden können, desto mehr pfadrelevante Daten können ermittelt werden. Auf der anderen Seite reduziert eine höhere Reichweite auch die Genauigkeit der ermittelten Tag-Position.

Wenn mehr als ein Reader verwendet wird, können sich überschneidende Lesebereiche genutzt werden, um Tags genauer zu lokalisieren. Dies kann jedoch zu dem Problem von Reader-Kollisionen führen. Auch hier führt ein sehr großer Überschneidungsbereich zum Verlust der Positionsinformation eines Tags.

Die Größe des Radius bestimmt des Weiteren, wie lange ein Tag für die Detektion und weitere Kommunikation zur Verfügung steht. Eine höhere Reichweite ermöglicht somit prinzipiell eine höhere Fahrgeschwindigkeit, führt bei gleicher Tag-Dichte aber unter Umständen zu mehr Tags, die erkannt werden müssen.

4.2.2 Umgebung

Nicht allein die Einrichtung des Fahrzeugs mit den Lesegeräten bestimmt die Möglichkeiten für die Orientierung. Die Umgebung, in der sich das Fahrzeug bewegt, spielt eine wichtige Rolle.

Tag-Dichte

Je mehr Tags erkannt werden können, desto mehr Informationen sind verfügbar, um einem Pfad präzise folgen zu können, und die Aufgabe einem Pfad aus diskreten Punkten zu folgen, wandelt sich zu der Aufgabe, einem kontinuierlichen Pfad zu folgen. Auf der anderen Seite verlängern mehr Tags die Inventory-Dauer und erfordern mehr Schreib- und Leseoperationen. Daher ist es nicht immer von Vorteil, die Tag-Dichte zu erhöhen.

Platzierung der Tags

Die Tags können zufällig oder gleichmäßig verteilt werden. Außerdem kann die Platzierung gemäß einer vorgegebenen Anordnung erfolgen. Eine symmetrische Verteilung bietet unter Umständen Vorteile, da diese bei der Entwicklung der Navigationsalgorithmen berücksichtigt und ausgenutzt werden kann.

Tag-Technologie

Die RFID-Tags können unterschiedliche Merkmale aufweisen. Unter anderem können sie sich im verfügbaren Speicherplatz und in der Kommunikationsgeschwindigkeit unterscheiden. Die Wahl der Tags muss mit dem RFID-Reader abgestimmt sein, wobei viele RFID-Reader von Haus aus mehrere Protokolle des gleichen Frequenzbandes unterstützen.

4.2.3 Anforderungen an das Navigationssystem

Ein dritter Einflussfaktor ist durch die Anforderungen gegeben, die an das Navigationssystem gestellt werden können. Besondere Bedeutung haben die Fahrgeschwindigkeit und die Drehrate.

Fahrgeschwindigkeit

Da die Kommunikationsgeschwindigkeit zwischen Tags und Lesegerät begrenzt ist, kann eine zu hohe Fahrgeschwindigkeit (bei der Teaching- oder der Follow-Phase) zu unvollständigen Inventorys oder unterbrochenen Schreib- oder Lesevorgängen führen. Dies führt in der Konsequenz zu einer geringeren Datenbasis, die verwendet werden kann.

Drehrate/Drehradius

Der zulässige Drehradius mit dem ein Fahrzeug Kurven fahren darf, ist ein kritischer Wert für den Folge-Algorithmus. Ist die Tag-Dichte in Bezug auf den Drehradius und die Geschwindigkeit zu gering, ist es nicht möglich dem Pfad zu folgen, wenn die Drehung während der physikalischen Lücken zwischen den Tags vorgenommen wird und ohne dass diese für das Fahrzeug vorhersehbar ist.

4.2.4 Informationen, die dem Algorithmus zur Verfügung stehen

Zu guter Letzt ist von Interesse, welche zusätzlichen Informationen den Algorithmen zur Verfügung stehen. Hierunter werden Daten verstanden, die nicht unmittelbar durch das RFID-System geliefert werden.

Sensoren

Zusätzliche Sensoren (neben den RFID-Readern) können genutzt werden, um bessere Voraussetzungen für die Pfad-Erkennung zu schaffen. Beispielsweise könnte ein Kompass die absolute Fahrtrichtung ermitteln oder mittels der Odometrie könnte die Drehgeschwindigkeit bestimmt werden. Auch inertiale Systeme können wichtige Daten liefern. Derartige Informationen können sowohl beim Teaching als auch für das Following verwendet werden.

Erzeugte Daten

Durch die Auswertung des bisherigen Pfadverlaufes können die Algorithmen selbst weitere Informationen erzeugen. Diese Informationen können beispielsweise aus der Reihenfolge, in der Tags erkannt wurden, gewonnen werden (basierend auf der ansteigenden Sequenznummer) oder können Meta-Daten sein, wie die vermutete Position eines Tags.

4.3 Testumgebung

Für die praktische Überprüfung von Algorithmen wurde ein Testsystem bestehend aus Hard- und Software aufgebaut.

4.3.1 Hardware

Als Hardware wurde unter anderem ein Roboter auf Basis der LEGO-NXT-Plattform vorgesehen (Abbildung 4.2). Diese Plattform stellt verschiedene Schnittstellen zur Verfügung, über die weitere Sensoren und Motoren angeschlossen werden können. Es standen zwei Lesegeräte zur Auswahl: OpenPCD-Reader[1], die über das Two-Wire-Interface (TWI bzw. I^2C) angeschlossen wurden und seriell angesprochene RFID-Lesegeräte der Firma Feig, die in einer alternativen Konfiguration genutzt werden konnten. Beide Reader unterstützen das ISO-15693-Protokoll und haben eine Reichweite zwischen 10 und 15 cm. Von LEGO standen Motoren und Kompassmodule zur Verfügung.

Als RFID-Umgebung wurden ISO-15693-Tags verwendet, die symmetrisch auf mehreren $1\,m^2$ großen Matten verteilt wurden. Der Abstand zwischen den je 100 Tags betrug ca. 10 cm, so dass meist vier Tags in Reichweite waren. Die Matten konnten je nach Bedarf zu einer größeren Fläche zusammengefügt werden.

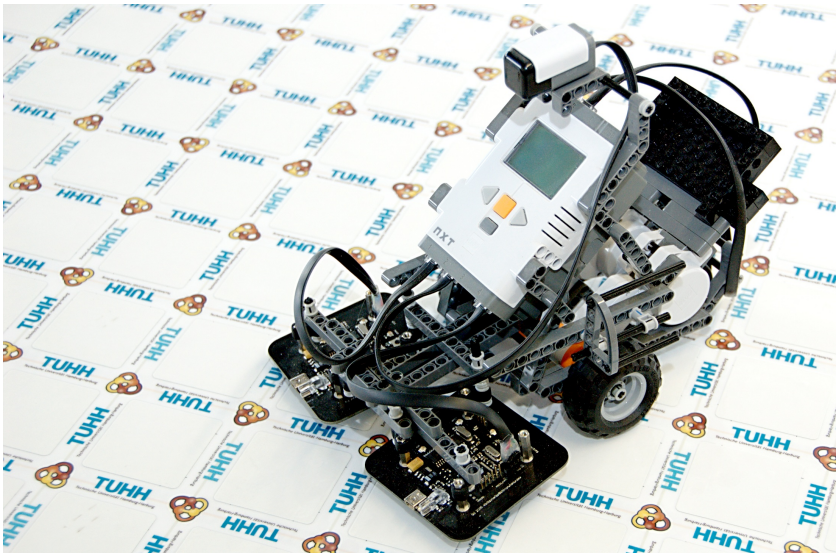


Abbildung 4.2: Roboter auf NXT-Basis

4.3.2 Software

Das leJOS-NXJ-Projekt bietet eine alternative Firmware zur vorinstallierten LEGO-Firmware für den Roboter, die dessen Programmierung in Java erlaubt. Darüber hinaus

[1] `http://www.openpcd.org`

werden verschiedene Klassen mitgeliefert, die eine komfortable Steuerung der Motoren erlauben. Aufgrund der Plattformunabhängigkeit von Java war es möglich, eine Abstraktionsschicht zu schaffen, die es Algorithmen erlaubt, sowohl auf simulierter als auch auf realer Hardware ausgeführt zu werden (Abbildung 4.3). Die Algorithmen haben Zugriff auf die Module Fahrzeugsteuerung, Sensoren (Kompass, RFID-Lesegerät etc.) und können Log-Daten zum Monitoring versenden. Die Implementierung der Module richtet sich nach der unterliegenden Hardware oder dem Simulator. Für die Evaluation der Algorithmen wurde auf dieser Basis eine Software entwickelt (Abbildung 4.4), die für folgende Aufgaben genutzt werden konnte:

- Simulation von Roboter, Sensoren und Tags zur Ausführung von Algorithmen im Simulator
- Steuerung des realen Roboters während der Teach-Phase, Kommunikation mit dem Roboter
- Ausgabe von Log-Daten und graphische Aufbereitung dieser zum Monitoring während der Teach- und Follow-Phasen eines realen Roboters

Der letzte Punkt ist von besonderer Wichtigkeit, da so ausgewertet werden kann, welche Tags geschrieben und welche Informationen dort hinterlegt werden bzw. welche Tags (während der Follow-Phase) erkannt und welche Daten von diesen gelesen werden. Zur Kommunikation zwischen Software und Roboter wurde die Bluetooth-Schnittstelle genutzt, über die Steuerbefehle und Log-Daten übertragen wurden.

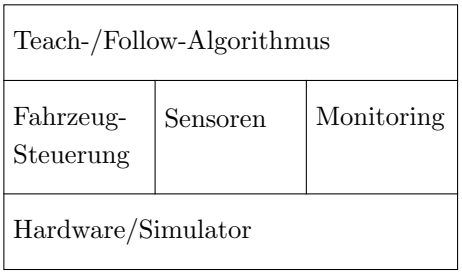


Abbildung 4.3: Schnittstellen zwischen Algorithmen und Hardware/Simulator

Im Simulator kann eine beliebige Anzahl von RFID-Tags frei platziert werden. Die Tags besitzen eine ID und einen beschreibbaren Speicher. Wird ein Inventory mit einem simulierten Lesegerät ausgeführt, liefert es die Tags zurück, deren Zentrum sich innerhalb der festgelegten Reichweite des Lesegerätes befinden. Dabei werden keine Kommunikationsfehler simuliert. Über die Fahrzeugsteuerung kann die Position und Ausrichtung des Fahrzeugs verändert werden, die vom Kompass auch reflektiert wird.

4.4 Allgemeiner Ansatz für ein Navigationsverfahren

Die geeignete Implementierung für Algorithmen für Following und Teaching hängt stark von den gewählten Voraussetzungen ab, die in Abschnitt 4.2 genannt wurden. Dennoch kann eine Struktur formuliert werden, die allgemein verwendet werden kann.

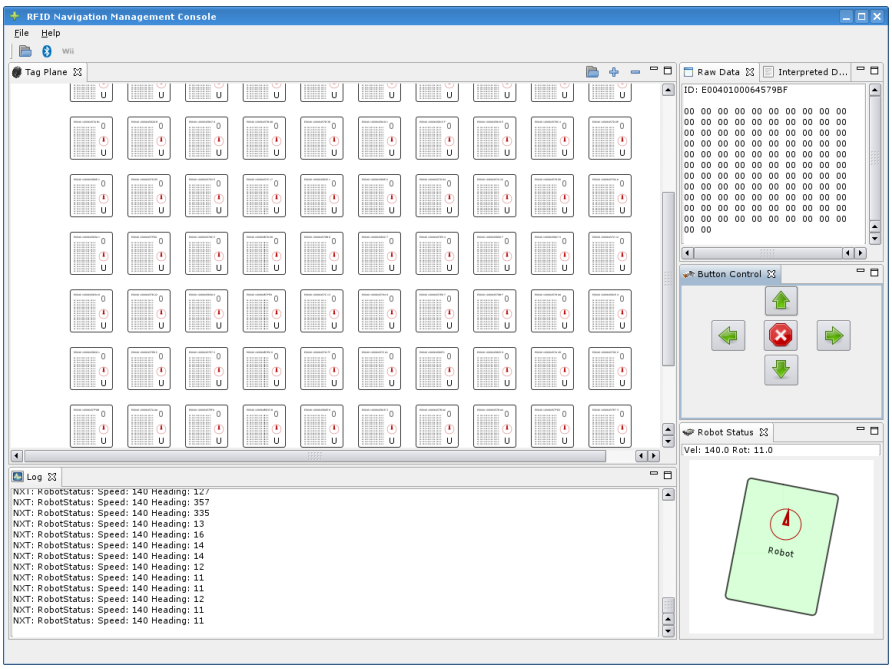


Abbildung 4.4: Steuerungs- und Auswertungssoftware für das Testsystem

4.4.1 Pfad-Folgen

Die Durchführung des Following kann in drei verschiedene Prozesse eingeteilt werden (siehe Abbildung 4.5).

Inventory-/Leseprozess: Der Inventory- und Leseprozess dient der Detektion der Tags in Reichweite und dem Auslesen der benötigten Informationen. Dabei können auch Möglichkeiten wie Tag-Caching benutzt werden, bei dem die Daten eines Tag nur einziges Mal gelesen und anschließend wiederverwendet werden.

Folgeprozess: Hier werden alle verfügbaren Daten genutzt, um die Position des Fahrzeuges und den Verlauf des Pfades zu bestimmen.

Steuerungsprozess: Der Steuerungsprozess legt die einzunehmende Geschwindigkeit und Fahrtrichtung fest. Für deren Berechnung benutzt er die Daten aus dem Folge-Prozess und den aktuellen Fahrzeugstatus.

4.4.1.1 Inventory-/Leseprozess

Der Inventory-Vorgang ist unter Berücksichtigung der Anforderungen des Follow-Algorithmus für die Durchführung des Inventorys verantwortlich. Er liefert je nach Voraussetzung alle Tags in Reichweite, neu hinzugekommene oder eine andere Untermenge. In allen Fällen werden anschließend die gespeicherten Daten der bislang unbekannten

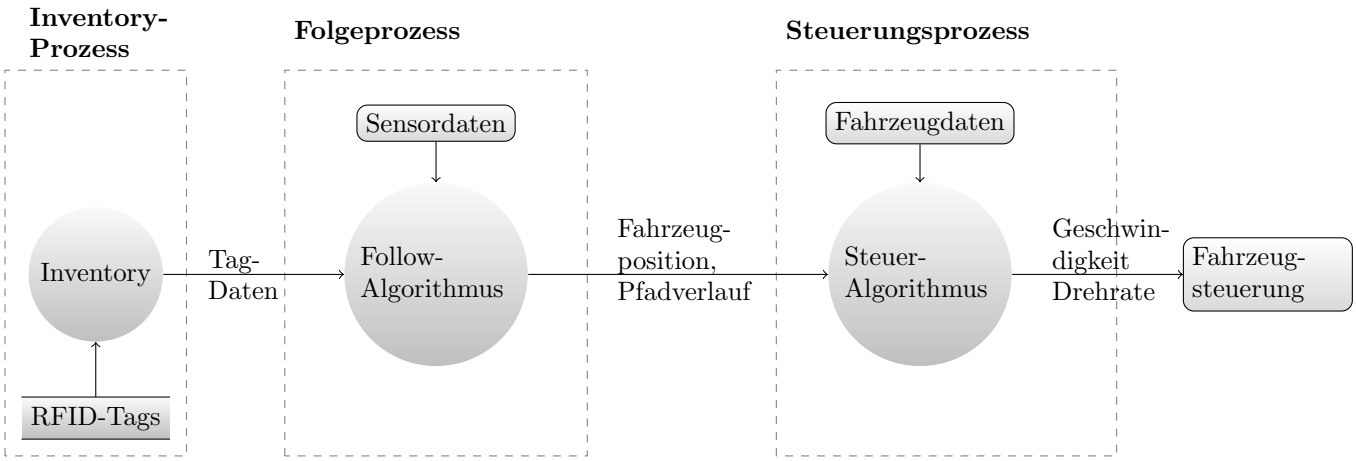


Abbildung 4.5: Datenfluss für Pfad-Folgen

Tags ausgelesen. Es ist möglich, dass die Antikollision und das Auslesen der Daten verschränkt[2] ausgeführt wird.

4.4.1.2 Folgeprozess

Die Hauptaufgabe dieses Bereiches besteht in der Bestimmung des zukünftigen Kurses und der momentanen Fahrzeugposition. Dafür müssen die folgenden Aufgaben bearbeitet werden:

Filterung
Die erste Aufgabe besteht darin, die detektierten Tags in Hinblick auf ihre Relevanz für die übergeordnete Aufgabe des Pfad-Folgens zu filtern. Alle Tags, von denen unvollständige oder korrupte Daten gelesen wurden, können entfernt werden. Des Weiteren können alle Tags aussortiert werden, die irrelevante Daten enthalten, beispielsweise solche, die zu anderen Pfaden gehören. Korrupte Daten können unter anderem erkannt werden, wenn eine Prüfsumme verwendet wird oder wenn die Daten einer Plausibilitätsprüfung unterzogen werden. Fehlerhafte Tag-Daten wären dann zu vermuten, wenn Tags, die redundante Informationen beinhalten sollten, widersprüchliche Daten vorweisen oder der berechnete Pfadverlauf unrealistisch ist.

Des Weiteren muss sichergestellt werden, dass zuvor benutzte Tags (die demzufolge „alte“ Daten enthalten) nicht erneut benutzt werden. Diese Informationen wurden bereits ausgewertet und haben zu der aktuellen Position geführt. Die Filterung soll aber die Daten liefern, die für den zukünftigen Pfadverlauf relevant sind. Die alten Daten könnten jedoch genutzt werden, um weitergehende Schlussfolgerungen über den Pfadverlauf anzustellen.

Am Ende der Filterung sollten nur die benötigten Tags stehen.

[2] Das bedeutet, dass der Inventory-Vorgang einen Augenblick lang angehalten wird, um die Daten eines Tags sofort nach seiner Detektion auszulesen.

Fahrzeug-Lokalisierung

Die korrekte Positionierung des Fahrzeuges auf dem Pfad war nicht nur originäres Ziel der gesamten Navigation, sondern ist darüber hinaus sinnvoll, um sicherzustellen, dass der Pfad auch zukünftig erfolgreich abgefahren werden kann. Drei Problemfälle müssen bewältigt werden (siehe Abbildung 4.6):

- Das Fahrzeug befindet sich auf dem Pfad, aber in einer falschen Orientierung.
- Die Orientierung ist korrekt, aber das Fahrzeug ist abseits des Pfades positioniert.
- Auch die Kombination der vorherigen Fälle ist möglich. Das Fahrzeug befindet sich abseits des Pfades und hat eine falsche Orientierung.

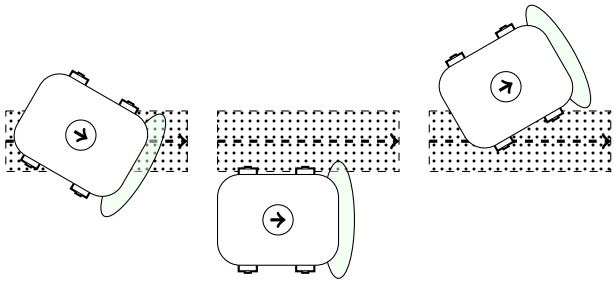


Abbildung 4.6: Mögliche Positionen auf dem Pfad, die eine Korrektur erforderlich machen

Zuerst soll die Abweichung der Position herausgefunden werden. Dieser Wert kann beispielsweise durch eine Zahl zwischen -1 und 1 dargestellt werden, welche auf die Größe und Richtung der Verschiebung quer zur Pfadrichtung hindeutet. Eine 0 bedeutet eine perfekte Platzierung, eine 1 hingegen die stärkst mögliche Verschiebung in Fahrtrichtung nach rechts, bei der der Pfad noch in Reichweite ist und -1 die stärkst mögliche detektierbare Verschiebung nach links. Für die Bestimmung der Positionsabweichung kann ausgewertet werden, welche Informationen auf eine korrekte Position hindeuten und welche dagegen. Die Größe der Positionsabweichung kann auch genutzt werden, um die Fahrgeschwindigkeit anzupassen und bei einer schlechten Platzierung die Folge-Geschwindigkeit zu verringern. Werden sowohl beim Teaching als auch beim Following nur jeweils ein Lesegerät bzw. eine Antenne genutzt, kann die Pfadposition nicht ohne Absuchen der Umgebung bestimmt werden.

Im zweiten Schritt soll bestimmt werden, wie die Abweichung der Orientierung zwischen Fahrzeug und Pfad ist. Dieses ist am einfachsten umsetzbar, wenn ein Kompass genutzt wird. Weitere Möglichkeiten sind in genauer Odometrie und inertialen Messsystemen zu sehen.

Pfad-Vorhersage

Insbesondere bei der Anwendung in Umgebungen mit einer geringen Dichte von RFID-Tags ist es nützlich, eine allgemeine Vorhersage zu haben, in welche Richtung der Pfad verlaufen wird. Ändert sich die Richtung des Pfades, kann mit einer gewissen Wahrscheinlichkeit davon ausgegangen werden, dass eine Kurve gefahren und das darauffolgende Tag auf eine abermalige Änderung der Richtung hinweisen wird. Daher ist es

in diesem Fall sinnvoll, ebenfalls eine Kurve einzuschlagen und nicht nur eine kurze Richtungskorrektur vorzunehmen

4.4.1.3 Steuerungsprozess

Der Steuer-Algorithmus ist dafür verantwortlich, das Fahrzeug in die richtige Richtung zu lenken. Zu diesem Zweck benutzt er die berechneten Werte des Folge-Algorithmus (relative Fahrzeugposition zum Pfad und Pfadrichtung), um ein gleichmäßiges Folgen und möglicherweise die Annäherung an den Pfad zu gewährleisten. Befindet sich das Fahrzeug neben dem Pfad, aber ändert sich der Pfad in diese Richtung, ist unter Umständen keine Korrektur des Kurses notwendig. Ändert sich der Pfad hingegen in die entgegengesetzte Richtung, ist vermutlich eine sehr starke Kurskorrektur vonnöten.

4.4.2 Allgemeiner Ansatz zum Pfad-Schreiben

Die Teaching-Phase hat zum Ziel, die nötigen Voraussetzungen zu schaffen, die es dem Follow-Algorithmus ermöglichen, den Pfad erfolgreich und zuverlässig folgen. Abbildung 4.7 zeigt die Prozesse, in die man das Teaching einteilen kann. In Abhängigkeit von den gegebenen oder gewählten Bedingungen werden fortlaufend Inventorys ausgeführt und zweckmäßige Informationen in die Tags geschrieben. Beim Schreiben können zwei unterschiedliche Aktionen identifiziert werden, das initiale Schreiben und das Aktualisieren eines Tags.

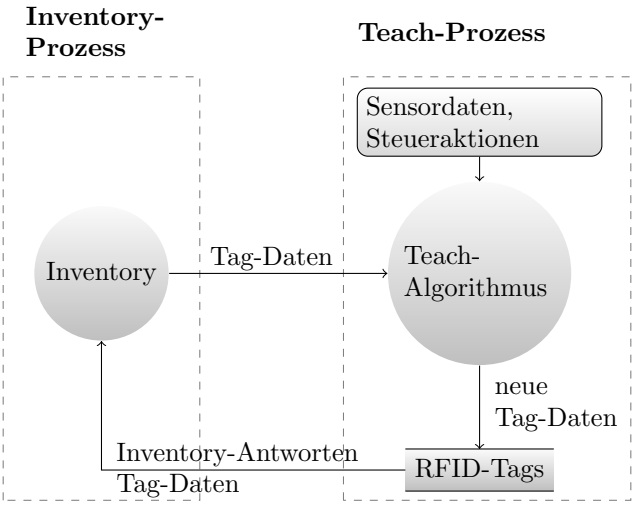


Abbildung 4.7: Datenfluss beim Pfad-Schreiben

4.4.2.1 Inventory-Prozess

Ähnlich wie das Inventory beim Folgen werden auch hier die Tags in Reichweite ermittelt und nötigenfalls ihre Daten gelesen. Im Gegensatz zum vorherigen Inventory sollte hier

allerdings angestrebt werden, alle Tags zu detektieren. Es ist nicht bekannt, welche Tags beim späteren Folgen erkannt werden können, so dass Einschränkungen die Folge wären, würde nur eine Teilmenge erfasst werden.

4.4.2.2 Teach-Prozess

Der Teach-Prozess nutzt die verfügbaren Sensordaten und die Steueraktionen des Fahrers als Eingaben, um die Daten zu ermitteln, die auf den Tags hinterlegt werden sollen.

Initiales Schreiben
Das Initiale Schreiben wird angewendet, wenn ein Tag zum ersten Mal detektiert wurde und dort somit noch keine Informationen zu dem aktuellen Pfad abgelegt wurden.

Aktualisieren
Wird ein Tag nochmalig detektiert, ist unter Umständen eine Aktualisierung nötig, wenn neue Informationen bezüglich des Tags oder des Pfades vorhanden sind. Dies kann zum Beispiel der Fall bei Kurven sein, wo kurveninnere Tags einen längeren Zeitraum in Reichweite des Lesegerätes sind oder wenn Tags mehrmalig (beispielsweise von verschiedenen Lesegeräten) detektiert wurde und neue Informationen über ihre Position bereitstehen.

4.5 Teach- und Follow-Algorithmen für zwei Reader und Kompass

Aufbauend auf dem vorherigen Ansatz wurde eine Lösung entwickelt und real getestet. Es kommen zwei RFID-Lesegeräte und ein Kompassmodul zum Einsatz.

4.5.1 Ansatz

Um das Problem der Lokalisierung lösen, wurden zwei Lesegeräte vorgesehen. Hiermit ist es während der Teach-Phase möglich, die Tags in verschiedene Gruppen einzuteilen: Tags die mit dem rechten Lesegerät erkannt wurden und solche, die mit dem linken Lesegerät erreicht werden konnten. Da sich die Lesebereiche überschnitten, konnte zusätzlich eine dritte Gruppe der Tags identifiziert werden, die in der Reichweite beider Lesegeräte waren und somit eine mittlere Position besitzen. Auf die Tags wurden die Sequenznummer, die Pfad-ID, die Fahrtrichtung und die Information, mit welchem Lesegerät das Tag detektiert wurde, gespeichert. In Abbildung 4.8 ist die mögliche Konfiguration eines Tag-Feldes nach der Teach-Phase aufgezeichnet.

Durch die Klassifizierung in linke, mittlere und rechte Tags kann das Fahrzeug während Follow-Phase bestimmen, wie seine Position auf dem Pfad ist. Werden beispielsweise mit dem linken Lesegerät als rechts markierte Tags erkannt und mit dem rechten Gerät

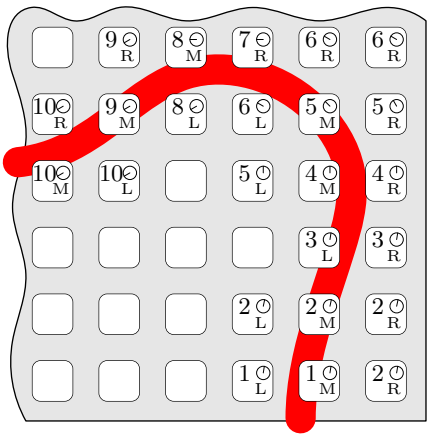


Abbildung 4.8: Beispiel für einen einzelnen Pfad eines Roboters in einer Umgebung aus RFID-Tags: nur die Sequenznummern, die Richtung und die Tag-Positionen (**L**inks, **M**itte, **R**echts) wurden eingezeichnet, die Pfad-ID wurde fortgelassen

keine, dann befindet sich das Fahrzeug rechts neben dem Pfad. Durch die Markierung mittlerer Tags soll eine weitere Verbesserung der Lokalisierung erreicht werden.

Ändert sich die Richtung über einen bestimmten Wert hinaus und ist ein bereits beschriebenes Tag noch in Reichweite, wird dieses Tags aktualisiert, um frühzeitig Pfadänderungen erkennen zu können. Die Aktualisierung ist auch nötig, um mittlere Tags zu markieren. Diese können erst als zentral erkannt werden, wenn sie bei einem zweiten Inventory-Durchgang von dem anderen Lesegerät erkannt wurden. Die Sequenznummer bleibt jeweils unverändert, unabhängig davon, welcher Aktualisierungsfall eingetreten ist.

Im Folgenden wird beschrieben, wie Teaching und Following funktionieren. Es wird dabei angenommen, dass die beiden Lesegeräte nebeneinander an der Vorderseite des Fahrzeugs montiert sind.

4.5.2 Teaching

Im Teaching-Modus steuert eine Person das Fahrzeug entlang des gewünschten Pfades. Die beiden Fälle beim Schreiben haben folgende Bedeutungen:

Initiales Schreiben: Alle Werte - Sequenznummer, Drehrate, Fahrzeugrichtung und die Position des Readers (links oder rechts) - werden in die Tags geschrieben. Wenn zumindest ein neu detektiertes Tag geschrieben wurde, wird die Sequenznummer inkrementiert.

Aktualisieren: Da der linke und der rechte Reader zeitlich versetzt genutzt werden müssen, um Reader-Kollisionen zu vermeiden, ist es nicht möglich, die Position mittlerer Tags sofort zu bestimmen. Wie in Abbildung 4.9 dargestellt, erkennt der

linke Reader anfangs drei Tags. Während das Fahrzeug weiterfährt, erkennt nun der rechte Reader drei Tags, kann bestimmen, dass ein mittleres Tag existiert und aktualisiert die Position dieses Tags.

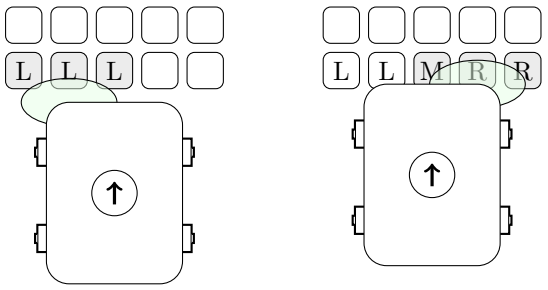


Abbildung 4.9: Aktualisieren der Tag-Position

Algorithm 4: Teaching

```
Data: PATH_ID
oldTags = {}
while true do
    foreach reader ∈ {leftReader, rightReader} do
        currentTags = reader.inventory()
        newTags = filterNew(currentTags, oldTags)
        foreach tag ∈ newTags do
            reader.write(tag, sequenceNo, PATH_ID, richtung, reader.position)
        end
        middleTags = filterMiddle(currentTags, oldTags)
        foreach tag ∈ middleTags do
            reader.update(tag, richtung, middle)
        end
        updateTags = filterUpdate(currentTags, oldTags, richtung)
        foreach tag ∈ middleTags do
            reader.update(tag, richtung)
        end
        oldTags += newTags
    end
    if Tags wurden beschrieben then
        sequenceNo++
    end
end
```

In Algorithmus 4 ist der Ablauf der Teach-Phase angegeben. Der Reader führt fortlaufend Inventorys durch (`reader.inventory()`), welche alle in Reichweite liegenden Tags ermitteln. Die IDs der entdeckten Tags (`currentsTags`) werden mit den IDs der vorher detektierten und in einem Speicher abgelegten IDs (`oldTags`) verglichen (`filterNew()`). Die neu erschienenen Tags (`newTags`) bilden dann die Gruppe, in welche die Daten geschrieben werden (`reader.write()`). Somit wird die Sequenznummer gleichzeitig in

mehreren Tags abgelegt (siehe Abbildung 4.8), sofern die Gruppe aus mehr als einem Tag besteht.

Im nächsten Schritt wird geprüft, ob Tags vorhanden sind, die bereits zuvor von einem anderen Lesegerät erkannt wurden (`filterMiddle()`). In diesem Fall handelt es sich um mittlere Tags, und deren Position muss aktualisiert werden (`reader.updatePosition()`).

Anschließend werden die Tags bestimmt, die bereits Informationen enthalten, aber deren Richtungsinformation aufgrund einer Kurvenfahrt nicht mehr gültig ist (`filterUpdate()`). Bei jedem dieser Tags wird die Richtungsinformation aktualisiert.

Die Sequenznummer wird erhöht, wenn neue Tags gefunden und beschrieben wurden und das Prozedere wird wiederholt. Während des gesamten Vorganges bewegt sich das Fahrzeug kontinuierlich fort. Bleibt das Fahrzeug stehen, können keine neuen Tags ermittelt werden. Folglich werden keine Tags beschrieben und die Sequenznummer wird nicht inkrementiert.

4.5.3 Folgen

Im Following-Modus (Algorithmus 5) soll ein FTS anschließend den zuvor abgelegten Pfad nachvollziehen. Es wird angenommen, dass dem Fahrzeug die ID des zu folgenden Pfades bekannt ist und es sich an einer Position befindet, an der der Pfad in Reichweite ist. Das Fahrzeug startet mit einer Inventory-Operation (`reader.inventory()`) und überprüft für jedes erkannte Tag (`currentTags`), ob es bereits zuvor erkannt und sein Inhalt ausgelesen wurde. Ist dies nicht der Fall, wird das Tag gelesen (`reader.read()`). Wurde das Tag zuvor gelesen, ist es nicht erforderlich dies erneut zu tun, da sich der Inhalt während eines Follow-Durchlaufes nicht ändert.

Im nächsten Schritt (`filterRelevant()`) wird bestimmt, welche der erkannten Tags relevant sind. Damit erreichbare Tags als relevant erkannt werden, müssen sie die korrekte Pfad-ID tragen und die Sequenznummer s_i muss die folgende Bedingung erfüllen:

$$s_{i-1} < s_i < s_{i-1} + m$$

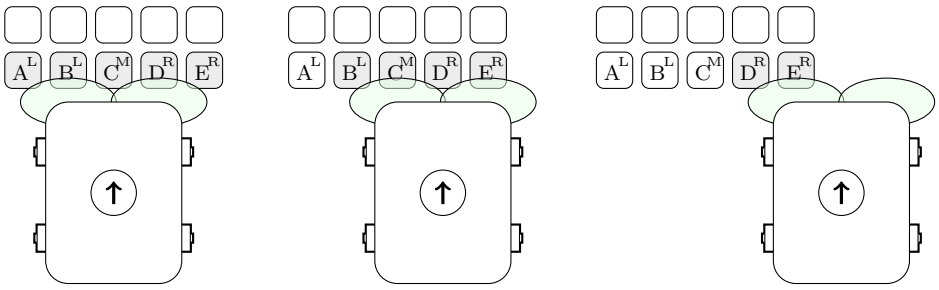
Der Index i bezeichnet dabei den jeweiligen Durchlauf. Zum einen muss die neue Sequenznummer größer sein, als die, des vorherigen Durchlaufs. Zum anderen darf sie keinen größeren Wert als $s_{i-1} + m$ besitzen. Die Variable m steht dabei für die Anzahl der Sequenznummer-Schritte, die übersprungen werden können, um eine höhere Folgegeschwindigkeit zu erreichen. Sie sollte nicht zu hoch sein, um zu verhindern, dass das FTS bei sich selbstkreuzenden Pfaden abbiegt oder auf einen anderen Pfad-Abschnitt überspringt, der nahe dem aktuellen liegt.

Durch die Analyse der Tag-Positionen (L/R/M) und des Richtungswertes wird das FTS in die Lage versetzt, seine Position zu korrigieren (`estimateRobotPosition()`). Als letztes wird die höchste Sequenznummer der verwandten Tags (`highestSequenceNo()`) als neue Sequenznummer gesetzt.

Unterschiedliche und sich kreuzende Pfade werden durch verschiedene Pfad-IDs ermöglicht. Ein Pfad darf sich selbst kreuzen, wenn die Größe von m berücksichtigt wird. Diese wiederum hängt vom Drehradius und von der Tag-Dichte ab.

Algorithm 5: Following

```
Data: PATH_ID, m
while true do
    foreach reader ∈ {leftReader, rightReader} do
        currentTags = reader.inventory()
        foreach tag ∈ currentTags do
            if tag wurde zuvor noch nicht gelesen then
                reader.read(tag)
            end
        end
        relevantTags = filterRelevant(currentTags, sequenceNo, m, PATH_ID)
        estimateRobotPosition(relevantTags, reader.position)
    end
    sequenceNo = highestSequenceNo(relevantTags)
end
```



(a) Optimal positioniertes Fahrzeug, Pfadabweichung = 0

(b) Fahrzeug ist etwas zu weit rechts positioniert, Pfadabweichung = 0,25

(c) Sehr starke Pfadabweichung nach rechts, Pfadabweichung = 1

Abbildung 4.10: Bestimmung der Pfadabweichung anhand der Tag-Positionen

Für die Lokalisierung in `estimateRobotPosition()` wird ausgewertet, welche Informationen auf eine korrekte Position hindeuten und welche dieser Annahme widersprechen. In Abbildung 4.10 sind mögliche Fälle angegeben. Werden mit jedem Lesegerät Tags erkannt, deren Soll-Positionen der Position des jeweiligen Lesegeräte entspricht, befindet sich das Fahrzeug optimal positioniert, wie in Abbildung 4.10(a). Eine geringe Abweichung, wie in Abbildung 4.10(b), kann erkannt werden, wenn die Lesegeräte zwar Tags mit den richtigen Positionen erkennen, aber auch solche dabei sind, die auf eine verschobene Position hindeuten. In Abbildung 4.10(c) lassen hingegen alle erkannten Tags auf eine falsche Position schließen. In Tabelle 4.1 findet sich eine Beispielrechnung für die Pfadabweichung aus der Abbildung.

In Form einer Matrix wurde notiert, welches Lesegerät welche Tag-Positionen erkannt hat. Die Diagonale zeigt somit die Anzahl der exakt positionierten Tags an. Die rechte obere Dreiecksmatrix beinhaltet die Tags, die auf eine Verschiebung nach rechts hinweisen, die linke untere Dreiecksmatrix die Tags, die auf eine Verschiebung nach links

hindeuten. Daraus kann die Abweichung der Position berechnet werden:

- Tags die mit dem Lesegerät der richtigen Position erkannt wurden, werden nicht betrachtet. Sie weisen auf keine Abweichung hin. (Weiße Kästchen in der Tabelle)
- Mittlere Tags, die nicht von beiden Lesegeräte erkannt wurden, werden mit 0,5 bewertet. (Graue Kästchen in der Tabelle)
- Linke oder rechte Tags, die von beiden Lesegeräten erkannt wurden, werden ebenfalls mit 0,5 bewertet. (Graue Kästchen in der Tabelle)
- Tags, deren Position mit den entgegen gesetzten Lesegeräten erkannt wurden, werden mit 1 bewertet. (Schwarze Kästchen in der Tabelle)

Tags aus der rechten oberen Dreiecksmatrix werden mit einem positiven Vorzeichen versehen, Tags aus der linken unteren Dreiecksmatrix mit einem negativen. Anschließend werden die Werte aufsummiert und durch die Anzahl der beteiligten Tags geteilt. Wurde jedes Tag von seinem entsprechenden Lesegerät detektiert (d.h. nur die Diagonale ist besetzt), beträgt das Ergebnis genau 0.

Lesegerät	erkannte Tags	L	M	R (Anzahl)
nur links	A,B	2	0	0
links und rechts	C	0	1	0
nur rechts	D,E	0	0	2
(0 + 0 - 0 - 0) · 0,5 + (0 - 0) · 1 = 0				
→ Pfadabweichung = $\frac{0}{5} = 0$				
nur links	B,C	1	1	0
links und rechts	D	0	0	1
nur rechts	E	0	0	1
(1 + 1 - 0 - 0) · 0,5 + (0 - 0) · 1 = 1				
→ Pfadabweichung = $\frac{1}{4} = 0{,}25$				
nur links	B,C	0	0	2
links und rechts		0	0	0
nur rechts		0	0	0
(0 + 0 - 0 - 0) · 0,5 + (2 - 0) · 1 = 2				
→ Pfadabweichung = $\frac{2}{2} = 1$				

Tabelle 4.1: Beispielberechnung der Pfadabweichung für Abbildung 4.10

Die Richtungsabweichung kann sofort erkannt werden, indem die in den Tags abgelegte Richtung mit der momentanen Richtung des Fahrzeug verglichen wird. Für die Fahrzeugsteuerung kann anschließend ein Steuerungswert als eine Zahl zwischen -1 und 1 berechnet werden. Ein Wert von 1 lässt das Fahrzeug maximal nach links steuern, eine -1 dagegen maximal nach rechts, eine 0 bedeutet eine gerade Fahrt. Die Variable

a nimmt einen Wert zwischen 0 und 1 ein, der das Verhältnis zwischen Richtungsabweichung und Pfadabweichung bestimmt. Für $a = 0{,}5$ haben Richtungsabweichung und Pfadabweichung einen gleich großen Einfluss. Damit ergibt sich folgende Berechnung:

$$\text{Steuerungswert} = a \cdot \text{Richtungsabweichung} + (1 - a) \cdot \text{Pfadabweichung}$$

Je nach Anzahl der Tags in Reichweite, Genauigkeit des Kompasses und anderer Parameter kann eine andere Berechnung des Steuerwertes von Vorteil sein.

4.5.4 Praktischer Test

Bei dem praktischen Test wurde das Fahrzeug über die RFID-Matten gesteuert. Die Steuerung geschah per Hand und es sollten verschiedene Wege geschrieben werden.

Der praktische Test zeigte, dass der beschriebene Ansatz das kontinuierliche Folgen ohne Verlust des Pfades ermöglicht. Dabei wurde keine Pfadvorhersage genutzt, sondern das Fahrzeug richtete sich immer gemäß der aktuellen Pfadrichtung und Pfadabweichung aus. Es zeigte sich allerdings auch, dass es bei zu engen Kurven (bei einem Drehradius von weniger als 20 cm) zum Verlust der Pfades kommen kann, weil das Fahrzeug geradeaus über die Kurve hinausfährt. Hier würde eine Pfadvorhersage sinnvoll sein, da sie es dem Fahrzeug erlaubt würde, rechtzeitig die Richtung zu ändern. Bei engen Kurven war außerdem zu beobachten, dass der gleiche Pfad nach verschiedenen Teach-Vorgängen unterschiedlich zuverlässig erkannt wurde. Dies ist darauf zurückzuführen, dass die Tags bei verschiedenen Teach-Vorgängen zu jeweils unterschiedlichen Zeitpunkten erkannt werden und demzufolge jedes Mal andere Daten auf den Tags abgelegt werden. Bei größeren Radien (von mehr als 50 cm) hatten unterschiedliche Teach-Vorgänge dagegen keinen erkennbaren Einfluss auf die Zuverlässigkeit beim Folgen.

Des Weiteren zeigte sich, dass die Benutzung eines Kompasses in Innenräumen zwar möglich ist, aber auch zu Problemen führen kann, wenn er durch andere Gegenstände beeinflusst wird. Als Alternative zu einem magnetischen Kompass steht die Verwendung eines GPS-basierten Kompasses. Dies bietet sich jedoch im wesentlichen im Freien an, kann aber sinnvoll sein, wenn beispielsweise große Elektro-Motoren den Einsatz eines herkömmlichen Kompasses verhindern. Richtungsdaten können ebenfalls von inertialen Messsysteme erhalten werden. Diese erfordern allerdings die Justierung in regelmäßigen Abständen.

4.6 Weitere Möglichkeiten

4.6.1 Benutzung von Odometrie

Eine Verbesserung des vorherigen Ansatzes kann erreicht werden, wenn während der Teach-Phase Odometriedaten erhoben und auf den Tags abgelegt werden. Dabei ist insbesondere die Steuerrichtung und die Drehrate des Fahrzeuges von Interesse, da sie im

Gegensatz zum Kompass (siehe Tabelle 4.2) einen frühzeitigen Aufschluss über den Verlauf des Pfades zulässt. Des Weiteren kann über die Verwendung der Drehrate bestimmt werden, ob Änderungen auf die der Kompass hinweist begründet sind, ob es sich um Messfehler handelt. Die Drehrate kann statt aus Odometriedaten auch von inertialen Messsystemen erhalten werden.

Sowohl der Teach- als auch der Follow-Algorithmus funktionieren wie zuvor, jedoch wird die Drehrate zusätzlich vom Steueralgorithmus genutzt. Dieser Ansatz wurde in [Man09] erfolgreich getestet.

Die Benutzung der Informationen über die zurückgelegte Strecke kann Vorteile bieten, wenn der zu erwartende Abstand zwischen den Tags bekannt ist. Der Steueralgorithmus kann dann unabhängig vom Inventory-Algorithmus bestimmen, ob der Pfad verloren wurde, wenn eine zu lange Strecke keine neuen Informationen vom Folgeprozess geliefert wurden.

	Kompass	Drehrate
Auskunft Fahrtrichtung	momentan	zukünftig
Langfristige Genauigkeit	+	-
Kurzfristige Genauigkeit	-	+

Tabelle 4.2: Vergleich der Merkmale von Kompass und Drehrate

4.6.2 Rückwärts-Schreiben

Eine Variante der Teach-Phase besteht darin, den Pfad in der umgekehrten Richtung abzufahren. Dies wurde ebenfalls in [Man09] getestet. Durch die umgekehrte Richtung besteht die Möglichkeit, das Schreiben der momentanen Pfadinformationen hinauszuzögern und die Fahrzeuge beim Following somit früher über den zukünftigen Pfadverlauf informieren zu können. Fahrtrichtung und Drehrichtung müssen ebenfalls jeweils gedreht werden, der Follow-Algorithmus kann wie zuvor bestehen bleiben.

4.6.3 Benutzung von Antennen-Arrays

Die Benutzung von zwei Lesegeräten erlaubt nur eine sehr eingeschränkte Lokalisierung der Tags. Ist eine höhere Genauigkeit erwünscht, die damit in der Folge auch zu einer höheren Genauigkeit bei der Fahrzeugpositionierung in der Follow-Phase führt, können Antennen-Arrays genutzt werden. In solchen Systemen nutzt ein Lesegerät eine Vielzahl von Spulen. Mit ihrer Hilfe kann bestimmt werden, wo das stärkste Antwortsignal empfangen wird und sich das Tag folglich befindet. Ein solches System wird beispielsweise in [NBOF06] verwendet. Bei der Verwendung eines Antennenarrays mit bis zu 32 Spulen und einer Antennenbreite von 500 mm kann die Position eines Tags quer zur Fahrbahn bis auf eine Genauigkeit von ±5 mm bestimmt werden [Gö10].

5 Inventory für Navigationssysteme

Die im vorherigen Kapitel vorgestellten Antikollisionsverfahren wurden nicht in Hinblick auf Navigationsverfahren entwickelt und führen dort zu niedrigen Fahrgeschwindigkeiten. Daher wird im Folgenden untersucht, wie sich die Algorithmen für diesen Verwendungszweck optimieren lassen. Des Weiteren wird ein neuer Ansatz besprochen, der über die Speicherung von Masken ein effizienteres Verfahren verspricht.

5.1 Anforderungen an das RFID-System für eine schnelle Tag-Detektion

Um die Kommunikation des Lesegerätes mit einem von mehreren RFID-Tags in Reichweite zu ermöglichen, muss ihm dessen ID bekannt zu sein. Da dies normalerweise nicht gegeben ist, werden in einem ersten Schritt alle Tags in Reichweite identifiziert. Zu diesem Zweck sendet der Reader ein Inventory-Kommando, welches die Tags veranlasst, mit ihrer ID zu antworten. Da es den Tags nicht möglich ist, untereinander zu kommunizieren, führt die gleichzeitige Übertragung der Antworten mehrerer Tags zu Kollisionen, welche die Identifizierung der Tags zu diesem Zeitpunkt verhindern.

Ein Antikollisionsalgorithmus versucht diese Kollisionen aufzulösen. Hierfür existieren verschiedene Ansätze, die diese Aufgabe unterschiedlich schnell lösen. Sie eint, dass mit steigender Tag-Zahl der Zeitaufwand ebenfalls steigt. Die Verwendung von RFID in mobilen Anwendungen führt zu erhöhten Anforderungen an die Kommunikationsgeschwindigkeit, da sich das Lesegerät bewegt und sich die Tags nur einen beschränkten Zeitraum in Reichweite befinden. Die mögliche Fahrgeschwindigkeit eines FTS hängt eng mit der erreichbaren Kommunikationsgeschwindigkeit zusammen. Daher ist die Verfügbarkeit effizienter Antikollisionsalgorithmen von besonderer Bedeutung [BT09].

5.1.1 Beispiel: Die Ermittlung eines einzelnen Tags

Für die Ermittlung eines einzelnen Tags, welches zentral auf dem Pfad des Readers platziert ist, bei einer realistischen Detektionszeit von $t_{\text{inv}} = 0{,}030\,s$ (bei ISO-15693) sowie einer Reichweite von $r = 0{,}15\,m$ ist eine maximale Fahrgeschwindigkeit von

$$\begin{aligned} v &= \frac{2r}{t_{\text{inv}}} \\ &= \frac{0{,}30\,m}{0{,}030\,s} \\ &= 10\,m/s \end{aligned}$$

möglich, um das Tag noch erkennen zu können. Dies gilt jedoch nur dann, wenn der Beginn des Inventory-Vorgangs mit dem Zeitpunkt zusammenfällt, in dem das Tag in die Reichweite des Lesegerätes gelangt. Da dies unwahrscheinlich ist, folgt aus dem Nyquist-Shannon-Abtasttheorem (siehe Abbildung 5.1), dass zumindest zwei Detektionsdurchläufe notwendig sind, um ein Tag mit Sicherheit zu entdecken (allerdings ohne zu berücksichtigen, dass Kommunikationsfehler auftreten können). Daher reduziert sich die Geschwindigkeit auf

$$v_{\mathrm{s}} = \frac{r}{t_{\mathrm{inv}}}$$

und somit für das Beispiel auf $5\,m/s$.

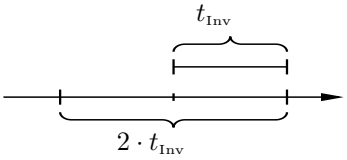


Abbildung 5.1: Befindet sich ein Tag länger als $2 \cdot t_{\mathrm{Inv}}$ in Reichweite des Lesegerätes, ist die Detektion sicher möglich, unabhängig davon, zu welchem Zeitpunkt der Inventory-Durchlauf startet

5.1.2 Ermittlung vieler Tags

Die maximale Fahrgeschwindigkeit reduziert sich weiterhin, wenn sich mehrere Tags innerhalb des Lesebereichs befinden. Die Anzahl der Anfragen, um n Tags mit einem einfachen Antikollisionsalgorithmus und bei fortlaufenden Tag-IDs zu entdecken, kann als die Anzahl der Knoten eines balancierten Binärbaumes berechnet werden (Abbildung 5.2). Die Höhe h eines solchen Baumes ist gegeben durch [Meh88]:

$$h = \log_2(n)$$

Die Anzahl der Knoten k berechnet sich als [Meh88]:

$$\begin{aligned} k &= 2^{h+1} - 1 \\ &= 2^{\log_2(n)+1} - 1 \end{aligned}$$

Für die Überprüfung eines jeden Knoten wird t_{inv} angenommen, so dass die Dauer eines kompletten Antikollisionsdurchlaufs bei n Tags wie folgt abgeschätzt werden kann:

$$\begin{aligned} t_{\mathrm{inv}}(n) &= k \cdot t_{\mathrm{inv}} \\ &= \left(2^{\log_2(n)+1} - 1\right) t_{\mathrm{inv}} \end{aligned}$$

Die maximale Geschwindigkeit bei $t_{\mathrm{inv}} = 0{,}030\,s$ und $r = 0{,}15\,m$ für $n = 4$ Tags reduziert

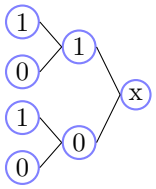


Abbildung 5.2: Balancierter Baum mit vier Tags, die Höhe beträgt 2, werden alle 7 Knoten abgelaufen, kann jedes Tag erkannt werden

sich somit auf:

$$\begin{aligned} v_{\mathrm{s}}(n) &= \frac{r}{t_{\mathrm{inv}}(n)} \\ v_{\mathrm{s}}(4) &= \frac{r}{\left(2^{\log_2(4)+1}-1\right) t_{\mathrm{inv}}} \\ &= \frac{0{,}15\,m}{7 \cdot 0{,}030\,s} \\ &\approx 0{,}71\,m/s \end{aligned}$$

Diese deutlich geringere Geschwindigkeit zeigt, dass andere Inventory-Ansätze nötig sind, um höhere Fahrgeschwindigkeiten zu erreichen. Zusätzlich zu der reinen Detektion muss die Zeit berücksichtigt werden, die erforderlich ist, um die eigentlichen Nutzdaten aus den Tags zu lesen. Des Weiteren darf nicht davon ausgegangen werden, dass sich alle Tags in der Pfadmitte befinden.

5.1.3 Ermittlung von Tags in Randbereichen

Da sich der Reader bewegt und das Lesefeld als kreisförmig angenommen werden kann, hängt die Strecke x, auf der sich ein Tag innerhalb der Reichweite des Readers befindet, vom Versatz y zur Pfadmitte ab (siehe Abbildung 5.3). Damit zusammenhängend ändert sich auch die Zeitdauer, die für Detektion und Kommunikation zur Verfügung steht.

Die Strecke x kann berechnet werden als:

$$\begin{aligned} r^2 &= \left(\frac{x}{2}\right)^2 + y^2 \\ x &= 2\sqrt{r^2 - y^2} \end{aligned}$$

Somit reduziert sich die erreichbare Fahrgeschwindigkeit weiterhin. Durch die Erweiterung des Radius könnte zwar eine längere Zeitspanne erzielt werden, während der sich die Tags in Reichweite befinden und die Detektion möglich ist. Auf der anderen Seite steigt mit einer größeren Reichweite aber auch die Anzahl der Tags, die detektiert werden müssen, sofern die Tag-Dichte unverändert bleibt. Daher liefert dieses Vorgehen nicht zwangsläufig einen Vorteil. Auch dies unterstreicht die besondere Bedeutung beschleunigter Antikollisionsalgorithmen, die insbesondere dann nötig sind, wenn Tags aus Randbereichen ebenfalls erkannt werden sollen.

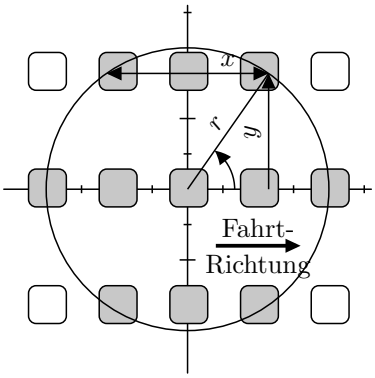


Abbildung 5.3: RFID-Tags in Reichweite des Lesegerätes

Schritt	Maske	Antwort	Bedeutung
1	ø	`1x1x xxx0 xxxx`	Kollisionen, Erweiterung der Maske
2	`0`	`101x 1010 xx10`	Kollisionen, Erweiterung der Maske
3	`00`	ø	keine Antwort
4	`10`	`101x 1010 xx10`	Kollisionen, Erweiterung der Maske
5	`010`	`1011 1010 1010`	**Tag 4 erkannt**
6	`110`	`1010 1010 0110`	**Tag 1 erkannt**
7	`1`	`1x1x 01x0 xxx1`	Kollisionen, Erweiterung der Maske
8	`01`	`1010 0100 0101`	**Tag 2 erkannt**
9	`11`	`1111 0110 1011`	**Tag 3 erkannt**

Tabelle 5.1: Vollständiges Inventory mit stdRec

5.2 Verbesserung bestehender Algorithmen

Im Folgenden werden Verbesserungen zu Algorithmen aus Abschnitt 3.5.3 untersucht. Auf der einen Seite handelt es sich um Modifikationen, die das Inventory für traditionelle Anwendungen verbessern, aber andererseits werden auch neue Möglichkeiten in Hinblick auf RFID-Navigation aufgezeigt.

5.2.1 Leere Zweige auslassen

Eine erste sehr offensichtliche Verbesserung besteht darin, Anfragen zu überspringen, die zu keinen neuen Erkenntnissen führen. In Abschnitt 3.7.2 (siehe auch Tabelle 5.1) wird im dritten Schritt die Folge `00` gesendet, ohne dass es zu Antworten von Tags kommt. Ein genauerer Blick auf die Antwort im vorherigen Schritt 2 zeigt, dass in den letzten beiden Bits `xx10` keine Kollisionen zu verzeichnen sind und es sich bei dem vorletzten

Bit um eine 1 handelt. Aus diesem Grunde kann Schritt 3 ausgelassen werden, welcher auf Tags prüft, die bekanntermaßen nicht existieren.

5.2.2 Zur Position der Kollision springen (Algorithmus 6: JTC)

Die zweite Verbesserung ähnelt der vorherigen und lässt Anfragen aus, die vorhersehbar zu bereits bekannten Ergebnissen früherer Anfragen führen. Dies kann im vierten Schritt beobachtet werden (Tabelle 5.1), welcher zu der gleichen Antwort führt wie Schritt 2. Seit Schritt 2 ist bekannt, dass die letzten beiden Bits kollisionsfrei sind. Insofern ist es unnötig diese beiden Bits erneut zu überprüfen. Schritt 4 kann somit ebenfalls ausgelassen werden.

Algorithmus 6 gibt aufbauend auf stdRec einen erweiterten Algorithmus an, der leere Zweige auslässt und direkt zur Kollisionsposition springt.

Algorithm 6: JTC

```
Data: mask
response = sendRequest(mask)
if Antwort hat Kollisionen then
    mask = unterere kollisionsfreie Bitfolge der Antwort
    JTC(0 + mask)
    JTC(1 + mask)
else
    if Antwort enhält ID then
        return ID
    end
end
```

5.2.3 Nicht mit leerer Maske starten (Algorithmus 7: Q2 bzw. Qx)

Kann angenommen werden, dass sich mehrere Tags in Reichweite des Lesegerätes befinden, ist es sinnvoll den Algorithmus nicht mit einer leeren Maske beginnen zu lassen, da diese Maske mit einer hohen Wahrscheinlichkeit zu einer kollisionsbehafteten Antwort führen wird. Kann eine Gleichverteilung der Tag-IDs angenommen werden, so ist die Wahrscheinlichkeit einer Kollision im ersten Bit bei n Tags gegeben durch:

$$\begin{aligned} P(\text{Kollision}) &= \overline{P(\text{keine Kollision})} \\ &= \overline{P(\text{alle ersten Bits} = 0) \vee P(\text{alle ersten Bits} = 1)} \\ &= 1 - (0{,}5^n + 0{,}5^n) \\ &= 1 - 0{,}5^{n-1} \end{aligned}$$

Somit liegt die Wahrscheinlichkeit, dass nur drei Tags im ersten Bit kollidieren, bereits bei bei 75%. Daher ist es bei drei Tags sinnvoll, einen Algorithmus mit den Masken 0

und `1` zu starten. In Abhängigkeit davon, wie viele Tags erwartet werden können, kann die Länge der initialen Masken erweitert werden. Die Nomenklatur sieht vor, dass das x von Qx für die Länge der initialen Maske steht.

Bei einer Länge von zwei Bits würden die Masken `00`, `01`, `10`, `11` verwendet, wie es in Algorithmus 7 angewandt wird. Als eigentlicher Antikollisionsalgorithmus (`antikollision()`) kann ein beliebiger Algorithmus zum Einsatz kommen, der Kollisionen behandelt.

Algorithm 7: Q2

```
startMasks = {00, 01, 10, 11}
foreach mask ∈ startMasks do
    antikollision(mask)
end
```

Selbstverständlich führt dieser Ansatz unter Umständen zu Fehlanfragen, wenn kein Tag zu den anfänglichen Anfragen passt. Auf diese Problematik geht der folgende Abschnitt 5.2.4 näher ein. Es sollte jedoch beachtet werden, dass die Ausführung einer Anfrage, die zu keiner Antwort führt, schneller ist, als eine, auf die Tags antworten. Im ersten Fall müssen keine IDs empfangen werden, und der Algorithmus kann sofort mit der nächsten Anfrage fortfahren.

Die Verwendung einer initialen Maske bietet auch dann Vorteile, wenn die unteren Bit-Positionen der Tag-IDs mit einer hohen Wahrscheinlichkeit identisch sind. Dies kann der Fall sein, wenn diese Bits Meta-Informationen wie Hersteller o.ä. enthalten [CSK08].

5.2.4 Tag-IDs/Distribution berücksichtigen

Es kann beobachtet werden, dass die Inventory-Geschwindigkeit nicht nur von der Anzahl der Tags abhängt, sondern auch von der Struktur der verwendeten Tags-IDs. Während die Tag-IDs der Gruppe

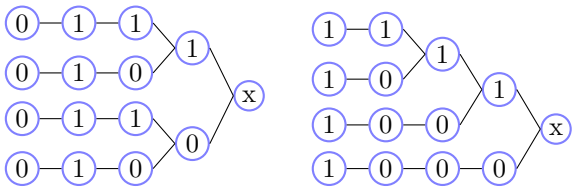
Tag 1: `0111` Tag 2: `0101`
Tag 3: `0110` Tag 4: `0100`

in einem balancierten Baum resultieren (siehe Abbildung 5.4(a)) und nur kurze Maskenlängen von maximal 2 Bits erfordern, formen die folgenden Tag-IDs (siehe auch Abbildung 5.4(b)) einen unbalancierten Baum, der längere Masken erforderlich macht:

Tag 1: `1111` Tag 2: `1011`
Tag 3: `1001` Tag 4: `1000`

Auf der anderen Seite kann ein erstes Tag (`1000`) der zweiten Gruppe sehr schnell nur durch die Übertragung einer Ein-Bit-Maske (`0`) erkannt werden.

Für den Q2-Algorithmus stellt die Verteilung aus Abbildung 5.4(a) den Optimalfall dar. Mit jeder Anfrage kann augenblicklich ein Tag erkannt werden, ohne dass weitere



(a) Balanciert (b) Unbalanciert

Abbildung 5.4: Verschiedene ID-Bäume

Antikollisionsschritte nötig sind. In Kombination mit der twoAtOnce-Erweiterung (bei der zwei Tags bei einer verbleibenden Kollision gleichzeitig detektiert werden) wäre Q1 in diesem Falle sogar noch schneller, da sich die IDs in jedem Zweig nur um ein Bit unterscheiden und somit mit zwei Anfragen alle vier Tags erkannt werden könnten.

Abbildung 5.4(b) stellt dagegen einen ungünstigen Fall für Q2 dar, da nur zwei Anfragen (`01` und `00`) direkt ein Tag liefern, `10` führt zu keinem Tag und `11` liefert eine Kollision, die aufgelöst werden muss. Ein noch problematischerer Fall ergibt sich mit der Tag-Verteilung aus Abbildung 5.5. Keine der Anfragen von Q2 liefert direkt ein Tag und drei der vier Anfragen (`01`, `10`, `11`) führen sogar nicht einmal zu einer Antwort.

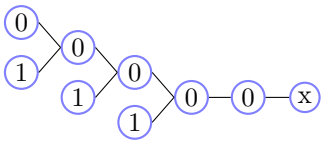


Abbildung 5.5: Ungünstiger Fall für Q2; in der Abbildung wurden nur die relevanten Bits für das Inventory (die identifizierende Maske) angegeben

Bei Navigationsanwendungen wird das Inventory immer für stationäre und umliegende Tags durchgeführt. Um die Inventory-Geschwindigkeit zu erhöhen, kann diese Erkenntnis ausgenutzt werden, indem nur Tags nebeneinander platziert werden, deren Tag-IDs mit möglichst wenig Schritten detektiert werden können. Außerdem ist es möglich IDs so auszuwählen, dass sie sich in möglichst wenig Bits unterscheiden, um so die Geschwindigkeit gemäß Abschnitt 3.7.3 zu erhöhen. Entscheidet man sich für diese Option, schränkt man allerdings die Möglichkeit ein, bei Einrichtung des Tag-Feldes die Tags zufällig zu verteilen.

5.3 Speicherung von Teil-IDs für Pfadverläufe

In diesem Abschnitt wird eine neue Möglichkeit vorgestellt, wie bei RFID-Navigationssystemen der Umstand stationärer Tags ausgenutzt werden kann [BTZ12].

Ein Antikollisionsalgorithmus verbraucht den Großteil seiner Zeit damit, nach den identifizierenden Masken zu suchen. Offensichtlich könnte das Inventory beschleunigt werden oder sogar in seiner bisherigen Form wegfallen, wenn die identifizierenden Masken bereits bei Beginn des Inventorys bekannt wären. In diesem Falle könnten diese Masken –

ähnlich wie bei dem Ansatz *Masken erinnern* (Abschnitt 3.7.5) – direkt gesendet werden, um so an die vollständige ID zu gelangen. Deshalb sollen im folgenden Ansatz die identifizierenden Masken der Tags, die als nächstes detektiert werden sollen, auf den jeweils vorherigen Tags gespeichert werden. Neben einem schnelleren Inventory hat dieser Ansatz zum Vorteil, dass von Beginn an nur die Tags erkannt werden, die auch zu dem gewünschten Pfad gehören und alle anderen Tags nicht mehr erkannt und ausgelesen werden müssen.

Durch die Benutzung der identifizierenden Maske ist es möglich, ein Tag zu identifizieren, ohne die vollständige ID speichern zu müssen. Es wäre zwar auch denkbar, die ID in den Tags abzulegen, aber zum einen würde dies erheblich mehr Speicherplatz verbrauchen und zum zweiten würde die Übertragung der (im Vergleich zu den Masken sehr langen) ID die Fahrgeschwindigkeit wieder verringern.

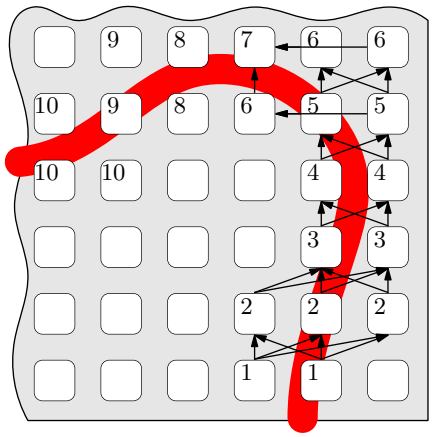


Abbildung 5.6: Ansatz zur Speicherung von Teil-IDs für einen PFID-Pfad

Dieser Ansatz ist nur dann praktikabel, wenn während der Teaching-Phase der folgende Verlauf und die folgenden Tag-IDs der Strecke bekannt sind, damit die nötigen Daten in die entsprechenden Tags geschrieben werden können.

Abbildung 5.6 zeigt beispielhaft eine Anwendung dieses Ansatzes. Der Pfad ist in 10 Schritte unterteilt. Alle Tags, die zum selben Schritt gehören, speichern die gleichen identifizierenden Masken, die es erlauben, die Tags des nächsten Schrittes zu erkennen. Beispielsweise zeigen alle Tags des zweiten Schrittes auf die des dritten. Das bedeutet, dass alle Tags des zweiten Schrittes zwei Masken speichern, um die beiden Tags des dritten Schrittes zu identifizieren.

5.3.1 Identifizierende Maske

Im Gegensatz zu der vollständigen ID, die in jedem Fall eindeutig ist, bezieht sich die Eindeutigkeit der identifizierenden Maske nur auf eine bestimmte Menge von Tags, die festgelegt werden muss. Ändert sich diese Menge, ändern sich auch die identifizierenden Masken (vergleiche Abschnitt 3.7.1.3). Die Tag-Menge wird bei stationären Tags durch

den Bereich festgelegt, in dem die Maske gültig sein soll. Dieser Bereich wiederum wird durch die Reichweite des Lesegerätes bestimmt, welche dadurch eine große Relevanz bei diesem Ansatz hat.

Des Weiteren muss man sich mit der Frage beschäftigen, zu welchen Zeitpunkt erwartet werden kann, dass ein Tag detektiert wird. Je größer der Bereich oder die Menge der Tags ist (die auch durch die Dichte beeinflusst wird), desto länger wird auch die identifizierende Maske sein.

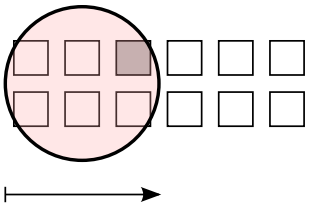


Abbildung 5.7: Sofortige Detektion eines Tags, wenn es in die Reichweite des Lesegerätes gelangt. Das Lesegerät bewegt sich von links nach rechts, das erkannte Tag ist hervorgehoben, der Kreis beschreibt die Reichweite des Lesegerätes.

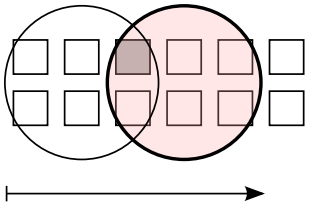


Abbildung 5.8: Detektion eines Tags zu einem beliebigen Zeitpunkt, während es sich in Reichweite befindet. Die beiden Kreise stellen die möglichen Extrempositionen der Lesebereiche dar.

Optimalfall: Im optimalen Fall (Abbildung 5.7) wird ein Tag genau in dem Augenblick detektiert, wenn es in den Lesebereich des Readers eintritt. Dies bedeutet, dass die Maske nur innerhalb des momentanen Lesebereiches eindeutig sein muss.

Durchschnittlicher Fall: Es ist jedoch wahrscheinlicher, dass ein Tag zu einem unbestimmten Zeitpunkt detektiert wird, während es sich in Reichweite des Readers befindet (Abbildung 5.8). In diesem Fall muss die Maske innerhalb des gesamten Bereiches eindeutig sein, die der Reader abdecken könnte, innerhalb dessen das Tag in Reichweite ist.

Ungünstigster Fall: Ebenso kann es sein, dass ein Tag nicht erkannt wird, während es sich in Reichweite des Readers befindet. In diesem Fall wird die benutzte Maske dazu führen, dass ein nicht gewünschtes Tag oder gar kein Tag erkannt wird. Um zu vermeiden, dass ein falsches Tag detektiert wird, müsste die Maske innerhalb eines unbestimmten, großen Areals gültig sein. Da dies nicht sinnvoll ist, sollte

dieser Fall verhindert werden, was durch die Verringerung der Fahrgeschwindigkeit erreicht werden kann.

5.3.2 Weitere Möglichkeiten

In früheren Ansätzen wurde eine Pfad-Identifikationsnummer benutzt, um herauszufinden, ob ein Tag zum verfolgten oder zu einem anderen Pfad gehört [BVT08, MZ05]. Wenn jedes Tag nur einen Datensatz zu einem Pfad speichert, kann in Betracht gezogen werden, auf die Pfad-Identifikationsnummer zu verzichten, da der neue Ansatz von vornherein nur die relevanten Tags liefert und keine nachträgliche Unterscheidung anhand der Pfad-Identifikation erfordert. Es muss jedoch angemerkt werden, dass diese Form der Nutzung nicht von hoher Robustheit zeugt. Fielen beispielsweise die beiden Tags im dritten Schritt (Abbildung 5.6) aus, wäre es nicht mehr möglich den Pfad weiterzuverfolgen. Daher kann diese Möglichkeit nur genutzt werden, wenn eine hohe Redundanz gegeben ist und die Detektion eines Tags des folgenden Schrittes garantiert werden kann. Allerdings könnte die Einbringung neuer Tags dazu führen, dass eine Maske nicht mehr eindeutig ist und fälschlicherweise die Daten anderer Tags eines anderen Pfades genutzt würden, so dass das nachträgliche Hinzufügen von Tags nicht erlaubt werden dürfte.

Da die Benutzung der Masken der folgenden Tags bereits zu einer natürlichen Ordnung führt, wäre es des Weiteren möglich, die Sequenznummer wegzulassen. Auf der anderen Seite kann dies zu Komplikationen führen, wenn Tags für mehrere Datensätze genutzt werden sollen und sich selbstkreuzende Pfade erlaubt sind. In diesem Falle könnte nicht ohne weiteres unterschieden werden, welcher Datensatz der richtige ist.

5.3.3 Herausforderungen

Nicht nur die Detektion von Tags benötigt Zeit, auch das Lesen der Tag-Daten muss berücksichtigt werden. Im Allgemeinen gilt, dass die Übertragung von mehr Daten mehr Zeit benötigt. Dieses Prinzip trifft in dieser Form bei ISO-15693-Tags möglicherweise nicht zu, da deren Daten blockweise organisiert sind. Das bedeutet, dass immer nur ein ganzer Block (bestehend aus mehreren Bytes) angesprochen und übertragen werden kann, unabhängig davon, wie viele Bits oder Bytes man tatsächlich benötigt. Dennoch kann man davon ausgehen, dass das Ablegen langer und vieler Masken dazu führen wird, dass mehr als ein Block genutzt wird. Des Weiteren muss beachtet werden, dass der Speicherplatz von Tags begrenzt ist und unter Umständen auch andere Informationen gespeichert werden müssen.

5.4 Algorithmus

Im Folgenden wird ein konkreter Algorithmus für das Inventory entwickelt, der zuvor beschriebene Ansätze umsetzt. Er basiert im wesentlichen auf der Maskenspeicherung, aber nutzt nötigenfalls die anderen Optimierungen für die Durchführung vollständiger Antikollisionsläufe.

5.4.1 Vorüberlegungen

Wenn der Algorithmus der Maskenspeicherung (MS) über keinerlei Informationen über die Einrichtung des Tag-Feldes oder die momentane Geschwindigkeit des Lesegerätes verfügt, entsteht das Problem, dass der Algorithmus entscheiden muss, was geschieht, wenn keine Tags erkannt wurden. Dieser Umstand kann zwei Ursachen haben:

Fahrzeug ist zu wenig gefahren (aufgrund geringer Geschwindigkeit) Dies bedeutet, dass die Maske des nächsten Schrittes (noch) nicht gültig ist und kann die folgenden Konsequenzen nach sich ziehen:

- Tags antworten nicht, weil sich die erwarteten Tags noch nicht in Reichweite befinden.
- Kollisionen entstehen; allerdings nur dann, wenn der Bereich, in dem die Maske eindeutig sein soll, zu klein gewählt ist und alte, zuvor detektierte Tags im aktuellen Antikollisionsdurchlauf erneut teilnehmen.

Fahrzeug ist zu weit gefahren (aufgrund hoher Geschwindigkeit) Dies bedeutet, dass die Maske nicht (mehr) gültig ist und kann zu folgenden Fällen führen:

- Tags antworten nicht, da vollständig andere Tags involviert sind und sich das ursprünglich erwartete Tag nicht mehr in Reichweite befindet.
- Kollisionen entstehen, da andere Tags nun ebenfalls zu dieser Maske passen (sofern der Bereich, in dem die Maske eindeutig sein soll, zu klein gewählt wurde) oder vollständig andere Tags involviert sind und das ursprünglich erwartete Tag sich nicht mehr in Reichweite befindet.

Im ersten Fall, wenn sich der Reader noch nicht weit genug vorwärts bewegt hat, sollte der Algorithmus idealerweise die gleiche Maske im nächsten Durchgang erneut nutzen. Hat sich das Fahrzeug dagegen inzwischen zu weit bewegt und das Tag befindet sich nicht mehr in Reichweite, müsste ein komplettes, nicht MS-basiertes Inventory durchgeführt werden.

Der MS-Algorithmus ist nur anhand der Daten, die das Inventory liefert, jedoch nicht in der Lage zu entscheiden, welcher Fall eingetreten ist. Um die gleiche Maske wiederholt zu benutzen, müsste sichergestellt sein, dass das Fahrzeug tatsächlich noch nicht weit genug gefahren ist und die verwendete Maske absehbar zu einer Antwort führen wird. Anderenfalls können folgende Tags unter Umständen nicht erkannt werden.

Wenn Kollisionen auftraten, könnte der MS-Algorithmus in beiden Fällen alternativ auch diese mit einem Standard-Antikollisionsalgorithmus lösen.

5.4.2 Mask-storing Algorithmus (Algorithmus 8: MS)

Der im Folgenden vorgestellte Antikollisionsalgorithmus geht im Zweifelsfalle vom zweiten Fall aus (das Fahrzeug ist zu weit gefahren), da das Interesse dem Verhalten bei hohen Geschwindigkeiten gilt. Tritt der erste Fall ein, wird dieser Ansatz ebenfalls funktionieren, jedoch eine geringere Detektionsrate zeigen.

Algorithm 8: MS

```
if not isEmpty(identMasks) then
    startMasks = identMasks
else
    startMasks = {}
end
foreach startMask ∈ startMasks do
    foundTags += anticollision(startMask)
end
clear(identMasks)
foreach foundTag ∈ foundTags do
    if isOld(foundTag) then
        continue
    end
    readData(foundTag)
    if belongsToPath(foundTag) then
        nextMasks = extractMasksFromData()
        if isNewData(nextMasks) then
            identMasks += nextMasks
        end
    end
end
```

Der MS-Algorithmus (Algorithmus 8) funktioniert wie folgt. Wenn der Algorithmus zum ersten Mal startet oder der vorherige Durchlauf keine identifizierenden Masken (`identMasks`) geliefert hat (`isEmpty()`), wird `startMasks` als leere Maske gesetzt. Anderenfalls – und das ist der erwünschte Fall – werden die zuvor gefundenen Masken als `startMasks` gesetzt. Für jede der `startMasks` wird ein gängiger Antikollisionsalgorithmus (`anticollision()`) genutzt, um die komplette ID zu bestimmen. Für den Fall, dass die `startMask` eine gültige identifizierende Maske darstellt, sollte der Algorithmus sofort in der Lage sein, das Tag zu identifizieren. Wenn dieses aufgrund von Kollisionen nicht möglich ist (z.B. auch dann, wenn eine leere Maske als Startmaske genutzt wurde), werden die Kollisionen unter Berücksichtigung der zuvor besprochenen Optimierungen behandelt.

Alle gefundenen Tags werden in dem Feld `foundTags` gespeichert. Nach diesem Schritt wird `identMasks` geleert, da dieses Feld genutzt wird, um die neu gefundenen identifizierenden Masken zu speichern.

Als nächstes wird jedes gefundene Tag einzeln behandelt. Zuerst wird überprüft, ob das Tag bereits in einem vorherigen Durchlauf genutzt wurde (`isOld()`). In diesem Falle wird mit dem nächsten Tag fortgefahren, da nur neue Informationen von Wert sind. Außerdem gilt, dass in dem Falle, dass die Daten eines solchen Tags von Interesse wären, die Daten nicht noch mal ausgelesen werden müssten und aus einen Cache geladen werden könnten.

Nach dem Auslesen der Tag-Daten wird entschieden, ob das Tag zu dem gewünschten Pfad gehört (`belongsToPath()`), und die gespeicherten Masken werden im positiven Falle extrahiert (`extractMasksFromData()`).

Abschließend wird geprüft, ob die gefundenen Masken nicht zu Tags gehören, die bereits im aktuellen Durchlauf gefunden wurden (`isNewData()`). Dieser Fall kann eintreten, wenn das Lesegerät Tags mehrerer verschiedener Schritte erkennt (Abbildung 5.9). In diesem Fall würden Masken früherer Schritte auf bereits gefundene Tags zeigen. Diese Filterung ist insbesondere wichtig, wenn ein Durchlauf ohne bekannte identifizierenden Masken gestartet wurde.

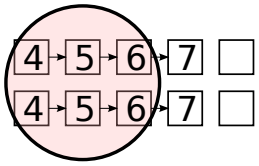


Abbildung 5.9: Wenn der Reader Masken verschiedener bzw. alter Schritte liest (Schritt 4 oder 5), müssen die Daten gefiltert werden. Die Daten der Tags der Schritte 4 und 5 liefern die Masken für den fünften bzw. sechsten Schritt. Von Interesse sind aber die Masken, die in den Tags von Schritt 6 gespeichert sind, da nur sie die unbekannten Tags des siebten Schrittes liefern.

Der benutzte Standard-Antikollisionsalgorithmus sollte dahingehend verändert werden, dass er stoppt, sobald das erste Tag gefunden wurde und nicht versucht wird, alle Tags in Reichweite zu finden. Dies verbessert das Startverhalten, da die initiale Detektion und das Lesen der Daten aller erreichbaren Tags viel Zeit beansprucht. Nachdem das erste Tag gefunden wurde, kann der Algorithmus so von seiner Stärke profitieren, die folgenden Tags effizient zu finden.

6 Evaluation der Algorithmen für sequentielles Inventory

In diesem Kapitel wird analysiert, wie sich die zuvor besprochenen Algorithmen tatsächlich verhalten. Dazu wird als erstes ein genauerer Blick auf die möglichen ID-Verteilungen geworfen und die Frage beantwortet, wie balancierte und unbalancierte ID-Bäume erzeugt werden können. Diese Bäume werden im weiteren Verlauf bei den Simulationen benutzt.

In Unterkapitel 6.2 wird die Detektionswahrscheinlichkeit eines Tags bei einem mobilen Lesegerät berechnet. Das Ergebnis dieser Berechnung ist nicht unmittelbar notwendig für die Evaluation, soll aber das Verständnis der Ergebnisse erleichtern.

In Unterkapitel 6.3 werden Simulationsergebnisse der in Kapitel 5 besprochenen Algorithmen vorgestellt und im darauffolgenden Unterkapitel 6.4 bewertet.

6.1 Tag-ID-Verteilung

Die Verteilung der Tag-IDs hat einen wesentlichen Einfluss auf den Ablauf und die Möglichkeiten des Inventorys [BT]. Bei den Tag-Distributionen können die folgenden Fälle grob unterschieden werden:

Inkrementierte Tag-IDs: Die Tag-IDs werden systematisch so verteilt, dass sich aufeinanderfolgende IDs um den Wert 1 unterscheiden. Dies führt zu einem balancierten Baum, der beim Inventory kurze Masken zulässt. Durch die Inkrementierung erhält man ein hohes Wissen über die übrigen Tags, sobald ein erstes Tag detektiert ist. Des Weiteren führt das Hochzählen von IDs dazu, dass sich die höheren Bitpositionen nicht unterscheiden und somit weniger Kollisionen auftreten, was die gleichzeitige Detektion zweier Tags ermöglicht (vergleiche Abschnitt 3.7.3).

Unbalancierte Tag-IDs: Die IDs sind so gewählt, dass ein stark unbalancierter ID-Baum entsteht. Dies hat Masken unterschiedlicher Länge zur Folge, die im Schnitt länger sind. Es kann aber auch eine Anzahl Masken entstehen, die sehr kurz sind und so die schnelle Detektion der zugehörigen Tags erlauben. Für einen solchen Baum müssen sich die IDs an den niedrigeren Bitpositionen gleichen und auf möglichst verschiedener Höhe vom Hauptzweig abzweigen.

Zufällige Tag-IDs: Durch die Wahl zufälliger Tag-IDs entsteht mit hoher Wahrscheinlichkeit ebenfalls ein Baum, der einem balancierten Baum ähnlich sieht.

6.1.1 Balancierte Bäume mit dem Gilbert-Modell

Das Gilbert-Modell kann genutzt werden, um eine Verteilung der Tag-IDs über zwei Parameter (p und q) zu steuern [KYK08]. Die IDs werden hochgezählt und für jede ID wird entschieden, ob sie für ein Tag ausgewählt oder übersprungen wird. Das Gilbert-Modell erlaubt so die Generierung unterschiedlich gut balancierter Bäume, bei denen die identifizierenden Knoten größtenteils auf der gleichen Höhe liegen. Für das Erstellen unbalancierter Strukturen mit sehr unterschiedlichen Maskenlängen ist das Modell nicht geeignet.

Zu Beginn befindet sich das Modell im Zustand *Tag* (Abbildung 6.1), was bedeutet, dass die ID für ein Tag ausgewählt wird. Mit der Wahrscheinlichkeit p wird auch die folgende ID genommen und das Modell bleibt in dem Zustand *Tag*. Mit der Wahrscheinlichkeit $1-p$ wird dagegen der Zustand zu *NoTag* gewechselt und die ID wird ausgelassen. Der Wert von q bestimmt die Wahrscheinlichkeit, dass nach einer nicht-gewählten ID die folgende ID ebenfalls nicht gewählt wird. Anschaulich bestimmt p damit die durchschnittliche Menge aufeinanderfolgender ausgewählter IDs und q den durchschnittlichen Abstand zwischen diesen.

Tabelle 6.1 zeigt eine Übersicht, welche Auswirkungen mögliche Kombinationen von p und q haben. Weisen sowohl p als auch q einen hohen Wert auf, führt das dazu, dass die IDs eher gruppenweise ausgewählt werden. Der Abstand zwischen diesen Gruppen entspricht der Gruppengröße. Werden p und q reduziert, werden die Gruppen sowie der Abstand dazwischen kleiner, so dass bei $p = 5$ und $q = 5$ die IDs abwechselnd ausgewählt und übersprungen werden. Sind p und q asymmetrisch gewählt, führt ein großer Wert von p erneut zu einer hohen Konzentration von Tags, nur dass nun der Abstand zwischen den Gruppen sehr gering ist. Tauschen die beiden Variablen ihre Größe, treten die IDs regelmäßig aber einzeln auf.

Experiment	p	q	Eigenschaften
Symmetrisch	95	95	Hohe Konzentration von Tag-IDs wahrscheinlich
	80	80	↑
	50	50	Keine Struktur
	20	20	↓
	5	5	Abwechselnde Tag-IDs
Asymmetrisch	95	5	Hohe Konzentration von Tag-IDs, geringer Abstand
	80	20	↑
	65	35	|
	35	65	|
	20	80	↓
	5	95	Abwechselnde/Vereinzelte Tag-IDs

Tabelle 6.1: Verschiedene Werte für das Gilbert-Modell [KYK08]

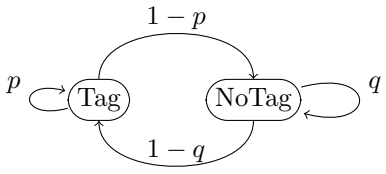


Abbildung 6.1: Gilbert-Modell für die balancierte Bäume

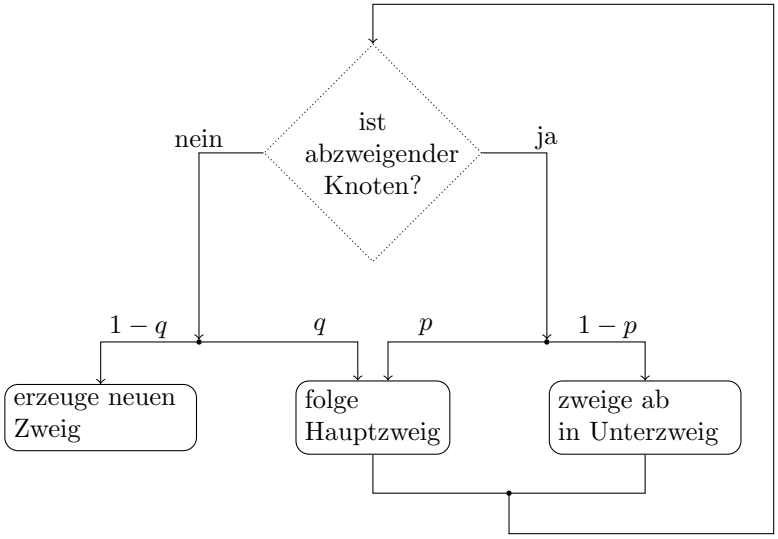


Abbildung 6.2: Modell für die Erstellung unbalancierter Bäume

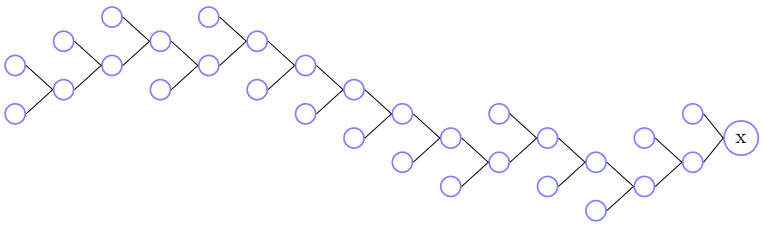
6.1.2 Modell für unbalancierte Bäume

Basierend auf der Idee des Gilbert-Modells wurde ein neues Modell entwickelt, um unbalancierte Bäume zu erstellen (Abbildung 6.2). Mit diesem Modell ist es möglich einen Hauptzweig zu erzeugen, von dem Unterzweige abzweigen. Der Hauptzweig ist derjenige Zweig, der das größte Gewicht hat. Das Gewicht bestimmt sich aus der Anzahl der Blätter, der Pfad zu diesen ist gleichbedeutend mit den IDs.

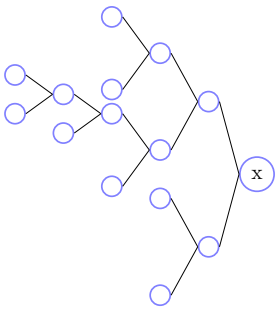
Die Generierung beginnt mit einer zufällig gewählten Tag-ID, die den Hauptzweig bildet. Eine neue ID wird in den Baum dadurch eingefügt, dass der bestehende Baum von der Wurzel aus abgelaufen und bei jedem abzweigenden Knoten die Entscheidung getroffen wird, ob mit der Wahrscheinlichkeit p dem Hauptzweig gefolgt wird oder dieser mit der Wahrscheinlichkeit $1 - p$ verlassen wird. Zweigt ein Knoten nicht ab, wird der Zeig weiterverfolgt (mit der Wahrscheinlichkeit q) oder ein neuer Zweig wird erzeugt $(1 - q)$. Die verbleibenden Bits werden zufällig ausgewählt.

Abbildung 6.3 zeigt typische Beispiele unbalancierter Bäume, die mit dem Modell erzeugt wurden. In Abbildung 6.3(a) unterscheiden sich die IDs an verschiedenen Bit-Positionen und fast alle Masken haben unterschiedliche Längen. Die resultiert in einem langgezogenen Baum und wird durch ein großes p verursacht. Um Gegensatz dazu erzeugt ein kleineres p einen Baum, der ausgeglichener ist (Abbildung 6.3(b)).

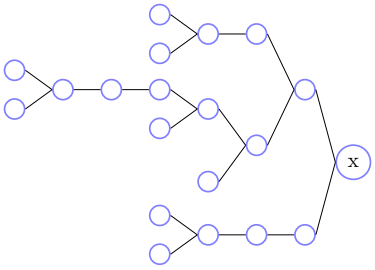
Der Wert von q hingegen legt fest, wie lang die Zweigabschnitte zwischen den abzweigenden Knoten sind. Für die ersten beiden Bäume aus Abbildung 6.3 ist der Wert von q daher sehr gering, so dass keine Abschnitte wie in Abbildung 6.3(c) entstanden sind. JTC (Abschnitt 5.2.2) erkennt die abzweigungsfreien Abschnitte und benötigt für die Ermittlung von Tags aus 6.3(b) genauso viele Schritte wie für 6.3(c), so dass sich die Laufzeit nur aufgrund unterschiedlicher Maskenlängen unterscheiden sollte.



(a) großes p



(b) kleines p



(c) großes q

Abbildung 6.3: Verschiedene unbalancierte ID-Bäume; die höheren Bits wurden nicht dargestellt, sondern nur die niederwertigen Bits bis zum identifizierenden Knoten

6.2 Detektionswahrscheinlichkeit für Tags bei inkrementierten IDs

Im Folgenden wird die Detektionswahrscheinlichkeit für Tags bei einem bewegten Lesegerät untersucht, wenn ein Tag bei $t = 0$ in die Reichweite des Lesegerätes gelangt. Dabei wird von inkrementierten IDs und der Vereinfachung ausgegangen, dass die Detektionswahrscheinlichkeit während eines Inventorys gleichverteilt ist. Des Weiteren wird eine konstante Inventorydauer $t_{\min}$ und somit eine konstante Anzahl von Tags innerhalb des Lesebereiches vorausgesetzt. Übertragungsfehler werden nicht berücksichtigt.

6.2.1 Detektion eines Tags bei synchronem Start

Die geringste Zeit $t_{\min}$, die notwendig ist, bevor ein Tag detektiert werden kann, ist durch die benötigte Zeit gegeben, um zumindest eine Inventory-Anfrage zu stellen und

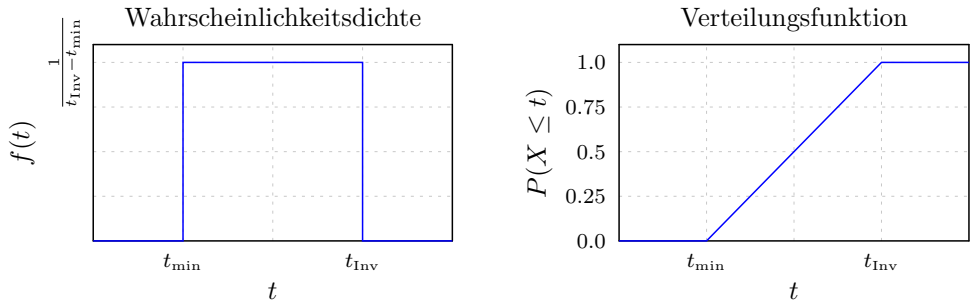


Abbildung 6.4: Wahrscheinlichkeitsdichte und Verteilungsfunktion einer Gleichverteilung auf dem Intervall t_{min} und t_{Inv}

die Antwort des Tags zu empfangen. Im Gegensatz dazu steht die maximale Zeit, die durch $t_{\text{max}} = 2 \cdot t_{\text{Inv}}(n)$ gegeben ist. Die Zeit $t_{\text{Inv}}(n)$, die für das Inventory benötigt wird, hängt im Wesentlichen von der Anzahl der beteiligten Tags n ab, soll im Weiteren aber als konstant angesehen werden.

Die Zufallsvariable X bezeichnet den Zeitpunkt, an dem ein Tag detektiert wird. Demzufolge gibt $P(X \leq t)$ die Wahrscheinlichkeit an, dass ein Tag bis zu dem Zeitpunkt t erkannt wurde. Damit ist es die Integration von 0 bis t:

$$P(X \leq t) = \int_0^t f_X(t)dt$$

Dabei bezeichne $f_X(t)$ die Wahrscheinlichkeitsdichte, also die Wahrscheinlichkeit, dass ein Tag zum Zeitpunkt t detektiert wird. Obwohl $f_X(t)$ nicht bekannt ist, wird angenommen, dass sich die Wahrscheinlichkeit bei der gleichmäßigen Struktur der inkrementierten IDs während des gesamten Inventorys nicht wesentlich ändert und somit gleichverteilt ist. Dies resultiert in Abbildung 6.4 und kann formal ausgedrückt werden als:

$$P(X \leq t) = \begin{cases} 0, & \text{falls } t \leq t_{\text{min}} \\ \frac{t-t_{\text{min}}}{t_{\text{Inv}}-t_{\text{min}}}, & \text{falls } t_{\text{min}} < t < t_{\text{Inv}} \\ 1, & \text{falls } t \geq t_{\text{Inv}} \end{cases} \tag{6.1}$$

6.2.2 Wiederholte Inventorys

Für ein bewegtes Lesegerät beschreibt das vorherige Modell die Detektionswahrscheinlichkeit nur unzureichend. Zum einen wird das Inventory in einer fortlaufenden Schleife ausgeführt und zum anderen kann nicht davon ausgegangen werden, dass ein Tag dann in den Lesebereich des Reader gelangt, wenn auch gerade ein Antikollisionszyklus startet. Beides muss berücksichtigt werden.

Die Zufallsvariable Y beschreibe den Zeitpunkt, bis zu dem ein Tag detektiert wird. Dabei gelte, dass sich das Tag im Lesebereich des Readers befindet, das Inventory zu einem beliebigen Zeitpunkt startet und Y durch t_{R} beschränkt ist.

In Abhängigkeit von der Größe von t_{R} müssen vier Fälle unterschieden werden:

- Es kann ein vollständiges Inventory ausgeführt werden, während sich das Tag in Reichweite befindet: $2 \cdot t_{\mathrm{Inv}} \leq t_{\mathrm{R}}$
- Es kann ein vollständiges oder teilweises Inventory (d.h. das Tag befindet sich nicht bis Ende des Inventory-Durchlaufes in Reichweite) ausgeführt werden: $t_{\mathrm{Inv}} \leq t_{\mathrm{R}} < 2 \cdot t_{\mathrm{Inv}}$
- Es ist nur ein teilweises Inventory möglich: $t_{\mathrm{min}} \leq t_{\mathrm{R}} < t_{\mathrm{Inv}}$
- Kein Inventory kann durchgeführt werden: $t_{\mathrm{R}} < t_{\mathrm{min}}$

Die folgenden Berechnungen gelten auch ohne Annahme einer Gleichverteilung für $P(X \leq t)$. Abschnitt 6.2.3 zeigt das zusammenfassende Ergebnis, wenn diese angenommen wird.

6.2.2.1 Vollständiges Inventory ($2 \cdot t_{\mathsf{Inv}} \leq t_{\mathsf{R}}$)

Wie bereits in Abschnitt 5.1.1 erläutert, beträgt die Wahrscheinlichkeit ein Tag zu erkennen 1, wenn $t_{\mathrm{R}} \geq 2 \cdot t_{\mathrm{Inv}}$ ist. Das Inventory kann zu jedem beliebigen Zeitpunkt starten, aber das Tag wird zumindest während eines Durchlaufes erkannt.

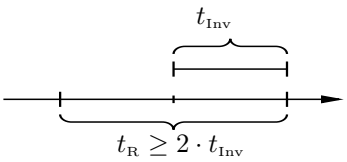


Abbildung 6.5: Vollständiges Inventory möglich

Es gilt:

$$P(Y = t_{\mathrm{R}}) = 1 \tag{6.2}$$

6.2.2.2 Vollständiges oder teilweises Inventory ($t_{\mathsf{Inv}} \leq t_{\mathsf{R}} < 2 \cdot t_{\mathsf{Inv}}$)

Wenn t_{R} reduziert wird, verringert sich auch die Detektionswahrscheinlichkeit. Das Ereignis ein Tag in diesem Fall zu erkennen, setzt sich aus zwei disjunkten Ereignissen zusammen: Wenn das Inventory vor $t_{\mathrm{R}} - t_{\mathrm{Inv}}$ startet, kann das Inventory vollständig ausgeführt und das Tag sicher detektiert werden (siehe Abbildung 6.6). Anderenfalls kann das Tag möglicherweise während eines unvollständiges Inventorys erkannt werden.

Die Zeitspanne $t_{\mathrm{R}} - t_{\mathrm{Inv}}$ nach Ablauf von t_{Inv} darf nicht berücksichtigt werden. Beginnt das Inventory in dieser Zeit, ist bereits einmal t_{Inv} verstrichen, so dass das Tag sicher detektiert werden konnte.

Die Wahrscheinlichkeit ein Tag innerhalb von t_{Inv} (welches im Zeitraum von t_{R} liegt) zu detektieren, kann wie folgt berechnet werden. Mit einer Wahrscheinlichkeit von $\frac{t_{\mathrm{R}} - t_{\mathrm{Inv}}}{t_{\mathrm{Inv}}}$

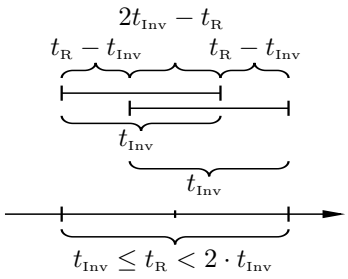


Abbildung 6.6: Startet das Inventory nach Eintritt des Tags in den Lesebereich bis spätestens $t_{\mathrm{R}} - t_{\mathrm{Inv}}$, kann es vollständig ausgeführt werden. Anderenfalls wird es nur teilweise ausgeführt, solange sich das Tag noch in Reichweite befindet.

startet das Inventory bis $t_{\mathrm{R}} - t_{\mathrm{Inv}}$ und kann sicher detektiert werden. Mit einer Wahrscheinlichkeit von $\frac{2t_{\mathrm{Inv}} - t_{\mathrm{R}}}{t_{\mathrm{Inv}}}$ startet das Inventory danach und wird nur unvollständig ausgeführt. Die Wahrscheinlichkeit ein Tag zwischen $t_{\mathrm{R}} - t_{\mathrm{Inv}}$ und t_{Inv} zu erkennen, berechnet sich aus dem Integral der Wahrscheinlichkeiten in diesem Zeitraum und muss auf die Zeitdauer skaliert werden. Dies führt zu:

$$\begin{aligned} P(Y = t_{\mathrm{R}}) &= P(\text{vollständiges Inventory}) + P(\text{teilweises Inventory}) \\ &= \frac{t_{\mathrm{R}} - t_{\mathrm{Inv}}}{t_{\mathrm{Inv}}} + \frac{2t_{\mathrm{Inv}} - t_{\mathrm{R}}}{t_{\mathrm{Inv}}} \cdot \frac{1}{2t_{\mathrm{Inv}} - t_{\mathrm{R}}} \int_{t_{\mathrm{R}} - t_{\mathrm{Inv}}}^{t_{\mathrm{Inv}}} P(X \leq t) dt \\ &= \frac{t_{\mathrm{R}} - t_{\mathrm{Inv}}}{t_{\mathrm{Inv}}} + \frac{1}{t_{\mathrm{Inv}}} \int_{t_{\mathrm{R}} - t_{\mathrm{Inv}}}^{t_{\mathrm{Inv}}} P(X \leq t) dt \end{aligned} \tag{6.3}$$

6.2.2.3 Teilweises Inventory ($t_{\mathsf{min}} \leq t_{\mathsf{R}} < t_{\mathsf{Inv}}$)

Wird t_{R} kleiner als t_{Inv}, ist es nicht mehr möglich, das Tag während eines vollständigen Inventorys zu erkennen. Des Weiteren muss beachtet werden, dass das Inventory nur mit einer Wahrscheinlichkeit von $\frac{t_{\mathrm{R}}}{t_{\mathrm{Inv}}}$ beginnt, während sich das Tag in Reichweite des Lesegerätes befindet (siehe Abbildung 6.7). Es ergibt sich die folgende Gleichung:

$$\begin{aligned} P(Y = t_{\mathrm{R}}) &= \frac{t_{\mathrm{R}}}{t_{\mathrm{Inv}}} \cdot \frac{1}{t_{\mathrm{R}}} \int_{0}^{t_{\mathrm{R}}} P(X \leq t) dt \\ &= \frac{1}{t_{\mathrm{Inv}}} \int_{0}^{t_{\mathrm{R}}} P(X \leq t) dt \end{aligned} \tag{6.4}$$

6.2.2.4 Kein Inventory ($t_{\mathsf{R}} < t_{\mathsf{min}}$)

Wenn t_{R} zu klein wird, ist es nicht mehr möglich, das Tag zu detektieren.

$$P(Y = t_{\mathrm{R}}) = 0 \tag{6.5}$$

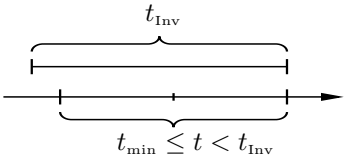


Abbildung 6.7: Inventory kann nur teilweise durchgeführt werden

6.2.3 Graphische Darstellung bei einer Gleichverteilung

Die Zusammenfassung der soeben gefundenen Gleichungen (6.2, 6.3, 6.4 und 6.5) ergibt:

$$P(Y = t_{\text{R}}) = \begin{cases} 0, & \text{falls } t_{\text{R}} < t_{\text{min}} \\ \frac{1}{t_{\text{Inv}}} \int_0^{t_{\text{R}}} P(X \leq t)dt, & \text{falls } t_{\text{min}} \leq t_{\text{R}} < t_{\text{Inv}} \\ \frac{t_{\text{R}} - t_{\text{Inv}}}{t_{\text{Inv}}} + \frac{1}{t_{\text{Inv}}} \int_{t_{\text{R}} - t_{\text{Inv}}}^{t_{\text{Inv}}} P(X \leq t)dt, & \text{falls } t_{\text{Inv}} \leq t_{\text{R}} < 2 \cdot t_{\text{Inv}} \\ 1, & \text{falls } 2 \cdot t_{\text{Inv}} \leq t_{\text{R}} \end{cases} \tag{6.6}$$

Die Integration der Gleichverteilung aus Gleichung 6.1 ergibt unter Berücksichtigung der Integrationskonstanten:

$$\int P(X \leq t)dt = \begin{cases} 0, & \text{falls } t \leq t_{\text{min}} \\ \frac{\frac{1}{2}t^2 - t_{\text{min}}t + \frac{1}{2}t_{\text{min}}^2}{t_{\text{Inv}} - t_{\text{min}}}, & \text{falls } t_{\text{min}} < t < t_{\text{Inv}} \\ t - \frac{1}{2}(t_{\text{Inv}} + t_{\text{min}}), & \text{falls } t \geq t_{\text{Inv}} \end{cases} \tag{6.7}$$

Die Gleichungen 6.6 und 6.7 ergeben die graphische Darstellung in Abbildung 6.8 für die Detektionswahrscheinlichkeit eines Tags bei wiederholten Inventorys und einer Gleichverteilung innerhalb eines Inventorys.

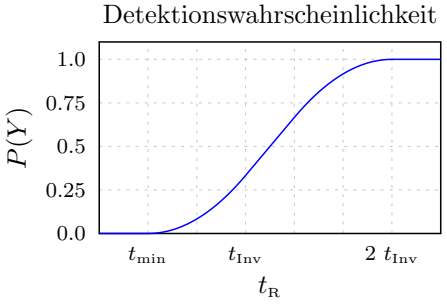


Abbildung 6.8: Wahrscheinlichkeit ein Tag bis zum Zeitpunkt t_{R} zu erkennen

6.3 Simulation

Für die Evaluation von Antikollisionsalgorithmen wurde ein Simulator entwickelt, mit dem Algorithmen für ISO-15693 getestet werden können. Eine beliebige Anzahl von Tags und ein Lesegerät können frei positioniert werden. Den Algorithmen stehen Funktionen zur Verfügung, mit denen sie Pakete verschicken können, die gemäß ISO-15693 vorgesehen wurden. Das Lesegerät kann nur mit den Tags kommunizieren, die sich in der festgelegten Reichweite befinden. Innerhalb dieser ist die Kommunikation ohne Übertragungsfehler möglich. Die Reichweite wurde als kreisförmig mit festem Radius angenommen, wobei dies prinzipiell variabel ist. Die Tags verhalten sich, wie in der Standardisierung festgelegt.

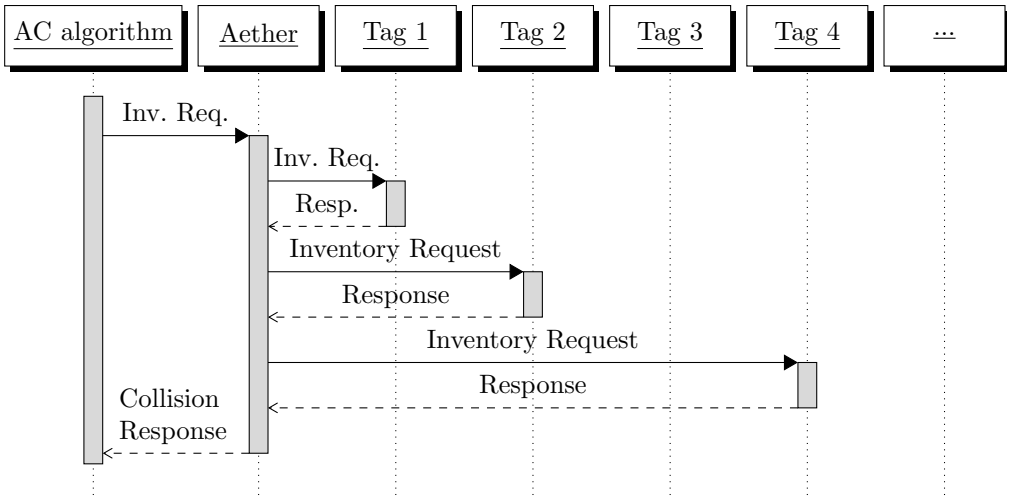


Abbildung 6.9: Sequenzdiagramm für die Simulation

Abbildung 6.9 zeigt den prinzipiellen Ablauf der Simulation. Der Antikollisionsalgorithmus sendet eine Anfrage an den Äther, welcher die physikalischen Gegebenheiten simuliert. Die Anfrage des Algorithmus wird nun an die Tags weitergeleitet, die sich tatsächlich in Reichweite befinden. In der Abbildung sind dies die Tags 1, 2 und 4. Das dritte Tag befindet sich außerhalb des Lesebereiches. Die Tags ermitteln die Antworten gemäß ihrem Zustand und sendet diese zurück an den Äther. Dabei kann die Antwort auch leer sein, wenn die Inventory-Anfrage nicht auf die ID des Tags passt. Dies entspricht keiner Antwort. Der Äther sammelt und überlagert die Antworten und berechnet auf diese Weise die entstehenden Kollisionen. Anschließend wird die kollisionsbehaftete Antwort dem Algorithmus übergeben. Dieser wertet die Antwort nun aus und kann eine erneute Anfrage an den Äther stellen.

Der Simulator war ursprünglich vorgesehen, um den Zeitaufwand von Algorithmen bei unbewegtem Lesegerät und unbewegten Tags zu ermitteln. Da die Übertragungsrate sowie alle Timings festgelegt wurden, ist somit unter der Voraussetzung fehlerfreier Kommunikation und Hardware eine gute Abbildung der Realität möglich. Für Betrachtung eines mobilen Lesegerätes wurde der Simulator erweitert. Dazu wird die Position des Lesegerätes nach jedem Paketübertragungsschritt angepasst. Da ein kompletter

Inventory-Durchlauf aus vielen Schritten besteht, ist auf diese Weise die Simulation des tatsächlichen Verhaltens möglich. Die gesamte Kommunikation läuft immer über den Äther, so dass dieser die lokale Uhrzeit bestimmen kann. Anhand der Uhrzeit und der Geschwindigkeit des Lesegerätes kann dessen Position bestimmt werden und infolgedessen auch die erreichbaren Tags.

Die folgenden Abschnitte beinhalten die Simulationen unterschiedlicher Algorithmen unter verschiedenen Voraussetzungen. Als erstes werden Algorithmen untersucht, die nicht lesend auf Tags zugreifen und bei verschiedenen Geschwindigkeiten sowie verschiedenen ID-Distributionen ausgeführt wurden. Des Weiteren wurde der Einfluss unterschiedlicher unbalancierter Bäume simuliert. Im Anschluss wird die Maskenspeicherung genauer untersucht, welche nun auch lesend auf die Tags zugreift. Abschließend werden Wechselwirkungen verschiedener Parameter wie Tag-Dichte und Reichweite betrachtet.

6.3.1 Geschwindigkeitssimulation für verschiedene Antikollisionsalgorithmen

Ziel der Geschwindigkeitssimulation war es, einen Algorithmus danach zu beurteilen, wie viele Tags er in der Lage zu detektieren ist, wenn er mit konstanter Geschwindigkeit über ein Feld von Tags bewegt wird. Zu diesem Zweck wurden 30 Tags in einem Gitter von 3x10 Tags auf einer Länge von einem Meter und einer Breite von 0,25 m platziert. Der Reader bewegte sich auf einer festgelegten Strecke über die Tags und führte fortlaufend Inventorys aus. Die Lesereichweite wurde auf 0,14 m festgelegt. Dieser Aufbau entspricht der bereits real verwendeten Umgebung. Lediglich die Lesereichweite ist ein wenig höher, damit sich mehr Tags gleichzeitig in Reichweite befinden und der Einfluss des verwendeten Antikollisionsalgorithmus besser hervortritt. Jede Simulation wurde 100 Mal durchgeführt und das Ergebnis gemittelt.

Abbildung 6.10 zeigt die Ergebnisse für verschiedene Modifikationen des Standard-ISO-15693-Algorithmus:

- stdRec: der in der ISO-15693-Spezifikation vorgegebene Algorithmus (siehe Abschnitt 3.7.2)
- JTC: Verringerung der Anzahl der Anfragen durch Identifikation der Kollisionspositionen (siehe Abschnitt 5.2.2)
- +2@1 (JTC+2@1): zusätzliche Verbesserung durch gleichzeitige Identifikation von zwei Tags, wenn nur eine Kollision auftritt (siehe Abschnitt 3.7.3)
- +Q2 (JTC+2@1+Q2): weitere Optimierung durch Bereitstellen von vier initialen Masken (siehe Abschnitt 5.2.3)

Aufgrund der statischen Natur der Simulation (die zeitabhängige Position des Lesegerätes ändert sich nicht) sind die resultierenden Kurven sehr uneben, wenn hochgezählte Tag-IDs simuliert wurden. Die ähnlichen identifizierenden Masken bei inkrementierten IDs verhindern größere Variationen bei wiederholten Durchläufen.

Zur Kollisionsposition zu springen (JCP) liefert hier keine erkennbare Verbesserung gegenüber dem Standard-Algorithmus. Dies lässt sich darauf zurückführen, dass bei

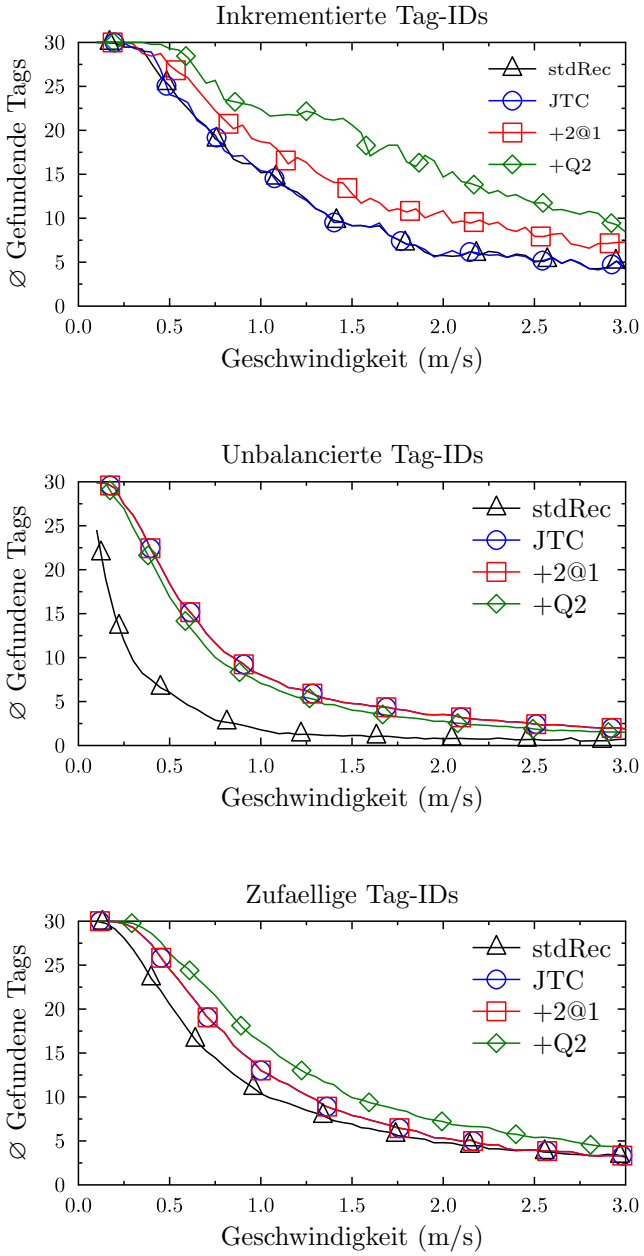


Abbildung 6.10: Geschwindigkeitssimulation rekursiver Algorithmen mit verschiedenen Tag-ID-Verteilungen

inkrementierten IDs keine Kollisionen in höheren Bitpositionen auftreten und diese Optimierung damit nie zur Anwendung kommt.

Zwei Tags in einem Schritt zu identifizieren (+2@1) führt hingegen zu einem deutlich erkennbaren Vorteil mit bis zu 88% mehr Tags (bei einer Geschwindigkeit von $2\,m/s$). Diese Modifikation ist bei aufeinanderfolgenden IDs von besonderer Bedeutung, da sich hier je zwei IDs in einer Position unterscheiden.

Wird zusätzlich eine Start-Query (+Q2) mit vier Masken benutzt (`00`, `01`, `10`, `11`), zeigt sich, verglichen mit Rec/JTC, sogar eine Verbesserung von bis zu 188% (für v =1,95 m/s). Auch diese Modifikation profitiert von der Tag-ID-Verteilung, da sie Fehlanfragen von +Q2 unwahrscheinlich macht. Durch die inkrementierten IDs sind die möglichen Zweige in geringer Höhe mit hoher Wahrscheinlichkeit besetzt, so dass +Q2 die Anzahl der Anfragen reduziert.

Bei unbalancierten ID-Verteilungen kann die größte Verbesserung erreicht werden, wenn direkt zur Kollisionsposition gesprungen wird. Kollisionen in höheren Bitpositionen treten bei dieser ID-Verteilung häufig auf. Die gleichzeitige Identifikation zweier Tags führt zu keiner Veränderung gegenüber JTC. Initiale Masken verschlechtern hier das Ergebnis leicht, was auf Anfragen zurückzuführen ist, die zu keinen Ergebnissen führen. Die wenigsten Tags im Vergleich liefert erneut stdRec. Dies liegt in der ungünstigen Struktur unbalancierter Bäume für Antikollisionsalgorithmen, die auf der Breitensuche basieren, begründet. Sie erfordern längere Masken und detektieren die Tags im Schnitt später (siehe auch den folgenden Abschnitt 6.3.2), so dass die Leistung aller Algorithmen bei dieser ID-Distribution insgesamt schlechter ist, als bei inkrementierten IDs.

Bei zufällig verteilten Tag-IDs ist die Gesamtleistung der Algorithmen wieder höher, aber immer noch geringer als bei inkrementierten IDs. Es kann beobachtet werden, dass die Ergebnisse aller Algorithmen näher zusammenliegen. Zu den Kollisionspositionen zu springen liefert aufgrund auftretender Kollisionen in höheren Bits ein besseres Ergebnis als stdRec. Zwei Tags gleichzeitig zu ermitteln führt wie bereits zuvor zu keiner erkennbaren Veränderung. Da durch die zufällig verteilten IDs ein balancierterer ID-Baum entsteht, hat +Q2 hier erneut Vorteile.

Abbildung 6.11 vergleicht das Verhalten zweier Antikollisionsalgorithmen, die nicht rekursive Ansätze verfolgen, mit dem jeweils besten Ergebnis aus der vorherigen Simulation.

- Breitensuche: steht im Gegensatz zu den Algorithmen, die auf stdRec aufbauen und somit auf einer Tiefensuche basieren
- RemM: benutzt die Resultate eines vorherigen Durchlaufs, um wiederholt auftretende Tags schneller zu finden (siehe Abschnitt 3.7.5)

Bei inkrementierten Tag-IDs kann erneut beobachtet werden, dass +Q2 das beste Ergebnis liefert. Die Wiederbenutzung von Masken führt zu keinem großen Vorteil gegenüber der Breitensuche, da insbesondere mit höherer Geschwindigkeit die Wahrscheinlichkeit sinkt, dass ein Tag in einem folgenden Inventory-Durchlauf erneut auftritt. Dennoch ist das Ergebnis besser als erwartet. Dies ist darauf zurückzuführen, dass die identifizierenden Masken inkrementierter ID sehr kurz sind und deren Wiederverwendung daher mit

Schritt	Tiefensuche		Breitensuche	
	Maske	Ergebnis	Maske	Ergebnis
1	∅	Kollision	∅	Kollision
2	0	Kollision	0	Kollision
3	00	Kollision	1	Tag
4	000	Tag	00	Kollision
5	100	Tag	10	Tag
6	10	Tag	000	Tag
7	1	Tag	100	Tag

Tabelle 6.2: Tiefen- und Breitensuche im Vergleich

recht hoher Wahrscheinlichkeit auch dann Tags liefert, wenn sich die Menge der Tags inzwischen geändert hat, aber die Masken nun zu anderen IDs passen.

Die Breitensuche hat einen sichtbaren Vorteil bei einer unbalancierten Verteilung. Hier führt sie dazu, dass Tags im Antikollisionsprozess früher gefunden werden. Bemerkenswert ist der Unterschied zur Tiefensuche stdRec aus der vorherigen Abbildung 6.10. Beide Algorithmen haben im Allgemeinen für dieselben Tag-IDs die gleiche Ausführungszeit, da in beiden Algorithmen alle Knoten des ID-Baumes jeweils einmal besucht werden. Die Knoten werden jedoch zu unterschiedlichen Zeitpunkten besucht.

Die JTC+2@1-Modifikation liefert im Vergleich zur Breitensuche leicht weniger Tags. Ein deutlich schlechteres Ergebnis jedoch zeigt RemM. Durch die ID-Struktur ist es im Gegensatz zu den inkrementierten IDs unwahrscheinlich, dass die hier längeren Masken nochmals ein Tag identifizieren.

Zufällig verteilte IDs führen erneut zu kürzeren identifizierenden Masken, so dass die Leistung von RemM im Vergleich zur vorherigen ID-Verteilung wieder besser ist. Ein leicht besseres Ergebnis zeigt die Breitensuche. Die Modifikation mit Q2 zeigt erneut das beste Ergebnis.

6.3.2 Unbalancierte Bäume

Die Simulation verschiedener unbalancierter Bäume hatte zum Ziel, das Verhalten der Tiefensuche mit dem der Breitensuche zu vergleichen.

Beide Algorithmen haben die gleiche Ausführungszeit, aber es kann beobachtet werden, dass sie keine konstante Detektionsrate haben. In Abbildung 6.12 ist ein Beispiel eines unbalancierten Baumes angegeben. Für diese IDs ist in Tabelle 6.2 der Ablauf der Breiten- und Tiefensuche gegenübergestellt. Es ist dabei zu erkennen, dass bei der gegebenen ID-Struktur die Breitensuche bereits im dritten Schritt ein Tag liefert, während dies der Tiefensuche erst im vierten Schritt gelingt.

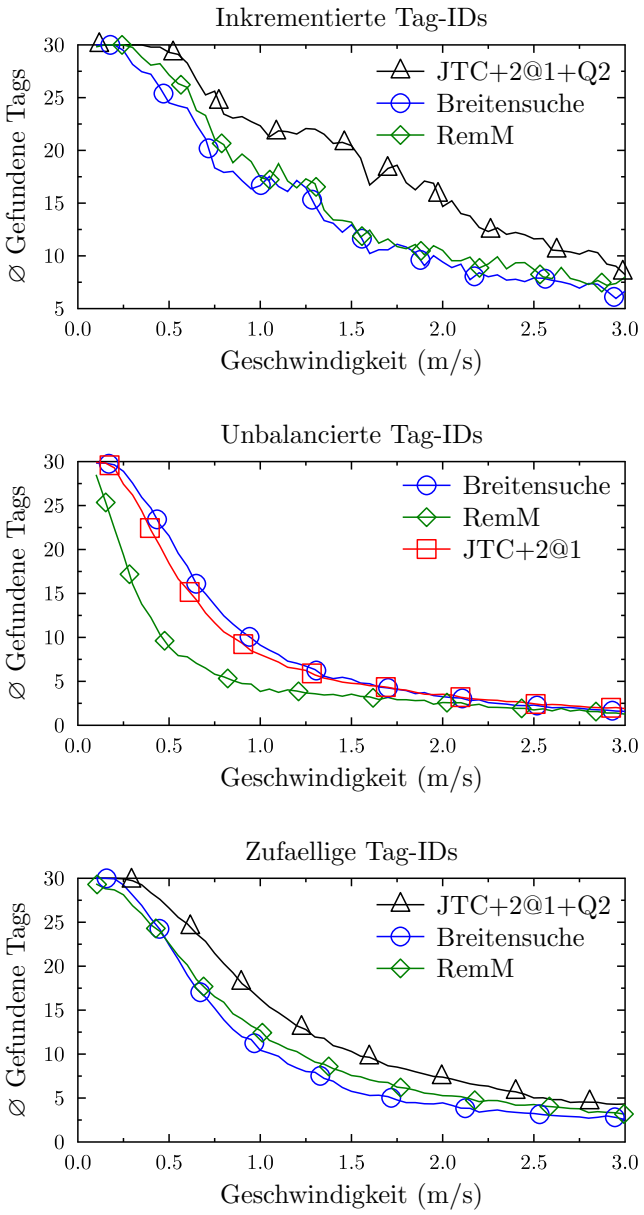


Abbildung 6.11: Geschwindigkeitssimulation unterschiedlicher Algorithmen mit verschiedenen Tag-ID-Verteilungen

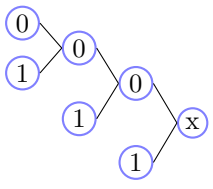


Abbildung 6.12: Unbalancierter Baum der relevanten Masken für die Detektion

Abbildung 6.13 zeigt das Ergebnis der Simulation. Es wurden die Zeiten für jedes einzelne Tag ermittelt, die benötigt wurden, um es unter 30 Tags bei verschiedenen unbalancierten Tag-Distributionen zu detektieren. Für jede Anzahl gefundener Tags wurde die bis dahin benötigte Zeit aufgetragen. Jede Simulation wurde 1000 mal durchgeführt und gemittelt. Diese nun höhere Anzahl an Durchläufen war insbesondere für $p = 0{,}8$ notwendig, um ein vergleichbares Ergebnis zu erreichen.

Wie zu erkennen ist, verfügt die Breitensuche in der ersten Simulation über Vorteile, da sie anfangs weniger Zeit benötigt, um Tags zu erkennen. Der hohe p-Wert führt zu sehr unbalancierten Bäumen. Wird dieser Wert reduziert, nähern sich die bei Graphen an, und der resultierende ID-Baum ist weniger ungleichmäßig. Bei $p = 0{,}6$ handelt es sich fast um einen balancierten Baum, und die Leistung der Tiefensuche liegt erkennbar über der der Breitensuche.

6.3.3 Maskenspeicherung

Ein Ziel der folgenden Simulation war es, den Ansatz der Maskenspeicherung (MS) mit anderen, herkömmlicheren Ansätzen zu vergleichen. Es sollte herausgefunden werden, wie sich die Algorithmen im Vergleich bei verschiedenen Geschwindigkeiten verhalten. Die Simulation sollte zeigen, dass der Algorithmus Vorteile hat, wenn nur eine Teilmenge der möglichen Tags erkannt werden soll. Die Detektion einer Teilmenge ist sinnvoll, da nicht vorausgesetzt werden kann, dass sich das Fahrzeug in der Follow-Phase exakt auf dem Pfad befindet und nur Tags erkannt werden können, die auch Pfadinformationen enthalten. Aufgrund des Detektionsverfahrens von MS kann vermutet werden, dass dieser Algorithmus hier Vorteile bietet.

Als zweite Aufgabe galt es zu zeigen, welche Einflüsse die Reichweite des Lesegerätes und die Tag-Dichte auf das Ergebnis haben. Beide Parameter haben einen direkten Einfluss auf die Anzahl der Tags innerhalb des Lesebereiches.

Des Weiteren sollte untersucht werden, ob der Algorithmus robust ist und auch bei Ausfällen in der Lage ist, weitere Tags zu erkennen. Dies ist hier von besonderem Interesse, weil MS darauf beruht, dass Informationen aus vorherig detektierten Tags verwendet werden.

Der implementierte Algorithmus benutzt keine Pfad-Identifikationsnummer, um zwischen nahen oder kreuzenden Pfaden zu unterscheiden, da für die folgenden Simulationen immer nur ein Pfad zur Zeit benutzt wurde. Ebensowenig wurden Sequenznummern genutzt, um eine Reihenfolge der Tags zu bestimmen. Werden diese Daten benötigt, reduziert sich der zur Verfügung stehende Speicher. Dieser ist jedoch ohnehin variabel. Der benutztbare Speicher wurde auf einen Block der Größe 32 Bit festgelegt. Dies entspricht der Blockgröße gehandelter Tags. In diesem Block können bis zu sieben Einträge mit einer jeweiligen maximalen Größe von sieben Bit abgelegt werden. Dabei darf die Blockgröße von 32 Bit jedoch nicht überschritten werden.

Wie in den Simulationen zuvor, wurde ein symmetrisches Tag-Feld genutzt. Die exakten Positionen der Tags wurden jedoch innerhalb ihres festgelegten Areals zufällig variiert.

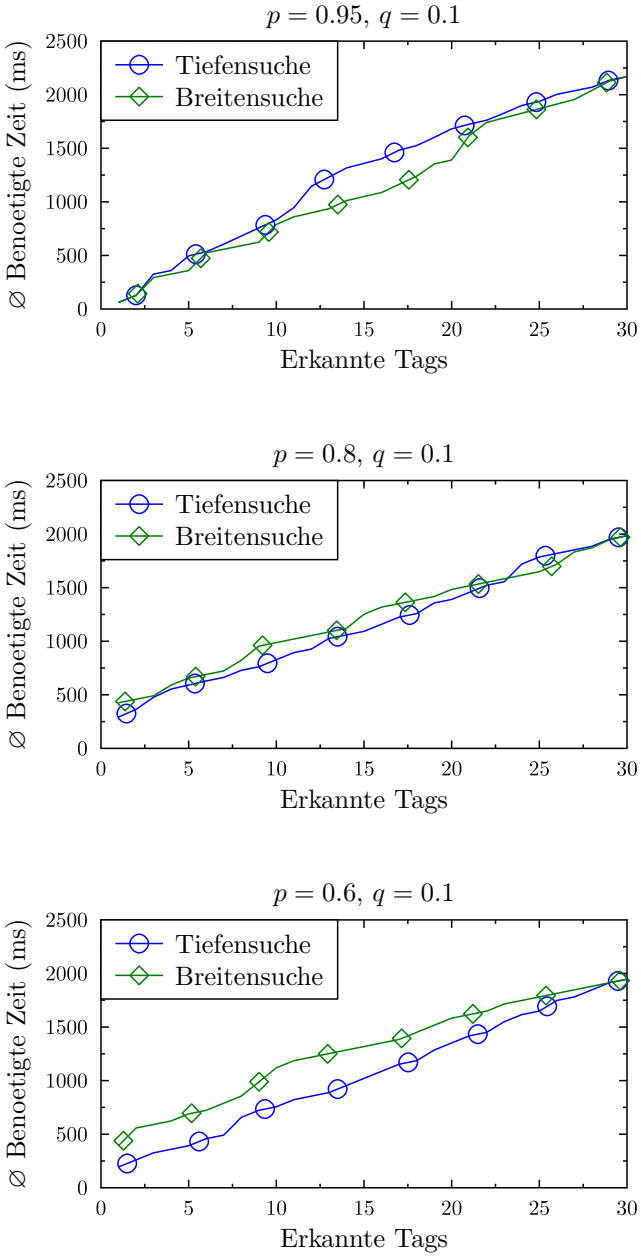


Abbildung 6.13: Breiten- und Tiefensuche im Vergleich bei verschiedenen ID-Verteilungen

Damit wird verhindert, dass große Sprünge bei der Anzahl der involvierten Tags auftreten, wenn die Reichweite des Lesegerätes nur wenig verändert wird.

6.3.3.1 Geschwindigkeit

In der Geschwindigkeitssimulation soll wie bei der vorherigen Geschwindigkeitssimulation überprüft werden, wie sich die Anzahl der Detektionen ändert, wenn sich das Lesegerät mit steigender Geschwindigkeit über das Tag-Feld bewegt.

Dazu wurden drei leicht unterschiedliche MS-Algorithmen gewählt und mit stdRec sowie Q2 verglichen. Die drei MS-Varianten wurden wie folgt gewählt:

- MS steht für den Algorithmus, der stdRec nutzt, um Kollisionen zu behandeln. Für den Fall, dass keine vorherigen Masken erkannt wurden, wird nach der ersten verfügbaren gesucht und die Suche abgebrochen (so wie beschrieben in Abschnitt 5.4.2).
- MS-Q2t ist eine Variante, die Q2 für die Kollisionsbehandlung verwendet.
- MS-Q2f ist nahezu identisch, nur mit dem Unterschied, dass hier bei einem Inventory ohne Start-Masken versucht wird, alle verfügbaren Tags zu erkennen.

Es sollte beachtet werden, dass die Simulationen zur Maskenspeicherung, verglichen mit den vorherigen Abschnitten, höhere Anforderungen an die Algorithmen stellen. Hier wird ein Tag erst dann als detektiert betrachtet, wenn seine ID erkannt wurde und auch der Inhalt gelesen werden konnte. Das zusätzliche Lesen erfordert zusätzlich Zeit und wird kritisch, wenn ein Tag relativ spät erkannt wurde, bevor der Lesebereich wieder verlassen wird. Diese zusätzliche Anforderung ist notwendig, da MS aufgrund seiner Funktionsweise lesend auf die Tags zugreifen muss. Um die Vergleichbarkeit der Algorithmen zu gewährleisten, müssen alle Algorithmen den Inhalt auslesen. Dieses ist bei Verwendung in den anvisierten Navigationsverfahren ohnehin notwendig.

Die Simulationsumgebung wurde so eingerichtet, dass ein Feld aus 60 Tags und einer Länge von zwei Metern existiert. Die Breite des Pfades sowie der Durchmesser des Lesebereiches wurden jeweils auf 0,30 m gesetzt. Im Vergleich zu den vorherigen Simulationen ist die zurückgelegte Strecke hier länger. Dies ist notwendig, damit die Initialisierungsphase von MS (zumindest beim Start muss ein vollständiges Inventory ausgeführt werden) keinen relevanten Einfluss auf das Simulationsergebnis hat.

Abbildung 6.14 zeigt die Ergebnisse der Simulation. In einem ersten Durchlauf wurde versucht alle Tags in Reichweite zu erkennen. Im zweiten Durchlauf wurde die Reichweite des Readers auf 0,24 m erweitert (bzw. der Durchmesser auf 0,48 m), was bedeutete, dass um die 100 Tags in Reichweite waren. Es sollte aber nur die Teilmenge gesucht werden, die aus den gleichen 60 Tags aus dem vorherigen Durchlauf besteht. Durch die Erweiterung der Reichweite mussten die Masken angepasst und eine neue Spur geschrieben werden. Die Spurbreite hat sich zwar nicht geändert, aber durch die höhere Reichweite musste der Algorithmus nun auch Tags behandeln, die nicht Ziel der Detektion waren und somit den Antikollisionsprozess stören (Abbildung 6.15).

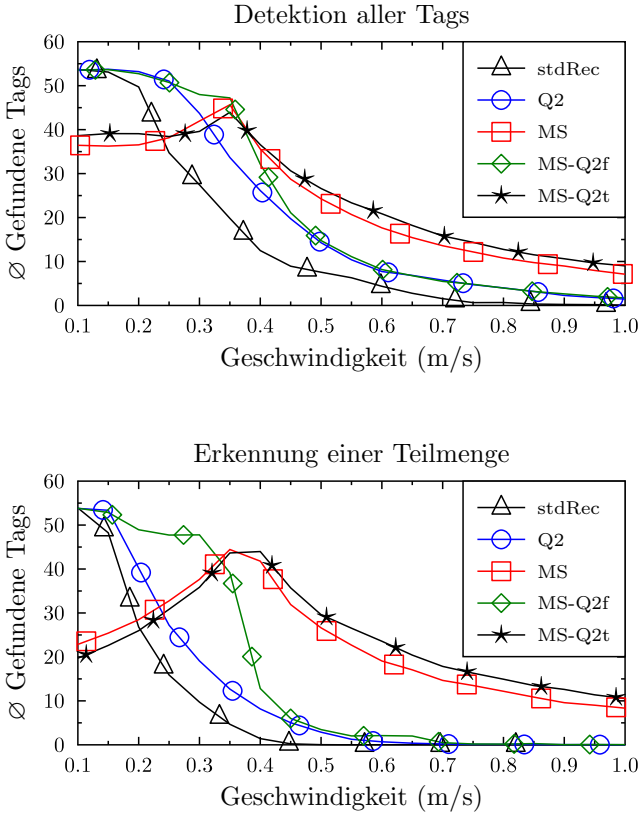


Abbildung 6.14: Detektion entlang eines Pfades

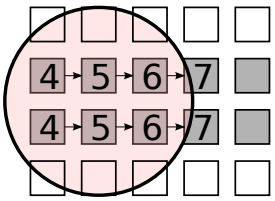


Abbildung 6.15: Das Lesegerät erreicht mehr Tags, als für das Folgen des Pfades relevant sind. Die zusätzlichen Tags beeinträchtigen die Leistung der Antikollisionsalgorithmem, die ein vollständiges Inventory durchführen (stdRec oder Q2). Die MS-Algorithmen brauchen diese zusätzlichen Tags nicht zu erkennen.

Die erste bemerkenswerte Beobachtung ist, dass der MS-Algorithmus ein globales Maximum bei einer optimalen Geschwindigkeit aufweist. Dies konnte a priori erwartet werden. Bei dieser Geschwindigkeit tritt der Fall, dass eine Maske „nicht richtig“ ist (vergleiche Abschnitt 5.4.1), nicht ein. Wie erwartet zeigt Q2 ein besseres Verhalten als stdRec. Für höhere Geschwindigkeiten hat jedoch MS deutliche Vorteile. Der Vergleich zwischen den verschiedenen MS-Algorithmen zeigt, dass sich die Q2f-Variante besser für geringere Geschwindigkeiten eignet, aber für höhere Geschwindigkeiten eine schlechtere Leistung aufweist. MS-Q2t verhält sich ähnlich wie MS, hat aber leichte Vorteile.

Bei geringeren Geschwindigkeiten steht dem Inventory vergleichsweise viel Zeit zur Verfügung. Da sich das Fahrzeug langsam bewegt, sind beim nächsten Inventory-Durchlauf noch nicht die Tags des nächsten Schritts in Reichweite. Während MS und MS-Q2t immer versuchen genau ein Tag zu detektieren, wenn ihnen keine vorherigen Masken zur Verfügung stehen, sucht MS-Q2f in diesem Fall von vornherein nach allen möglichen Tags. Daher findet er auch die relevante Tags mit einer höheren Wahrscheinlichkeit. Bei höheren Geschwindigkeiten dagegen reicht die Zeit nicht aus, um ein vollständiges Inventory auszuführen und anschließend die gefundenen Masken noch verwenden zu können, weil das Fahrzeug dann bereits zu weit gefahren ist. Daher ist es hier sinnvoller nur nach jeweils einem Tag zu suchen, so das MS-Q2f hier Nachteile hat.

Wenn dem MS-Algorithmus Informationen über den Abstand zwischen den Tags und über das eigene Fahrverhalten zur Verfügung stehen, ist es möglich, das Verhalten von Q2t für geringere Geschwindigkeiten zu verbessern. Beispielsweise könnte in diesem Fall auf die Kollisionsbehandlung von Q2f gewechselt werden.

Es kann beobachtet werden, dass die MS-Algorithmen MS und MS-Q2t bei höheren Geschwindigkeiten nicht so sehr durch die Erweiterung des Lesebereiches negativ beeinflusst werden. Dagegen verhalten sich stdRec und Q2 deutlich schlechter. MS-Q2f zeigt erneut seine Stärken für geringe Geschwindigkeiten und seine Schwächen bei höheren.

Dadurch, dass MS und MS-Q2t von vornherein nur auf IDs prüfen, müssen sie selten einen kompletten Inventory-Durchlauf durchführen. Dies wird auch durch eine größere Reichweite nicht beeinflusst. Die herkömmlichen Algorithmen müssen dagegen alle Tags überprüfen. Bei MS resultiert die Verwendung einer größeren Reichweite in einem größeren Speicherverbrauch auf den Tags, da die größere Reichweite zu längeren identifizierenden Masken führt, die gespeichert werden müssen.

6.3.3.2 Einfluss der Reichweite des Lesegerätes

Bei der Untersuchung der Lesereichweite können zumindest zwei wichtige Parameter verändert werden: die Reichweite des Lesegerätes sowie die Breite des geschriebenen Pfades. Wie im vorherigen Abschnitt gesehen, muss die Pfadbreite nicht zwangsläufig der Leseeeichweite entsprechen, wenn die Reichweite höher ist und der Pfad nur auf einer Teilmenge der erreichbaren Tags hinterlegt wird. Für alle folgenden Untersuchungen wurde MS-Q2t benutzt, der bei hohen Geschwindigkeiten die meisten Tags erkannt hat.

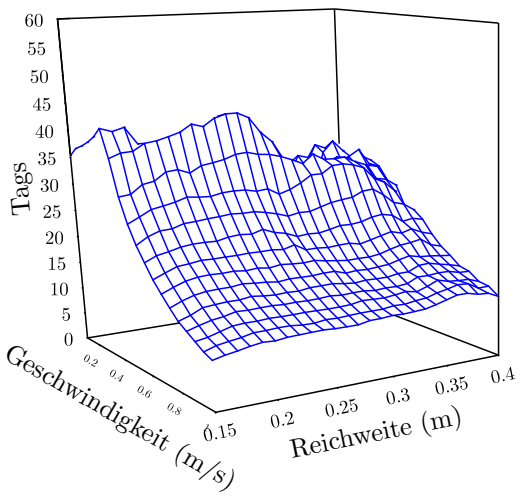


Abbildung 6.16: Vergleich des Einflüsse von Geschwindigkeit und Reichweite bei der Maskenspeicherung

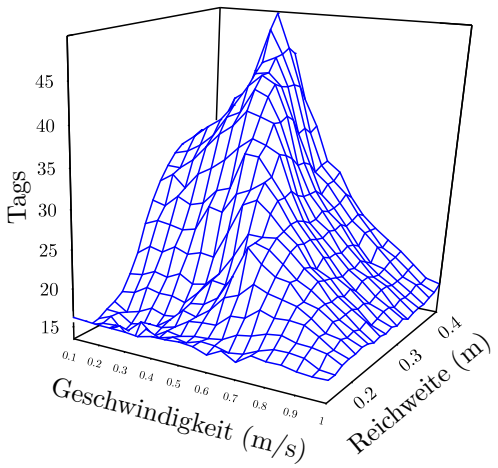


Abbildung 6.17: Vergleich der Einflüsse von Geschwindigkeit und Reichweite/Pfadbreite bei der Maskenspeicherung

Im vorherigen Abschnitt wurde gezeigt, dass sich das Verhalten von MS nicht bedeutend verschlechtert, wenn die Reichweite erhöht und die Pfadweite unangetastet gelassen wird. Abbildung 6.16 stellt dieses Verhalten dar. Die Voraussetzungen sind die gleichen, wie in den vorherigen Simulationen. Die Reichweite wurde schrittweise von 0,15 m bis 0,40 m erweitert. Die Anzahl der relevanten Tags, die Pfadinformationen beinhalten, ändert sich während der gesamten Simulation nicht. Mit steigender Reichweite verlängern sich jedoch die identifizierenden Masken. Stehen keine Masken für die schnelle Detektion zur Verfügung, wird nach wie vor ein herkömmliches Inventory ausgeführt.

Es kann beobachtet werden, dass die Anzahl detektierter Tags mit steigender Reichweite sinkt. Dies wird durch die erhöhte Anzahl von Tags verursacht, die behandelt werden müssen, wenn der Pfad verloren wurde und ein komplettes Inventory durchgeführt werden muss. Eine höhere Reichweite bedeutet allerdings ebenfalls, dass sich ein Tag länger im Lesebereich befindet. Daher ändert sich die Erkennungsrate nur langsam.

Als nächstes wurde untersucht, wie sich die Anzahl der detektierten Tags ändert, wenn die Reichweite und in gleichem Maße die Pfadbreite erhöht wird. Im Gegensatz zur vorherigen Simulation stiegt hier mit der Reichweite auch die Anzahl der Tags mit Pfadinformationen. Abbildung 6.17 zeigt das Ergebnis. Bei höheren Reichweiten werden bei geringen und mittleren Geschwindigkeiten deutlich mehr Tags erkannt. Bei höheren Geschwindigkeiten nähern sich alle Werte einem gemeinsamen Wert an. Die Pfadweite zu erhöhen, erhöht ebenfalls den Speicherverbrauch auf den Tags.

6.3.3.3 Möglichkeiten der Skalierung

In der folgenden Simulation wurde eine Umgebung genutzt, in der Reichweite, der Tag-Abstand und die Pfad-Länge synchron skaliert wurden. Dies bedeutet, dass in jedem Durchlauf potenziell die gleiche Anzahl von Tags in Reichweite ist.

Abbildung 6.18 zeigt, dass mit der Skalierung der genannten Parameter auch die Geschwindigkeit, bei der die meisten Tags erkannt werden können, skaliert. Aber es sollte beachtet werden, dass die Anzahl der Tags pro Strecke geringer ist, da die Tags über die ebenfalls skalierte Strecke gezählt wurden.

6.3.3.4 Wechselwirkungen zwischen Geschwindigkeit, Reichweite und Dichte

Die Erweiterung der Reichweite des Lesegerätes oder die Erhöhung der Tag-Dichte sind ähnliche Operationen, die dennoch in verschiedenen Ergebnissen resultieren können. Beide führen zu einer höheren Anzahl von Tags, die sich im Lesebereich befinden. Die Reichweite zu erweitern, führt jedoch auch zu einer längeren Zeit, in der sich die Tags in Reichweite befinden. Die Erhöhung der Dichte stellt dagegen höhere Anforderungen an den Antikollisionsalgorithmus, da in der gleichen Zeit mehr Tags erkannt werden müssen. In dieser Simulation sollte daher betrachtet werden, wie sich Änderungen an der Reichweite und der Dichte im direkten Vergleich verhalten.

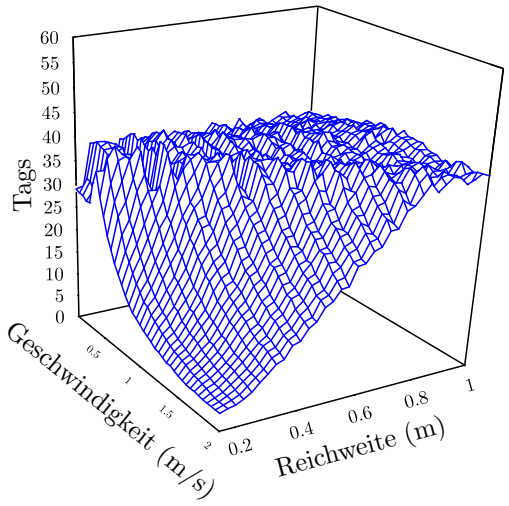


Abbildung 6.18: Geschwindigkeit vs. Reichweite; als Parameter wurde nur die Reichweite eingezeichnet, Tag-Abstand und Pfad-Länge wurden anhand dieses Parameters skaliert.

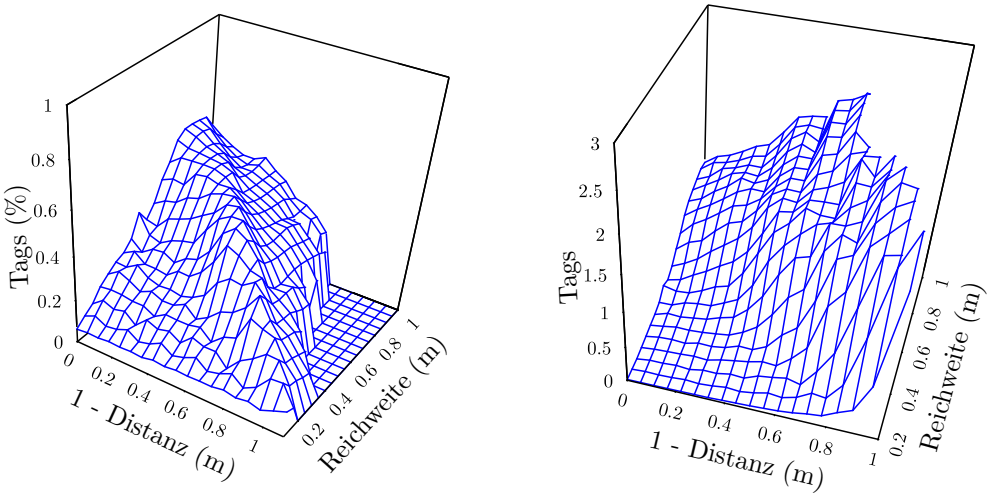


Abbildung 6.19: Reader-Reichweite vs. Tag-Dichte; die linke Grafik zeigt den prozentualen Anteil der erkannten Tags, die rechte die über die Geschwindigkeiten gemittelte absolute Anzahl der Tags pro Meter.

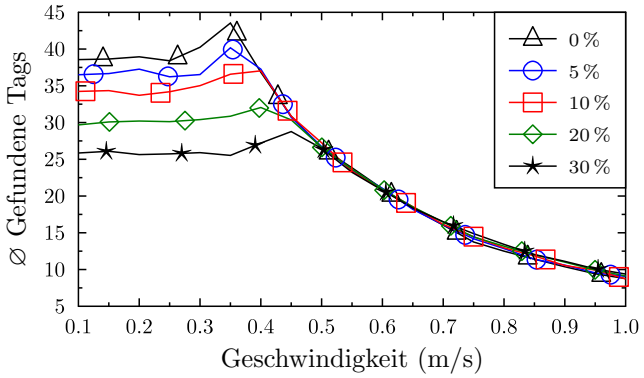


Abbildung 6.20: Verhalten der Maskenspeicherung bei Einbringung von Fehlern

In Abbildung 6.19 werden die Auswirkungen der Reichweiten- und der Dichteveränderung direkt miteinander verglichen. Die Dichte ist als der resultierende Abstand zwischen den Tags ausgedrückt, welcher zwischen 1,0 m und 0,1 m liegt. Die Reichweite liegt ebenfalls im Bereich von 1,0 m und 0,1 m. Für jedes Reichweiten-Dichte-Paar wurde der Anteil der Tags bestimmt, der bei Geschwindigkeiten zwischen 0,1 m/s und 7,0 m/s durchschnittlich ermittelt wurde. Dieser Wert wurde auf der x-Achse der ersten Grafik aufgetragen. Die zweite Grafik basiert auf den Daten derselben Simulation, nur zeigt die x-Achse hier die absolute Anzahl gefundener Tags pro Meter, die ebenfalls über die verschiedenen Geschwindigkeiten gemittelt wurden.

Bei hohen Reichweiten und einer hohen Dichte sind sehr viele Tags im Lesebereich des Readers. In diesen Fällen ist keine Evaluation möglich, da der vorgesehene Speicher der RFID-Tags für diese Simulation nicht ausreicht. Der x-Wert ist dort auf Null gesetzt worden.

Es kann beobachtet werden, dass der höchste Anteil erkannter Tags für eine geringe Dichte und eine hohe Reichweite erreicht werden kann. In diesem Fall muss der Antikollisionsalgorithmus nur wenige Tags erkennen und hat zusätzlich aufgrund der hohen Reichweite viel Zeit dafür. Aber es darf nicht vergessen werden, dass eine geringe Dichte ebenfalls weniger Tags pro Strecke bedeutet, selbst wenn die Detektionsrate hoch ist. Dies kann aus der zweiten Grafik entnommen werden. Bei hoher Dichte und hoher Reichweite können die meisten Tags erkannt werden, obwohl die Detektionsrate gering ist.

6.3.3.5 Robustheit

Als letztes sollte untersucht werden, wie sich MS verhält, wenn Fehler auftreten. Zu diesem Zweck wurden zufällig 5%, 10%, 20% und 30% der Tags entfernt. Die sonstigen Voraussetzungen entsprechen denen, die im vorherigen Abschnitt genannt wurden. Die Wegnahme von Tags ist eine der größten Herausforderungen, die man an MS stellen kann, da dies zu Lücken im Pfad führen kann. Dies ist kritisch, da die Funktionsweise von MS darauf basiert, dass die Daten vorherig erkannter und gelesener Tags genutzt werden.

Abbildung 6.20 zeigt das Ergebnis dieser Simulation. Die Anzahl der erkannten Tags sinkt, da sich auch die Gesamtanzahl der erkennbaren Tags reduziert. Für höhere Geschwindigkeiten kann kein substanzieller Unterschied beobachtet werden. Dennoch führt das Entfernen vieler Tags zu Lücken im Pfad. Bei geringeren Geschwindigkeiten kann daher gesehen werden, dass die Anzahl der maximal erkannten Tags prozentual ein wenig mehr als die Anzahl der erkennbaren Tags sinkt. Dennoch kommt es zu keinem signifikanten Einbruch der Detektionsrate.

Dabei muss beachtet werden, dass die Reduzierung der Tag-Dichte auch einen positiven Effekt auf das Inventory hat. Da weniger Tags in Reichweite sind, sinkt die Detektionsdauer, so dass die Detektionswahrscheinlichkeit je Tag steigt.

6.4 Ergebnisse

Der Vergleich verschiedener Algorithmen bei der Geschwindigkeitssimulation hat gezeigt, dass die Kombination aus Q2 mit JTC und 2@1 das beste Verhalten aufweist. Dennoch kann Q2 die Leistung unter Umständen auch verschlechtern, wenn wie bei unbalancierten ID-Verteilungen Anfragen gestellt werden, die zu keinen Antworten führen. Dieser Nachteil erscheint im Gegensatz zu den Vorteilen, die der Algorithmus ansonsten verschafft, jedoch unwesentlich.

Die genauere Untersuchung von unbalancierten ID-Bäumen hat gezeigt, dass die Breitensuche eine geringere durchschnittliche Detektionszeit bei sehr langezogenen Bäumen aufweist. Dieser Vorteil verringert sich jedoch rasch, sobald die Bäume ausgeglichener werden. Bei eher balancierten Bäumen ist dagegen die Tiefensuche überlegen.

Zusammenfassend muss festgehalten werden, dass beim Vergleich der ID-Distributionen die inkrementierten IDs am besten abschneiden. Dies kann darauf zurückgeführt werden, dass diese balancierten Bäumen am gleichmäßigsten aufgebaut sind und die meisten Informationen über die Positionen der identifizierenden Knoten bieten. Zudem besteht die Möglichkeit, zwei Tags in einem Schritt zu detektieren. Die unbalancierten Bäume stellen ein Inventory dagegen vor deutlich größere Herausforderungen und sind daher weniger geeignet. Zufällig verteilte IDs erlauben mit den getesteten Algorithmen eine Leistung, die im Mittelfeld liegt.

Die Benutzung der Maskenspeicherung hat zwei wesentliche Vorteile. Zum einen erhöht sich die Geschwindigkeit, da es nicht länger erforderlich ist, eine zeitaufwendige Suche durch den gesamten ID-Baum durchzuführen. Zweitens müssen weniger Tags detektiert werden, da die gespeicherten Masken von Beginn an zwischen gewünschten und unnötigen Tags unterscheiden. Dies zeigt sich am deutlichsten bei der Geschwindigkeitssimulation. Sie zeigt, dass der Ansatz der Maskenspeicherung den anderen Algorithmen überlegen ist. Dies steigert sich weiterhin, wenn die Anforderungen dahingehend geändert werden, dass nur noch eine Teilmenge von Tags erkannt werden soll. Die Vorteile der Maskenspeicherung erkauft man sich jedoch mit einem höheren Speicherbedarf auf den Tags.

Die Vergrößerung der Pfadbreite (und Anpassung der Reichweite) führt zu signifikant mehr erkannten Tags bei geringeren Geschwindigkeiten. Die Anzahl fällt aber ab, wenn

die optimale Geschwindigkeit überschritten wurde. Daher sollte dies nicht in Betracht gezogen werden, wenn ein konstantes Verhalten nötig ist.

Durch die Skalierung der Dimensionen wird auch das Ergebnis skaliert. Wenn Reichweite, Dichte und Pfadlänge vergrößert werden, wächst auch die Anzahl der detektierten Tags linear. Dies ist ein sehr vielversprechendes Ergebnis, da es die Übertragung eines kleineren Modells auf eine reale Anwendung ermöglicht.

Bei einer vorgegebenen Reichweite wird die beste Leistung der Detektionsrate dann erreicht, wenn der Tag-Abstand auf einen ähnlichen Wert gesetzt wird. Ist die Effektivität der Tag-Nutzung nicht das oberste Ziel, können die meisten Tags bei einer hohen Dichte und einer großen Reichweite erreicht werden. Dies führt allerdings zu einem erhöhten Speicherverbrauch, wenn Maskenspeicherung benutzt wird.

Das Fazit kann gezogen werden, dass die Reichweite und die Dichte jeweils nur einen geringen Einfluss auf die Detektionsrate haben und nur die Kombination beider Parameter einen wesentlichen Einfluss auf das Verhalten hat. Um höhere Geschwindigkeiten zu erreichen, ist es nötig, alle Dimensionen zu skalieren.

Der Test der Robustheit liefert das interessante Ergebnis, dass der MS-Algorithmus bei hohen Geschwindigkeiten kaum durch induzierte Fehler beeinflusst wird. Auf der anderen Seite zeigt dies, dass die obere Grenze der detektierbaren Tags erreicht ist und dass die Anzahl der Tags in Reichweite keinen Einfluss mehr auf das Ergebnis hat.

Nicht simuliert wurde, wie sich die Maskenspeicherung verhält, wenn Tags fehlerhafte Daten enthalten. Dies hat jedoch vergleichbare Konsequenzen, wie der Ausfall von Tags. Wurden fehlerhafte Daten gelesen, können im nächsten Durchlauf nicht die gewünschten Tags erkannt werden, weil die falschen Masken entweder keine oder irrelevante Tags liefern. Wie beim Ausfall von Tags erfordert dies einen vollständigen Inventory-Durchlauf.

Zusammenfassend empfiehlt sich die Verwendung der Maskenspeicherung, wenn genügend Speicherplatz auf den Tags vorhanden ist. Des Weiteren muss eine Möglichkeit vorhanden sein, während der Teach-Phase die benötigten Masken zu bestimmen. Können diese Voraussetzungen nicht erfüllt werden, sollte ein Algorithmus verwendet werden, der wie Q2 mit initialen Masken arbeitet. Die Optimierungen JTC und 2@1 sollten immer benutzt werden.

7 Zusammenfassung

7.1 Fazit

In vielen Bereichen werden heutzutage autonome Transportsysteme eingesetzt und benötigt. Eine der großen Herausforderungen hierbei ist die Umsetzung des Navigationsverfahrens. Beschreibbare RFID-Tags bieten sich als eine Möglichkeit an, um ein flexibles und leicht installierbares System zu schaffen.

Es existieren verschiedene Ansätze, die RFID für Navigationssysteme nutzen. Dabei wird meist die ID eines Tags genutzt, um eine Position zu erkennen. Eine weitere Möglichkeit besteht darin, den Pfadverlauf betreffende Informationen direkt in den Tags abzulegen. Dies wird ebenfalls in einigen Ansätzen verwandt. Dennoch sind keine Verfahren bekannt, die eine kontinuierliche Navigation nur anhand der Nutzung dezentral hinterlegter Daten realisieren und auf weitere Infrastruktur sowie eine absolute Lokalisierung verzichten. Ziel dieser Arbeit war es, die Realisierbarkeit eines solches Systems nachzuweisen und die notwendigen Voraussetzungen zu überprüfen.

Es wurde ein allgemeiner Ansatz entwickelt, der als Basis für Algorithmen dienen kann, die das Schreiben und Folgen eines Pfades umsetzen. Mittels einer Testumgebung wurde gezeigt, dass ein Roboter mit zwei Lesegeräten und einer Richtungsbestimmung (beispielsweise mittels Kompass) in der Lage ist, einen Pfad zu schreiben, der zu einem späteren Zeitpunkt kontinuierlich und autonom verfolgt werden kann. Werden weniger als zwei Lesegeräte bzw. Antennen genutzt, können auftretende Positionierungsungenauigkeiten nicht erkannt werden und führen zu häufigen Pfadverlusten, die ein flüssigen Folgen verhindern. Die Richtungsbestimmung ist ebenfalls eine wichtige Voraussetzung. Des Weiteren kann die Verwendung einer eingeschränkten Pfadvorhersage (beispielsweise mittels Odometrie) nötig sein, wenn die Tagdichte in Bezug auf die Drehrate zu gering wird.

Es stellte sich heraus, dass die erreichbare Fahrgeschwindigkeit der Roboter im Wesentlichen durch die Kommunikationsgeschwindigkeit des Lesegerätes mit den Tags limitiert wird. In diesem Zusammenhang ist das Inventory von Bedeutung, welches den Kommunikationsaufbau ermöglicht. Seine Aufgabe besteht darin, die IDs der erreichbaren Tags zu erkennen. Die Dauer des Inventorys hängt vom verwendeten Algorithmus und der Anzahl der zu bestimmenden Tags ab. Im weiteren Verlauf der Arbeit wurden Ansätze untersucht, um das Inventory und damit einhergehend die Fahrgeschwindigkeit zu beschleunigen.

Es zeigte sich, dass die vorhandenen Algorithmen, die auf der Traversierung des Baums der möglichen Tags-IDs basieren, für Navigationssysteme nicht die besten Ergebnisse

liefern. Es wurden verschiedene Optimierungen eingeführt, um diese Algorithmen zu verbessern.

- Durch Analyse der Tag-Antworten während des Inventorys können unnötige Anfragen ausgelassen werden. Dies wird im wesentlichen durch die Benutzung der Manchesterkodierung ermöglicht, welche die bitgenaue Identifizierung von Kollisionen erlaubt. Derartige Verbesserung können in fast allen Ansätzen verwendet werden, die Kollisionen behandeln.
- Aufgrund der ortsfesten Tags kann sich zu Nutze gemacht werden, dass die ungefähre Anzahl der zu erwartenden Tags bekannt ist. Dadurch können während des Inventory-Vorgangs von Beginn an wahrscheinliche Zweige geprüft und die Anzahl der Anfragen reduziert werden.
- Eine weitere Option besteht darin, über die Verteilung der Tag-IDs Einfluss auf den Verlauf des Inventorys zu nehmen. Auch dies kann in Betracht gezogen werden, weil die Tags stationär sind. Als Verteilungen sind IDs möglich, die balancierte Bäume (durch inkrementierte Tag-IDs) oder unbalancierte Bäume formen. Auch die Vergabe zufälliger IDs ist denkbar.

Des Weiteren wurde ein neuer Algorithmus entwickelt, der durch Benutzung von Teilen der Tag-IDs (Masken) das Inventory speziell für die untersuchten Navigationssysteme beschleunigt. Die benötigten Masken werden dazu jeweils auf den zuvor liegenden Tags gespeichert und erlauben so die schnellere Identifizierung der folgenden zugehörigen Tags.

Die Eignung der betrachteten Algorithmen und der Optimierungen wurde mittels Simulation evaluiert. Hiermit wurden auch die Einflüsse untersucht, die sich durch verschiedene Reichweiten der Lesegeräte und die Dichte der Tags ergeben. Die simulierte Umgebung entsprach der des Testaufbaus.

Die eingeführten Optimierungen zeigten gegenüber dem ursprünglichen Algorithmus (der in ISO-15693 beispielhaft angegeben wurde) deutliche Verbesserungen. Insbesondere die zusätzliche Verwendung initialer Masken von zwei Bit Länge (Q2) führte in den meisten Fällen zu der höchsten Anzahl erkannter Tags.

Im Vergleich mit der Maskenspeicherung, bei deren Simulation neben der Tag-Detektion auch das Auslesen von Daten berücksichtigt wurde, zeigt die Maskenspeicherung allerdings die besten Ergebnisse. Dies wurde insbesondere deutlich, wenn nur eine Teilmenge der erreichbaren Tags erkannt werden sollte. Nachteilig bei der Maskenspeicherung ist, dass das Schreiben eines Pfades komplexer ist und mehr Daten auf den Tags abgelegt werden müssen. Letzteres kann dann kritisch werden, wenn viele weitere Daten abgelegt werden sollen.

Die Verwendung spezieller Tag-ID-Distributionen (wie unbalancierte Bäume) wies keine besonderen Vorteile auf. Dennoch hat sich gezeigt, dass die Verteilung einen erkennbaren Einfluss auf die Leistung des Inventorys hat. Die schnellste Erkennung ist bei der Vergabe inkrementierter IDs möglich. Für die Leistung der Maskenspeicherung ist die Wahl der ID-Distribution nicht von besonderer Relevanz. Hier wird durch die Verteilung der IDs nur die Länge der zu speichernden Masken beeinflusst. Soll eine spezielle

Tag-Distribution verwendet werden, verliert man die Freiheit, die Tags bei der Einrichtung der Umgebung beliebig zu verteilen. Die einfachste Möglichkeit verspricht hier die Verteilung zufälliger IDs, die ebenfalls schnell zu erkennen sind.

7.2 Ausblick

In dem Testaufbau wurde die Machbarkeit des angestrebten Systems mit Hilfe zweier RFID-Lesegeräte nachgewiesen. Für den produktiven Einsatz bietet sich dagegen vermutlich die Verwendung von Antennen-Arrays an, um eine genauere Lokalisierung der Spur zu ermöglichen. Dies wurde nicht getestet.

Um die Nutzung in Industrieumgebungen zu ermöglichen, ist es notwendig, über eine verlässliche Richtungsbestimmung zu verfügen. In dieser Arbeit wurde ein Kompass benutzt, welcher sich aber als anfällig für Störungen erwiesen hat. Sind die Richtungsänderungen bezogen auf die physikalischen Lücken im Pfad (die von der Tag-Dichte abhängen) nur gering, kann auch in Betracht gezogen werden, keine separate Richtungsbestimmung zu verwenden und die Fahrtrichtung nur anhand einer genauen Bestimmung des Abstandes zum Pfad zu ermitteln. Anderenfalls sind beispielsweise inertiale Messsysteme in Betracht zu ziehen.

Die simulierten Antikollisionsalgorithmen wurden nur teilweise einem praktischen Test unterzogen (siehe auch [Kha09]). Für eine ausführlichere Betrachtung wird ein geeignetes RFID-Lesegerät benötigt, das den Austausch der Algorithmen erlaubt. Dies ist mit dem OpenPCD-Reader prinzipiell möglich, dennoch funktioniert die Firmware nicht fehlerfrei und die Auslagerung der Ausführung von Algorithmen auf dem PC führt zu zu hohen Latenzen.

Ebenfalls nicht praktisch getestet wurde, wie eine Spur erzeugt werden kann, die für das Inventory mit Maskenspeicherung benötigt wird. In der Simulation wurde der Pfad dazu von Beginn zum Ende abgefahren, wobei mehrere Lesebereiche verwendet, die berechneten Daten aber nur an der Rückseite des Fahrzeuges auf die Tags geschrieben wurden. Alternativ können die benötigten Daten erhoben werden, wenn das Fahrzeug die Strecke in der Teach-Phase in umgekehrter Richtung abfährt. Eine Strecke in entgegengesetzter Richtung zu schreiben wurde zwar getestet, jedoch nicht in Zusammenhang mit der Maskenspeicherung.

Mögliche Optimierungen ergeben sich, wenn ein Pfad mehrfach abgefahren wird. Haben sich die Daten auf den Tags nicht verändert, können die Daten einmal gelesener Tags in einem Cache gespeichert werden und müssen nicht erneut ausgelesen werden. Somit kann eine höhere Folgegeschwindigkeit erreicht werden. Des Weiteren ist es möglich, mit jeder Fahrt die Positionen der Tags genauer abzuschätzen. Ansätze hierfür liefern beispielsweise [HBF+04, PH09, KNS+08]. Diese können für eine verbesserte Positionierung genutzt werden.

A Anhang

Abkürzungsverzeichnis

2@1	Two at once
AFI	Application Family Identifier
CTA	Container Terminal Altenwerder
EPC	Electronic Product Code
FTS	Fahrerlose Transportsysteme
JTC	Jump to collision position
MS	Maskenspeicherung
Q2	Initiale Masken von 2 Bit Länge
RemM	Remember Masks
RFID	Radio Frequency Identification
stdRec	Rekursiver Standard-Algorithmus

Symbolverzeichnis

α	Statistische Sicherheit
c	Anzahl von Kollisionen
$f_X(t)$	Wahrscheinlichkeit, dass ein Tag zum Zeitpunkt t detektiert wird
h	Höhe eines ID-Baumes
k	Anzahl von Knoten eines ID-Baumes
l	Länge einer Tag-ID

N	Anzahl von Slots
n	Anzahl von Tags
R	Rundenanzahl
r	Radius des Lesebereiches des Readers
t	Zeit
t_{Inv}	Zeit, um ein einzelnes Tag zu detektieren
$t_{\mathrm{Inv}}(n)$	Zeit, um ein Tag aus n Tags zu detektieren
t_{min}	Minimale Zeit, die benötigt wird, bevor ein Tag detektiert werden kann
t_{R}	Dauer, die sich ein Tag in Reichweite des Lesegerätes befindet
v	Geschwindigkeit
v_{S}	Geschwindigkeit für sichere Detektion
X	Zeitpunkt, an dem ein Tag detektiert wird
Y	Zeitpunkt, bis zu dem ein Tag detektiert wird

Abbildungsverzeichnis

Literaturverzeichnis

[AC09] ABC-Consulting: *RFID Einsatz für Infrastrukturmessstellen.* Expertise, 2009

[BAQZY06] Biao, L. ; Ai-Qun, H. ; Zhong-Yuan, Q.: Trends and Brief Comments on Anti-collision Techniques in Radio Frequency Identification System. In: *6th International Conference on ITS Telecommunications Proceedings*, 2006

[Bau09] Baum, M.: *Transpondergestütze Fahrzeugleitsysteme.* PZH Produktionstechnisches Zentrum GmbH, 2009

[BM04] Bohn, J. ; Mattern, F.: Super-Distributed RFID Tag Infrastructures. In: *Proceedings of the 2nd European Symposium on Ambient Intelligence (EUSAI).* Eindhoven, Niederlande : Springer-Verlag, 2004

[BNA+07] Baum, M. ; Niemann, B. ; Abelbeck, F. ; Fricke, D. ; Overmeyer, L.: Qualification Tests of HF RFID Foil Transponders for a Vehicle Guidance System. In: *Intelligent Transportation Systems Conference*, 2007

[BT] Bosien, A. ; Turau, V.: *RFID Anti-collision Algorithms for AGV Navigation*

[BT09] Bosien, A. ; Turau, V.: RFID for mobile applications. In: *Proceedings of the Workshop on Self-Organising Wireless Sensor and Communication Networks.* Hamburg, 2009

[BTZ12] Bosien, A. ; Turau, V. ; Zambonelli, F.: Approaches to Fast Sequential Inventory and Path Following in RFID-enriched environments. In: *International Journal of Radio Frequency Identification Technology and Applications*, Inderscience, 2012

[Bun04] Bundesamt für Sicherheit in der Informationstechnik (Hrsg.): *Risiken und Chancen des Einsatzes von RFID-Systemen.* SecuMedia Verlags-GmbH, 2004

[BVT08] Bosien, A. ; Venzke, M. ; Turau, V.: A rewritable RFID environment for AGV navigation. In: *Proceedings of the 5th International Workshop on Intelligent Transportation (WIT).* Hamburg, 2008

[CeB09] CeBIT Forum AutoID/RFID: RFID bei der METRO Group. In: *RFID im Blick* (2009). `http://www.cebit-forum.rfid-im-blick.de/2009020988/Sponsoren/METRO-Group/rfid-bei-der-metro-group.html`, Abruf: 29.10.2011

[CJK05] Choi, H. ; J., J. C. ; Kim: Improved Bit-by-Bit Binary Tree Algorithm in Ubiquitous ID System. In: Aizawa, K. (Hrsg.) ; Nakamura, Y. (Hrsg.) ; Satoh, S. (Hrsg.): *Advances in Multimedia Information Processing*. Springer Berlin / Heidelberg, 2005

[CSK08] Cho, J. ; Shin, J. ; Kim, S.: RFID Tag Anti-Collision Protocol: Query Tree with Reversed IDs. In: *10th International Conference on Advanced Communication Technology*, 2008. – ISSN 1738–9445

[Dem07] Demmel, M.: Automatische Spurführung von Landmaschinen - Systeme, Einsatzbereiche, Wirtschaftlichkeit. (2007). `http://www.alb-bayern.de/wissen/Landtechniktag_2007/Spurfuehrungssysteme_Demmel.pdf`, Abruf: 29.10.2011

[DNRK10] Deyle, T. ; Nguyen, H. ; Reynolds, M. ; Kemp, C.: RFID-Guided Robots for Pervasive Automation. In: *Pervasive Computing, IEEE*, 2010. – ISSN 1536–1268

[Eng01] Engels, D.: The reader collision problem / Auto-ID Labs. 2001. – Forschungsbericht

[Fac04] Fachhochschule Köln: Intelligenter Kühlschrank und Verschlusszeitenmessung von Digitalkameras auf dem Tag der Technik im MediaPark / Fachhochschule Köln. 2004. – Forschungsbericht

[Fin06] Finkenzeller, K.: *RFID Handbuch*. Carl Hanser, 2006

[FM05] Fleisch, E. (Hrsg.) ; Mattern, F. (Hrsg.): *Das Internet der Dinge*. Springer, 2005

[Fut08] Future Shape Pressemitteilung: Navi Floor - wissen, wo es lang geht. (2008). `http://www.future-shape.de/downloads/press/NF_d_3650.pdf`, Abruf: 29.10.2011

[GH07] Guo, Z. ; Hu, B.: A Dynamic Bit Arbitration Anti-Collision Algorithm for RFID System. In: *International Workshop on Anti-counterfeiting, Security, Identification*, IEEE, 2007

[GKT10] Gressmann, B. ; Klimek, H. ; Turau, V.: Towards Ubiquitous Indoor Location Based Services and Indoor Navigation. In: *Proceedings of the 7th IEEE Workshop on Positioning Navigation and Communication (WPNC2010)*. Dresden, 2010

[Gö10] Götting KG: *1,5-dim. Positionier- und Identsystem outdoor*. Datenblatt, 2010

[HBF+04] Hähnel, D. ; Burgard, W. ; Fox, D. ; Fishkin, K. ; Philipose, M.: Mapping and Localization with RFID Technology. In: *In Proceedings of International Conference on Robotics and Automation*, 2004

[Hes09] Hess, E.: RFID-Technologie und Kryptographie. (2009). `http://www.afcea.de/fileadmin/downloads/Mittagsforen/06.11.2009/Hess.pdf`, Abruf: 29.10.2011

[Hol87] Hollier, R. (Hrsg.): *Automated Guided Vehicle Systems*. Springer, 1987

[Hua06] Huang, X.: An Improved ALOHA Algorithm for RFID Tag Identification. In: Gabrys, B. (Hrsg.) ; Howlett, R. (Hrsg.) ; Jain, L. (Hrsg.): *Knowledge-Based Intelligent Information and Engineering Systems*. Springer Berlin / Heidelberg, 2006

[ISO99] ISO/IEC: *ISO/IEC 14443-3 Part 3: Initialization and anticollision*. 1999

[ISO00] ISO/IEC: *ISO/IEC FCD 15693-3 Part 3: Anti-collision and transmission protocol*. 2000

[KCR10] Klair, D. ; Chin, K. ; Raad, R.: A Survey and Tutorial of RFID Anti-Collision Protocols. In: *IEEE Communications Surveys & Tutorials*, 2010

[Kha09] Khan, S.: *Optimization of RFID Anti-Collision Algorithms*, Technische Universität Hamburg-Harburg, Diplomarbeit, 2009

[KKLA08] Kim, S. ; Kim, Y. ; Lee, S. ; Ahn, K.: An Improved Anti Collision Algorithm using Parity Bit in RFID System. In: *Seventh International Symposium on Network Computing and Applications*, IEEE, 2008

[KKPS08] Kämpke, T. ; Kluge, B. ; Prassler, E. ; Strobel, M.: Robot Position Estimation on a RFID-Tagged Smart Floor. In: Laugier, C. (Hrsg.) ; Siegwart, R. (Hrsg.): *Field and Service Robotics*. Springer Berlin / Heidelberg, 2008

[KNS+08] Kodaka, K ; Niwa., H. ; Sakamoto, Y. ; Otake, M. ; Kanemori, Y. ; Sugano, S.: Pose estimation of a mobile robot on a lattice of RFID tags. In: *International Conference on Intelligent Robots and Systems (IROS)*, IEEE/RSJ, 2008

[KYK08] Koh, H. ; Yun, S. ; Kim, H.: Sidewalk: A RFID Tag Anti-Collision Algorithm Exploiting Sequential Arrangements of Tags. In: *International Conference on Communications*, IEEE, 2008

[Lah06] Lahiri, S.: *RFID Sourcebook*. IBM Press, 2006

[Lan05a] Landt, J.: The history of RFID. In: *IEEE Potentials*, 2005

[Lan05b] *Kapitel* Die Privatsphäre im Ubiquitous Computing - Datenschutzaspekte der RFID-Technologie. In: Langheinrich, M.: *Das Internet der Dinge*. Springer, 2005

[LL06] Liu, L. ; Lai, S.: ALOHA-Based Anti-Collision Algorithms Used in RFID System. In: *International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, 2006

[LLS00] Law, C. ; Lee, K. ; Siu, K.-Y.: Efficient memoryless protocol for tag identification. In: *Proceedings of the 4th international workshop on Discrete algorithms and methods for mobile computing and communications*, ACM, 2000

[LNC05] Leong, K. ; Ng, M. ; Cole, P.: The reader collision problem in RFID systems. In: *International Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications (MAPE)*, IEEE, 2005

[LXXL05] Liu, L. ; Xie, Z. ; Xi, J. ; Lai, S.: An Improved Anti-collision Algorithm in RFID System. In: *2nd International Conference on Mobile Technology, Applications and Systems*, IEEE, 2005

[LYP+09] Lee, E. ; Yoo, Y. ; Park, C. ; Kim, M. ; Gerla, M.: Installation and evaluation of RFID readers on moving vehicles. In: *Proceedings of the sixth international workshop on VehiculAr InterNETworking*. New York, USA : ACM, 2009. – ISBN 978–1–60558–737–0

[Man09] Manteuffel, F.: *Navigation von Robotern in einer RFID-Umgebung*, Technische Universität Hamburg-Harburg, Bachelorarbeit, 2009

[Mat03] *Kapitel* Vom Verschwinden des Computers - Die Vision des Ubiquitous Computing. In: Mattern, F.: *Total vernetzt*. Springer, 2003

[Meh88] Mehlhorn, K.: *Datenstrukturen und effiziente Algorithmen*. B. G. Teubner Stuttgart, 1988

[Met06] Metro AG: Der Dynamik des globalen Handels begegnen. In: *RFID-Newsletter* (2006). `http://www.future-store.org/fsi-internet/html/de/2366/index.html`, Abruf: 29.10.2011

[MLS06] Myung, J. ; Lee, W. ; Shih, T.: An Adaptive Memoryless Protocol for RFID Tag Collision Arbitration. In: *IEEE Transactions on Multimedia*, 2006

[MLSS07] Myung, J. ; Lee, W. ; Srivastava, J. ; Shih, T.: Tag-Splitting: Adaptive Collision Arbitration Protocols for RFID Tag Identification. In: *IEEE Transactions on Parallel and Distributed Systems*, 2007. – ISSN 1045–9219

[MQZ06] Mamei, M. ; Quaglieri, R. ; Zambonelli, F.: Making tuple spaces physical with RFID tags. In: *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*. New York, USA : ACM, 2006. – ISBN 1–59593–108–2

[MTS08] Melski, A. ; Thoroe, L. ; Schumann, M.: RFID – Radio Frequency Identification. In: *Informatik-Spektrum*, Springer Berlin / Heidelberg, 2008. – ISSN 0170–6012

[MZ05] Mamei, M. ; Zambonelli, F.: Physical deployment of digital pheromones through RFID technology. In: *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2005

[MZ07] Mamei, M. ; Zambonelli, F.: Pervasive pheromone-based interaction with RFID tags. In: *Transactions on Autonomous and Adaptive Systems*. New York, USA : ACM, 2007. – ISSN 1556–4665

[NBOF06] Niemann, B. ; Baum, M. ; Overmeyer, L. ; Fricke, D.: Aufbau von fahrerlosen Transportsystemen (FTS) durch eine dezentrale Datenstruktur. In: *Logistics Journal*, 2006

[OBNF06] Overmeyer, L. ; Baum, M. ; Niemann, B. ; Fricke, D.: Prüftechnik für RFID bei Hochgeschwindigkeitsanwendungen im Transportwesen. In: *Logistics Journal*, 2006

[Pec08] Peck, M.: RFID Tags Guide the Blind. In: *IEEE Spectrum* (2008). `http://spectrum.ieee.org/biomedical/devices/rfid-tags-guide-the-blind`, Abruf: 29.10.2011

[PFP+04] Philipose, M. ; Fishkin, K. ; Perkowitz, M. ; Patterson, D. ; Fox, D. ; Kautz, H. ; Hahnel, D.: Inferring activities from interactions with objects. In: *IEEE Pervasive Computing*, 2004. – ISSN 1536–1268

[PH09] Park, S. ; Hashimoto, S.: Autonomous Mobile Robot Navigation Using Passive RFID in Indoor Environment. In: *IEEE Transactions on Industrial Electronics*, 2009. – ISSN 0278–0046

[RABB07] Randall, J. ; Amft, O. ; Bohn, J. ; Burri, M.: LuxTrace: indoor positioning using building illumination. In: *Personal and Ubiquitous Computing*, Springer London, 2007. – ISSN 1617–4909

[RGD+09] Raoui, Y. ; Goller, M. ; Devy, M. ; Kerscher, T. ; Zollner, J. ; Dillmann, R. ; Coustou, A.: RFID-based topological and metrical self-localization in a structured environment. In: *International Conference on Advanced Robotics (ICAR)*, 2009

[RK09] Roussos, G. ; Kostakos, V.: RFID in pervasive computing: State-of-the-art and outlook. In: *Pervasive and Mobile Computing* (2009). – ISSN 1574–1192

[Sed02] Sedgewick, R.: *Algorithmen*. Pearson Education, 2002

[Sto48] Stockman, H.: Communication by Means of Reflected Power. In: *Proceedings of the Institue of Radio Engineers*, 1948. – ISSN 0096–8390

[Tex00] Texas Instruments: RFID Tracks Containers Throughout Europe. In: *RFID News* (2000). `http://www.ti.com/rfid/docs/manuals/RFIDNews/Tiris_NL20.pdf`, Abruf: 29.10.2011

[Vor06] Vorwerk Pressemitteilung: Vorwerk präsentiert RFID smart floor auf der CeBIT. (2006). `http://www.vorwerk-teppich.de/sc/vorwerk/vorwerk_cebit_2006_rfid.html`, Abruf: 29.10.2011

[Wis07] Wischnewski, R.: *Transportsysteme mit Spurführung in der Virtuellen Produktion*. VDI Verlag GmbH, 2007

[ZLTT10] Zhang, X. ; Lakafosis, V. ; Traille, A. ; Tentzeris, M.: Performance analysis of fast-moving RFID tags in state-of-the-art high-speed railway systems. In: *IEEE International Conference on RFID-Technology and Applications (RFID-TA)*, 2010

[ZT11] Zhang, X. ; Tentzeris, M.: Applications of Fast-Moving RFID Tags in High-speed Railway Systems. In: *International Journal of Engineering Business Management*, 2011

Publikationsliste

- Arne Bosien, Marcus Venzke und Volker Turau. A rewritable RFID environment for AGV navigation. In *Proceedings of the 5th International Workshop on Intelligent Transportation (WIT'08)*. Hamburg, März 2008.
- Arne Bosien und Volker Turau. RFID for mobile applications. In *Proceedings of the Workshop on Self-Organising Wireless Sensor and Communication Networks*. Hamburg, Oktober 2009.
- Arne Bosien, Volker Turau und Franco Zambonelli. Approaches for fast sequential inventory and path following in RFID-enriched environments
  Special Issue on Advances in RFID Technology, *International Journal of Radio Frequency Identification Technology and Applications, Vol. 4, No. 1*, Inderscience, 2012
- Arne Bosien und Volker Turau. RFID Anti-collision Algorithms for AGV Navigation im Review-Verfahren für: *The Open Transportation Journal*

Lebenslauf

Persönliche Daten

Name	Bosien
Vorname	Arne
Geburtsdatum	13. Mai 1981
Geburtsort	Henstedt-Ulzburg, Deutschland

Schulbildung

1987 - 1991	Grundschule Rönneburg in Hamburg
1991 - 1997	Gesamtschule Harburg in Hamburg
1997 - 2000	Gymnasiale Oberstufe der Gesamtschule Harburg in Hamburg Abschluss: Abitur

Zivildienst

08.2000 - 06.2001	Mobile soziale Hilfsdienste in Hamburg

Studium

10.2001 - 03.2007	Informatik-Ingenieurwesen, Technische Universität Hamburg-Harburg Abschluss: Diplom-Ingenieur

Promotion

04.2007 - 12.2009	Wissenschaftlicher Mitarbeiter Technische Universität Hamburg-Harburg
01.2010 - 07.2010	Promotionsstipendiat Università degli Studi di Modena e Reggio Emilia, Italien
08.2010 - 03.2011	Promotionsstudent, Technische Universität Hamburg-Harburg

Berufstätigkeit

seit 03.2011	Softwareingenieur, Xcontrol GmbH