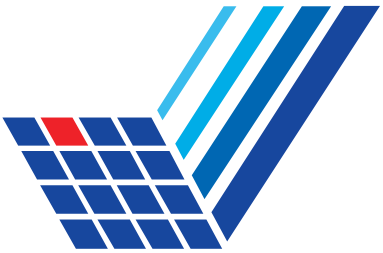
UNIVERSITÄT DORTMUND
FACHBEREICH INFORMATIK

Diplomarbeit

Planung, Implementierung und Evaluierung eines Algorithmus zur Lokalisierung von Gabelstaplern

Michael Wachter

INTERNE BERICHTE
INTERNAL REPORTS



Diplomarbeit
am Fachbereich Informatik
der Universität Dortmund

4. September 2007

Gutachter:

Prof. Dr.-Ing. Uwe Schwiegelshohn
Prof. Dr.-Ing. Gernot A. Fink

GERMANY · D-44221 DORTMUND

Inhaltsverzeichnis

Kapitel 1

Einleitung

In diesem Kapitel wird eine kurze Einleitung in die Thematik der Gabelstaplerlokalisierung gegeben. Dazu wird geklärt, warum sie überhaupt benötigt wird und welche grundsätzlichen Möglichkeiten es hierzu gibt. Danach wird die Aufgabenstellung wiedergegeben und die Gliederung der Diplomarbeit deutlich gemacht.

1.1 Motivation

Die Verfolgung von Waren vom Erzeuger bis zum Verbraucher wird nicht nur aufgrund neuer gesetzlicher Regelungen immer wichtiger. Im Lebensmittelbereich muss es beispielsweise möglich sein, verdorbene Ware komplett zurückzurufen. Zum Zwecke einer Chargenverfolgung muss also für jede Einheit jederzeit feststellbar sein, wo die Ware produziert wurde, wo sie gerade lagert oder ob sie schon verkauft wurde.

Auf ihrem Weg zum Verbraucher wird die Ware häufig in großen Verteilerzentren zwischengelagert. In diesen Umschlaglagern wird die Ware auf Paletten angeliefert, mit Hilfe von Gabelstaplern abgeladen, auf einem freien Platz im Lager zwischengelagert, um dann später wieder auf einen anderen LKW aufgeladen zu werden.

Um dabei eine lückenlose Chargenverfolgung zu gewährleisten, muss die Chargenverfolgungssoftware nachhalten können, welche Waren angekommen sind, wo sie sich gerade im Lager befinden und auf welchen LKW sie am Ende aufgeladen wurden. Dies kann man entweder durch manuelles Eintragen in eine Datenbank oder über automatisches Lokalisierungssystem erreichen.

Dazu könnte man nun jede Palette mit einem Lokalisierungssystem versehen, was aber mit einem erheblichen Aufwand verbunden wäre. Da allerdings die Paletten meistens sehr schwer sind, werden sie ausschließlich mit Gabelstaplern bewegt. Zur Lokalisierung der Paletten reicht es daher aus, die Position des Gabelstaplers beim Auf- und Abladen der Paletten zu kennen. Aus dieser Position und mittels zusätzlicher Sensoren zur Erkennung des Auf- und Abladevorgangs sowie einer Kennzeich-

nung der Palette über Barcodes oder RFID, lässt sich dann die Position der Paletten nachhalten.

1.2 Verschiedene Lokalisierungssysteme

Um nun die Position eines Fahrzeuges zu bestimmen, gibt es verschiedene Möglichkeiten. Außerhalb von Gebäuden ist es am einfachsten, das Global Positioning System GPS zu verwenden. Dieses ermöglicht theoretisch eine Genauigkeit von ca. 1m, ist aber vom amerikanischen Militär für die allgemeine Nutzung auf eine Genauigkeit von ca. 15m eingeschränkt. Diese Einschränkung kann mit „differential GPS“ überwunden werden. Dabei wird die Differenz zwischen dem GPS-Empfänger auf dem Fahrzeug und einer fest installierten Referenzstation gemessen und damit die relative Position des Fahrzeugs berechnet [1]. Dadurch kann nun wieder eine Genauigkeit von ca. 1m erreicht werden.

Innerhalb von Gebäuden ist das GPS-System wegen der starken Signaldämpfung nicht zu gebrauchen, weshalb es andere Systeme gibt, die eine Lokalisierung mit „Pseudolites“ ermöglichen. Pseudolites sind dabei in der Halle montierte Sender, mit denen es einem Empfänger möglich ist, seine Position über die Messung der Signallaufzeit zu mehreren der Pseudolites zu berechnen. Die Firma Symeo stellt ein solches System her, das Genauigkeiten im Bereich von 10cm ermöglicht. Ein anderes solches Projekt ist „Locata“ von Locata Corporation und der University of New South Wales, das Genauigkeiten um 1cm ermöglicht [2]. Der Nachteil dieser Systeme ist der hohe Hardwareaufwand, der zu hohen Kosten führt. So sind bei einer Beispielsinstallation [1] mit dem Symeo-System zur Gabelstaplerlokalisierung in einer Halle 12 Pseudolites verwendet worden.

Günstigere Lokalisierungssysteme, die auch in einer Halle funktionieren, arbeiten auf Basis von WLAN-Accesspoints [3]. Dabei wird versucht, anhand von Signalstärke mehrerer Accesspoints die Position des Fahrzeuges zu bestimmen. Da hier nur mit der Signalstärke und nicht mit der Laufzeit gearbeitet wird, wird nur eine Genauigkeit von 3,25m erreicht.

Wieder andere Lokalisierungssysteme arbeiten mit Ultraschall. Das Cricket-System [4] besteht aus aktiven Sendern, die z.B. an der Hallendecke angebracht sind, und einem passiven Empfänger, der über die Laufzeit des Signals den Abstand zu jedem Sender berechnet, um daraus die Position zu bestimmen. Mit diesem System wurden in [5] Genauigkeiten von 45cm erreicht.

Des weiteren gibt es Laserscanner, die schon ein komplettes Lokalisierungssystem beinhalten und an ihrer Schnittstelle Positionsdaten liefern. Der SICK NAV 200 [6] verwendet dazu Reflektoren, die an festen Stellen in der Halle angebracht werden. Hat der Laserscanner mindestens 3 Reflektoren zuverlässig erkannt, so kann er seine eigene Position in der Halle berechnen. Leider ist es gerade in einem oben beschrie-

[1] http://www.symeo.com/LPR-A_DE.pdf

benen Lager häufig nicht möglich, genügend Reflektoren zu detektieren, da die Ware zwischen Laserscanner und Reflektor stehen kann. Eine Möglichkeit, möglichst viele Reflektoren zu detektieren, ist es daher, den Laserscanner so auf eine Stange zu montieren, dass dieser die Reflektoren über die Ware hinweg sehen kann. Dieses ist bei einem langsam fahrenden Fahrzeug wie in Abbildung 1.1 möglich. Bei einem schnell gefahrenen Gabelstapler würde der Laserscanner so stark wackeln, dass er Probleme hätte, die Reflektoren zuverlässig zu erkennen. Daher ist es sinnvoll, die Markierungen für das Lokalisierungssystem entweder auf dem Boden oder an der Decke anzubringen.

Abbildung 1.1: Der Nav200 wurde auf einer langen Stange montiert, damit er die Reflektoren sehen kann. [6]

Dies macht z.B. das NorthStar-System der Firma Evolution-Robotics. Es verwendet zur Lokalisierung an die Hallendecke projizierte Infrarot-Markierungen, welche dann von einer Infrarotkamera auf dem Fahrzeug erfasst werden. Dabei soll eine Genauigkeit von bis zu 15cm erreicht werden.

In dieser Diplomarbeit sollen Magnete als Markierungen im Boden verwendet werden, die im Darüberfahren mittels Magnetfeldsensoren erkannt werden. Diese haben den Vorteil, dass sie einfach im Boden angebracht werden können und in der Regel, d.h. überall dort, wo der Gabelstapler fahren kann, nicht von anderen Gegenständen verdeckt sind. Zur Erkennung dieser Magnete wurde von der Firma Innovent ein Magnetfeldsensor entwickelt, der in dieser Diplomarbeit als Hauptsensor eingesetzt wird. Dieser Sensor ist dabei eine modifizierte Variante eines in der Medizin verwendeten Systems zur Darmuntersuchung [7].

Zum Tracking der Position auch zwischen den Magneten, sollen Odometriesensoren verwendet werden, um die Positionsänderung zu berechnen. Die Magnet-Markierungen werden somit dazu benötigt, die Position einmal zu finden und dann immer wieder zu verifizieren.

1.3 Aufgabenstellung

Ziel der Diplomarbeit ist es, einen Lokalisierungsalgorithmus für einen Gabelstapler zu entwickeln, der mit Hilfe des Magnetfeldsensors und Odometriesensoren die Position des Gabelstaplers berechnet.

Dazu müssen zunächst einige Odometriesensoren ausgewählt werden, die dann zusammen mit dem Magnetfeldsensor auf einem Testfahrzeug montiert werden.

Das Testfahrzeug ist danach so mit Hardware und Software auszustatten, dass es möglich ist, die Sensoren auszulesen und den Lokalisierungsalgorithmus zu entwickeln.

Schließlich muss ein Lokalisierungsalgorithmus ausgewählt und implementiert werden, der dann auf seine Brauchbarkeit für das Gabelstapler-Lokalisierungsproblem untersucht wird.

1.4 Gliederung der Arbeit

Die Diplomarbeit gliedert sich in zwei große Bereiche:

Zuerst werden in Kapitel 2 das Testfahrzeug, die zur Verfügung stehenden Sensoren sowie das Sensorinterface vorgestellt. Das zugehörige zur Entwicklung verwendete Software-Framework wird in Kapitel 3 beschrieben.

Ab Kapitel 4 wird nun der Lokalisierungsalgorithmus beschrieben. Zuerst werden verschiedene Ansätze vorgestellt und die Auswahl eines bestimmten Algorithmus begründet. Danach wird dessen konkrete Implementierung in Kapitel 5 beschrieben und in Kapitel 6 die Ergebnisse der Evaluierung vorgestellt.

Kapitel 2

Hardware

In diesem Kapitel wird die verwendete Hardware beschrieben. Die Hardware besteht aus einem Testfahrzeug, an dem verschiedenen Sensoren samt Ausleseelektronik montiert sind.

2.1 Sensoren

Es wurden verschiedene Sensoren zur Verfügung gestellt, die auf ihre Brauchbarkeit zur Lokalisierung getestet werden sollen. Diese unterteilen sich dabei in Odometriesensoren und den Magnetfeldsensor. Die Odometriesensoren sollen es ermöglichen, möglichst genau die Bewegung des Gabelstaplers zu messen. Außerdem sollten die Sensoren wegen der rauhen Fahrweise von Gabelstaplerfahrern möglichst robust sein, da z.B. in der Lagerhalle herumliegende Verpackungsteile überfahren werden können, die die Sensorik nicht zerstören dürfen.

Die meisten dieser Sensoren senden ihre Messdaten digital über den „CAN-Bus“. Dieser ist ein im Automobilbereich sehr verbreitetes asynchrones, serielles Bussystem, das von Bosch zur Vernetzung von Mikrocontrollern und Sensoren entwickelt wurde. Der CAN-Bus benötigt drei Kupferadern und ermöglicht eine Datenübertragungsrate von bis zu 1 Mbit/s.

Die Daten selber werden in Paketen, so genannten CAN-Frames versandt. Diese bestehen aus einer Kennung, einem Längenfeld, maximal 8 Byte Daten und einem CRC Prüfwert. Die Kennung bezeichnet dabei den Inhalt des Frames, nicht den Empfänger. Ein Sensor z.B. sendet seine Messwerte immer mit der Kennung 0xC0 und alle Steuerrechner am CAN-Bus, die den Messwert dieses Sensors benötigen, warten auf CAN-Frames mit dieser Kennung. Außerdem dient die Kennung des CAN-Frames zur Prioritätsbestimmung. Niedrigere Kennungen haben eine höhere Priorität.

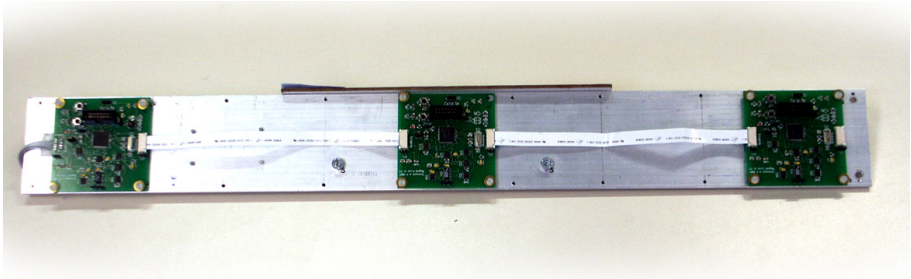


Abbildung 2.1: Das Magnetlineal: Drei Magnetfeldsensoren detektieren den Magneten. Es lässt sich somit feststellen, ob und wo über einen Magneten gefahren wurde.

2.1.1 Magnetlineal

Als Markierungen zur Lokalisierung sollen Magnete verwendet werden. Um zu erkennen, ob der Gabelstapler über einen Magneten gefahren ist, werden mehrere Magnetfeldsensoren verwendet, die in einer Reihe an einem Aluminium-Profil angebracht sind. Diese Anordnung ist in Abbildung 2.1 gezeigt und wird im Folgenden als „Magnetlineal“ bezeichnet.

Das Magnetlineal wurde von der Firma Innovent geliefert und in einer komplexeren Variante ursprünglich in der Medizin für Darmuntersuchungen verwendet [7].

Das Magnetlineal existiert in zwei Versionen. Eine frühere Version lieferte die Messwerte als analoge Signale, die noch mit einem zusätzlichen A/D-Wandler abgetastet werden mussten. Dies führte wegen der Kabelwege vom Sensor bis zum A/D Wandler zu starkem Rauschen. Daher wurde eine Version mit integriertem A/D-Wandler und digitalem Interface entwickelt.

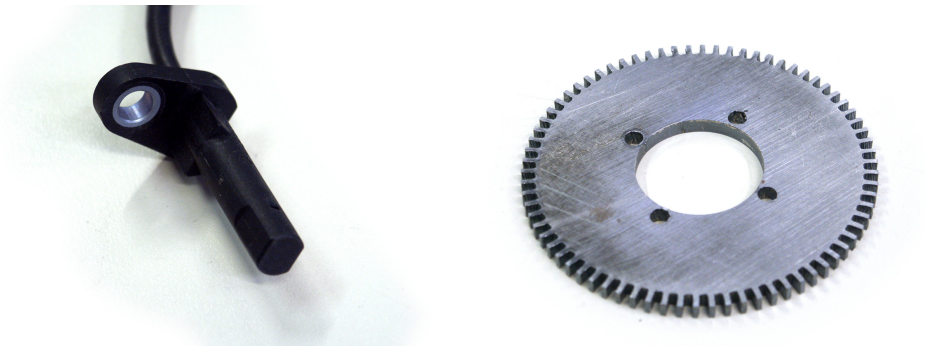
Die überarbeitete Version liefert ihre Sensormesswerte über ein CAN-Interface und benötigt keine weitere externe Beschaltung. Sie arbeitet zwar noch mit dem gleichen Magnetfeldsensor HMC 1020Z von Honeywell, wegen der kurzen analogen Signalwege vom Sensor zum integrierten A/D-Wandler ist nun der SNR ca. 15dB höher.

In einem Abstand von ca. 30cm vom Boden montiert, ist es mit dem Magnetlineal anhand der Messwerte der hier drei Sensoren möglich, nicht nur zu erkennen, ob sich das Magnetlineal über einen Magneten bewegt hat, sondern auch, wo genau sich der Magnet unter dem Magnetlineal zu diesem Zeitpunkt befand.

2.1.2 Radsensoren

Zur Bestimmung des zurückgelegten Weges des Fahrzeuges sind Radsensoren notwendig, die die Umdrehungen der Räder möglichst genau bestimmen. Außerdem sollten sie möglichst robust sein.

Daher werden zur Messung der Radrotation Radsensoren vom Typ DF11i von Bosch [8] verwendet. Dieser in Abbildung 2.2(a) zu sehende Radsensor wird auch von vielen



(a) Der Radsensor (b) Die Encoderscheibe

Abbildung 2.2: Das Radsensorsystem besteht aus dem Radsensor DF11i und einer Encoderscheibe

Automobilherstellern für ASP und ESP eingesetzt. Er ist in einem robusten Gehäuse untergebracht und bietet eine hinreichende Genauigkeit.

Der DF11i arbeitet mit einem Magneten, Hallsensoren und einer Encoderscheibe (Abbildung 2.2(b)) und kann erkennen, ob sich ein Zahn der Encoderscheibe in der Nähe befindet oder nicht. Dieses Messverfahren bietet den Vorteil, dass die Encoderscheibe selbst nicht magnetisiert werden muss und somit relativ einfach passend für das Testfahrzeug gebaut werden kann. Außerdem bietet der Sensor wegen seiner zwei Hallsensoren die Möglichkeit, ebenfalls die Drehrichtung zu bestimmen.

Wann immer die Kante eines Zahns der Encoderscheibe an dem Sensor vorbeikommt, generiert dieser ein Signal, dessen Länge die Drehrichtung, Stillstand oder einen zu großen Sensorabstand angibt. Die einzelnen Impulslängen dafür sind in Tabelle 2.1 angegeben. Durch Mitzählen der Signale und den Reifenumfang lässt sich somit die zurückgelegte Strecke berechnen.

Signallänge	Bedeutung
45μs	Abstand zwischen Sensor und Encoderscheibe zu groß
90μs	Schnelle Linksdrehung
180μs	Schnelle Rechtsdrehung
360μs	Langsame Linksdrehung
720μs	Langsame Rechtsdrehung
1440μs	Stop-Signal (wird alle 700ms wiederholt)

Tabelle 2.1: Signallängen des Radsensors und deren Bedeutung

Die Encoderscheibe wurde passend zu den im Testfahrzeug (Kapitel 2.2) herrschenden Platzverhältnissen hergestellt, wobei die Anzahl der Zähne so gewählt wurde, dass der Sensor sie gerade noch zuverlässig detektiert und somit die höchstmögliche Auflösung erzielt wird. Diese Auflösung beträgt bei einem Raddurchmesser von 20cm ca. 4,1 mm gefahrene Strecke.

Der DF11i verfügt nur über zwei Anschlüsse für seine Versorgungsspannung von 12V. Das Signal wird vom Sensor über seine Stromaufnahme übertragen. Dabei entspricht eine Stromaufnahme von 5,9mA bis 8,5mA einem Low-Pegel und eine von 11,8mA bis 16,8mA einem High-Pegel.

2.1.3 Lenkwinkelsensor

Es ist möglich, nur mit Hilfe der Radsensoren die Positionsänderung des Gabelstaplers auch in Kurvenfahrten zu berechnen, aber es ist anzunehmen, dass sich durch die zusätzliche Verwendung eines Lenkwinkelsensors bessere Ergebnisse erzielen lassen.

Daher wurde der Lenkwinkelsensor LWS40 von Bosch [9] (Abbildung 2.3) ausgewählt, um den Lenkwinkel zu messen. Er bietet die Möglichkeit, die Winkelstellung der Lenkachse eines Fahrzeuges mit einer Genauigkeit von 0,1° zu bestimmen. Außerdem ist er „Multi-Turn-fähig“, d.h. er kann mehrere Umdrehungen der Lenkachse unterscheiden. Sein Messbereich erstreckt sich von -780° bis +780°. Die Messdaten liefert er über sein CAN-Interface.

Eine besondere Fähigkeit dieses Sensors gegenüber ähnlichen Sensoren ist es, sofort nach dem Einschalten den richtigen Lenkwinkel zu liefern. Er muss also nur bei mechanischen Änderungen am Testfahrzeug neu kalibriert werden. Dies geschieht über einen speziellen CAN-Frame, der an den Sensor gesendet werden muss. Der Sensor nimmt dann die aktuelle Position als 0°-Position an. Da der Lokalisierungsalgorithmus sehr auf die Daten des Lenkwinkelsensors vertraut, muss diese Kalibrierung mit großer Sorgfalt vorgenommen werden.

Abbildung 2.3: Der Lenkwinkelsensor LWS40 liefert die Winkelstellung der Lenkachse mit einer Genauigkeit von 0,1°

2.1.4 Beschleunigungssensor

Desweiteren könnte bei der Odometrieberechnung ein Beschleunigungssensor sinnvoll sein, um die mit den anderen Sensoren berechnete Odometrie zu verifizieren.

Der Beschleunigungssensor SC-MM3.22 von Bosch [10] ist ein „Sensor-Cluster“, der mit mehreren Sensoren die Beschleunigung und Drehrate misst (Abbildung 2.4). Die Beschleunigung wird in Fahrtrichtung (X-Achse) und senkrecht dazu (Y-Achse) gemessen, die Drehrate für eine Drehung um die Z-Achse. Die Messwerte werden im Sensor digitalisiert und über ein CAN-Interface ausgegeben.

Die Beschleunigung wird dabei in einem Bereich von -2,8g bis +2,8g mit einer Genauigkeit von 5% erfasst, die Drehrate im Bereich von -125°/s bis +125°/s mit einer Genauigkeit von 4%.

Abbildung 2.4: Der Bosch MM3 ist ein Sensorcluster mit Beschleunigungs- und Drehratensensoren

Abbildung 2.5: Das Testfahrzeug

2.2 Das Testfahrzeug

Da ein richtiger Gabelstapler für Tests in der Halle des Instituts für Roboterforschung zu groß und unhandlich ist, musste ein vereinfachtes Testfahrzeug gebaut werden. Dieses Fahrzeug musste die Möglichkeit bieten, die verschiedenen Sensoren, Elektronik zum Auslesen, Referenzsysteme sowie die Spannungsversorgung zu montieren. Außerdem sollte das Fahrzeug einfach zu fertigen sein, damit es in der internen Werkstatt des Instituts gebaut werden konnte.

Aus diesen Anforderungen heraus wurde ein „Bollerwagen“ gewählt. Er wurde aus Bosch-Profil gefertigt, damit die Sensoren und die Aufbaufläche einfach zu montieren waren. In Abbildung 2.5 ist dieses Testfahrzeug gezeigt.

Die Unterschiede zu einem Gabelstapler sind nicht so gravierend. Zwar ist die Lenkgeometrie an einem Gabelstapler mit der dort verwendeten Ackermannlenkung komplexer und durch den Verzicht auf den Antrieb müssen auch keine eventuell durchdrehenden Räder berücksichtigt werden, dies ist aber bei der Implementierung nicht so wichtig, da sich nur die Berechnung der Odometrie und einige Parameter des Lokalisierungsalgorithmus ändern.

Die Größe des Bollerwagens wurde so gewählt, dass der Wagen möglichst klein und wendig wird. Dies ist nötig, da als Referenzsystem unter anderem eine Deckenkamera verwendet wird und somit der Bereich, in dem getestet werden kann, nur ca. 4m x 6m groß ist. Somit ergaben sich die in Tabelle 2.2 angegebenen Dimensionen des Fahrzeugs.

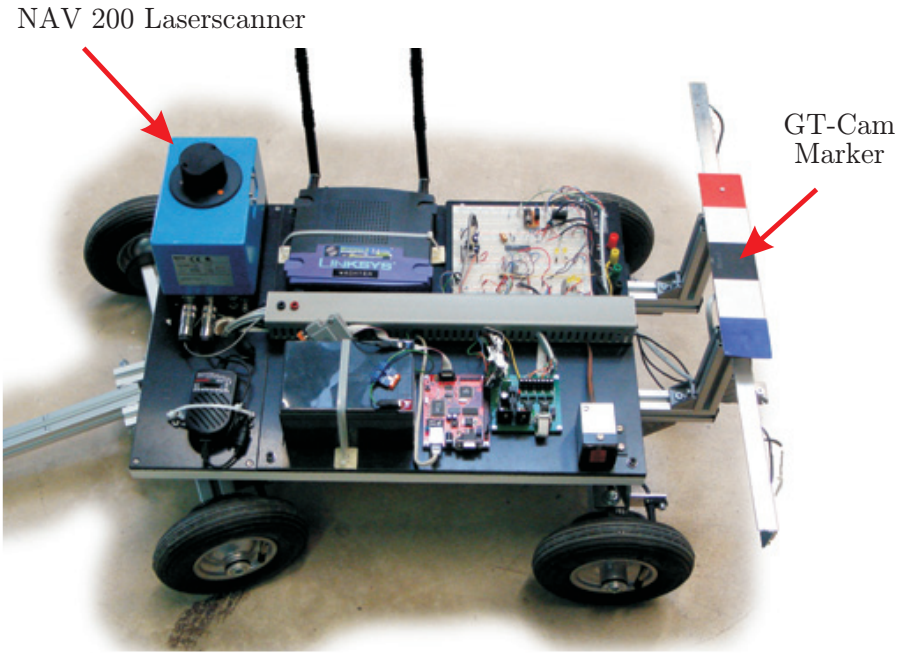


Abbildung 2.6: Der NAV200 und das GTCam Deckenkamarasystem dienen als Referenz zum Test des Lokalisierungsalgorithmus

2.3 Referenzsysteme

Um die Qualität der Lokalisierung überprüfen zu können, sind zwei Referenzsysteme vorhanden.

Der SICK-NAV 200 (Abbildung 2.6) kann Reflektoren erkennen und beinhaltet ein Lokalisierungssystem, das, ausgehend von den bekannten Positionen der Reflektoren, seine eigene Position berechnet. Dabei gelingt es ihm, falls er genügend Reflektoren sieht, seine Position bis auf eine Genauigkeit von 2 cm zu bestimmen.

Als zweites Referenzsystem dient die „GT-CAM“, das Deckenkamerasystem der Microsoft Hellhounds [11], [12], das ursprünglich als Referenzsystem für Laufevolution im Roboterfußball entwickelt wurde. Dieses System besteht aus zwei Deckenkameras, die einen Marker (Abbildung 2.6) auf den Rücken der Roboter erkennen können. Zur richtigen Berechnung der Position muss die Höhe des Markers bekannt sein. Da sich die Markerhöhe beim Testfahrzeug von der beim Roboterfußball unterscheidet,

Länge (bei heruntergeklappter Lenkstange)	150 cm
Breite	67 cm
Höhe	43 cm
Aufbaufläche Länge	60 cm
Aufbaufläche Breite	40 cm
Reifendurchmesser	20 cm

Tabelle 2.2: Grundlegende Dimensionen und Daten des Testfahrzeuges

wurde das GT-CAM Programm so angepasst, dass die Höhe des Markers für jeden Marker getrennt angegeben werden kann. Die Genauigkeit der GT-CAM beträgt ca. 7 mm.

2.4 Sensorboard

Um den Rechner, auf dem der Lokalisierungsalgorithmus läuft, mit den Sensordaten versorgen zu können, wurde ein System entwickelt, das die Messwerte der verschiedenen Sensoren ausliest, vorverarbeitet und über WLAN zur Verfügung stellt. Diese Daten können dann von dem Rechner, auf dem der Lokalisierungsalgorithmus läuft, empfangen und weiterverarbeitet werden, wobei dieser nicht auf dem Testfahrzeug montiert sein muss.

In Abbildung 2.7 ist dieses System dargestellt. Wo sich die einzelnen Komponenten auf dem Testfahrzeug befinden, ist in Abbildung 2.8 zu sehen.

Den Kern bildet dabei das „Ethernut-Board“ [13]. Dies ist ein Embedded-Board mit einem Ethernet-Interface und einem ATMEGA 128 Prozessor. Es hat den Vorteil, dass es verschiedene Anschlussmöglichkeiten für die Sensoren bietet und außerdem ein passendes Betriebssystem vorhanden ist, welches die Ansteuerung des Ethernet-Interfaces bereits enthält.

Obwohl das Ethernut-Board über verschiedene Anschlussmöglichkeiten verfügt, ist trotzdem weitere Hardware zum Anschluss der Sensoren erforderlich:

- Die Anbindung des CAN-Busses an das Sensorboard wird mit einem MCP2515 [14] realisiert. Der MCP2515 ist ein CAN-Bus Controller, der über eine SPI-Schnittstelle mit dem Mikrocontroller kommuniziert. Die SPI-Schnittstelle ist eine einfache und dennoch schnelle serielle Schnittstelle, die auch auf dem ATMEGA 128 vorhanden ist. Somit können mit dem Ethernut-Board CAN-Frames gesendet und empfangen werden.
- Zum Auslesen der Radsensoren wurde von der elektronischen Werkstatt des IRF eine Schaltung entwickelt, die die Stromaufnahme misst und in passende Pegel für den ATMEGA 128 umwandelt. Diese Signale wurden mit interruptfähigen Eingängen des ATMEGA verbunden, so dass eine präzise Zeitmessung der Signallängen möglich ist.
- Der NAV-200 ist an eine der integrierten seriellen Schnittstellen des ATMEGA 128 angebunden. Da der ATMEGA aber mit anderen Signalpegeln als die RS232 Schnittstelle arbeitet, werden diese Pegel mit einem MAX232 entsprechend angepasst.

Außerdem wurde das Ethernut-Board mit einem WRT54GS Access-Point verbunden. Damit ist eine drahtlose Übertragung der Messdaten an den Entwicklungsrechner möglich.

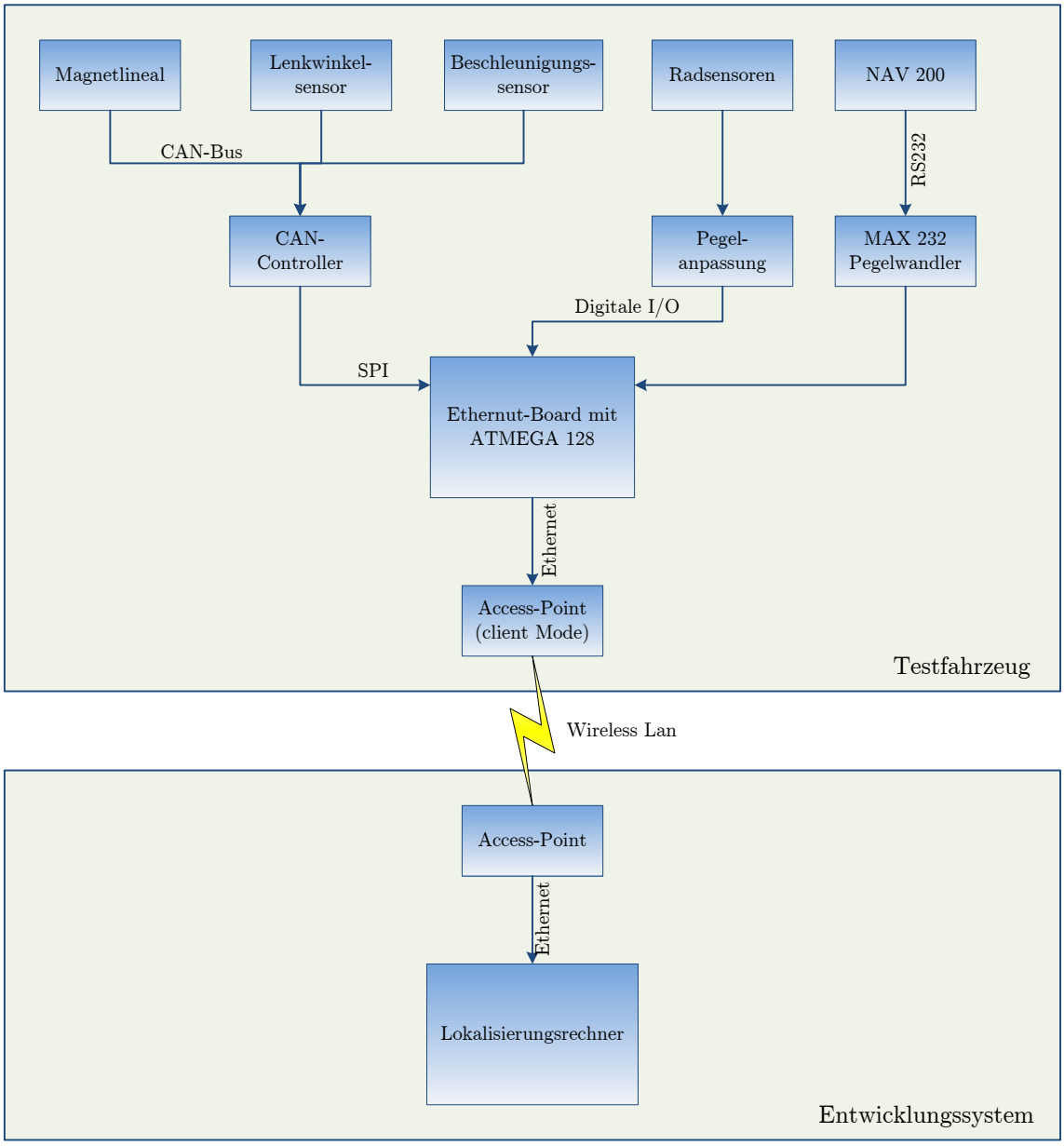


Abbildung 2.7: Hardwareaufbau: Die Sensoren werden über einige Interface-Bausteine vom Ethernut-Board ausgelesen. Dieses sendet die Messdaten über Ethernet und Wireless-LAN zu dem Rechner, auf dem der eigentliche Lokalisierungsalgorithmus läuft

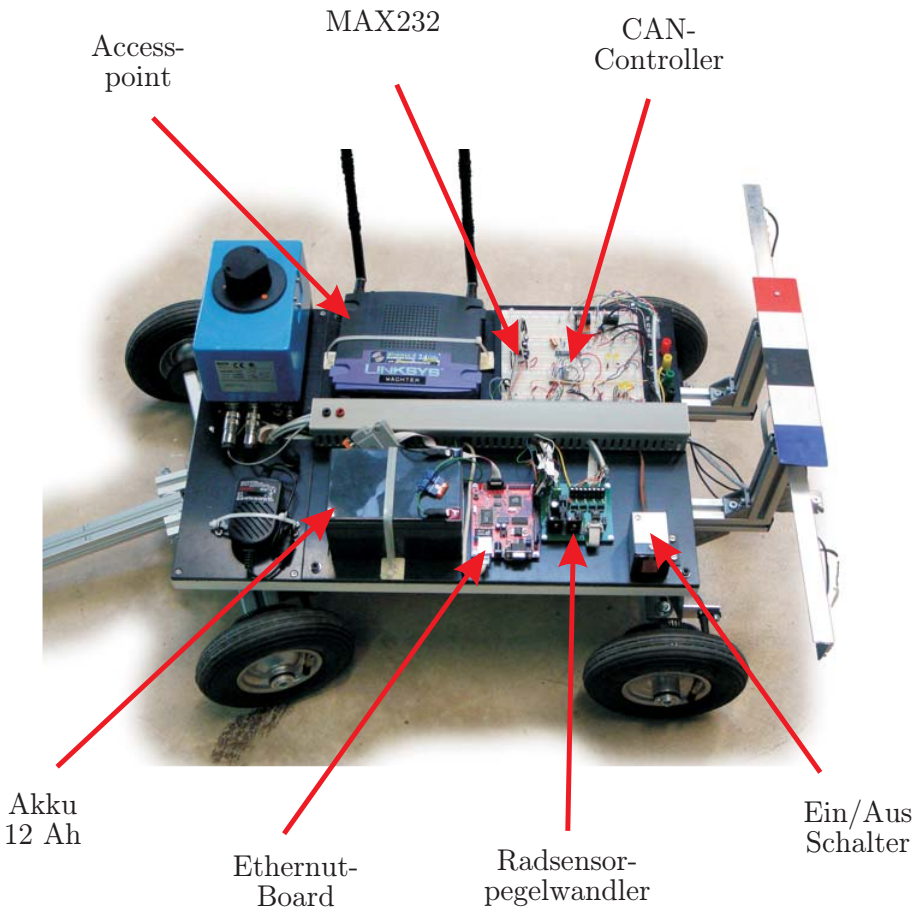


Abbildung 2.8: Positionen der einzelnen Komponenten des Systems zur Übertragung der Sensordaten

Die Stromversorgung für all diese Komponenten wird über einen 12Ah Blei-Gel-Akku sichergestellt. Insgesamt hat das Testfahrzeug eine Stromaufnahme von ca. 1,7A, was eine Laufzeit von ca. 7 Stunden ermöglicht.

Kapitel 3

Software-Framework

Für den Lokalisierungsalgorithmus ist eine Ausführungsumgebung notwendig, die ein rasches Implementieren und ausgiebiges Testen ermöglicht.

Daher wurde der Ansatz gewählt, die Messwerte der Sensoren auf dem Testfahrzeug nur auszulesen und dann auf den Entwicklungsrechner zu übertragen, auf dem der Lokalisierungsalgorithmus läuft. Dies bietet den Vorteil, dass sich der Aufwand, eine neue Version des Algorithmus zu starten, auf einen Tastendruck in der Entwicklungsumgebung beschränkt und es außerdem möglich ist, die Debugging-Möglichkeiten der Entwicklungsumgebung zu nutzen.

Zum ausgiebigen Testen ist es sinnvoll, die Eingangsdaten, Zwichenergebnisse sowie Ausgangsdaten des Algorithmus visualisieren und Parameter ändern zu können. Außerdem ist es sinnvoll, eine Möglichkeit zum Aufzeichnen und Wiedergeben von Messwerten sowie einen Simulator für das Testfahrzeug zur Verfügung zu haben.

Aus diesen Erfordernissen heraus wurde ein zweigeteiltes Software-Framework implementiert. Zum Auslesen der Sensoren wird ein Programm benötigt, das die Messwerte der Sensoren ausliest, vorverarbeitet und über Netzwerk versendet. Dieses Programm heißt „SensorRead“ und läuft auf dem in Kapitel 2.4 beschriebenen Sensorboard.

Auf dem Entwicklungsrechner läuft das Programm „CargoControl“, das die oben beschriebene Kontroll-, Simulations- und Debugfunktionalität enthält und in das der Lokalisierungsalgorithmus eingebettet ist.

3.1 SensorRead

Das Sensorboard, auf dem SensorRead läuft, basiert auf einem ATMEGA 128, einem 8 Bit Microcontroller mit 128kb Flash Rom zur Aufnahme von Programmen.

Der ATMEGA kann sowohl in Assembler als auch in C programmiert werden, wobei mehrere Compiler zur Verfügung stehen. Wegen seiner freien Verfügbar-

keit wurde das WINAVR-Packet verwendet. Es enthält einen GCC-Cross-Compiler für die ATMEGA Architektur sowie alle weiteren zur Erzeugung von ATMEGA-Programmen unter Windows notwendigen Dateien und Bibliotheken. Der Build-Prozess wurde in Microsoft Visual-Studio integriert. Das hat den Vorteil, dass für SensorRead und CargoControl die selbe Entwicklungsumgebung benutzt werden kann und wichtige Funktionen von Visual-Studio, wie das Syntax-Highlighting und IntelliSense, somit auch für die Programmierung des ATMEGA zur Verfügung stehen.

Normalerweise benötigen Programme für den ATMEGA kein Betriebssystem. Auf dem Ethernut-Board ist aber eine Ethernet-Schnittstelle vorhanden, deren Benutzung ein solches notwendig macht. Als Betriebssystem kommt hier „Nut-OS“ zum Einsatz, da es frei zur Verfügung steht und die Netzwerkfunktionalität integriert ist. Neben Treibern für das Netzwerk-Interface bietet Nut-OS außerdem ein einfaches, non-preemtive Scheduling, so dass verschiedene Aufgaben auf verschiedene Threads verteilt werden können.

Nut-OS wird als Library zu dem eigentlichen Programm gelinkt und zusammen mit ihm auf den ATMEGA übertragen.

3.1.1 Framework des Sensorboards

Das Framework des Sensorboards (Abbildung 3.1) teilt sich in drei große Bereiche auf. Zunächst die Schnittstellentreiber, dann die Messwertverarbeitung und schließlich der Versand der Daten.

Schnittstellentreiber werden für die Radsensoren, das CAN-Interface und die serielle Schnittstelle des NAV200 benötigt. Die ersteren mussten dabei implementiert werden, für die serielle Schnittstelle ist der Treiber in Nut-OS vorhanden.

Die von den Schnittstellentreibern gelieferten Daten werden nun weiter verarbeitet, um danach im richtigen Datenformat im Speicher für aktuelle Daten gespeichert zu werden. Dieser Speicher wird in regelmäßigen Zeitabständen von 1/10 Sekunde zu einem Netzwerkpaket konvertiert und mit den von Nut-OS zur Verfügung gestellten Funktionen versandt.

3.1.2 Schnittstellentreiber

Um die Längen der Signale der Radsensoren präzise messen zu können, wurde der ATMEGA so programmiert, dass bei jeder Flanke des Signals ein Interrupt ausgelöst wird. Mittels eines zusätzlichen Timers, der mit der Taktfrequenz des ATMEGA von 14,576 MHz läuft, kann die Länge des Signals so genau gemessen werden, dass die in Kapitel 2.1.2 beschriebenen Signale der Radsensoren voneinander unterschieden werden können. Da die Verarbeitung dieser Sensordaten lediglich im Inkrementieren bzw. Dekrementieren des aktuellen Zählerstandes besteht, wird dieses gleich in

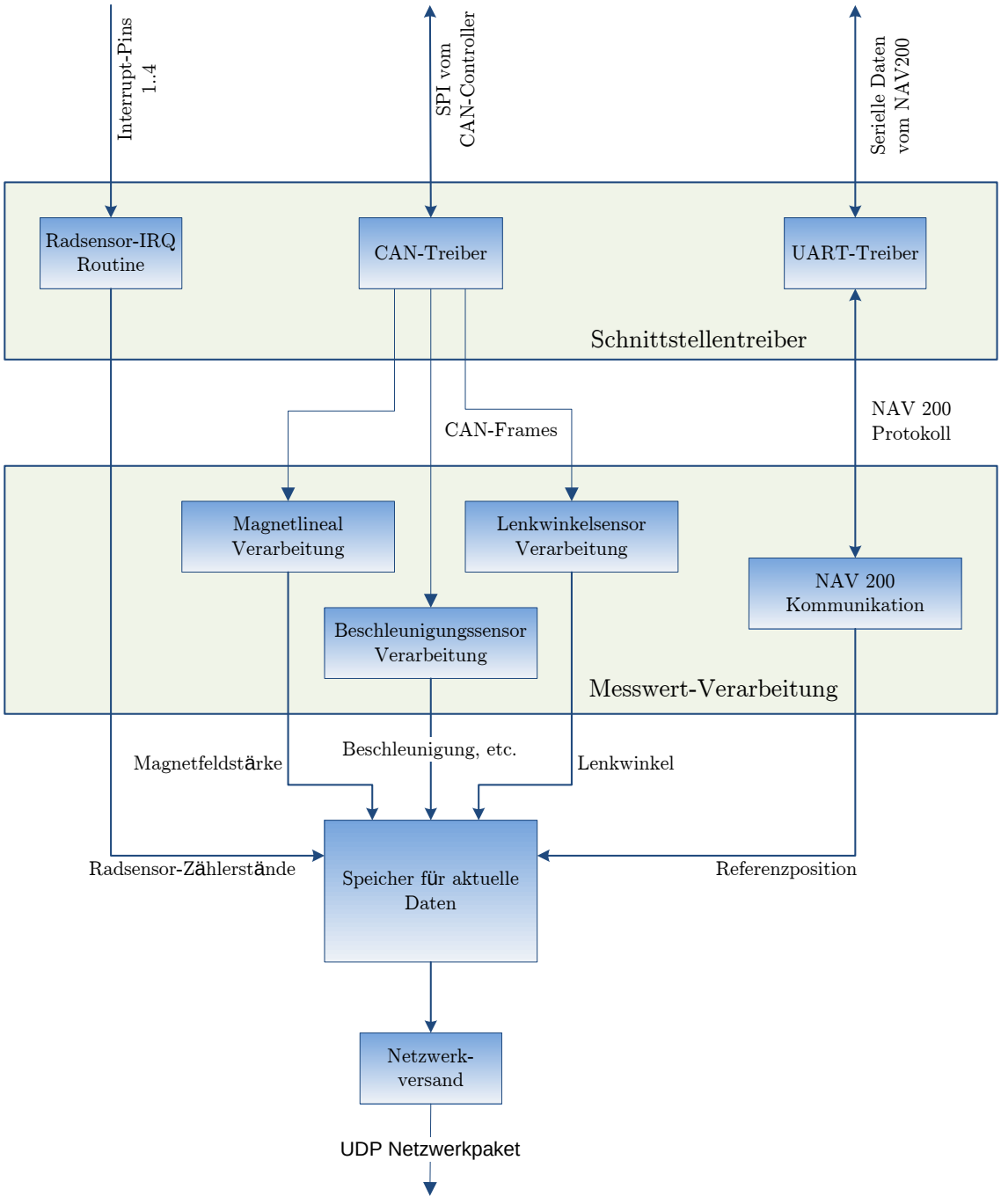


Abbildung 3.1: Das Sensorread-Framework dient zum Auslesen und Versenden der Messdaten der einzelnen Sensoren.

der Interrupt-Routine mit erledigt und es gibt keine weitere Messwert-Verarbeitung mehr für die Radsensoren.

Der CAN-Treiber ist für die Kommunikation mit dem CAN-Controller zuständig. Sobald ein CAN-Frame empfangen wurde, wird abhängig von seiner Frame-ID die entsprechende Funktion zur Verarbeitung dieses CAN-Frames aufgerufen. Diese Funktionen müssen in der Initialisierungsphase des Frameworks beim CAN-Treiber registriert werden.

Der Schnittstellentreiber für die serielle Schnittstelle ist in Nut-OS integriert. Er wird nach der Initialisierungsphase über Standard-C Ein/Ausgabebefehle angesprochen.

3.1.3 Messwert-Verarbeitung

Die verschiedenen von den Sensoren empfangenen CAN-Frames werden in der Messwert-Verarbeitung weiter verarbeitet. Diese Verarbeitung besteht hauptsächlich darin, nur die benötigten Daten der CAN-Frames auszulesen und dabei eventuelle Formatkonvertierungen (Byte-Order) durchzuführen. Die so ermittelten Messwerte werden an die entsprechende Stelle im Speicher für aktuelle Daten geschrieben.

Für das Magnetlineal wird allerdings noch ein Offset berechnet, so dass für alle Magnetfeldsensoren der Wert 0 zurückgeliefert wird, wenn kein Magnet in der näheren Umgebung vorhanden ist. Dazu werden nach der Initialisierung die ersten 16 Messwerte jedes Magnetfeldsensors gespeichert. Danach wird der Mittelwert als Offset verwendet, der dann von allen folgenden Messwerten subtrahiert wird.

Um die aktuelle Position vom NAV200 zu erhalten, muss das entsprechende Kommunikationsprotokoll unterstützt werden. Dieses sieht zunächst eine Initialisierung vor, danach sendet der NAV200 seine Positionsdaten, sobald er einen entsprechenden Befehl erhält. Die empfangenen Daten werden wie bei den anderen Sensoren passend konvertiert und in den Speicher für aktuelle Daten geschrieben.

3.2 CargoControl

CargoControl ist die auf dem Entwicklungsrechner laufende Software, in die der Lokalisierungsalgorithmus eingebettet ist. Damit der Lokalisierungsalgorithmus ausgiebig evaluiert werden kann, enthält CargoControl eine grafische Benutzeroberfläche (Abbildung 3.2), die es ermöglicht, die Sensormesswerte, den internen Zustand und das Ergebnis des Lokalisierungsalgorithmus zu visualisieren und dessen Parameter zur Laufzeit zu verändern.

Zur Entwicklung von CargoControl wurde C# und das Microsoft .NET-Framework verwendet, da dieses eine rasche Entwicklung der GUI und des Lokalisierungsalgorithmus ermöglicht.

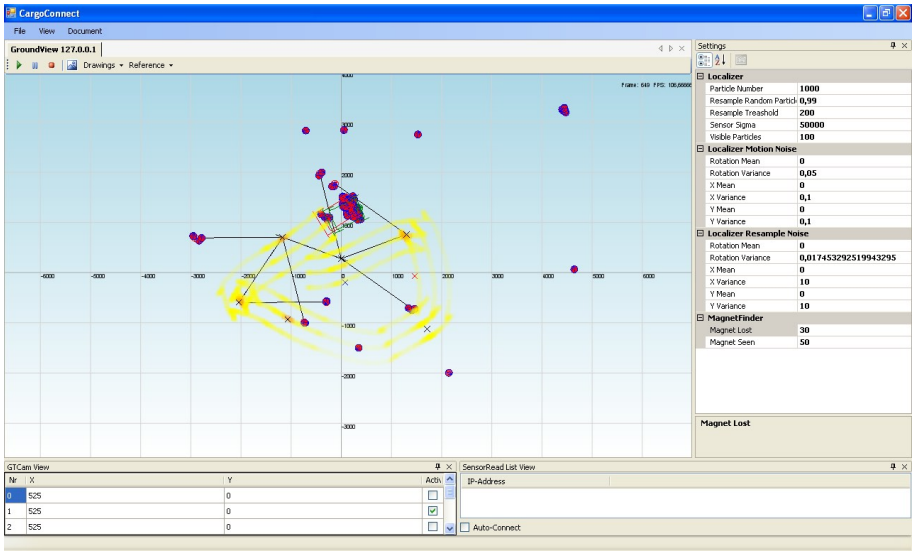


Abbildung 3.2: Cargo-Control Hauptfenster

CargoControl ist nach dem unter Windows verbreiteten Document/View Konzept aufgebaut. Dabei wird die Datenhaltung und die Logik (Document) von der Anzeige (View) der Daten getrennt. Ein Hauptmerkmal von Document/View Anwendungen ist, dass es möglich ist, gleichzeitig verschiedene Dokumente geöffnet zu halten und verschiedene Ansichten (Views) auf diese Dokumente zuzulassen.

3.2.1 SensorReadConnection

Die einzige Art von Dokument ist die SensorReadConnection. In ihr sind der eigentliche Lokalisierungsalgorithmus, die Verbindung zum Sensorboard, der Simulator sowie das Debug-System enthalten.

Der Lokalisierungsalgorithmus hat als Ausgabe nur eine Position. Um auch den internen Status darstellen zu können, gibt es das Debug-System, das alle anzeigbaren oder veränderbaren Werte verwaltet.

Grundsätzlich gibt es dabei drei Arten von Debug-Daten:

- DebugGraphs dienen zum Anzeigen von sich zeitlich verändernden Werten. Der Haupteinsatzzweck ist die Anzeige der Messwerte aller am Testfahrzeug vorhandenen Sensoren. Allerdings ist es auch möglich, sie für beliebige andere Werte, wie z.B. Zwischenergebnisse der Odometrieberechnung zu verwenden.

- DebugDrawings dienen zum Anlegen von Zeichnungen meist aus internen Daten des Lokalisierungsalgorithmus. Es lassen sich z.B. die Positionen der Magnete, die aktuell berechnete Position des Lokalisierungsalgorithmus und viele weitere Informationen zeichnen.

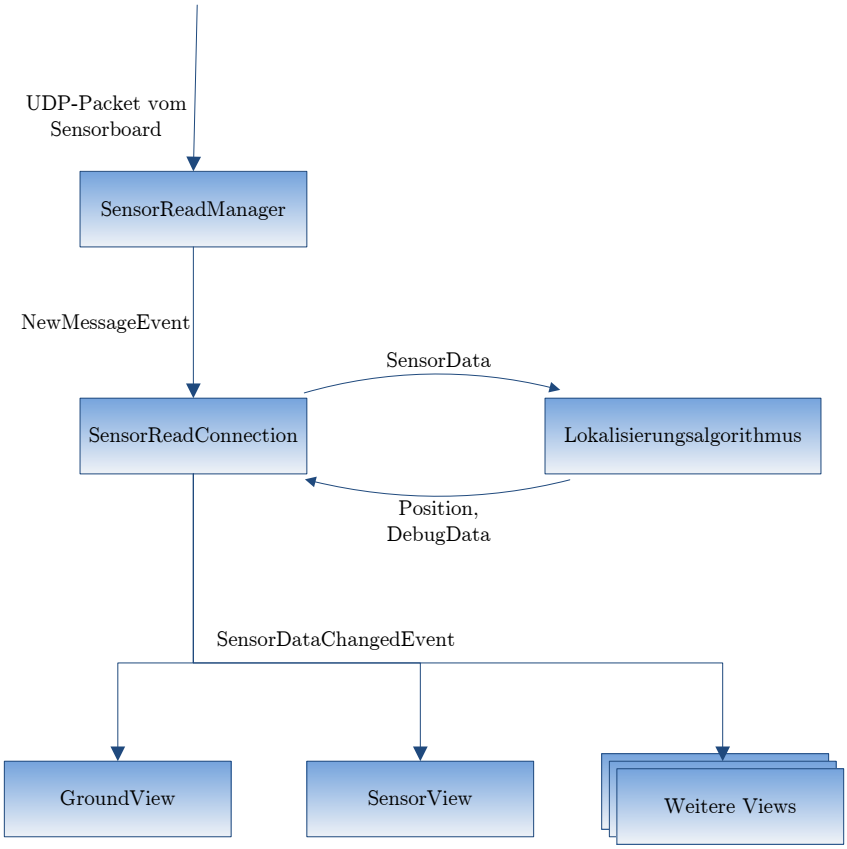


Abbildung 3.3: Verarbeitung eines ankommenden Datenpakets vom Sensorboard.

- DebugValues dienen im Gegensatz zu DebugGraphs und DebugDrawings nicht nur zum Anzeigen, sondern auch zum Verändern von Werten. Dadurch ist es möglich, zur Laufzeit z.B. die Parameter des Lokalisierungsalgorithmus zu ändern, um somit sofort deren Auswirkungen sehen zu können.

In Abbildung 3.3 ist dargestellt, was geschieht, wenn neue Sensormesswerte vom Sensorboard empfangen werden :

1. Der SensorReadMagager empfängt das Paket und löst ein Event aus.
2. Die SensorReadConnection reagiert auf dieses Event.
3. Die DebugGraphs für die Sensormesswerte werden aktualisiert.
4. Der Lokalisierungsalgorithmus wird ausgeführt. Dieser berechnet die Position und aktualisiert weitere Debug-Daten.
5. Es wird ein Event ausgelöst, auf das alle Views warten. Sie greifen auf die Debug-Daten zu und zeichnen ihren Inhalt neu.

Neben der Möglichkeit, die Daten vom Sensorboard zu empfangen, bietet Cargo-Control auch die Möglichkeit, Sensordaten in eine Datei aufzuzeichnen und später

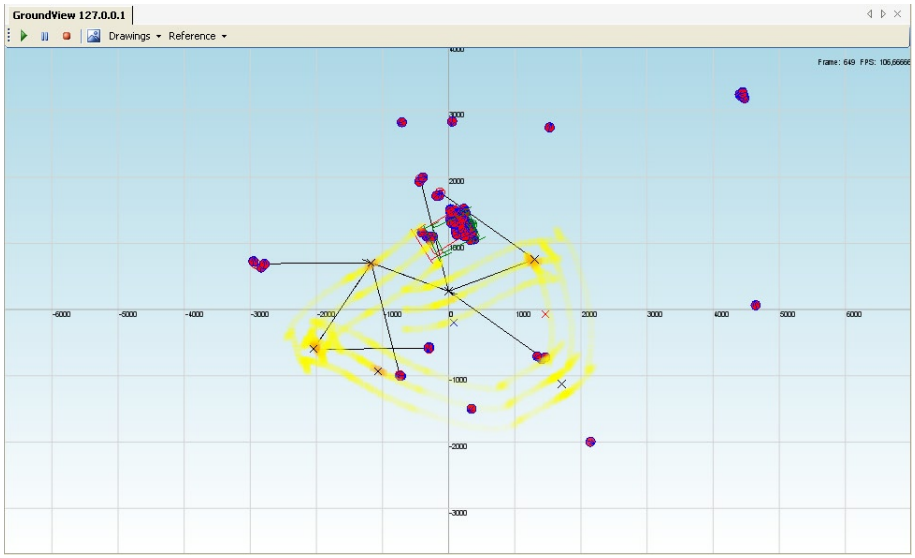


Abbildung 3.4: Der GroundView dient zur Anzeige von Debug-Drawings.

wiederzugeben. Dadurch wird es möglich, die Auswirkungen von verschiedenen Varianten oder Parametrisierungen des Lokalisierungsalgorithmus besser vergleichen zu können, da immer mit den selben Daten gearbeitet wird.

Außerdem ist es möglich, die Messwerte der verschiedenen Sensoren zu simulieren. Der Simulator, der in der SensorReadConnection enthalten ist, füllt die gleiche Datenstruktur wie der SensorReadMaganer aus, so dass alle nachfolgenden Schritte für den Simulator und das reale Testfahrzeug identisch sind. Der Simulator ist in Kapitel 3.2.3 genauer beschrieben.

3.2.2 Verschiedene Views

Um die von der SensorReadConnection erzeugten Debug-Daten anzuzeigen, gibt es verschiedene Views :

- Der GroundView dient zum Anzeigen aller Debug-Drawings, die sich auf absolute Positionen in der Ebene beziehen. In Abbildung 3.4 sieht man z.B. die vom Lokalisierungsalgorithmus berechnete Position, Positionen von Magneten, die Partikelmenge und die gemessenen Sensorwerte.
- Der SensorView dient dazu, die aktuellen Messwerte der verschiedenen Sensoren sowie andere interne Variablen des Lokalisierungsalgorithmus als Graphen über die Zeit anzuzeigen. In Abbildung 3.5 sind z.B. die Messwerte der Magnetfeldsensoren dargestellt. Die Zeiteinheit auf der X-Achse sind „Frames“, die hier einem Durchlauf des Lokalisierungsalgorithmus oder 0,1s entsprechen.
- Der LogView ermöglicht es, die aktuellen Werte aller DebugGraphs in eine Textdatei aufzuzeichnen, die dann später von anderen Programmen wie z.B. Gnuplot weiter verarbeitet werden kann.

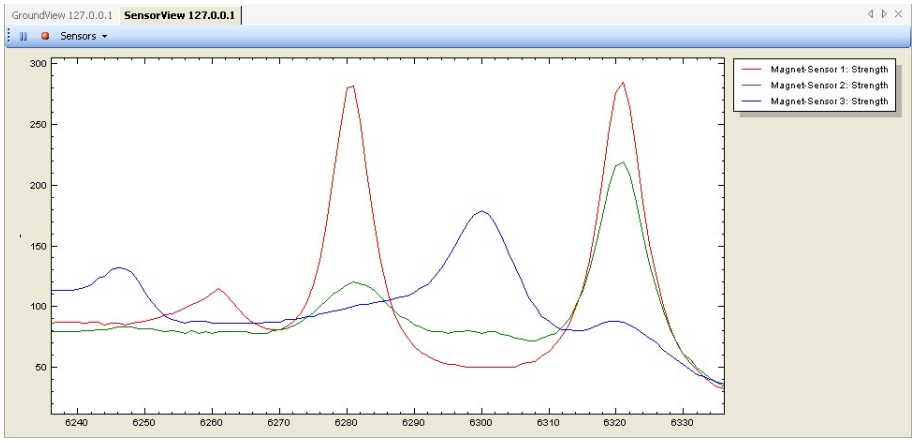


Abbildung 3.5: Sensor View. Hier mit Messwerten der Magnetsensoren.

- Der OdometryTester ist ein spezieller View zum Testen der Odometrie des Testfahrzeugs. Er dient zur Messung der Abweichung zwischen der durch ein Referenzsystem gegebenen und durch Odometrie (Kapitel 5.1.1) ermittelten Position.

3.2.3 Simulator

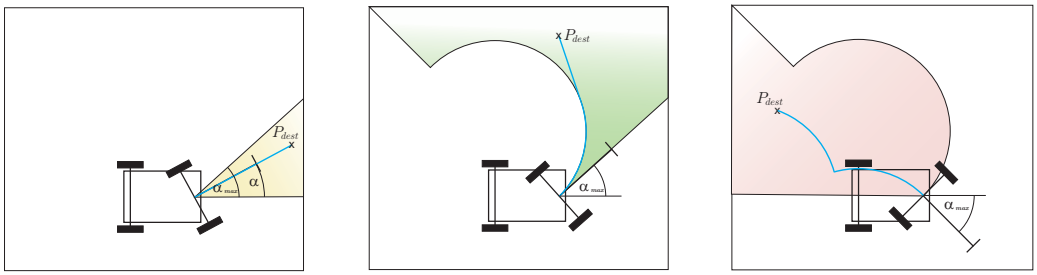
Der Simulator bietet die Möglichkeit, den Algorithmus auch ohne Testfahrzeug zu testen. Er erhält dazu über den GroundView eine Zielposition, zu der sich der simulierte Wagen dann abhängig von seiner aktuellen Position hinbewegt. Der Simulator läuft mit derselben Frequenz von 10Hz, mit der auch das Sensorboard sendet. In jedem Zeitschritt berechnet er zunächst die neue Position des Fahrzeuges und danach die Messwerte der einzelnen Sensoren.

Da das Fahrzeug einen maximalen Lenkwinkel besitzt, kann es den Zielpunkt nicht immer direkt durch Vorwärtsfahren erreichen. Um nun keine physikalisch unmöglichen Fahrbewegungen zu simulieren, wird in Abhängigkeit von der relativen Zielpositon P_{dest} zum Fahrzeug einer der unten angegebenen Fälle ausgeführt. (Abbildung 3.6)

Dabei ist α der Winkel zum Zielpunkt und α_{max} der maximale Lenkwinkel.

1. P_{dest} befindet sich vor dem Fahrzeug und $\alpha \leq \alpha_{max}$. Das Fahrzeug fährt vorwärts mit der Lenkstange zum Zielpunkt ausgerichtet. (Abbildung 3.6(a))

2. Das Ziel ist nicht direkt, aber noch durch Vorwärtsfahren erreichbar. Dies ist dann der Fall, wenn es sich nicht innerhalb des kleinst möglichen Wendekreises des Fahrzeuges befindet. In diesem Fall wird der maximale Lenkwinkel eingeschlagen und vorwärts gefahren. (Abbildung 3.6(b)) Mit dieser Fahrbewegung können auch Punkte hinter dem Fahrzeug angefahren werden. Bei Positionen



(a) 1. Fall: das Ziel ist direkt ansteuerbar, da es in einem Winkel $\alpha \leq \alpha_{max} liegt$

(b) 2. Fall: das Ziel ist nicht direkt ansteuerbar, aber durch Vorwärtsfahren erreichbar. Daher ist der Lenkwinkel $\alpha = \alpha_{max}$.

(c) 3. Fall: das Ziel ist durch Vorwärtsfahren nicht mehr erreichbar. Daher wird mit Lenkwinkel $-\alpha_{max}$ rückwärts gefahren

Abbildung 3.6: 3 von 6 möglichen Fällen beim Ansteuern eines Zieles P_{dest}. Blau dargestellt der Weg, der zum Ziel gefahren wird. Die anderen Fälle ergeben sich durch Spiegelung an der X-Achse.

hinter dem Fahrzeug, die mit einem Lenkwinkel $\alpha \leq \alpha_{max}$ erreichbar sind, ist es aber sinnvoller zu wenden.

Fährt das Fahrzeug nun, so bewegt sich der relative Zielpunkt irgendwann in den Bereich des ersten Falles, wodurch sich der Wagen erst entlang des Wendekreises und dann gerade auf das Ziel zubewegt. (Siehe blaue Fahrtstrecke in Abbildung 3.6(b))

3. Das Ziel ist nicht mehr durch Vorwärtsfahren zu erreichen. Dies ist dann der Fall, wenn es sich innerhalb des Wendekreises befindet. In diesem Fall wird mit negativem Maximallenkwinkel rückwärts gefahren.

   Das Fahrzeug fährt nun so lange rückwärts, bis sich der relative Zielpunkt wieder im Bereich des zweiten Falles befindet, was automatisch zu einer „Wende“ führt. (Abbildung 3.6(c))

   Der rückwärtige Bereich für den Lenkwinkel $\alpha \leq \alpha_{max}$ ist ebenfalls diesem Fall zugeordnet. In diesem Fall wird also ebenfalls gewendet.

Es wurden nur die Fälle für Punkte rechts vom Fahrzeug angegeben. Die anderen ergeben sich durch Spiegelung an der X-Achse.

Um nun die neue Position zu erhalten, wird zunächst die zu fahrende Geschwindigkeit ermittelt. Dazu ist eine Maximalgeschwindigkeit festgelegt und die maximale Beschleunigung begrenzt worden. Befindet sich das simulierte Fahrzeug in der Nähe des Zielpunktes, wird außerdem die Geschwindigkeit wieder bis auf 0m/s reduziert. Mit dieser Geschwindigkeit wird nun die neue Position des Fahrzeuges berechnet. Dabei gibt der Lenkwinkel zusammen mit dem Abstand der Achsen den Mittelpunkt der Kreisbahn vor, auf der sich der Wagen bewegt. Aus der geforderten Geschwindigkeit zusammen mit der Zeitdifferenz wird die zurückgelegte Strecke auf der Kreisbahn und damit die neue Position berechnet.

Die Simulation der einzelnen Sensoren wird nun anhand der neuen Position wie folgt durchgeführt:

- Der Lenkwinkelsensor ist sehr einfach zu simulieren, da nur der berechnete Lenkwinkel zurückgeliefert wird.
- Für die Simulation des Messwertes eines Magnetfeldsensors wird zunächst seine absolute Position mit Hilfe der festen relativen Position auf dem Wagen und der Position des Wagens selbst berechnet. Danach wird der Abstand zu jedem einzelnen Magneten bestimmt und mittels der in Kapitel 5.2.1 noch genau erläuterten Formel der Messwert bestimmt.
- Zur Simulation der Radsensoren muss zunächst die Positionsänderung an den Aufsetzpunkten der Räder bestimmt werden. Da der Mittelpunkt, um den sich der Wagen dreht, schon berechnet ist, lässt sich diese direkt aus dem Verhältnis der Radien und der Änderung der Position bestimmen.
- Ebenso wird der Beschleunigungssensor simuliert. Zunächst wird die Positionsänderung am entsprechenden Montagepunkt berechnet. Daraus lässt sich mit der Frequenz des Simulators die Geschwindigkeit und mit der Geschwindigkeitsdifferenz zum letzten Simulatordurchlauf die Beschleunigung errechnen.

Kapitel 4

Lokalisierungsalgorithmen

Bei der Gabelstaplerlokalisierung mit Magneten ist die Situation die, dass der Gabelstapler durch die Halle fährt und dabei gelegentlich an einem Magneten vorbeikommt. Es wird zunächst einmal mittels der Radsensoren, des Lenkwinkelsensors und des Beschleunigungssensors die Odometrie des Gabelstaplers gemessen. Die Odometrie gibt dabei an, in welche Richtung und wie weit sich der Gabelstapler seit der letzten Messung bewegt hat. Durch Aufsummierung lässt sich schon eine einfache, relative Positionsbestimmung durchführen. Die Odometrie ist natürlich Messungenauigkeiten unterworfen, die sich durch die Aufsummierung immer stärker auf die Position auswirken. Daher muss sie immer wieder korrigiert werden, was immer dann möglich ist, wenn mit dem Magnetlineal ein Magnet gemessen wird. Außerdem ist es notwendig, nicht nur eine relative Position, sondern die absolute Position in der Lagerhalle zu ermitteln.

Bei verschiedenen Lokalisierungsproblemen, wie z. B. Tracking für Radar [15], Ortung mit Ultraschall [5] oder bei der Positionsbestimmung von mobilen Robotern [16] [17], haben sich probabilistische Algorithmen als sehr effizient und robust erwiesen.

Probabilistische Algorithmen zeichnen sich dadurch aus, dass alle wichtigen Größen nicht als exakter Wert, sondern als Wahrscheinlichkeitsverteilung angenommen werden, wodurch im Lokalisierungsalgorithmus Ungenauigkeiten ebenfalls modelliert werden können.

Im folgenden Kapitel wird die Gruppe der Bayes-Filter erläutert und einige Implementierungsvarianten auf ihre Brauchbarkeit zur Lokalisierung von Gabelstaplern hin untersucht. Hierzu gibt das Buch „Probabistic Robotics“ [18] einen sehr guten und kompletten Überblick, weshalb es als Hauptquelle in diesem Bereich verwendet wurde.

4.1 Bayes-Filter

Zunächst einmal sind einige Begriffsdefinitionen zur probabilistischen Beschreibung des Lokalisierungsproblems notwendig.

Die aktuelle reale Position des zu lokalisierenden Gabelstablers wird als **Zustand** oder **State** bezeichnet. Er ändert sich im Laufe der Zeit t und wird daher als x_t bezeichnet. Aufgabe des Lokalisierungsalgorithmus ist es, diesen Zustand zu schätzen.

Das „Wissen“, das der Lokalisierungsalgorithmus über den State zum Zeitpunkt t besitzt, wird als **Belief** bezeichnet und ist eine Wahrscheinlichkeitsverteilung $Bel(x_t) = p(x_t)$ für den State.

Der Belief ist abhängig von den Eingangsdaten, die der Lokalisierungsalgorithmus bekommt. Diese sind **Messdaten** oder **Measurement-Data** z_t und **Kontrolldaten** oder **Control-Data** u_t, die beide wiederum Zufallsvariablen sind und sich über die Zeit ändern.

Die Messdaten z_t sind die Informationen, die der Lokalisierungsalgorithmus mittels Sensoren über den Zustand erhält. Diese sind nur vom aktuellen Zustand x_t abhängig, da genau dieser gemessen wird. Sie sind allerdings einer Messungenauigkeit unterworfen.

Die Kontrolldaten beschreiben dabei die Aktionen des zu lokalisierenden Objektes. Bei einem mobilen Roboter ist das die Bewegungsanforderung, die das Kontroll-Programm berechnet. Bei einem Gabelstapler, der von einem Staplerfahrer gefahren wird, müssen die Kontrolldaten ebenfalls gemessen werden. Man spricht daher auch von **Odometrie**. Da die Auswirkungen der Kontrolldaten auf den State nicht unbedingt immer gleich sein müssen, sind auch sie mittels einer Zufallsvariablen repräsentiert, um diese Ungenauigkeiten zu modellieren.

In jedem Zeitschritt des Lokalisierungsalgorithmus ändert sich nun der Belief in Abhängigkeit von den Mess- und Kontrolldaten. Der aktuelle Belief ist daher von allen Kontrolldaten $u_{0:t}$ und Messdaten $z_{0:t}$ vom Zeitpunkt 0 bis zum aktuellen Zeitpunkt t abhängig. Dies wird über bedingte Wahrscheinlichkeiten ausgedrückt:

$$Bel(x_t) = p(x_t|u_{0:t}, z_{0:t})$$

Geht man davon aus, dass dabei zuerst die Kontrolldaten und dann die Messdaten einbezogen werden, so ergibt sich

$$\overline{Bel}(x_t) = p(x_t|u_{0:t}, z_{0:t-1})$$

als Belief vor der Einbeziehung der Messdaten. Dieser wird auch als **Vorhersage** oder **Prediction** bezeichnet.

4.1.1 Algorithmus

Für den Bayes-Filter wird nun angenommen, dass der Zustand x_t zum Zeitpunkt t alle nötigen Informationen über seine vorherigen Zustände $x_{0:t-1}$ vom Zeitpunkt 0 bis $t-1$ enthält - er heißt dann **komplett**. Diese Annahme wird auch als **Markov-Assumption** bezeichnet.

Unter Verwendung dieser Annahme, wird nun $Bel(x_t)$ mittels des **Bayes-Filters** berechnet.

Algorithmus 1 : Bayes-Filter

Eingabe : $Bel(x_{t-1}), u_t, z_t$
Ausgabe : $Bel(x_t)$
1 **forall** x_t **do**
2 $\quad \overline{Bel}(x_t) = \int p(x_t|u_t, x_{t-1}) Bel(x_{t-1}) dx_{t-1}$
3 $\quad Bel(x_t) = \eta p(z_t|x_t) \overline{Bel}(x_t)$
4 **end**

Der Algorithmus läuft so ab, dass er in jedem Zeitschritt zuerst in Zeile 2 mittels der Kontrolldaten die Vorhersage $\overline{Bel}(x_t)$ berechnet. Die Annahme eines kompletten Zustandes sorgt hier dafür, dass nur der Belief $Bel(x_{t-1})$ und die aktuellen Kontrolldaten u_t statt aller Kontroll- und Messdaten benötigt werden. Dieser Schritt wird auch **Control-Update** genannt.

Im zweiten Schritt in Zeile 3 werden nun die Messdaten z_t mit einbezogen, um den neuen Belief $Bel(x_t)$ zu berechnen. Dieser Schritt wird auch **Measurement-Update** genannt.

Die Wahrscheinlichkeitsverteilung $p(z_t|x_t)$ heißt dabei **Messwahrscheinlichkeit** und die $p(x_t|u_t, x_{t-1})$ **Zustandsübergangswahrscheinlichkeit**. Diese sind beide vom jeweiligen System abhängig. Außerdem ist der Bayes-Filter in dieser allgemeinen Form nicht rechnerisch handhabbar, was vor allem an der Tatsache liegt, dass er für alle möglichen x_t berechnet werden muss. Daher muss für alle verwendeten Wahrscheinlichkeitsverteilungen eine dem Problem angemessene Darstellungsvariante gefunden werden, was dann zu Kalman-Filtern, Histogram-Filtern und Partikel-Filtern führt, die weiter unten in diesem Kapitel vorgestellt werden.

Für die mathematische Herleitung des Bayes-Filters sei auf [18, Kapitel 2] verwiesen.

4.1.2 Bayes-Filter zur Gabelstaplerlokalisierung

Bayes-Filter eignen sich sehr gut, um den zu Beginn dieses Kapitels gegebenen Anforderungen bei der Gabelstaplerlokalisierung gerecht zu werden. Es ist möglich, aus der gemessenen Odometrie eine Zustandsübergangswahrscheinlichkeit und aus den Messwerten die Messwahrscheinlichkeit zu ermitteln. Der Bayes-Filter sorgt dann

für die entsprechenden Änderungen des Beliefs. Außerdem bietet der Bayes-Filter auch die Möglichkeit, eine absolute Position zu finden.

Daher wird nun beispielhaft anhand zweier Durchläufe die Arbeit des Bayes-Algorithmus bei einer einfachen, eindimensionalen Gabelstaplerlokalisierung in Abbildung 4.1 dargestellt. Der Gabelstapler fährt entlang eines Flurs (grau), in den drei Magnete eingelassen sind. Unter dem Flur sind nun (grün) der momentane Belief $Bel(x_t)$ und, wenn gerade ein Magnet gemessen wird, die Messwahrscheinlichkeit $p(z_t|x_t)$ (rot) dargestellt.

1. Zu Beginn in Abbildung 4.1(a) ist der Belief des Lokalisierungsalgorithmus gleichverteilt, da noch keine Magnete gemessen wurden.

2. Zur Ermittlung der Kontrolldaten wird die Odometrie des Gabelstaplers herangezogen. Wie diese konkret berechnet wird, wird in Kapitel 5.1.1 beschrieben. Die Odometrie liefert einen Bewegungsvektor zurück, mit dem sich die Zustandsübergangswahrscheinlichkeit als Verschiebung mit Rauschen annehmen lässt. Da der Belief vorher schon gleichverteilt war, ändert sich durch das erste Control-Update nichts.

3. Als nächstes wird in Abbildung 4.1(b) ein Magnet gemessen. Dies führt dazu, dass die Messwahrscheinlichkeit besagt, dass sich der Gabelstapler an jedem der drei Magnete befinden kann. Der neue Belief entspricht, da der alte Belief gleichverteilt war, nun der Messwahrscheinlichkeit

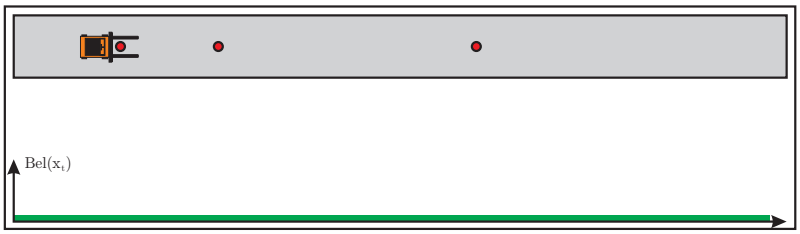
4. Nun ist der erste Durchlauf des Bayes-Filters vorbei und in Abbildung 4.1(c) wird das nächste Control-Update vorgenommen und der Belief wieder verschoben. Dabei ist zu beachten, dass die Wahrscheinlichkeitsverteilung des Beliefs wegen der Ungenauigkeit der Kontrolldaten abflacht.

5. Beim nun folgenden Measurement-Update in Abbildung 4.1(d) wird dieselbe Messwahrscheinlichkeit wie in Schritt 2 verwendet, um den neuen Belief zu errechnen. Da der vorherige Belief nicht gleichverteilt war, ergibt sich nun am zweiten Magneten eine besonders hohe Aufenthaltswahrscheinlichkeit für den Gabelstapler. Der Lokalisierungsalgorithmus hat die absolute Position im Flur gefunden.

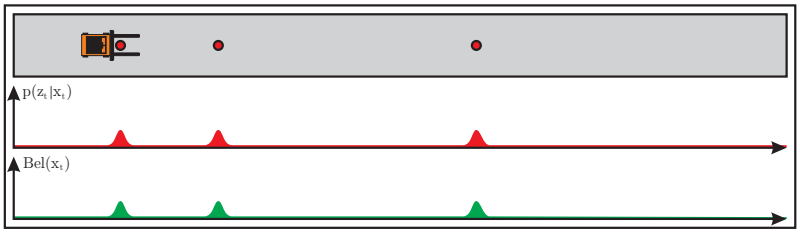
6. Beim nun folgenden nächsten Durchlauf wird der Belief wieder verschoben. Wegen der angenommenen Unsicherheit bei der Ausführung der Kontrolldaten flachen die Wahrscheinlichkeiten weiter ab. Abbildung 4.1(e)

4.2 Kalman-Filter

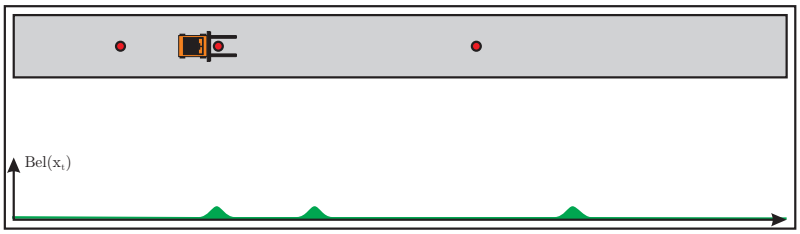
Der Kalman-Filter [19] ist eine konkrete Implementierung des Bayes-Filters, der Normalverteilungen zur Darstellung der Wahrscheinlichkeitsverteilungen verwendet.



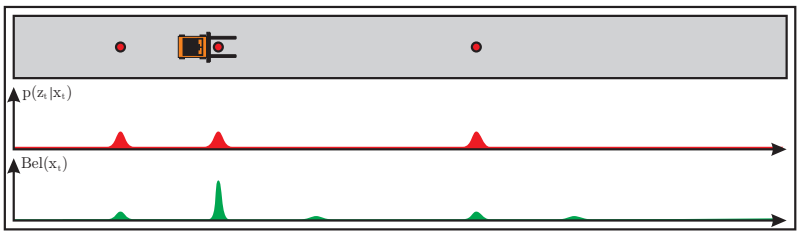
(a) Zu Beginn ist der Belief $Bel(x_t)$ gleichverteilt. Daran ändert auch das erste Control-Update nichts.



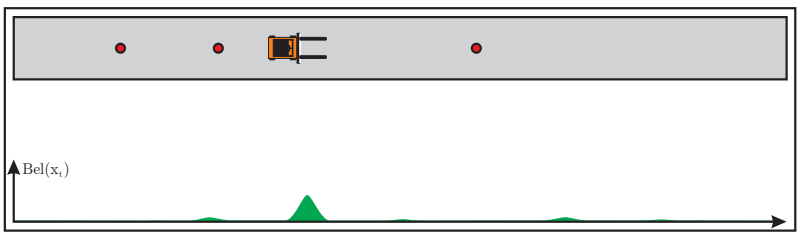
(b) Ein Magnet wird mit der Messwahrscheinlichkeit $p(z_t|x_t)$ gemessen, was bedeutet, dass sich der Stapler mit gleicher Wahrscheinlichkeit vor jedem Magneten befinden kann.



(c) Der Stapler fährt weiter. Durch das Control-Update wird der Belief $Bel(x_t)$ entsprechend verschoben.



(d) Ein Magnet wird gemessen, was wiederum bedeutet, dass der Stapler bei jedem Magneten sein kann. Zusammen mit dem vorherigen Belief ist aber die Wahrscheinlichkeit am zweiten Magneten am größten.



(e) Der Stapler fährt weiter, was wegen der Messungenauigkeit der Kontrolldaten zu einer Abflachung des Beliefs führt.

Abbildung 4.1: Ablauf des Bayes-Filters bei der Lokalisierung. Der Gabelstapler fährt entlang eines Flures (grau) und kommt dabei an Magneten(rot) vorbei.

4.2.1 Beschreibung

Wird der Belief durch eine Normalverteilung dargestellt, so hat er die Form

$$Bel(x_t) = det\,(2\pi\Sigma_t)^{-\frac{1}{2}}\, exp\left\{-\frac{1}{2}(x_t-\mu_t)^T\Sigma_t^{-1}(x_t-\mu_t)\right\}.$$

Dadurch lässt sich der Belief nur durch den Mittelwertvektor μ_t und die Kovarianzmatrix Σ_t beschreiben.

Damit der Belief zu jeder Zeit durch eine solche Normalverteilung zu beschreiben ist, müssen die Zustandsübergangswahrscheinlichkeit und die Messwahrscheinlichkeit lineare Funktionen mit normalverteiltem Rauschen sein.

Daher lässt sich die Änderung am Zustand x_t durch die Kontrolldaten als

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$$

beschreiben. A_t und B_t sind hierbei Matrizen, die den linearen Einfluss des alten States bzw. der Kontrolldaten auf den neuen State angeben und ε_t ist eine normalverteile Zufallsvariable mit der Kovarianzmatrix R, die die Ungenauigkeit wiedergibt.

Dadurch ergibt sich für die Zustandsübergangswahrscheinlichkeit die Normalverteilung

$$p(x_t|u_t,x_{t-1}) = det\,(2\pi R_t)^{-\frac{1}{2}}\, exp\left\{-\frac{1}{2}(x_t-A_tx_{t-1}-B_tu_t)^TR_t^{-1}(x_t-A_tx_{t-1}-B_tu_t)\right\}.$$

Genauso lassen sich auch die Messdaten z_t als

$$z_t = C_t x_t + \delta_t$$

darstellen, wobei C_t angibt, wie der aktuelle Zustand sich auf die Messung auswirkt und δ_t als gaussverteilte Zufallsvariable mit Kovarianzmatrix Q_t wiederum die Ungenauigkeit angibt. Die Normalverteilung für die Messwahrscheinlichkeit lautet dann wie folgt:

$$p(z_t|x_t) = det\,(2\pi Q_t)^{-\frac{1}{2}}\, exp\left\{-\frac{1}{2}(z_t-C_tx_t)^TQ_t^{-1}(z_t-C_tx_t)\right\}.$$

Da nun alle Wahrscheinlichkeitsverteilungen des Bayes-Filters als Normalverteilungen dargestellt sind, kann der neue Belief mit dem Kalman-Filter (Algorithmus 2) berechnet werden.

Algorithmus 2 : Kalman-Filter

Eingabe : $\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$

Ausgabe : μ_t, Σ_t

1 $\bar{\mu}_t = A_t\ \mu_{t-1}\ +\ B_t\ u_t$
2 $\bar{\Sigma}_t = A_t\ \Sigma_{t-1}\ A_t^T\ +\ R_t$
3 $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$
4 $\mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t)$
5 $\Sigma_t = (I - K_t C_t)\bar{\Sigma}_t$

Der Kalman-Filter berechnet zuerst $\bar{\mu}_t$ und $\bar{\Sigma}_t$. Sie repräsentieren die Vorhersage $\overline{Bel}(x_t)$. Der in der nächsten Zeile berechnete **Kalman-Gain** K_t gibt an, wie stark sich die Messung auf die Änderung des Beliefs auswirkt. Er ist ein Zwischenergebnis, das dann in den folgenden Zeilen zur Berechnung von μ_t und Σ_t benutzt wird.

4.2.2 Eignung für die Gabelstaplerlokalisierung

Der Kalman-Filter hat den großen Vorteil, dass er sehr schnell berechenbar ist und eine kontinuierliche Repräsentation des Beliefs liefert. Aber er hat auch einige Nachteile. Einer davon ist die Tatsache, dass er ein lineares System benötigt, was sehr häufig nicht der Fall ist. Dies kann man entweder ignorieren, falls die Nichtlinearität nicht zu stark ist, oder durch Abwandlungen, wie dem Extended Kalman Filter (EKF) begegnen. Die Haupteinschränkung des Kalman-Filters, dass er nur unimodale, d. h. mit nur einem lokalen Maximum behaftete, Beliefs verarbeiten kann, bleibt aber erhalten.

Es gibt nun viele Lokalisierungsaufgaben, die sich mit einem Kalman-Filter lösen lassen. Sie alle haben gemeinsam, dass sich ihre Zustandsübergangswahrscheinlichkeit, Messwahrscheinlichkeit und ihr Belief normalverteilt darstellen lassen. So lässt sich z.B. die Position eines Balls beim Roboterfußball mit einem Kalman-Filter gut verfolgen.

Für die Gabelstaplerlokalisierung ist ein Kalman-Filter allerdings nicht gut geeignet. Zwar lässt sich die über die Odometrie gemessene Zustandsübergangswahrscheinlichkeit problemlos als Normalverteilung darstellen. Auch besteht hier ein linearer Zusammenhang, da die Odometrie eine Verschiebung ist. Die Messwahrscheinlichkeit lässt sich allerdings nicht als Normalverteilung darstellen, da sich der Gabelstapler bei der Messung eines Magneten nicht nur über einem speziellen, sondern über jedem einzelnen Magneten befinden kann.

Eine Möglichkeit, dennoch den Kalman-Filter zur Lokalisierung zu benutzen, wäre nun, den zum Mittelwert $\bar{\mu}$ des Beliefs $\overline{Bel}(x_t)$ nächstliegenden Magneten für Messwahrscheinlichkeit zu benutzen. Dadurch könnte man zwar eine relative, nicht aber eine absolute Position bestimmen. Diese Problematik ist in Abbildung 4.2 dargestellt.

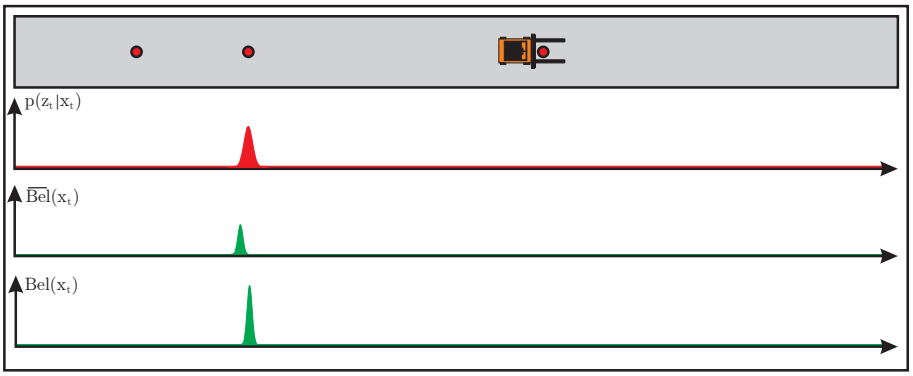


Abbildung 4.2: Der Stapler befindet sich an Magnet 3, die Vorhersage $\overline{Bel}(x)$ geht aber davon aus, dass er sich an Magnet 2 befindet. Wegen der Einschränkung auf Normalverteilungen kann der Kalman-Filter diesen Fehler nicht erkennen und die Wahrscheinlichkeit bleibt bei Magnet 2 sehr hoch.

4.3 Histogram-Filter

Der Histogram-Filter bedient sich einer anderen Darstellungsvariante für den Belief. In der Literatur wird er manchmal auch als „Hidden Markov Model“ bezeichnet. [20]

4.3.1 Beschreibung

Der Histogram-Filter teilt den Zustandsraum $dom(X_t)$ in viele kleine Bereiche $dom(X_t) = x_{1,t} \cup x_{2,t} \cup \ldots \cup x_{K,t}$ auf und berechnet für jeden dieser Bereiche eine Wahrscheinlichkeit $p_{k,t}$ mittels des „Diskreten Bayes-Filters“ (Algorithmus 3). Diese Bereiche $x_{k,t}$ sind konvex und bilden zusammen eine Partitionierung des Zustandsraums.

Algorithmus 3 : Diskreter Bayes-Filter

Eingabe : $\{p_{k,t-1}\}, u_t, z_t$
Ausgabe : $\{p_{k,t}\}$
forall k **do**
    $\bar{p}_{k,t} = \sum_i \; p(X_t = x_k | u_t, X_{t-1} = x_i) \; p_{i,t-1}$
    $p_{k,t} = \eta \; p(z_t | X_t = x_k) \; \bar{p}_{k,t}$
end

Hierbei ist $\{p_{k,t-1}\}$ eine endliche Menge von Wahrscheinlichkeiten, die jeweils zu einem Bereich $x_{k,t-1}$ gehören.

Nun stellt sich die Frage, wie aus dem kontinuierlichen Belief die Bereiche $x_{k,t}$ gewählt werden und welche Wahrscheinlichkeit $p_{k,t}$ sie zugewiesen bekommen.

Die einfachste Aufteilung teilt den Zustandsraum in eine feste Anzahl gleich großer Bereiche auf - es entsteht ein mehrdimensionales Gitter. Durch die Variation der

Anzahl der Gitterzellen kann nun zwischen Genauigkeit und Rechenzeitaufwand variiert werden. Man kann auch eine Aufteilung wählen, die sich an irgendwelchen problemspezifischen Eigenschaften orientiert und einige Bereiche des Zustandsraumes feiner und andere gröber aufteilen, was unter Umständen zu weniger Bereichen führt.

Außerdem gibt es mit Destiy-Trees dynamische Aufteilungsmöglichkeiten. Hierbei wird eine Baumstruktur aufgestellt, mit der in jedem Zeitschritt Bereiche mit hoher Wahrscheinlichkeit immer weiter unterteilt werden und benachbarte Bereiche mit niedriger Wahrscheinlichkeit wieder zusammenfasst werden.

4.3.2 Eignung für die Gabelstaplerlokalisierung

Der Histogram-Filter erfüllt in der Theorie alle Qualitätsanforderungen, die zur Gabelstaplerlokalisierung notwendig sind. Die Messwahrscheinlichkeit und die Zustandsübergangswahrscheinlichkeit lassen sich sinnvoll darstellen und bei einer hinreichenden Anzahl von Bereichen wird die Auflösung ebenfalls ausreichend hoch.

Aber was ist nun eine hinreichende Anzahl? Die Gabelstaplerlokalisierung ist ein dreidimensionales Problem und zwar muss die Position und die Orietentierung des Gabelstaplers bestimmt werden.

Wird eine Halle mit einer Fläche von 2550m^2 angenommen, in der die Position mit einer Genauigkeit von 10cm und die Oritentierung mit einer Genauigkeit von 10° bestimmt werden soll, so werden

$$50m \cdot 10\frac{1}{m} \cdot 50m \cdot 10\frac{1}{m} \cdot 360^{\circ} \cdot 36\frac{1}{^{\circ}} = 3,24 \cdot 10^9$$

Bereiche benötigt.

Von diesen vielen Bereichen besitzen zwar die meisten nur eine sehr kleine Wahrscheinlichkeit, weshalb es mit geschickten Programmiertricks vielleicht möglich wäre, den diskreten Bayes-Filter nicht für all diese 3,24 Milliarden Bereiche durchzuführen. Trotzdem ist ein statischer Ansatz auf jeden Fall unbrauchbar, da alle Bereiche gespeichert werden müssen, was mit 32-Bit Floating-Point Werten 12,96 GB Speicher verbrauchen würde.

Der dynamische Ansatz ist hier schon vielversprechender. Bei diesem müssen aber, besonders wenn die richtige Position noch nicht gefunden wurde, ebenfalls sehr viele Bereiche betrachtet werden.

Daher ist der Histogram-Filter zur Gabelstaplerlokalisierung unbrauchbar.

4.4 Partikel-Filter

Eine weitere Möglichkeit zur Darstellung der Wahrscheinlichkeitsverteilungen des Bayes-Filters ist die Verwendung von Partikeln. Partikelfilter wurden in den letzten Jahren für viele Lokalisierungsaufgaben vor allem in der mobilen Robotik verwendet [12] [16] [17] [21]. Sie werden auch „Monte-Carlo-Filter“ geannnt.

4.4.1 Beschreibung

Partikelfilter verwenden M sogenannte Partikel $x_t^{[m]}$ zur Repräsentation des Beliefs des Bayes-Filters. Alle Partikel zusammen bilden die Partikelmenge

$$X_t := x_t^{[1]}, x_t^{[2]}, \ldots, x_t^{[M]}.$$

Dabei spiegelt ein Partikel $x_t^{[m]}$ einen konkreten möglichen Zustand zum Zeitpunkt t wieder. Idealerweise sollte die Tatsache, dass ein Partikel $x_t^{[m]}$ in X_t enthalten ist, proportional zum Belief $Bel(x_t)$ sein. Das führt dazu, dass die Partikel an Stellen mit höherem Belief dichter zusammenliegen.

Algorithmus 4 : Partikel-Filter

Eingabe : X_{t-1}, u_t, z_t
Ausgabe : X_t

1 $\bar{X}_t = X_t = \emptyset$
2 **for** *m=0 to M* **do**
3 Generiere einen Partikel $x_t^{[m]} \approx p(x_t|u_t, x_{t-1}^{[m]})$
4 $w_t^{[m]} = p(z_t|x_t^{[m]})$
5 $\bar{X}_t = \bar{X}_t + \left\langle x_t^{[m]}, w_t^{[m]} \right\rangle$
6 **end**
7 **for** *m=0 to M* **do**
8 Wähle ein i zufällig mit Wahrscheinlichkeit $w_t^{[i]}$
9 Füge $x_t^{[i]}$ X_t hinzu.
10 **end**

Der Partikel-Filter hat als Eingabe die Partikelmenge X_{t-1}, die den Belief zum Zeitpunkt $t-1$ repräsentiert, sowie die aktuellen Kontroll- und Messdaten.

Zunächst wird in der ersten Schleife die Menge $\bar{X}_t$ mit Tupeln aus Partikeln $x_t^{[m]}$ und Gewichtungen $w_t^{[m]}$ für jedes Partikel gefüllt. Dazu wird zuerst ein neues Partikel $x_t^{[m]}$ auf Basis der Zustandsübergangswahrscheinlichkeit gewählt. Dies entspricht dem Control-Update des Bayes-Filters und alle hier generierten Partikel zusammen repräsentieren die Vorhersage $\overline{Bel}(x_t)$ des Bayes-Filters. Wie dies genau passiert, ist eine Frage des Problems und der Implementierung.

Danach wird diesem Partikel eine Gewichtung zugeordnet, die ein Maß dafür ist, wie gut das Partikel zur aktuellen Messung passt. Dies entspricht dem Measurement-Update des Bayes-Filters.

Im nächsten Schritt, der **Resampling** genannt wird, werden nun aus den im ersten Schritt generierten Partikeln zufällig Partikel ausgewählt, die der Ergebnis-Partikelmenge X_t hinzugefügt werden. Dabei ist die Wahrscheinlichkeit, dass ein Partikel ausgewählt wird, allerdings nicht uniform, sondern proportional zur Gewichtung dieses Partikels. Dies führt dazu, dass Partikel mit höherer Gewichtung häufiger ausgewählt werden als Partikel mit niedriger Gewichtung und sogar eine große Menge von Partikeln gar nicht mehr. Wie in [18, Kapitel 4] gezeigt wird, repräsentiert die neu entstandene Partikelmenge X_t den neuen Belief $Bel(x_t)$.

4.4.2 Überlegungen zur praktischen Realisierung

Möchte man nun mit dem Lokalisierungsergebnis weiter arbeiten, so ist es nötig, aus der Partikelmenge den Belief zu extrahieren, was **Destiny Extraxtion** genannt wird. Allgemein kann man einen kontinuierlichen Belief dadurch erhalten, dass man für jedes Partikel eine Normalverteilung mit einer gewissen Varianz und dem Partikel als Mittelwert annimmt und diese aufsummiert. Es können auch mittels verschiedener Clustering-Algorithmen mehrere Partikel zusammenfasst werden und eine Normalverteilung mit diesen Partikeln berechnet werden. Bei manchen Problemen ist es auch ausreichend, nur den Mittelwert der Partikel zu verwenden, da dieser bei der Weiterverarbeitung des Lokalisierungsergebnisses ausreicht.

Beim Resampling gibt es zu einem das Problem, dass sich der durch die Partikelmenge repräsentierte Belief immer etwas von dem Belief unterscheidet, den man bei genauer Rechnung erhalten würde. Das führt dazu, dass bei gleichen Eingangsdaten $u_{0:t}$ und $z_{0:t}$ der Belief in jedem Durchlauf anders verteilt ist. Diese **Sampling-Variance** ist dabei abhängig von der Anzahl der Partikel und der Resampling-Frequenz. Wird die Anzahl der Partikel erhöht, fällt die Sampling-Variance und der Partikelfilter liefert bessere Ergebnisse. Über die Entscheidung, ob das Resampling durchgeführt werden soll oder nicht, kann man beeinflussen, wie häufig der Fehler der Sampling-Variance Einfluss auf die Lokalisierung nimmt. Es ist z.B. meistens nicht sinnvoll, Resampling durchzuführen, wenn sich der Zustand nicht ändert oder keine neuen Messwerte vorliegen. Eine weitere Möglichkeit zum Verringern der Sampling-Varianz ist die Verwendung von Resampling-Algorithmen, die die Partikel nicht völlig unabhängig voneinander, sondern irgendwie anders sinnvoll auswählen. Daher werden in Kapitel 5.3 verschiedene Ansätze beim Resampling erleutert.

Ein weiteres Problem beim Resampling ist die Tatsache, das eine Situation entstehen kann, bei der sich kein Partikel in der Nähe des richtigen States befindet. Der Partikel-Filter liefert nun einen komplett falschen Belief. Diesem Problem kann man durch Einstreuen von zufälligen Partikeln begegnen. Da diese aber die Sampling-Variance erhöhen, muss hier abgewogen werden, wie viele zufällige Partikel gewählt werden. Es gibt aber auch die Möglichkeit, statt zufälliger Partikel sogenannte

Template-Partikel einzuführen [22] [23]. Diese Template-Partikel sind Partikel, die im nächsten Schritt eine hohe Wahrscheinlichkeit erhalten werden, aber unabhängig vom Lokalisierungsalgorithmus erzeugt werden. Die Erzeugung dieser Partikel im konkreten Fall der Gabelstaplerlokalisierung wird in Kapitel 5.3.3 erleutert.

4.4.3 Eignung für die Gabelstaplerlokalisierung

Der Partikelfilter scheint sehr gut für die Lokalisierung von Gabelstaplern geeignet. Zunächst ist es problemlos möglich, neue Partikel aus der Zustandsübergangswahrscheinlichkeit zu generieren. Da die Odometrie einen Verschiebungsvektor mit einer gewissen normalverteilten Ungenauigkeit liefert, müssen die Partikel entsprechend verschoben und normalverteiltes Rauschen addiert werden. Die Intensität dieses Rauschens ist abhängig von der Qualität der Sensoren und kann experimentell bestimmt werden.

Außerdem ist es möglich, durch Einstreuen von zufälligen Partikeln oder durch Template-Partikel von einer einmal falschen Lokalisierung wieder die richtige Lokalisierung zu finden.

Die Geschwindigkeit und die Qualität des Partikelfilters hängen von der Anzahl der Partikel ab. Da man diese verändern kann, ist es möglich, den Lokalisierungsalgorithmus genau auf die geforderten Bedürfnisse abzustimmen. Somit kann zwischen höherer Geschwindigkeit, die günstigere Hardware möglich macht, und der Genauigkeit abgewogen werden.

Wegen dieser Vorteile und der Nachteile des Kalman-Filters und des Histogram-Filters wurde ein Partikelfilter ausgewählt und implementiert.

Kapitel 5

Implementierung des Lokalisierungsalgorithmus

Die Gabelstaplerlokalisierung ist ein Lokalisierungsproblem in der Ebene. Dazu wird zunächst ein festes Bezugssystem beliebig definiert, zu dem nun alle absoluten Positionen angegeben werden. Dies kann bei einer Lagerhalle z.B. so geschehen, dass der Nullpunkt in die Mitte der Lagerhalle und die X-Achse parallel zu einer Wand gelegt wird.

Außerdem gibt es noch das lokale Koordinatensystem des Testfahrzeuges. Der Ursprung dieses Koordinatensystems liegt in der Lenkachse des Testfahrzeugs mit der X-Achse senkrecht zur Hinterachse. Diese Anordnung wurde gewählt, um die Odometrieberechnungen zu vereinfachen.

Die zu findende Position des Gabelstaplers besteht aus der X- und Y-Koordinate im absoluten Koordinatensystem sowie der Orientierung α als Winkel zwischen der absoluten X-Achse und der X-Achse des Fahrzeugkoordinatensystems. (Abbildung 5.1). Daher ist bei Verwendung eines Partikelfilters der Zustand x_t und damit jedes Partikel ein dreidimensionaler Vektor (x, y, α).

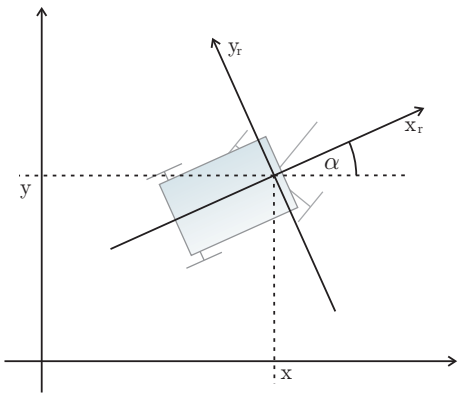


Abbildung 5.1: Koordinatensysteme und Positionsangabe bei der Gabelstaplerlokalisierung

5.1 Control-Update

Im ersten Schritt eines Durchlaufes des Partikelfilters wird die Vorhersage des Beliefs $\overline{Bel}(x_t)$ berechnet. Dazu werden die Kontrolldaten u_t in Form der Odometrie sowie ein Modell für die Zustandsübergangswahrscheinlichkeit, das „Motion Model“ benötigt.

5.1.1 Odometrie

Die Odometrie Δp wird aus den Messwerten der Radsensoren und des Lenkwinkelsensors berechnet und gibt die Änderung der Fahrzeugposition seit dem letzten Durchlauf des Algorithmus an.

Zur Berechnung der Odometrie wird der Lenkwinkel β und die seit dem letzten Update zurückgelegte Wegstrecke jedes einzelnen Radsensors $s_{vl}, s_{vr}, s_{hl}, s_{hr}$ (v = vorne, h = hinten, l = links, r = rechts) verwendet. Die zurückgelegte Wegstrecke ergibt sich dabei aus der Zählerdifferenz des Radsensors und der Strecke s_e pro Zahnkante der Encoderscheibe. Diese lässt sich aus dem Radradius r_r und der Anzahl x der Zähne der Encoderscheibe als

$$s_e = \frac{2\Pi r_r}{x} \tag{5.1}$$

berechnen und beträgt bei den verwendeten Rädern ca. 4,1 mm. Außerdem wird noch der Abstand zwischen Vorderachse und Hinterachse d benötigt.

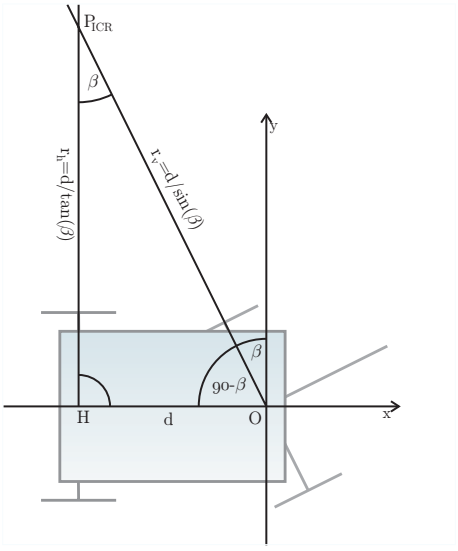


Abbildung 5.2: Berechnung des Kurvenmittelpunktes P_{ICR} mit Lenkwinkel β

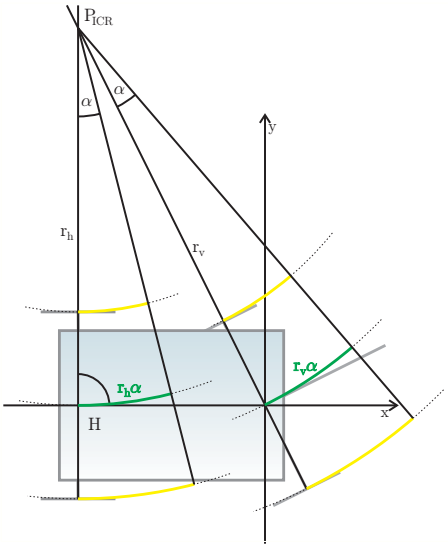


Abbildung 5.3: Berechnung des Drehwinkels α mit Hilfe des zurückgelegten Weges der Räder

Die Odometrie-Berechnungen werden alle in Koordinaten relativ zum Fahrzeug durchgeführt. Durch die Lage des Ursprungs in der Lenkachse des Testfahrzeugs und der gewählten Ausrichtung der X-Achse lassen sie sich an einigen Stellen besser vereinfachen, was zu einem geringeren Rechenaufwand führt.

Da im folgenden häufiger verschiedene Punkte in diesem Koordinatensystem verwendet werden, bezeichnet $O = (0, 0)$ den Ursprung und $H = (-d, 0)$ den Mittelpunkt der Hinterachse.

Immer wenn der Wagen eine Kurve fährt, ergibt sich durch den Lenkeinschlag dort, wo sich Geraden durch Vorder- und Hinterachse schneiden, der momentane Kurvenmittelpunkt $\mathrm{P_{ICR}}$ (instantaneous center of rotation). Da sich das gesamte Fahrzeug um diesen Punkt dreht, muss er zunächst berechnet werden. Der Punkt $\mathrm{P_{ICR}}$ wird nur mit dem Messwert des Lenkwinkelsensors bestimmt. Es wäre zwar auch möglich, die Radsensoren zu verwenden, aber die hohe Genauigkeit des Lenkwinkelsensors führt hier zu besseren Ergebnissen.

$\mathrm{P_{ICR}}$ liegt auf der Verlängerung der Hinterachse. Diese steht senkrecht zur X-Achse, weshalb die X-Koordinate $-d$ ist. Zur Berechnung der Y-Koordinate wird das Dreieck mit den Punkten $\mathrm{P_{ICR}}$, O und H betrachtet. Der Winkel in Punkt O ist dabei $90° - \beta$, der Winkel in Punkt H $90°$ und damit der Winkel in $\mathrm{P_{ICR}}$ ebenfalls der Lenkwinkel β. Dadurch lässt sich die Strecke $r_h = \frac{d}{tan(\beta)}$ berechnen. (Abbildung 5.2) Somit hat der Kurvenmittelpunkt die Koordinaten

$$\mathrm{P_{ICR}} = \left(-d, \frac{d}{\tan \beta}\right). \tag{5.2}$$

Die Räder bewegen sich, genauso wie der Rest des Wagens, auf einer Kreisbahn um $\mathrm{P_{ICR}}$. Der seit der letzten Berechnung der Odometrie zurückgelegte Weg eines

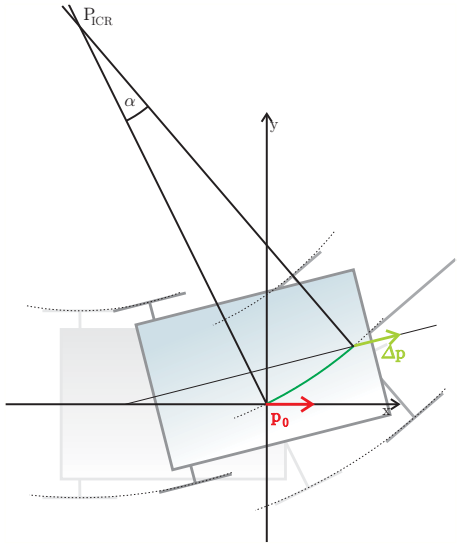


Abbildung 5.4: Berechnung der Änderung Δp der Position des Fahrzeuges durch Drehung der Nullposition um Punkt M mit Winkel α

beliebigen Punktes auf dem Wagen ist daher ein Kreisbogen der Länge $s = r \cdot \alpha$, wobei r der Abstand dieses Punktes zu $\mathrm{P_{ICR}}$ ist.

Zur Berechnung des Winkels α, den der Wagen gefahren ist, wird mit den Messwerten der hinteren Radsensoren der Winkel α_h berechnet. Dazu wird ein „virtuelles“ Rad in der Mitte der Hinterachse angenommen. Der Weg, den dieses Rad zurückgelegt hat (grün in Abbildung 5.3), ist das Mittel zwischen dem Weg des rechten Hinterrades und dem des linken Hinterrades (gelb). Es ist auch möglich, mit nur einem Rad zu rechnen, durch die Mittellung werden aber die Messfehler kleiner und es kann als Radius die schon berechnete Strecke r_h verwendet werden. Setzt man diesen Radius ein, ergibt sich

$$\begin{aligned} \frac{s_{hr} + s_{hl}}{2} &= \frac{d}{\tan\beta} \cdot \alpha_h \\ \Leftrightarrow \alpha_h &= \frac{(s_{hr} + s_{hl}) \tan\beta}{2d}. \end{aligned} \tag{5.3}$$

Analog lässt sich auch der Drehwinkel für die Vorderachse mit den vorderen Radsensoren und dem Radius $r_v = \frac{d}{\sin\alpha}$ berechnen.

$$\alpha_v = \frac{(s_{vr} + s_{vl}) \sin\beta}{2d} \tag{5.4}$$

Da im Idealfall, also ohne Messfehler bei den Radsensoren, $\alpha_v = \alpha_h$ ist, wird der Mittelwert $\alpha = \frac{\alpha_v + \alpha_h}{2}$ bei den weiteren Berechnungen verwendet.

Um die Odometrie zu erhalten, muss die Differenz Δp zwischen der alten relativen Fahrzeugposition und der neuen relativen Fahrzeugposition berechnet werden. Die

alte relative Fahrzeugposition ist die Nullposition $p_0 = (0, 0, 0)$, die neue Fahrzeugposition ergibt sich durch Drehung von p_0 um $\mathrm{P_{ICR}}$ mit dem Winkel α. (Abbildung 5.4). Dazu wird zunächst die homogene Matrix für eine Rotation um den $\mathrm{P_{ICR}}$ aus der Transformationsmatrix T für die Translation und der Rotationsmatrix R für den Winkel aufgestellt, wobei zu beachten ist, dass die dritte Komponente des Positionsvektors den Winkel darstellt:

$$\begin{aligned} M &= T \cdot R \cdot -T \\ &= \begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & \alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & -x \\ 0 & 1 & 0 & -y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & -x\cos\alpha + y\sin\alpha + x \\ \sin\alpha & \cos\alpha & 0 & -x\sin\alpha - y\cos\alpha + y \\ 0 & 0 & \alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{aligned} \tag{5.5}$$

Dann wird p_0 mit dieser Matrix multipliziert, um Δp zu erhalten:

$$\begin{aligned} \Delta p &= p_0 \cdot M \\ &= \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & -x\cos\alpha + y\sin\alpha + x \\ \sin\alpha & \cos\alpha & 0 & -x\sin\alpha - y\cos\alpha + y \\ 0 & 0 & \alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} -x\cos\alpha + y\sin\alpha + x \\ -x\sin\alpha - y\cos\alpha + y \\ \alpha \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} d\cos\alpha + \frac{d}{\tan\beta}\sin\alpha - d \\ d\sin\alpha - \frac{d}{\tan\beta}\cos\alpha + \frac{d}{\tan\beta} \\ \alpha \\ 1 \end{pmatrix} \end{aligned} \tag{5.6}$$

5.1.2 Motion Model

Zur Berechnung der Vorhersage $\overline{Bel}_x$ muss ein Modell aufgestellt werden, um die Zustandsübergangswahrscheinlichkeit $p(x_t|u_t, x_{t-1})$ auszudrücken.

Da die Verschiebung durch die Odometrie eine lineare Funktion ist, ändert sich der Zustand und damit jedes einzelne Partikel, das den Zustand repräsentiert, nach der Formel

$$x_t^{[m]} = x_{t-1}^{[m]} + \Delta p + \varepsilon. \tag{5.7}$$

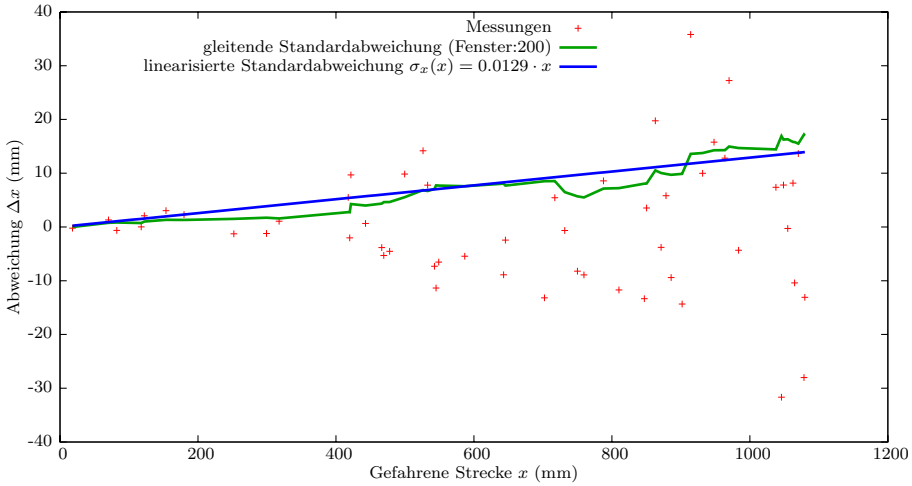


Abbildung 5.5: Messung der Abweichungen bei der X-Komponente. Bei größerer zurückgelegter Strecke ist auch die Abweichung größer.

Dabei sind Δp die Kontrolldaten in Form der Odometrie und ε eine gaußverteilte Zufallszahl. Jedes Partikel wird also um die seit dem letzten Durchlauf gefahrene Strecke und gefahrenen Winkel verschoben und gedreht, wobei zu jeder Komponente „Rauschen“ addiert wird. Wie stark dieses Rauschen für die einzelnen Komponenten der Odometriedaten ist, wurde experimentell bestimmt.

Zur Bestimmung der Abweichung der X-Komponente der Odometrie wurde das Testfahrzeug entlang einer Schiene geführt. Beim Start einer Messung wurde zunächst mittels des NAV-200 die aktuelle Position des Wagens bestimmt. Während der Wagen gezogen wurde, wurde immer wieder die Odometrie zu dieser Position addiert. Am Ende wurde dann die richtige Position noch einmal mit dem Referenzsystem gemessen und diese dann mit der durch die Odometrie gemessenen Position verglichen. Diese Messung wurde ca. 50 mal mit verschiedenen Geschwindigkeiten durchgeführt.

Anhand der Messwerte in Abbildung 5.5 erkennt man, dass die Abweichung abhängig von der zurückgelegten Strecke ist. Das ist auch einleuchtend, da bei jedem Durchlauf der Odometrieberechnung der Fehler hinzuaddiert wird. Daher wurde die Standardabweichung abhängig von der Strecke aus den Messwerten berechnet und mit der Funktion

$$\sigma_x(x) = 0,0129 \cdot x \tag{5.8}$$

approximiert.

Die Abweichung in Y-Richtung wurde auf dieselbe Art und Weise bestimmt und ist ebenfalls abhängig von der zurückgelegten Strecke (Abbildung 5.6). Die Tatsache, dass das Fahrzeug entlang einer Schiene gezogen wurde, ist dabei kein Problem, da hier der erwartete Messwert zwar 0 ist, aber davon abweichende Werte gemessen werden. Die Abweichung $\sigma_y(x) = 0,0117x$ hat auch ungefähr denselben

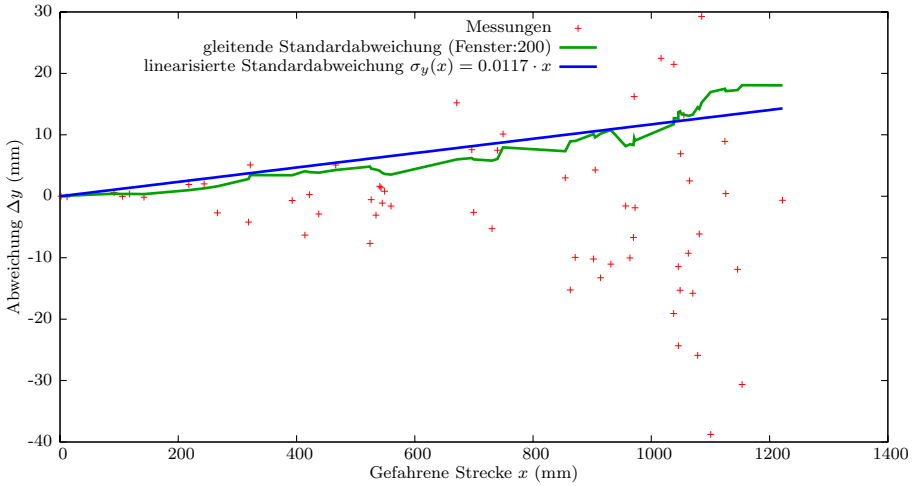


Abbildung 5.6: Messung der Abweichungen bei der Y-Komponente. Bei größerer zurückgelegter Strecke ist auch hier die Abweichung größer

Proportionalitätsfaktor. Dies ist auch einleuchtend, da die X-Komponente und die Y-Komponente durch die Drehung um P_{ICR} direkt voneinander abhängig sind.

Um die Abweichung der Orientierung zu bestimmen, wurde mit verschiedenen festen Lenkwinkeleinschlägen und damit mit festem P_{ICR} auf einer Kreisbahn unterschiedlich weit gefahren, um auch eine Abhängigkeit zur zurückgelegten Strecke erkennen zu können. Dabei wurde, ähnlich wie in den obigen Messungen, die vom Referenzsystem gelieferte Orientierung mit der über Odometrie berechneten verglichen.

Das Ergebnis dieser Messung ist in Abbildung 5.7 dargestellt. Es ist zu sehen, dass die Abweichung der Orientierung weder von der zurückgelegten Strecke, noch von dem verwendeten Lenkwinkel abzuhängen scheint. Dies steht aber im Widerspruch zu der Berechnung von Δp, da der Winkel α aus der zurückgelegten Strecke der Radsensoren berechnet wird. Es müsste sich, z.B. bei der Messung mit dem Lenkwinkel $\beta = 33,3°$ und einer zurückgelegten Strecke von s = 1000mm, eine Standardabweichung

$$\sigma_\alpha(s) = \frac{\sigma_s(s)}{r_v} = \frac{\sigma_s(s)}{\frac{d}{\sin(\beta)}} = \frac{0,0129 \cdot 1000mm}{\frac{550mm}{\sin(33,3°)}} \approx 0,0086 \approx 0,49° \tag{5.9}$$

ergeben.

Dieser Wert ist allerdings deutlich kleiner, als die Messgenauigkeit der GT-Cam, weshalb die obige Messung hauptsächlich die Winkelungenauigkeit der Deckenkamera wiedergibt und daher keine Abhängigkeiten von gefahrener Strecke und Lenkwinkeleinschlag erkennbar sind.

Problematisch bei der Betrachtung der Odometrie ist, dass der Reifenradius sich im Laufe der Zeit ändert, was durch Abrieb oder falschen Luftdruck geschehen kann. Da die Radsensoren eigentlich nicht die zurückgelegte Strecke, sondern den Dreh-

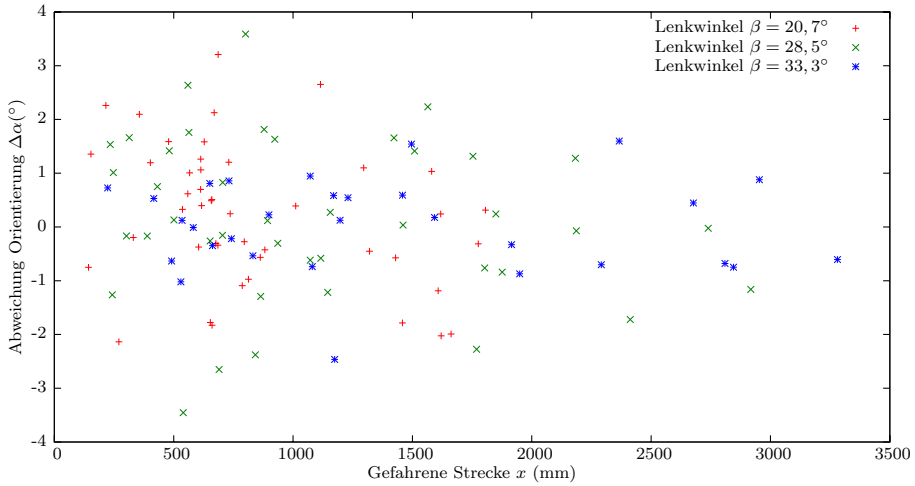


Abbildung 5.7: Messung der Abweichungen der Orientierung. Es ist im Gegensatz zu der X- und Y-Abweichung keine von der Strecke abhängige Abweichung erkennbar, da die Messungen der Deckenkamera nicht genau genug sind.

winkel der Räder messen, hat der Radius direkten Einfluss auf die Bestimmung der zurückgelegten Strecken $s_{vl}, s_{vr}, s_{hl}, s_{hr}$.

Bei einem falschen Radradius haben die Messwerte der Odometrie nicht nur eine von der zurückgelegten Strecke x abhängige Standardabweichung, sondern auch einen von x abhängigen Mittelwert. Die Messung der Strecke ist entweder zu kurz oder zu lang. Leider ist der momentane Radius und damit der Mittelwert nicht genau bekannt und wird daher in die Berechnung der Standardabweichung mit einbezogen.

Unter der Annahme, dass sich durch unterschiedlichen Luftdruck der Radius des Rades um bis zu 1cm ändert, kann die gemessene Strecke nun nicht mehr nur ca. 1,2%, sondern bis zu 10% von der richtigen Strecke abweichen. Daher wurde für das Motion-Model für jede der drei Komponenten ein Proportionalitätsfaktor von 0,1 bei der Berechnung der Standardabweichung verwendet.

Desweiteren wurden die Messwerte des Drehratensensor-Teils des Bosch MM3 mit der Orientierung von Δp verglichen. Im Normalfall war deren Abweichung mit weniger als einem Grad sehr klein. Nur wenn das Testfahrzeug mit Gewalt bei Volleinschlag der Lenkung noch weiter in die Kurve gezogen wurde, zeigten sich größere Abweichungen. Da dieser Fall bei einem Gabelstapler nicht vorkommt und die Abweichung im Normalfall einen deutlich geringeren Einfluss als der falsche Radradius hat, wurde die Abweichung im Motion-Model nicht weiter berücksichtigt.

Der Beschleunigungssensor war weder für die Odometrie, noch zu ihrer Validierung zu gebrauchen, da sein Rauschen im Vergleich zu den bei einer Testfahrt vorkommenden Beschleunigungen viel zu stark ist.

5.2 Measurement-Update

Im zweiten Schritt des Partikelfilters, dem Measurement-Update, wird für jedes Partikel eine Gewichtung bestimmt. Diese gibt die Wahrscheinlichkeit $p(z_t|x_t^{[m]})$ wieder, dass die aktuelle Messung z_t zu der angenommenen Position des aktuellen Partikels $x_t^{[m]}$ passt.

Die Gewichtung wird dabei aus der Differenz zwischen den gemessenen und den erwarteten Sensormesswerten bestimmt.

5.2.1 Bestimmung des erwarteten Messwertes

Zur Bestimmung der erwarteten Messwerte wurde zunächst eine Messung durchgeführt, um den Zusammenhang zwischen der Distanz eines Magneten und dem Messwert eines einzelnen Magnetfeldsensors des Magnetlineals zu bestimmen. Es wird die auf den Boden projizierte Distanz d verwendet, da der Lokalisierungsalgorithmus nur in der Ebene arbeitet. (Abbildung 5.8)

Um die Messwerte mit einer Formel anzunähern, wurde ein einfaches Modell des Magneten und seiner Feldstärke angenommen. Zunächst ist es so, dass der Sensor die gemessene Feldstärke in einen Messwert s (strength) umsetzt. Wie s mit der realen Feldstärke genau zusammenhängt, ist unbekannt, da dies genaue Kenntnis über z.B. Umsetzung der Feldstärke in eine Spannung im Sensor, die Verstärkung dieser Spannung und die Umsetzung der verstärkten Spannung in Zahlenwerte bei der Analog-Digital-Wandlung erfordern würde. All diese Faktoren werden daher in dem Faktor f zusammengefasst, der später mittels einer Messreihe approximiert wird.

Um den Messwert s berechnen zu können, wird vereinfacht angenommen, dass der Magnet mit seinem Nord- und Südpol aus zwei hypothetischen, unipolaren Magneten besteht, deren Feldstärkensumme die Gesamtfeldstärke ergeben. Die Feldstärke eines solchen unipolaren Magneten nimmt quadratisch zum Abstand ab, ist also $f \cdot 1/r^2$

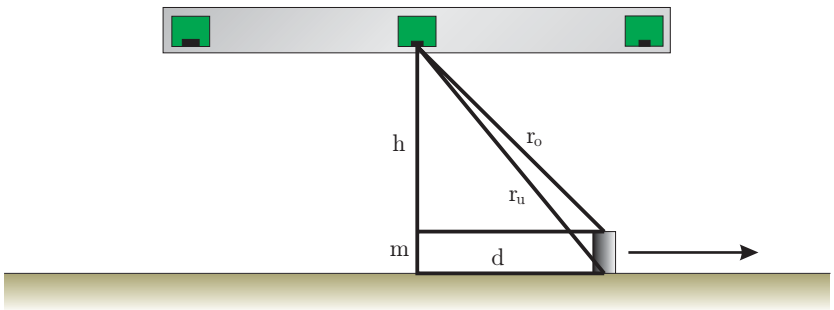


Abbildung 5.8: Bestimmung des erwarteten Messwertes abhängig vom Abstand d des Magneten auf dem Boden mittels der Höhe des Magneten m, der Höhe des Sensors über dem Magneten h, dem Radius zur Unterkante r_u und dem Radius zur Oberkante des Magneten

und die Feldstärkensumme ist, da die unipolaren Magnete entgegengesetzt gerichtet sind:

$$s = f\frac{1}{r_o^2} - f\frac{1}{r_u^2}. \tag{5.10}$$

Dabei ist r_o der Radius zur Mitte der Oberkante des Magneten und r_u der Radius zur Mitte der Unterkante (Abbildung 5.8). Sie können aus der Bodendistanz d, der Höhe h des Magnetlineals und der Höhe des Magneten berechnet werden:

$$r_o^2 = d^2 + h^2 \quad \text{und} \quad r_u^2 = d^2 + (h+m)^2 \tag{5.11}$$

Eingesetzt ergibt sich:

$$s(d) = f\frac{1}{d^2+h^2} - f\frac{1}{d^2+(h+m)^2} \tag{5.12}$$

Da der Sensor allerdings nur die Komponente s_x senkrecht zum Boden misst, ist der Messwert:

$$\begin{aligned} \frac{s_x(d)}{h} &= \frac{s(d)}{r_o} \\ \Leftrightarrow s_x(d) &= s(d)\cdot\frac{h}{r_o} = f\cdot\left(\frac{1}{d^2+h^2} - \frac{1}{d^2+(h+m)^2}\right)\cdot\frac{h}{\sqrt{d^2+h^2}} \end{aligned} \tag{5.13}$$

Mittels der gemessenen Werte wurde für diese Funktion der Parameter f bei gegebener Sensorhöhe $h = 285$mm und Magnethöhe $m = 15$mm approximiert (Abbildung 5.9) und zur Berechnung der erwarteten Sensorwerte für einen Magneten verwendet.

Um nun den Einfluss aller Magnete auf den Sensor zu bestimmen, wird für jeden Magneten die Distanz und der erwartete Messwert berechnet und alle zum Gesamtmesswert addiert. Dazu sind die Positionen aller Magnete in der MagnetMap gespeichert.

Damit diese Berechnung nicht für jedes Partikel und jeden Magnetfeldsensor erneut duchgeführt werden muss, werden die Ergebnisse für jede mögliche Sensorposition auf einem 50mm x 50mm Gitter vorberechnet und in einer entsprechenden Tabelle gespeichert.

5.2.2 Bestimmung der Partikelgewichtung

Zur Bestimmung der Partikelgewichtung aus den erwarteten und den gemessenen Messwerten wurden zwei verschiedene Ansätze ausprobiert.

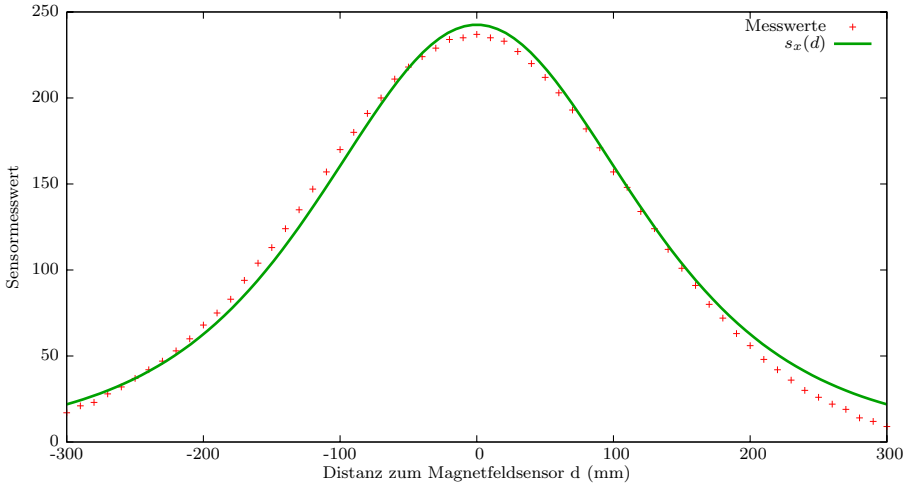


Abbildung 5.9: Messwerte für einen Magneten im Abstand d und deren Approximation durch die Funktion $s_x(d)$

Im ersten Ansatz wurde die Summe der Quadrate der Fehler verwendet. Seien S die Anzahl der Sensoren, e_i die erwarteten Messwerte für Sensor i und s_i die gemessenen Werte, so wird diese Summe mit

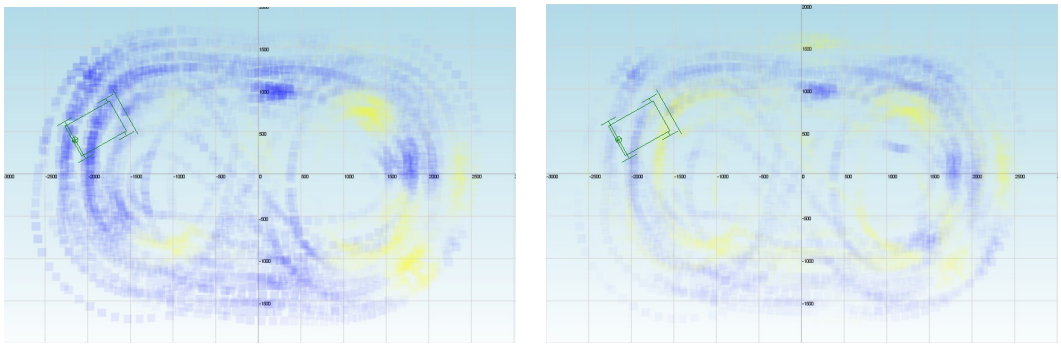
$$e = \sum_{i}^{S} (s_i - e_i)^2 \tag{5.14}$$

berechnet.

Dies entspricht allerdings noch nicht direkt der Gewichtung der Partikel, da eine eventuelle Messungenauigkeit noch nicht berücksichtigt ist. Dazu wird eine normalverteilte Wahrscheinlichkeitsdichte für das Auftreten einer gewissen Differenz e angenommen, deren Mittelwert $\mu = 0$ ist und deren Standardabweichung σ experimentell ermittelt werden kann. Diese Wahrscheinlichkeitsdichte wird dann als Gewichtung für die Partikel verwendet.

Bei ersten Tests zeigte sich allerdings, dass dieses Modell in der Praxis nicht sehr brauchbar ist. In der Halle, in der das System getestet wurde, lieferten die Sensoren des Magnetlineals auch ohne vorhandene Magnete stärker schwankende Messwerte, da sich im Boden aus irgendwelchen Gründen magnetisches Material befindet. (Abbildung 5.10(a)). Dies führte dazu, dass Partikel, die im Vergleich zu anderen Partikeln eine höhere Gewichtung hätten bekommen müssen, plötzlich eine niedrigere Gewichtung bekommen haben und verworfen wurden. Daher verlor der Lokalisierungsalgorithmus häufig seine Position.

Dieses Problem führte zu der zweiten Variante. Bei dieser Variante werden nicht die absoluten erwarteten Messwerte, sondern die Abweichungen zum Durchschnitt der Messwerte der einzelnen Sensoren des Magnetlineals verwendet:



(a) Unbearbeitete Messwerte (1. Variante) (b) Messwerte als Abweichung vom Mittelwert (2. Variante)

Abbildung 5.10: Messwerte der Magnetfeldsensoren in Abhängigkeit von der Position. (gelb: positives s_i, blau: negatives s_i, jeweils an den absoluten Positionen der Sensoren eingezeichnet)

$$\sum_{i}^{S} \left(\left(s_i - \frac{\sum_j^S s_j}{S} \right) - \left(e_i - \frac{\sum_j^S e_j}{S} \right) \right)^2 \tag{5.15}$$

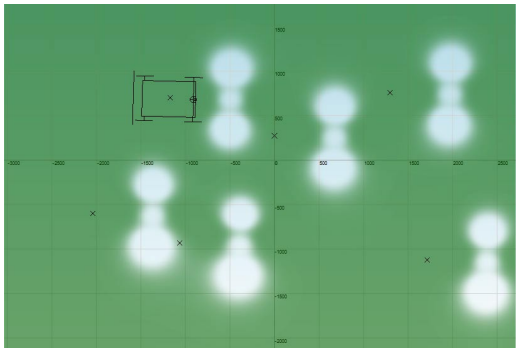
Wie zuvor wurde auch hier die Gewichtung über eine Normalverteilung bestimmt. Allerdings konnte wegen der geringeren Größe der Fehler eine kleinere Standardabweichung σ gewählt werden.

Diese zweite Variante wurde verwendet und die sich ergebende Gewichtung ist in den Abbildungen 5.11 und 5.12 für einige interessante Fälle exemplarisch dargestellt. Dabei ist die X-Achse und die Y-Achse variabel und die Orientierung auf die Orientierung der Referenzposition festgelegt, um eine zweidimensionale Darstellung zu ermöglichen.

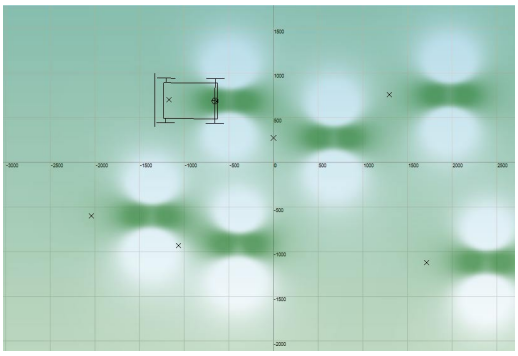
Abbildung 5.11 zeigt vier interessante Positionen beim Überfahren eines Magneten. Zunächst befindet sich der Wagen weit entfernt vom nächsten Magneten. Daher ist die Gewichtung überall hoch, außer an den Stellen, an denen ein Magnet gemessen werden würde. Diese Stellen befinden sich entsprechend der aktuellen Orientierung vor den Magneten (Abbildung 5.11(a)). Bewegt sich der Wagen nun weiter, so dass der Magnetfeldsensor in die Nähe eines Magneten gelangt, so werden die Positionen, für die kein Magnet gemessen werden dürfte, immer unwahrscheinlicher und die, für die ein Magnet gemessen werden dürfte, immer wahrscheinlicher. (Abbildung 5.11(b)). Die wahrscheinlichen Positionen werden nun immer weniger, bis sich der Magnet genau unter dem Magnetlineal befindet (Abbildung 5.11(c)).

Abbildung 5.12 zeigt die Gewichtung für verschiedene Magnetpositionen unter dem Magnetlineal. Abbildung 5.12(a) zeigt zunächst noch einmal den Fall aus Abbildung 5.11(c), bei dem sich der Magnet genau mittig unter dem Magnetlineal befindet.

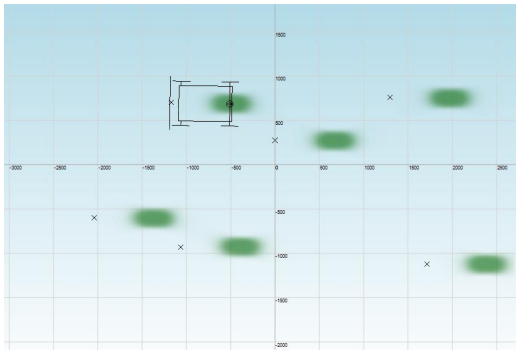
Wird der Magnet etwas weiter links überfahren (Abbildung 5.12(b)), so ist zu sehen, dass sich die Gewichtungsmaxima entsprechend verschieben. Direkt neben den



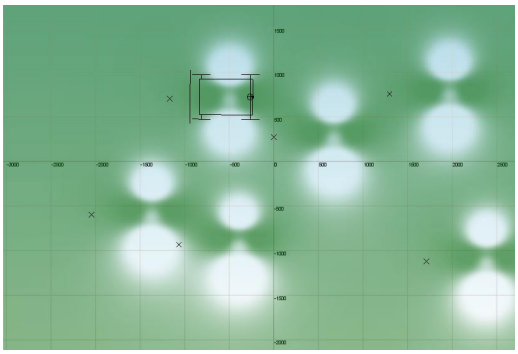
(a) Es wird kein Magnet gemessen. Daher kann sich der Wagen an jeder Position außer an denen befinden, wo ein Magnet gemessen werden würde.



(b) Der Wagen nähert sich einem Magneten. Die Wahrscheinlichkeit wird daher höher, dass er sich entsprechend der Montageposition des Magnetlineals vor einem Magneten befindet.

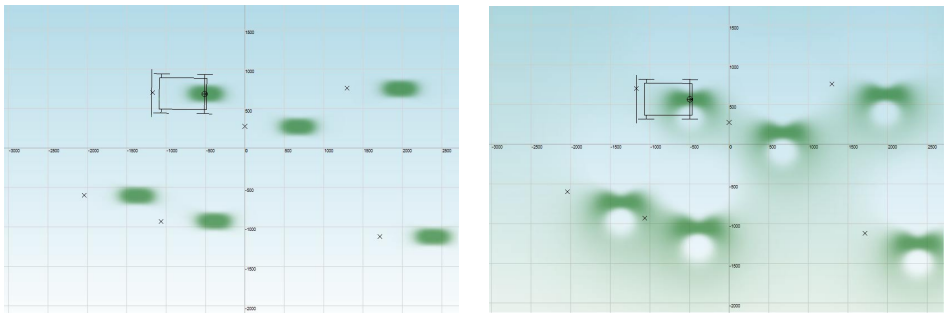


(c) Ein Magnet befindet sich direkt unter dem Wagen. Es ergeben sich wahrscheinliche Positionen bei jedem Magneten im Abstand der Montageposition des Sensors.



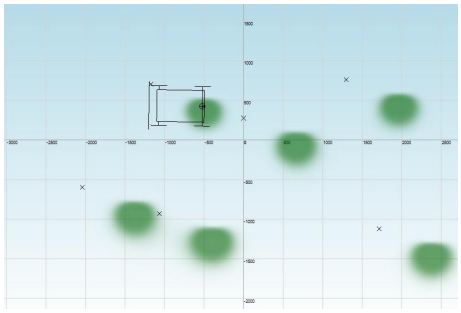
(d) Der Wagen fährt weiter.

Abbildung 5.11: Zeitliche Veränderung der durch das Sensor-Model berechneten Partikelgewichtung beim Überfahren eines Magneten. Der Wagen wurde an der Referenzposition eingezeichnet. Die schwarzen Kreuze markieren Magnetpositionen.



(a) Der Magnet wird mit der Mitte des Magnetlineals überfahren

(b) Der Magnet wird etwas weiter links überfahren



(c) Der Magnet wird ganz links überfahren

Abbildung 5.12: Partikelgewichtung für unterschiedliche Positionen eines Magneten unter dem Magnetlineal. Der Wagen wurde an der Referenzposition eingezeichnet

Maxima befinden sich allerdings auch starke Minima, da diese Positionen besonders unwahrscheinlich sind, da der linke Magnetfeldsensor dort einen sehr viel höheren Messwert hätte liefern müssen.

Befindet sich der Magnet direkt unter dem linken Magnetfeldsensor (Abbildung 5.12(c)), so sind die Maxima entsprechend noch weiter verschoben. Es ist auch zu sehen, dass die Positionen rechts vom Maximum wahrscheinlicher sind, da kein weiter links am Magnetlineal liegender Sensor vorhanden ist, der diese Positionen mit einem nicht erwarteten Messwert unwahrscheinlicher machen könnte.

5.3 Resampling

Der dritte Schritt des Partikelfilters ist das Resampling. Hier werden aus den Quellpartikeln $x_t^{[m]}$ mit der im Measurement-Update ermittelten Gewichtung $w_t^{[m]}$ neue Ergebnispartikel $\bar{X}_t^{[m]}$ ohne Gewichtung generiert, deren Verteilung dann den neuen Belief $Bel(x_t)$ repräsentiert. Dazu wurden verschiedene Möglichkeiten implementiert, die sich in der Sampling-Variance und Laufzeit unterscheiden.

Gemeinsam haben all diese Möglichkeiten, dass sie einige der Quellpartikel abhängig von ihrer Gewichtung auswählen. Dabei sollen Partikel mit stärkerer Gewichtung häufiger ausgewählt werden als Partikel mit niedrigerer Gewichtung. Da die Partikelanzahl M konstant bleiben soll, gibt es Quellpartikel, die mehrmals zu Ergebnispartikeln verarbeitet werden und Quellpartikel, die komplett aus der Partikelmenge herausfallen.

5.3.1 Zufällige Auswahl von Partikeln

Diese Variante entspricht genau der in Kapitel 4.4 theoretisch vorgestellten Variante, in der ein Partikel zufällig, aber mit zur Gewichtung proportionaler Wahrscheinlichkeit ausgewählt wird.

Dazu wird zunächst eine Zufallszahl r zwischen 0 und der Summe aller Gewichte $S = \sum_i w_t^{[i]}$ bestimmt, und der Quellpartikel $x_t^{[m]}$ mit dem größten m gesucht, für den die Summe seiner Vorgängerpartikel gerade noch kleiner als r ist (Abbildung 5.13):

$$\arg_m \max \sum_{i=1}^{m-1} w_t^{[i]} < r \tag{5.16}$$

Um dieses Partikel zu finden, wird ein binärer Suchbaum verwendet, in den jedes Partikel zusammen mit der Summe der Gewichtungen seiner Vorgängerpartikel eingetragen wird. Mit der Zufallszahl r wird nun die Summe gesucht, die gerade kleiner als r ist und der gefundene Quellpartikel wird als Ergebnispartikel in die neue Partikelmenge eingefügt.

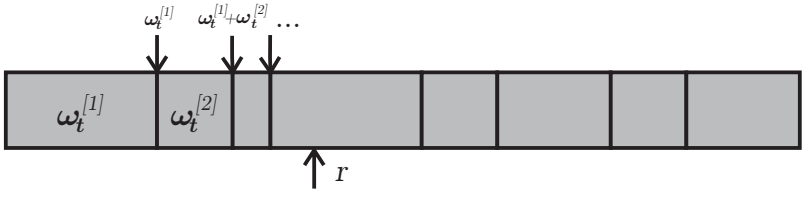


Abbildung 5.13: Mit der Zufallszahl r wird mittels eines binären Suchbaumes das Partikel $x_t^{[m]}$ mit $\sum_{i=1}^{m-1} w_i < r$ und größtem m gesucht.

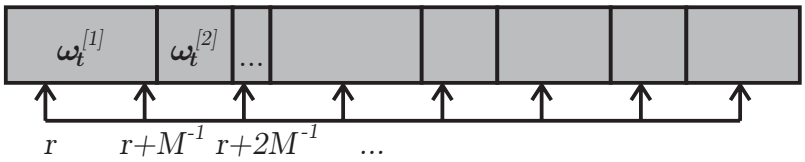


Abbildung 5.14: Auswahlverfahren beim Low-Variance-Sampler. Es wird das Quellpartikel verwendet, bei dem die Summe der Gewichtungen der Vorgängerpartikel gerade noch kleiner als $r + m \cdot M^{-1}$ ist.

Die Laufzeit für die Suche dieses Partikels beträgt wegen der Verwendung des binären Suchbaums $O(\log(M))$, wobei M die Anzahl der Partikel ist. Da für die neue Partikelmenge wiederum M Partikel benötigt werden, ist die Gesamtlaufzeit des Algorithmus $O(M\ \log(M))$.

Die zufällige Auswahl der Partikel führt außerdem zu einer hohen Sampling-Variance. Hier ist es z.B. unwahrscheinlich, dass bei gleicher Gewichtung aller Partikel auch alle genau einmal ausgewählt werden oder es kann vorkommen, dass Partikel mit sehr niedriger Gewichtung mehrmals ausgewählt werden.

5.3.2 Low-Variance Sampler

Dieser in [18] beschriebene Resampling-Algorithmus wählt die Partikel nicht völlig zufällig, sondern nur abhängig von einer Zufallszahl r aus. Dieser Algorithmus geht davon aus, dass die Summe der Gewichtungen der Quellpartikel = 1 ist, weshalb die Gewichtungen entsprechend normiert sein müssen. Er wählt nun für das Ergebnispartikel $x_t^{[1]}, ..., x_t^{[m]}$ dasjenige Quellpartikel aus, bei dem die Summe der Vorgängerpartikel gerade noch kleiner $r + m \cdot M^{-1}$ ist. (Abbildung 5.14)

Algorithmus 5 : Low variance sampler

Eingabe : Quellpartikelmenge X_t, Gewichtungen $w_t^{[1]} \dots w_t^{[M]}$
Ausgabe : Ergebnispartikelmenge $\bar{X}_t$

```
1  X̄_t = ∅
2  r = rand(0; M^-1)
3  c = w_t^[1]
4  i = 1
5  for m = 1 ... M do
6      U = r + (m − 1) · M^-1
7      while U > c do
8          i = i + 1
9          c = c + w_t^[i]
10     end
11     Füge x_t^[i] zu X̄_t hinzu.
12 end
```

Zur Auswahl der Partikel werden die Quellpartikelmenge X_t und die Ergebnis-Partikelmenge $\bar{X}_t$ parallel durchlaufen. i ist dabei die aktuelle Position in der Quell-Partikelmenge, c die Summe der Gewichte der bisher betrachteten Quell-Partikel und m die Position des Partikels in der Zielpartikelmenge.

Für gegebenes m wird in Zeile 6 die nächste Auswahlposition U bestimmt. Dann wird in der While-Schleife solange die Summe c und die Quellposition i erhöht, bis die Summe kleiner als U ist. Das so gefundene Partikel wird hinzugefügt. Nun kann es allerdings vorkommen, dass bei der Prüfung der While-Bedingung die Summe c schon größer als U ist. In diesem Fall wird i nicht erhöht, was bedeutet, dass das Partikel des letzten Durchlaufes noch einmal eingefügt wird.

Dieser Algorithmus hat gegenüber der zufälligen Auswahl einige Vorteile. Zunächst geht er systematisch bei der Auswahl der Partikel vor. Haben z.B. alle Partikel die gleiche Gewichtung, so werden alle auch genau einmal ausgewählt. Auch kann es nicht vorkommen, dass ein Partikel mit einer Gewichtung $w < M^{-1}$ häufiger als einmal ausgewählt wird. Daher ist die Sampling-Variance bei diesem Algorithmus kleiner.

Außerdem ist seine Laufzeit nur $O(M)$, wodurch er der zufälligen Auswahl mit der Laufzeit $O(M\ \log(M))$ vorzuziehen ist. Weniger Laufzeit beim Resampling bedeutet, dass entweder Hardware mit weniger Rechenleistung benötigt wird oder, bei gleicher Hardware, mit mehr Partikeln gearbeitet werden kann.

5.3.3 Unabhängige Partikel

Die oben angegebenen Resampling-Verfahren können beide eine verloren gegangene Position zunächst nicht wiederfinden, wenn sich die Partikel um eine falsche Position häufen und sich an der richtigen Position keine Partikel mehr befinden. Der Partikelfilter benötigt immer Partikel in der Nähe der richtigen Position, um diese wiederzufinden und es gibt verschiedene Möglichkeiten, diese Partikel unabhängig von dem vorherigen Belief zu erzeugen.

Die erste Möglichkeit, unabhängige Partikel zu erzeugen, ist das Hinzufügen von einem gewissen Prozentsatz zufälliger Partikel. Dies ist auch sinnvoll, da die Wahrscheinlichkeitsdichte für die Lokalisierung nirgendwo ganz null sein sollte, da es immer sein kann, dass die Lokalisierung fehlerhaft ist. Allerdings haben zufällig gewählte Partikel bei dem hier verwendeten Sensor-Modell den Nachteil, dass sie unter Umständen sehr lange in der Partikelmenge verbleiben, da sie erst beim Überfahren des nächsten Magneten als falsch erkannt werden und erst dann eine niedrige Gewichtung erhalten. Zusätzlich werden in jedem Resamplingschritt neue zufällige Partikel hinzugefügt, was die Sampling-Variance stark erhöht und letztendlich zum Verlieren der richtigen Position führen kann.

Erste Tests des Lokalisierungsalgorithmus haben ergeben, dass ca. 5% zufällige Partikel nötig sind, um eine Position zu finden. Dieser Wert war allerdings schon so hoch, dass die richtige Position häufig verloren wird.

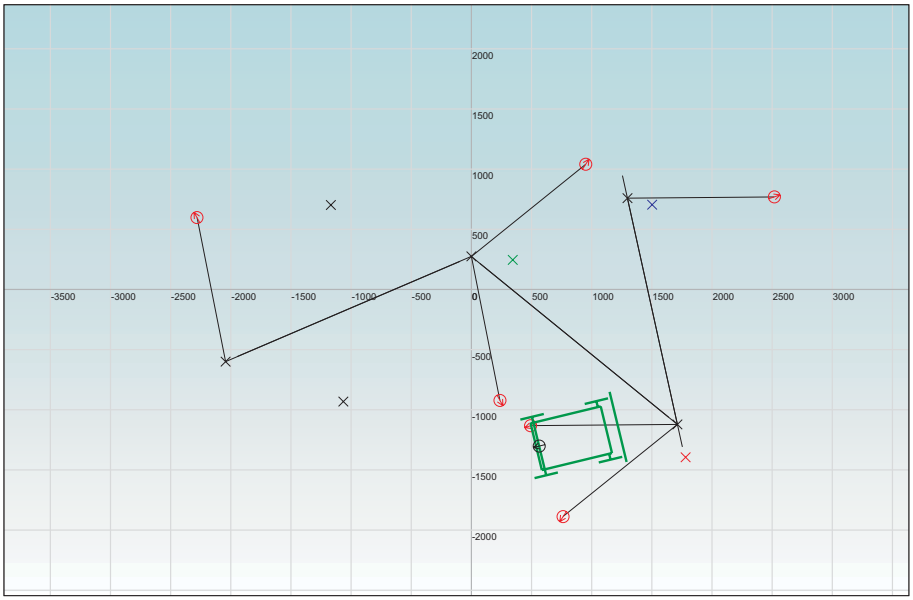


Abbildung 5.15: Es wurden 6 Template-Partikel (rote Kreise) mittels der letzten gemessenen Magnetpositionen (grünes und rotes Kreuz) ermittelt. Eines der Template-Partikel befindet sich erwartungsgemäß in der Nähe der realen Position (grüner Wagen). Schwarze Kreuze sind absolute Magnetfeldpositionen.

Dem wird mittels der schon in Kapitel 4.4.2 erwähnten Templates entgegengewirkt [22] [23]. Templates sind Partikel, die anstatt der zufällig ausgewählten Partikel beim Resampling eingefügt werden und dabei so gewählt wurden, dass sie über eine längere Zeit eine hohe Bewertung bekommen können.

Dies wird dadurch erreicht, dass, auf eine vom Partikelfilter unabhängige Art und Weise, all die Positionen ermittelt werden, auf denen sich das Fahrzeug befinden könnte, wenn nur die relativen Positionen der letzten beiden Magnete bekannt sind. Diese möglichen Positionen werden dann als Templates verwendet. Eines dieser Template-Partikel befindet sich dann in der Nähe der richtigen Position und wird beim Überfahren des nächsten Magneten „überleben“, während die anderen Template-Partikel wegen ihrer schlechten Gewichtung herausfallen. (Abbildung 5.15)

Um die Template-Partikel zu erzeugen, muss zunächst festgestellt werden, ob sich ein Magnet unter dem Magnetlineal befindet, was mittels der Messwerte der Magnetfeldsensoren durchgeführt wird. Ist einer der Messwerte größer als ein festeingestellter Grenzwert, so wird davon ausgegangen, dass ein Magnet vorhanden ist.

Nun wird die Position unter dem Magnetlineal über eine Best-Fit Methode bestimmt. Dazu wird für jede Position in Abstand von 1cm entlang des Magnetlineals zunächst der erwartete Messwert für jeden Magnetfeldsensor mit berechnet, mit dem gemessenen Messwert verglichen und die Position mit der geringsten Abweichung als Magnetposition angenommen. Diese Position wird relativ zum Fahrzeug gespeichert.

Entfernt sich der Sensor wieder vom Magneten, wird diese relative Position mittels der Odometrie verschoben. Sobald über den nächsten Magneten gefahren wird, kann daher der Abstand zwischen diesen beiden Magneten berechnet werden.

Nun werden in der MagnetMap, der Karte der absoluten Magnetpositionen, alle Paare von Magneten gesucht, deren Abstand zu dem über die Odometrie ermittelten Abstand passt.

Um aus diesen Magnetpaaren die Template-Partikel zu erhalten, werden alle relativen Positionen, also die der beiden Magnete und die Position des Wagens, zunächst so verschoben, dass die Position des zuletzt überfahrenen Magneten auf der Position eines Magneten des Paares liegt, und danach so gedreht, dass die Verbindungslinie zwischen den beiden relativen Magnetpositionen auf der Verbindungslinie der Magnete des Magnetpaares liegt. Somit ergibt sich an der verschobenen und gedrehten Position des Ursprungs des Fahrzeuges eine wahrscheinliche Position. Außerdem ergibt sich eine zweite Position, wenn die Magnetpositionen des gefundenen Magnetpaares vertauscht werden.

5.4 Bestimmung der Position

Zur Weiterverarbeitung des Lokalisierungsergebnisses ist es notwendig, aus dem als Partikelmenge dargestellten Belief eine einzige Position des Gabelstaplers zu ermitteln. Konkret gefordert ist hier die Position des Gabelstaplers, bestehend aus X-Koordinate, Y-Koordinate und Orientierung, sowie eine Validität, die angibt, wie „sicher sich“ der Algorithmus bei dieser Schätzung ist.

Eine einfache Möglichkeit, dies zu erreichen ist, den Mittelwert und die Standardabweichung der Partikelmenge zu bestimmen, den Mittelwert als Position zu verwenden und aus der Standardabweichung die Validität zu bestimmen.

Allerdings kommt es häufig vor, dass es nur einige wenige Bereiche oder Cluster des Zustandsraumes gibt, in denen sich sehr viele Partikel befinden. Kommt es z.B. dazu, dass sich 50% der Partikel in der Nähe der richtigen Position und die anderen 50% sich ebenfalls alle in einem kleinen Bereich, aber weit von der richtigen Position entfernt, befinden, liegt der Mittelwert irgendwo dazwischen, wo sich gar keine Partikel befinden.

Eine andere Idee war es, K-Mean-Clustering zu verwenden. Mit diesem Verfahren ist es möglich, die Partikelmenge in k verschiedene Cluster zu unterteilen, deren Mittelwerte zu bestimmen und den Cluster mit den meisten Partikeln auszuwählen. Allerdings muss vorher die Anzahl k der Cluster bekannt sein, die man finden möchte. Da man die Anzahl dieser Cluster aber im Voraus nicht kennt, ist K-Mean-Clustering hier nicht brauchbar.

Eine weitere Möglichkeit ist es, ein Gitter mit einer festen Zellenbreite von z.B. 15cm zu verwenden. Dann werden die Partikel durchlaufen und für jede Zelle die Anzahl Partikel gezählt, deren X- und Y-Position innerhalb dieser Zelle liegen. Dabei wird

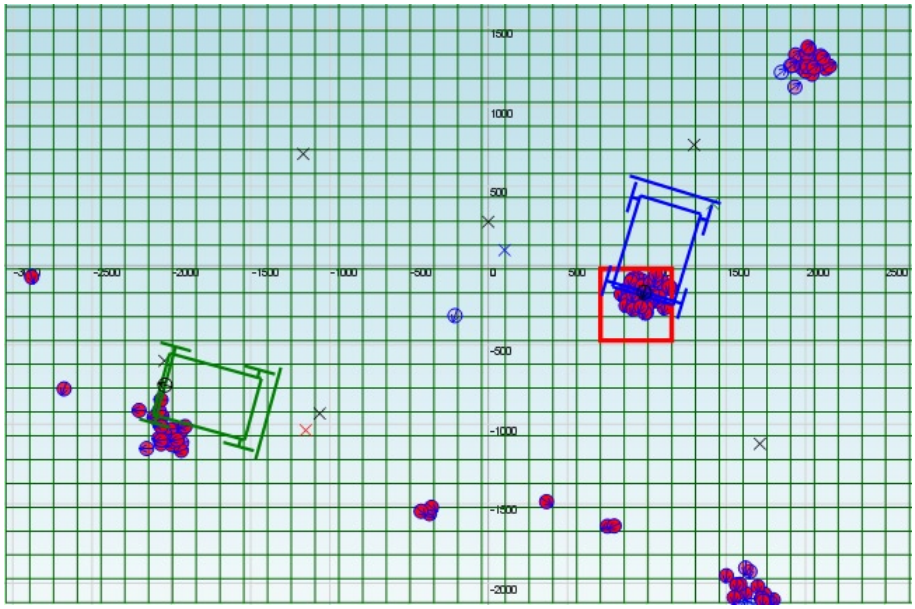


Abbildung 5.16: Bestimmung des größten Clusters von Partikeln mittels eines Gitters. Zur Bestimmung der Position (blauer Wagen) wird die Gitterzelle mit den meisten Partikeln und deren Nachbarzellen (rot umrandet) verwendet

auch die Zelle mit der höchsten Partikelanzahl gespeichert und zuletzt die Position als Mittelwert der Partikel aus dieser Zelle bestimmt.

Dieser Algorithmus hat den Vorteil, dass er den Cluster mit den meisten Partikeln unabhängig von der Anzahl der Cluster zurückliefert, hat allerdings das Problem, dass die Position „springt“, sobald sich der Cluster von einer Zelle in die nächste Zelle bewegt. Um dies abzumildern, werden zur Bestimmung des Mittelwertes nicht nur die Partikel der Zelle mit den meisten Partikeln, sondern auch alle Partikel aller benachbarten Zellen herangezogen. Diese Variante wurde letztendlich implementiert.

In Abbildung 5.16 ist eine Situation dargestellt, bei der der Partikelfilter die richtige Position (grüner Wagen) noch nicht gefunden hat. Es existieren vier große Cluster und die Position wurde mit den Partikeln aus dem rot umrandeten Bereich berechnet. Die mittlere Gitterzelle dieses Bereiches ist die Zelle mit den meisten Partikeln.

Als Validität wird das Verhältnis der zur Positionsbestimmung verwendeten Partikel zur Anzahl der Gesamtpartikel verwendet. Sobald der Lokalisierungsalgorithmus die richtige Position gefunden hat, sollte diese Validität sehr hoch sein, da sich viele Partikel in einem Cluster befinden. Ist diese Position allerdings die falsche Position, so ist die Validität weiterhin hoch. Erst wenn der nächste Magnet überfahren wird, werden alle Partikel dieses Clusters schlecht gewichtet und es existieren hoffentlich einige Template-Partikel, die wegen der hohen Gewichtung, die sie erhalten, für die Bildung neuer Cluster sorgen.

Kapitel 6

Evaluierung

In diesem Kapitel wird die Qualität des Lokalisierungsalgorithmus untersucht. Dazu werden zunächst verschiedene Qualitätskriterien aufgestellt und diese dann unter verschiedenen Bedingungen, d.h. mit verschiedenen Vorgaben für die Größe der Halle und mit verschiedenen Parametern für den Algorithmus untersucht.

Das Haupt-Qualitätskriterium bei einem Lokalisierungsalgorithmus ist die Genauigkeit, d.h. wie weit die vom Lokalisierungsalgorithmus berechnete Position von der realen Position entfernt ist. Es ist allerdings erst dann sinnvoll, diese Abweichung zu bestimmen, wenn der Lokalisierungsalgorithmus einmal eine absolute Position gefunden hat.

Daher ist ein zweites wichtiges Qualitätskriterium die Lokalisierungszeit. Dies ist die Zeit, bis die berechnete Position das erste Mal zuverlässig in der Nähe der richtigen Position liegt.

Außerdem kann es vorkommen, dass der Lokalisierungsalgorithmus eine richtige Lokalisierung verliert. In diesem Fall ist von Interesse, wie oft dies geschieht und wie schnell die richtige Position wieder gefunden wurde.

Zur Bestimmung dieser Qualitätsparameter wurden verschiedene Tests durchgeführt, die alle von der Möglichkeit, Sensormesswerte aufzuzeichnen und mehrmals wiedergeben zu können, Gebrauch machten. Dadurch, dass es möglich ist, den Lokalisierungsalgorithmus mit verschiedenen Parametern auf denselben Messdaten arbeiten zu lassen, wurde eine große Menge Testdurchläufe möglich, deren Ergebnisse besser untereinander vergleichbar sind.

Der erste Testdatensatz ist die Aufzeichnung einer Testfahrt mit dem Testfahrzeug. Diese fand auf dem Roboterfußball-Spielfeld unter der Deckenkamera statt und dauerte ca. 100 Sekunden. Dabei wurden 6 Magnete auf einer Fläche von 24m^2 verteilt. In Abbildung 6.1 ist dieser Versuchsaufbau fotografiert. Um die Magnete nicht im Boden versenken zu müssen, wurden sie in ein Holzbrett eingebaut. Dieses ist so abgeschrägt, dass mit dem Wagen darüber gefahren werden kann, ohne dass es sich verschiebt.

Abbildung 6.1: Testaufbau zur Aufzeichnung der Testdaten. Das Feld ist 6m x 4m groß. Es wurden 6 Magnete verwendett, die in abgeschrägte Holzbretter eingebaut wurden.

Weitere Testdatensätze wurden mit Hilfe des Simulators mit zwei verschiedenen Hallengrößen von 100m^2 und 400m^2 erzeugt, um eventuelle Auswirkungen der Größe zu erfassen. Da hier auch mit dem Aufzeichnungsverfahren gearbeitet wurde, war sichergestellt, dass für alle Tests mit derselben Hallengröße derselbe Weg mit dem Wagen im Simulator zurückgelegt wurde.

6.1 Magnetverteilung

Bevor die Tests durchgeführt werden konnten, musste zunächst eine Verteilung für die Magnete gewählt werden, die es ermöglicht, eine globale Position zu finden und diese zu verfolgen. Da die Magnete einzeln für sich gesehen nicht unterscheidbar sind, muss der Lokalisierungsalgorithmus sie anhand ihrer Abstände unterscheiden. Dies ist nur dann möglich, wenn es möglichst viele unterschiedliche Abstände zwischen zwei benachbarten Magneten gibt.

Ein regelmäßiges Gitter ist nicht brauchbar, da es zum einen periodisch ist und es zum anderen nur zwei verschiedene Abstände zwischen benachbarten Magneten gibt. Der Lokalisierungsalgorithmus hat damit keine Möglichkeit, eine globale Position zu finden.

Es gibt nicht-periodische Muster, die aus einigen wenigen verschiedenen Teilen zusammengesetzt werden können. Beim Penrose-Muster [24] sind dies nur zwei ver-

schiedene Vierecke. Die wenigen verwendeten Teile führen aber zu wenigen verschiedenen Abständen zwischen zwei benachbarten Magneten, weshalb solche Muster nicht zum Finden einer Position geeignet sind.

Mit einer zufälligen Verteilung der Magnete würde der Lokalisierungsalgorithmus funktionieren, aber es könnten große Bereiche in der Halle entstehen, auf denen sich kein Magnet befindet.

Im Simulator wurde daher ein regelmäßiges Gitter mit der Gitterweite g verwendet, die Magnete aber noch jeweils zusätzlich um maximal $\frac{1}{4}\ g$ zufällig verschoben. Damit ist ein Mindestabstand zweier benachbarter Magnete $\frac{1}{2}\ g$ und ein Maximalabstand von $\frac{3}{2} \cdot \sqrt{2}\ g$ sichergestellt und durch die zufällige Verteilung gibt es viele verschiedene Abstände zwischen benachbarten Magneten. Mit dieser Verteilung ist eine Lokalisierung problemlos möglich.

Bei dem realen Test wurden die Magnetpositionen manuell vergeben, was auch die beste Lösung bei einer späteren Anwendung sein dürfte. Bei der manuellen Verteilung ist es möglich, auf bestimmte lokale Gegebenheiten Rücksicht zu nehmen. Z.B. kann in der Nähe der LKW-Ladebuchten eine dichtere Magnetverteilung gewählt werden, damit die Position dort genauer bestimmt werden und eine eventuell verlorene Position schneller wiedergefunden werden kann, oder es können in langen Fluren, von denen kein Abbiegen möglich ist, Magnete eingespart werden. Allerdings ist immer darauf zu achten, dass die Abstände zwischen benachbarten Magneten unregelmäßig sind.

6.2 Laufzeit

Die Laufzeit des Lokalisierungsalgorithmus ist ausschließlich von der Partikelanzahl abhängig, da alle Teilschritte für ein einzelnes Partikel konstante Laufzeit benötigen. Die auf dem Entwicklungsrechner, einem Intel Core Duo mit 1,6 GHz, bei dem allerdings der Partikelfilter auf nur einem der beiden Prozessoren läuft, ermittelten Laufzeiten sind in Tabelle 6.1, getrennt für die 4 Phasen des Lokalisierungsalgorithmus, bei Verwendung von 1000 Partikeln angegeben.

	Laufzeit
Control-Update	2,4 ms
Sensor-Update	5,6 ms
Resampling	2,0 ms
Positionsbestimmung	0,4 ms
Gesamt:	10,4 ms

Tabelle 6.1: Laufzeiten der einzelnen Teile des Lokalisierungsalgorithmus bei 1000 Partikeln

Das Testfahrzeug liefert neue Messdaten mit einer Frequenz von 10 Hz, was bedeutet, dass für einen Echtzeit-Betrieb des Lokalisierungsalgorithmus dieser pro Durchlauf

maximal 100 ms benötigen darf. Daher können mit dieser Hardware und dieser Implementierung maximal ca. 9600 Partikel verarbeitet werden.

Allerdings ist die Implementierung des Lokalisierungsalgorithmus nicht auf Geschwindigkeit optimiert. Er verwendet beispielsweise weder die speziellen Assemblerbefehle der verwendeten CPU für Matrizenmultiplikationen, noch die speziellen Programmiertricks, die bei der Verwendung von C# notwendig sind, wenn man eine auf Geschwindigkeit optimierte Implementierung erstellen möchte. Es kann daher davon ausgegangen werden, dass mit einer hardwarenäheren, optimierten Implementierung in z.B. C++ möglich ist, eine deutlich höhere Partikelanzahl zu verwenden.

Bei allen folgenden Betrachtungen der Auswirkungen der Partikelanzahl ist es das Ziel, mit möglichst wenigen Partikeln zu arbeiten, damit entweder eine günstigere Hardware gewählt werden kann oder auf einer gegebenen Hardware mehr Rechenzeit für andere Aufgaben übrig bleibt.

6.3 Lokalisierungszeit

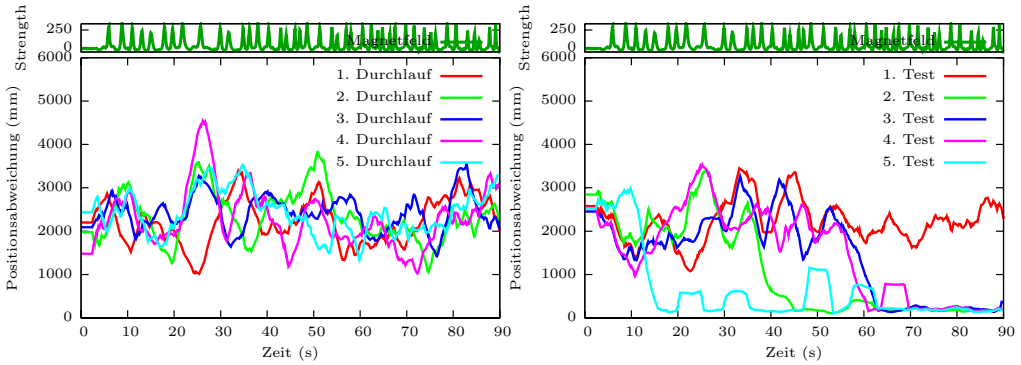
Sollen Aussagen über die Abweichung der richtigen Position von der berechneten Position gemacht werden, so muss zuerst gewartet werden, bis der Gabelstapler einmal richtig lokalisiert ist, d.h. die berechnete Position in der Nähe der richtigen Position liegt. Um diese Lokalisierungszeit bis zur Erstlokalisierung zu bestimmen, wurde der erste Zeitpunkt gesucht, zu dem die berechnete Position länger als 3 Sekunden näher als 50cm an der richtigen Position liegt.

Bei der Lokalisierungzeit ist es das Ziel, eine möglichst niedrige Partikelanzahl zu finden, mit der der Lokalisierungsalgorithmus möglichst schnell und zuverlässig eine Position in der Nähe der richtigen Position findet. Schnell bedeutet hier, dass die Position nach dem Überfahren einiger weniger Magnete gefunden wird und zuverlässig, dass dies nicht nur zufällig einmal, sondern fast immer geschieht.

6.3.1 Abhängigkeit von der Partikelanzahl

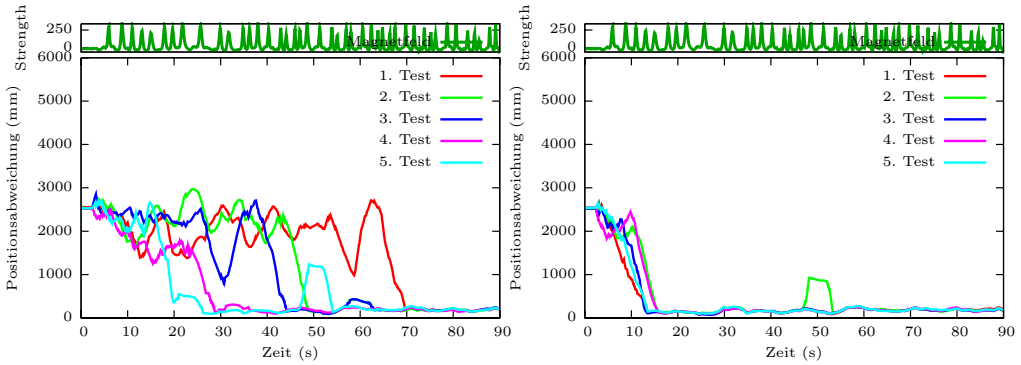
Es ist naheliegend, dass die Lokalisierungszeit abhängig von der Anzahl der verwendeten Partikel ist, da bei größerer Anzahl von Partikeln die Wahrscheinlichkeit höher ist, dass sich zufällig ein Partikel an der richtigen Position befindet.

Um diesen Zusammenhang zu verifizieren, wurde für alle Testfälle, d.h. die reale Fahrt mit dem Wagen auf dem $24m^2$-Feld und die simulierten Fahrten in einer simulierten $100m^2$ und $400m^2$ Halle, der Lokalisierungsalgorithmus mit einer Partikelanzahl zwischen 50 und 20000 Partikeln durchlaufen, wobei beim Resampling zunächst noch keine Templates (Kapitel 5.3.3) verwendet wurden. Jeder Test mit einer bestimmten Partikelanzahl wurde dabei 5 mal durchgeführt, da an verschiedenen Stellen des Lokalisierungsalgorithmus mit zufälligen Werten gearbeitet wird, was in jedem Durchlauf zu etwas anderen Ergebnissen führt.



(a) 50 Partikel: In keinem der Durchläufe wird eine richtige Position gefunden

(b) 350 Partikel: In 4 der 5 Durchläufe wird eine richtige Position gefunden



(c) 1000 Partikel: In allen Durchläufen wird eine richtige Position gefunden. Es dauert allerdings sehr lange.

(d) 3000 Partikel: In jedem Durchlauf wird innerhalb der ersten 20 Sekunden eine richtige Position gefunden.

Abbildung 6.2: Exemplarische Durchläufe des Lokalisierungsalgorithmus für verschiedene Partikelanzahlen. Dargestellt ist die Abweichung der Position in Abhängigkeit der Zeit.

Einige der Durchläufe mit den realen Messdaten sind in Abbildung 6.2 exemplarisch dargestellt. Dabei ist der Graph zweigeteilt. Die X-Achse ist in beiden Teilen gleich und gibt die Zeit an. Im oberen Teil ist die Summe der Messwerte der Magnetfeldsensoren angegeben. Dies zeigt, wann ein Magnet überfahren wurde, was bei den lokalen Maxima der Fall ist. Darunter ist die Differenz zwischen der berechneten Position und der richtigen Position für die 5 Durchläufe mit der gleichen Partikelanzahl angegeben.

Zunächst ist in Abbildung 6.2(a) mit 50 Partikeln zu sehen, dass der Lokalisierungsalgorithmus nie eine richtige Lokalisierung gefunden hat. In keinem der 5 Durchläufe wird eine Position in der Nähe der richtigen Position gefunden.

Schon nach leichter Erhöhung der Anzahl der Partikel (Abbildung 6.2(b)) wird in 4 der 5 Durchläufe die richtige Position gefunden, wobei einmal, allerdings rein zufällig, schon nach dem dritten Magneten.

Durch weitere Erhöhung der Partikelanzahl wird die Position im Durchschnitt immer früher und immer sicherer gefunden. Bei den 1000 Partikeln in Abbildung 6.2(c) wurde in jedem der Tests die richtige Position gefunden, wobei dies hier maximal 70 Sekunden dauerte.

Mit 3000 Partikeln schließlich gelingt es dem Algorithmus, in jedem der Tests innerhalb der ersten 15 Sekunden, d.h. nach Überfahren des zweiten Magneten, sich richtig zu lokalisieren. Abbildung 6.2(d)

Dieser Test wurde nicht nur für die oben verwendeten Partikelanzahlen durchgeführt, sondern für Partikelanzahlen von 50 bis 3000 im Abstand von 50 Partikeln. Das Ergebnis ist in Abbildung 6.3(a) dargestellt. Hier sieht man die minimale, die durchschnittliche und die maximale Lokalisierungszeit der 5 Durchläufe, die der Lokalisierungsalgorithmus benötigte. Der Testdatensatz der realen Testfahrt ist 103 Sekunden lang. Wenn der Lokalisierungsalgorithmus bis zu dieser Zeit keine Position gefunden hat, wird der Durchlauf mit 103 Sekunden bewertet.

Es ist zu sehen, dass es dem Lokalisierungsalgorithmus erst ab der Verwendung von 900 Partikeln zuverlässig, d.h. in jedem der 5 Durchläufe, gelingt, innerhalb des Testzeitraums eine Position in der Nähe der richtigen Position zu finden. Die schnellsten erfolgreichen Lokalisierungen traten schon nach ca. 8 Sekunden nach dem Überfahren des zweiten Magneten ein. In diesen Durchläufen war in einem der ersten Durchläufe des Algorithmus ein Partikel an der richtigen Position.

Bei einer Partikelanzahl von mehr als 2000 Partikeln liegt der Durchschnittswert der Lokalisierungszeit bei ca. 20 Sekunden oder 7 überfahrenen Magneten. Allerdings dauerte es in einem der Durchläufe mit 70 Sekunden, was 33 überfahrenen Magneten entspricht, immer noch sehr lange.

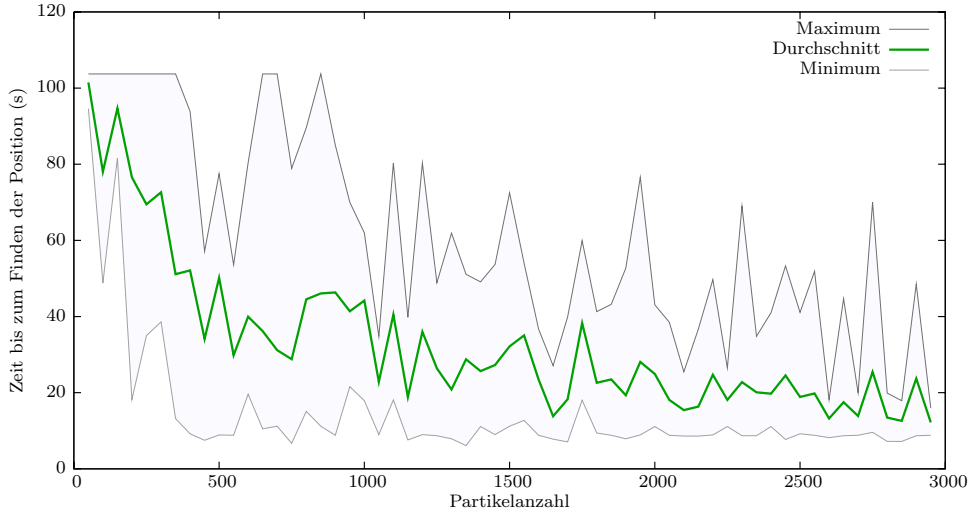
Derselbe Test wurde auch noch mit den simulierten Testdaten für verschiedene Hallengrößen von $100m^2$ (Abbildung 6.3(b)) und $400m^2$ (Abbildung 6.3(c)) durchgeführt, wobei bei $100m^2$ ca. 14000 und bei $400m^2$ mehr als 20000 Partikel notwendig sind, um die richtige Position zuverlässig innerhalb des Testzeitraums zu finden.

Diese deutliche Abhängigkeit der benötigten Partikelanzahl von der Hallengröße ist auch einleuchtend, da die Wahrscheinlichkeit, dass sich ein Partikel in der Nähe der richtigen Position befindet, bei größer werdender Halle kleiner wird.

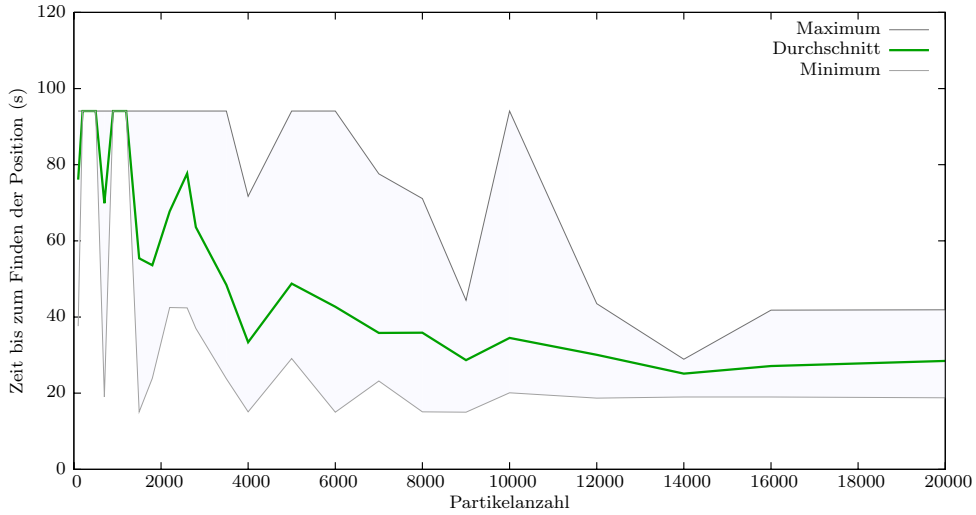
Nach der Betrachtung dieser Werte ist zu bemerken, dass eine Lokalisierung nur mit Hilfe von zufällig eingestreuten Partikeln zwar möglich ist, aber eine sehr hohe Partikelanzahl benötigt. Trotz der hohen Partikelanzahl, gerade in der größten Testhalle, ist die Lokalisierungszeit im Maximum immer noch viel zu hoch.

6.3.2 Lokalisierungszeit mit Hilfe von Templates

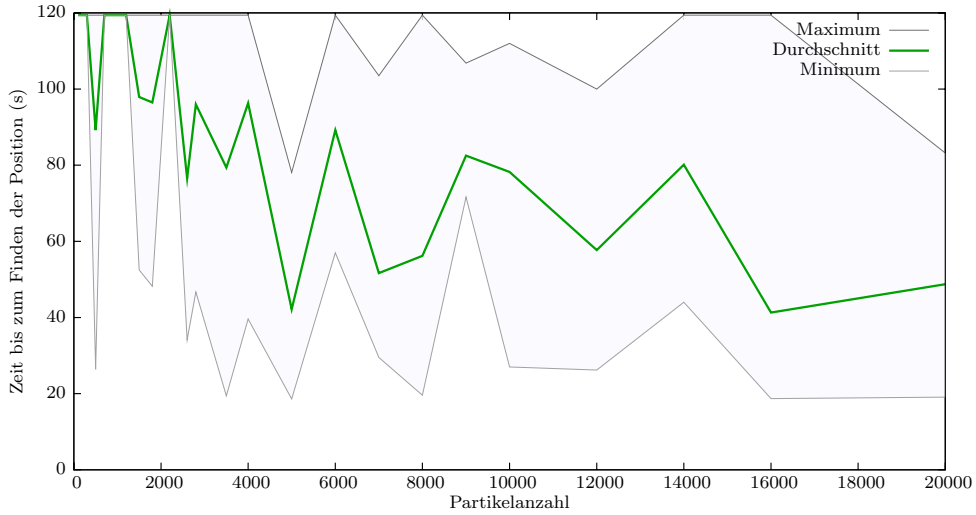
Um die Lokalisierungszeit zu verkürzen und mit weniger Partikeln schneller die richtige Position zu finden, wurden Template-Partikel (Kapitel 5.3.3) eingesetzt. Auch



(a) Reale Messdaten, 24m^2 Hallengröße



(b) Simulierte Messdaten, 100 m^2 Hallengröße



(c) Simulierte Messdaten, 400 m^2 Hallengröße

Abbildung 6.3: Lokalisierungszeit für die verschiedenen Hallengrößen in Abhängigkeit von der Partikelanzahl

hier wurden wieder die oben beschriebenen Tests auf denselben Daten durchgeführt, nur dass diesmal die Generierung von Template-Partikeln aktiviert war.

In Abbildung 6.4 sind die Ergebnisse dieser Testdurchläufe dargestellt. Es ist zu sehen, dass, sobald eine gewisse Mindestanzahl von Partikeln überschritten ist, die richtige Position sehr zuverlässig gefunden wird. Bei den realen Testdaten geschieht dies schon ab 100 Partikeln im Durchschnitt beim vierten überfahrenen Magneten. Bemerkenswert ist, dass auch die maximale Lokalisierungzeit ab 100 Partikeln immer unter 20 Sekunden liegt, womit eine schnelle Lokalisierung auch sehr wahrscheinlich ist. Diese Zeit entspricht ungefähr dem 6. überfahrenen Magneten.

Bei den größeren simulierten Testläufen zeigt sich ebenfalls, dass ab einer gewissen Partikelanzahl die Lokalisierungszeit ebenfalls nahezu konstant ist. Beim Unterschreiten dieser Partikelanzahl ist es dem Lokalisierungsalgorithmus nicht mehr möglich, alle von den Templates vorgegebenen Positionen zu verfolgen und eventuell beim Überfahren des nächsten Magneten bestätigen oder verwerfen zu können.

Im Vergleich zu den zufällig eingestreuten Partikeln ergibt sich bei der Verwendung von Templates eine wesentlich zuverlässigere Erstlokalisierung mit einer niedrigen maximalen Lokalisierungszeit bei der Verwendung von deutlich weniger Partikeln.

6.3.3 Abhängigkeit von der Anzahl unabhängiger Partikel

In jedem Durchlauf des Lokalisierungsalgorithmus müssen neue, vom vorherigen Belief unabhängige Partikel hinzugefügt werden, damit der Lokalisierungsalgorithmus eine eventuell verloren gegangene Position wieder finden kann. Diese sind entweder Templates oder zufällig ausgewählte Partikel.

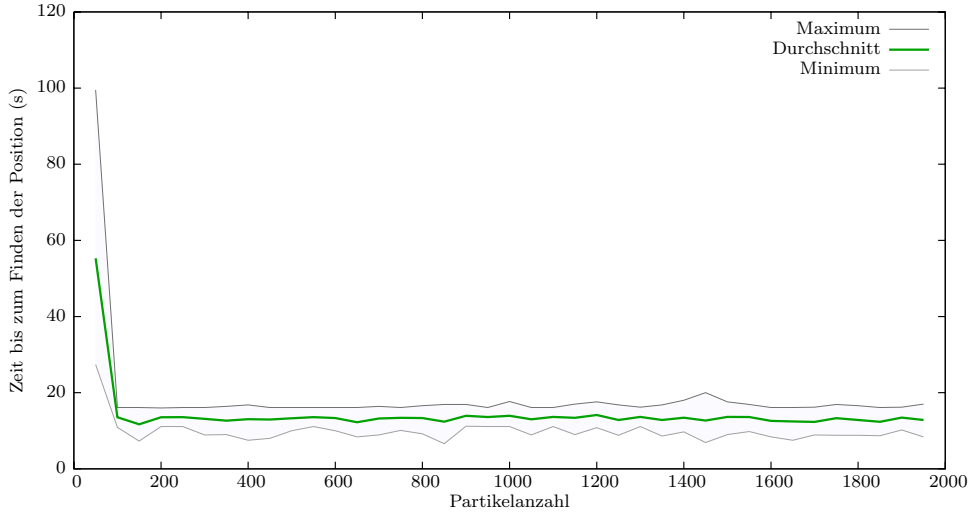
Das Verhältnis dieser neuen Partikel zur Gesamtzahl der Partikel hat dabei auch einen Einfluss auf die Lokalisierungsgeschwindigkeit.

In Abbildung 6.5 ist die Lokalisierungszeit in Abhängigkeit der Anzahl der neuen Partikel dargestellt. Es wurden dabei die realen Testdaten verwendet. Es ist zu sehen, dass die Anzahl der neuen Partikel nahezu keine Auswirkung auf die Lokalisierungszeit hat. Erst bei mehr als 10% neuer Partikel wird diese minimal größer.

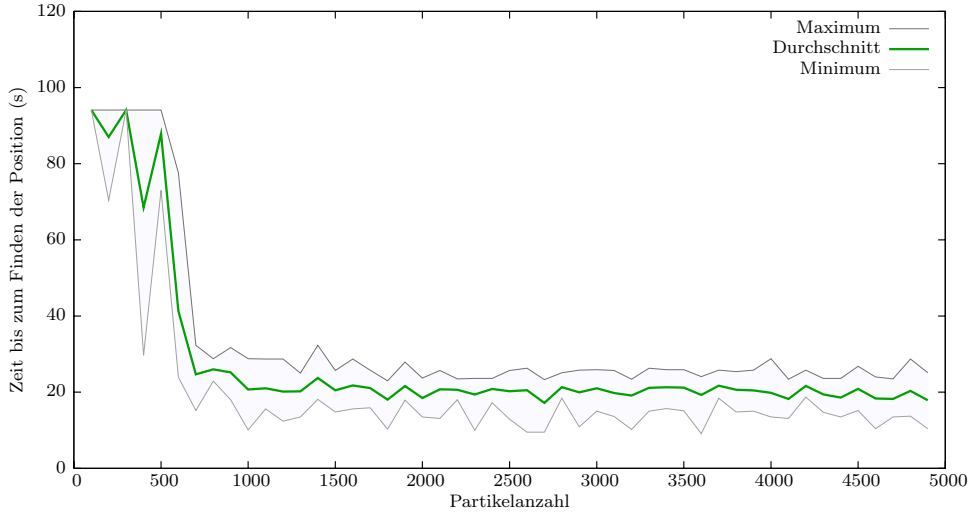
Dies ist insofern erstaunlich, als zu viele zufällige Partikel eigentlich einen negativen Einfluss auf das Finden der Position haben. Die Template-Partikel sorgen hier aber dafür, dass die Position trotzdem schnell gefunden wird.

6.3.4 Abhängigkeit vom Sensor-Model

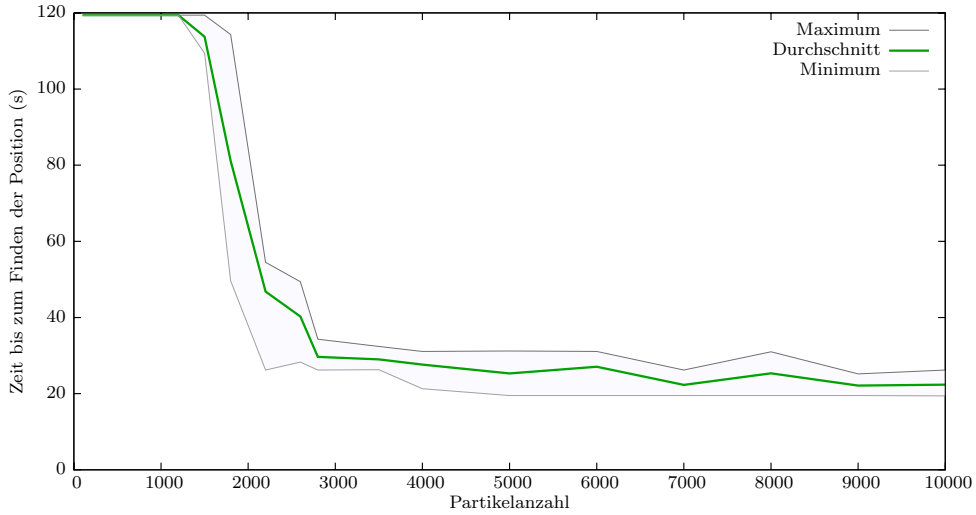
Das bei der Lokalisierung verwendete und in Kapitel 5.2.2 beschriebene Sensor-Model beinhaltet als Parameter die Standardabweichung σ für die Normalverteilung, die zur Berechnung der Partikelgewichtung verwendet wird. Über diese Standard-



(a) Reale Messdaten, 24m^2 Hallengröße



(b) Simulierte Messdaten, 100 m^2 Hallengröße



(c) Simulierte Messdaten, 400 m^2 Hallengröße

Abbildung 6.4: Lokalisierungszeit für verschiedene Hallengrößen in Abhängigkeit von der Partikelanzahl mit der Verwendung von Template-Partikeln

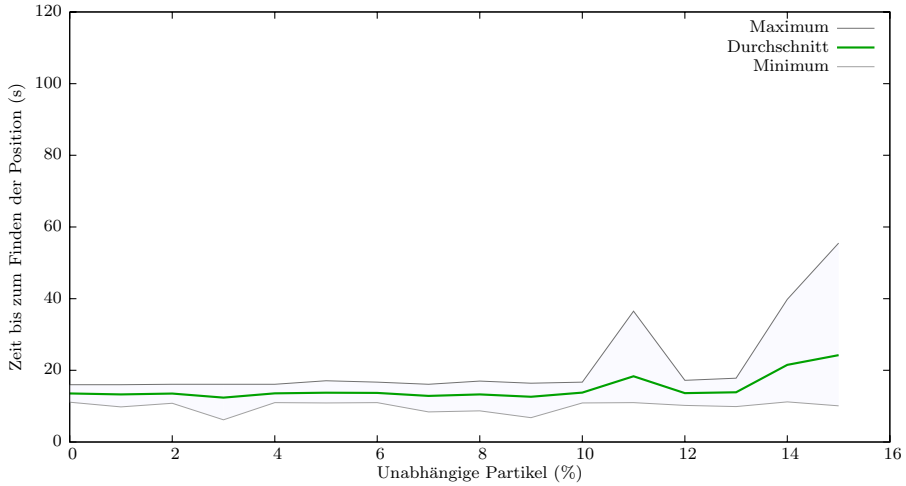


Abbildung 6.5: Lokalisierungszeit in Abhängigkeit der Anzahl unabhängiger Partikel mit den realen Testdaten

abweichung lässt sich steuern, wie stark die erwarteten Sensormesswerte von den realen Sensormesswerten abweichen dürfen.

Um einen guten Wert für σ zu erhalten, wurde die Lokalisierungszeit in Abhängigkeit von σ ermittelt. Dabei wurde eine Partikelanzahl von 3000 und 1% unabhängige Partikel verwendet.

In Abbildung 6.6 ist das Ergebnis dieser Messungen dargestellt. Bei sehr niedrigem $\sigma < 2000$ und sehr hohem $\sigma > 20000$ zeigt sich, dass der Lokalisierungsalgorithmus die Position nicht zuverlässig findet. Dazwischen sind die Lokalisierungszeiten aber auf dem normalen niedrigen Niveau aus den vorherigen Tests.

Die schlechte Lokalisierungszeit bei hohem σ ergibt sich dadurch, dass die Partikel für den Lokalisierungsalgorithmus kaum noch unterscheidbar sind, da sie alle eine ähnliche Bewertung erhalten. Bei niedrigem σ ergibt sich ein anderes Problem, das zu schlechten Lokalisierungszeiten führt. Ist σ zu klein, ist auch der Bereich zu klein, in dem ein Template-Partikel eine hohe Bewertung bekommen würde. Da die Erzeugung der Template-Partikel abhängig von den Ungenauigkeiten der Odometrie ist, kommt es bei einem immer kleiner werdenden Bereich immer häufiger vor, dass dieser vom Template nicht getroffen wird. Das Template bekommt dann eine niedrige Gewichtung und kann daher nichts zum Auffinden der richtigen Position beitragen.

Zusammengefasst ist eine schnelle, zuverlässige Lokalisierung bei der Verwendung von Templates mit mindestens 10 Partikeln/qm, einem σ zwischen ca. 3000 und 20000 und maximal 10% neuer Partikel möglich.

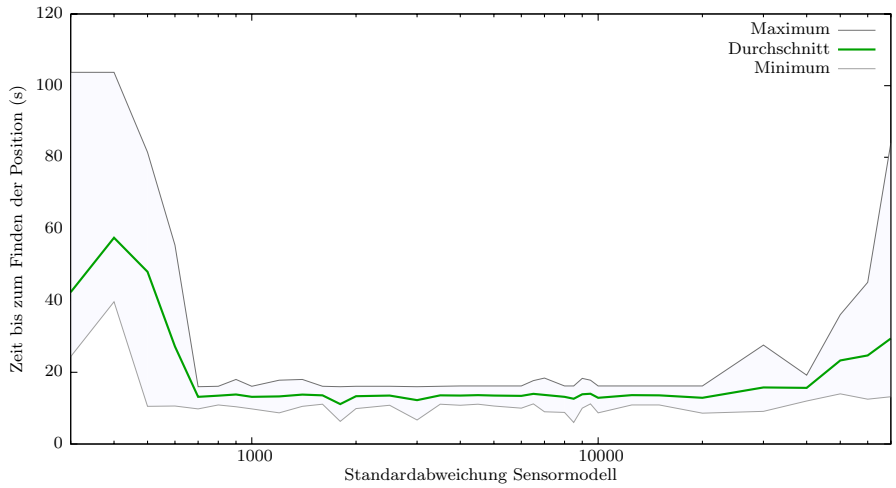


Abbildung 6.6: Lokalisierungszeit in Abhängigkeit der Standardabweichung σ des Sensor-Models auf dem 24m^2 großen Testfeld.

6.4 Verlieren der Lokalisierung

Ist der Gabelstapler einmal richtig lokalisiert, so ist als nächstes von Interesse, ob der Lokalisierungsalgorithmus diese Position richtig verfolgt und der Gabelstapler damit richtig lokalisiert bleibt, oder ob er sie wieder verliert. Daher wurden, wieder mit den aufgezeichneten Testdaten, einige Tests durchgeführt, um zu erkennen, wie häufig die Position verloren wird.

Um dies zu messen, muss zunächst definiert werden, wann der Gabelstapler nicht mehr richtig lokalisiert ist. Hier wird angenommen, dass dies der Fall ist, wenn die berechnete Position weiter als 50cm von der richtigen Position entfernt liegt.

Ist der Gabelstapler nicht mehr richtig lokalisiert, so ist es interessant zu wissen, wie lange dieser Zustand andauert. Dazu wird das Verhältnis t_{lok}/t_{ges} zwischen der Zeit t_{lok}, die der Stapler richtig lokalisiert ist, und der Gesamtzeit des Tests t_{ges} nach der ersten richtigen Lokalisierung betrachtet. Im Idealfall sollte $t_{lok}/t_{ges} = 1$ sein. Bei dem Test mit den realen Testdaten kommen Werte über 99,7% vor, was bedeutet, dass der Lokalisierungsalgorithmus bei der Testlänge von ca 100 Sekunden nur 0,3 Sekunden delokalisiert war.

6.4.1 Abhängigkeit von der Partikelanzahl

In Abbildung 6.7 ist t_{lok}/t_{ges} in Abhängigkeit von der Partikelanzahl für die realen Daten und die Simulatordurchläufe dargestellt.

Es ist zu sehen, dass ab einer bestimmten, von der Größe der Halle abhängigen Partikelanzahl das Verhältnis sehr nahe an 100% liegt. Die zum Überschreiten von 99,7% benötigte Partikelanzahl liegt bei den realen Testdaten auf dem 24m^2 Feld

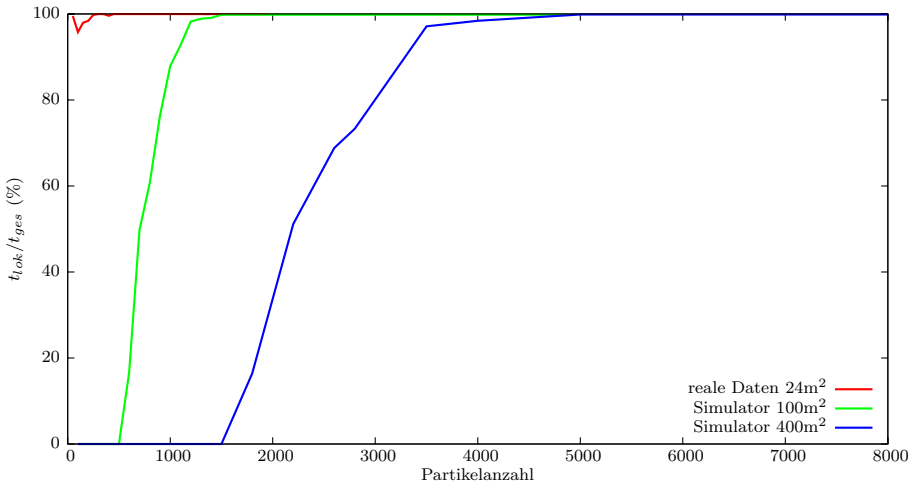


Abbildung 6.7: t_{lok}/t_{ges} in Abhängigkeit von der Partikelanzahl für verschiedene Hallengrößen

bei etwa 300, bei der simulierten 100m^2 Halle bei etwa 1500 und bei der 400m^2 Halle bei etwa 6000 Partikeln.

6.4.2 Anhängigkeit von der Anzahl unabhängiger Partikel

In Abbildung 6.8 ist das Verhältnis t_{lok}/t_{ges} in Abhängigkeit der Anzahl der unabhängigen Partikel dargestellt. Es ist gut zu sehen, dass zu viele neue Partikel der Lokalisierung sehr schaden. Nur bis 1% unabhängige Partikel kann der Lokalisierungsalgorithmus die Position gut bestimmen. Bei 5% ist der Algorithmus schon in 10% der Zeit nicht richtig lokalisiert. Noch mehr unabhängige Partikel führen zu noch schlechteren Ergebnissen.

6.4.3 Abhängigkeit vom Sensor-Model

Auch das Sensor-Model hat einen Einfluss darauf, wie häufig eine richtige Position verloren wird. In Abbildung 6.9 ist das Verhältnis t_{lok}/t_{ges} in Abhängigkeit der Standardabweichung σ des Sensor-Models dargestellt. Ähnlich wie bei der Lokalisierungszeit sieht man hier, dass σ in einem gewissen Bereich liegen muss, damit der Lokalisierungsalgorithmus sinnvoll funktioniert. Dieser Bereich beginnt hier bei ca. 3000 und endet bei ca. 20000.

Zusammengefasst sind, damit der Lokalisierungsalgorithmus nach der Erstlokalisation richtig lokalisiert bleibt, mindestens 15 Partikel/m^2, maximal 1% unabhängige Partikel und ein σ zwischen 3000 und 20000 erforderlich.

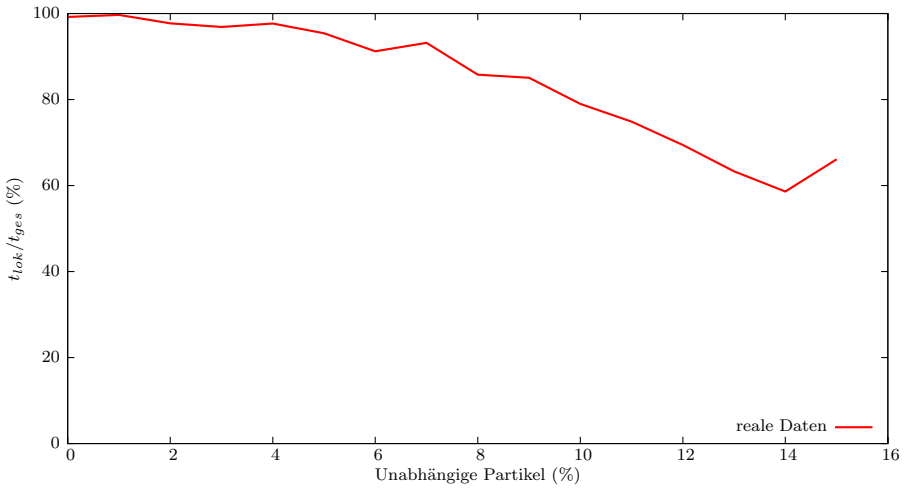


Abbildung 6.8: t_{lok}/t_{ges} in Abhängigkeit der Anzahl unabhängiger Partikel

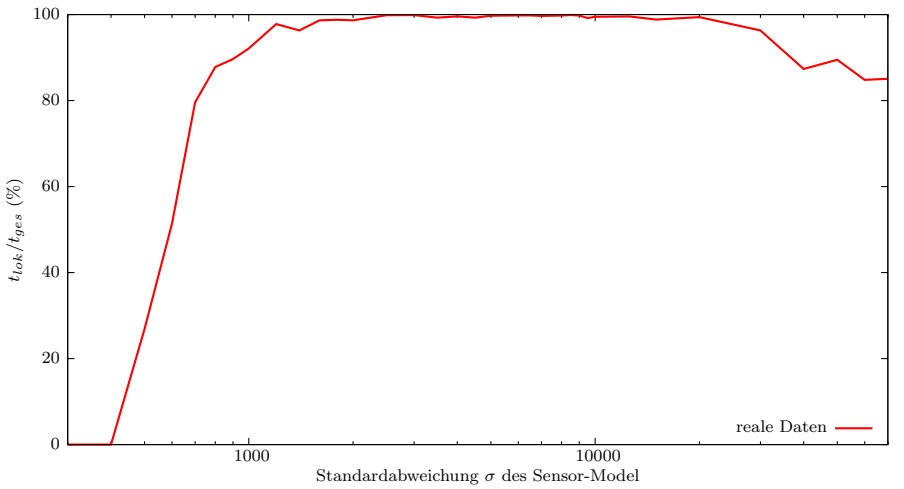


Abbildung 6.9: t_{lok}/t_{ges} in Abhängigkeit der Standardabweichung σ des Sensor-Models

6.5 Positionsabweichung

Nachdem nun die Lokalisierungszeit und das Verlieren der Lokalisation betrachtet wurde, wird die Abweichung der berechneten Position zu der Referenzposition in Abhängigkeit der verschiedenen Parameter untersucht. Die Position bei der Gabelstaplerlokalisierung besteht aus dem Punkt, an dem der Gabelstapler steht und seiner Orientierung, die leider unterschiedliche Einheiten haben und daher nicht direkt vergleichbar sind. Im Folgenden wird daher zwischen Positionsabweichung, die sich auf die X- und Y-Komponente bezieht und Orientierungsabweichung unterschieden.

Zur Untersuchung der Positions- und Orientierungsabweichung wird diese jeweils in jedem Frame nach der Erstlokalisierung berechnet. Für jeden Test mit seinen 5 Durchläufen sind dies ca. 4000 betrachtete Frames. Aus diesen wird dann der Mittelwert, das Minimum und das Maximum der Abweichungen bestimmt.

6.5.1 Abhängigkeit von der Partikelanzahl

In Abbildung 6.10 ist die durchschnittliche, die minimale und die maximale Positionsabweichung in Abhängigkeit von der Partikelanzahl dargestellt, wobei sich auch hier zeigt, dass eine gewisse Mindestpartikelanzahl von ca. 400 Partikeln vorhanden sein muss, damit der Lokalisierungsalgorithmus sinnvoll funktioniert und eine weitere Erhöhung der Partikelanzahl zu keiner weiteren Verbesserung mehr führt. Für die Orientierungsabweichung gilt das gleiche.

Bei mehr als 400 Partikeln beträgt die Positionsabweichung durchschnittlich ca. 120mm und die Orientierungsabweichung ca. 5°.

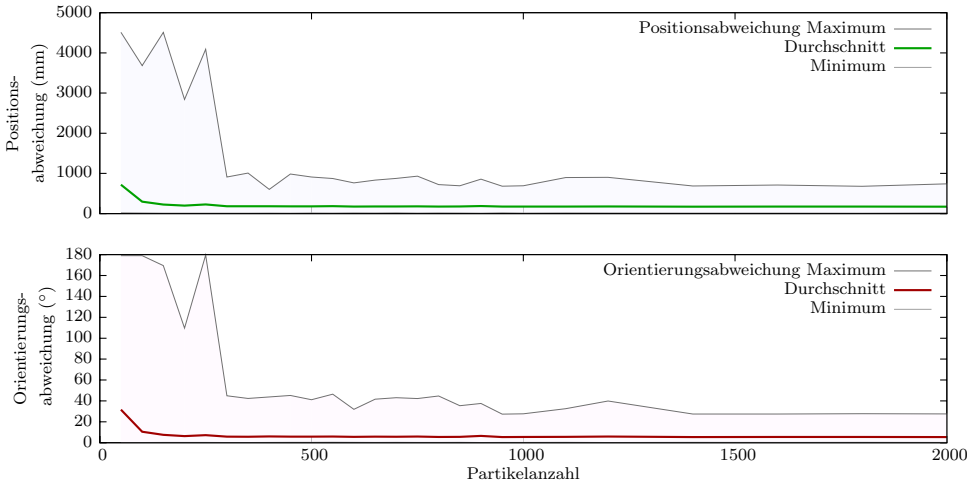


Abbildung 6.10: Positionsabweichung in Abhängigkeit von der Partikelanzahl

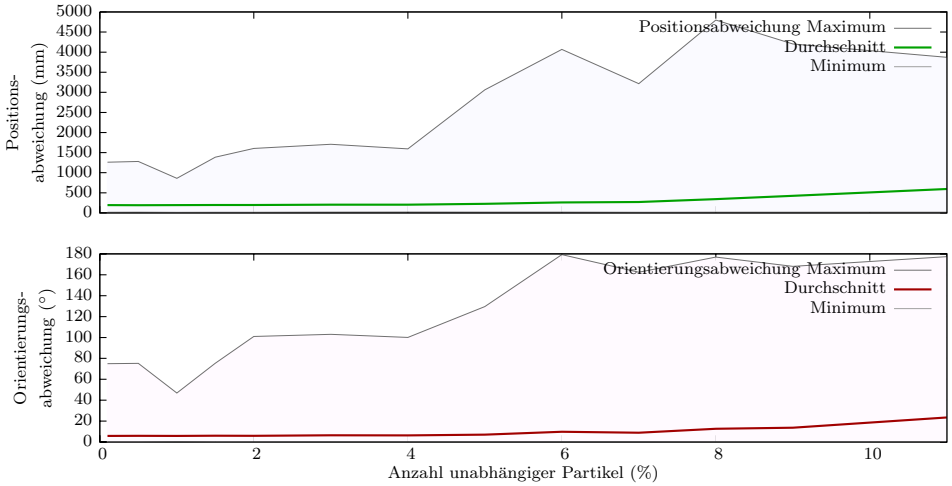


Abbildung 6.11: Positionsabweichung in Abhängigkeit der Anzahl neuer Partikel

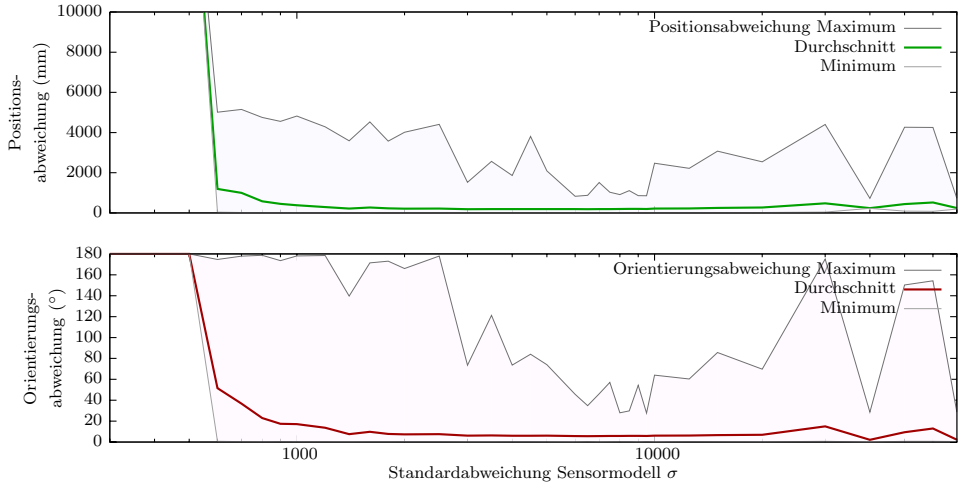


Abbildung 6.12: Positionsabweichung in Abhängigkeit des Sensor-Models.

6.5.2 Abhängigkeit von der Anzahl unabhängiger Partikel

Auch die Auswirkung der Anzahl der neuen Partikel auf die Positionsabweichung wurde untersucht. In Abbildung 6.11 ist zu sehen, dass sowohl die maximale, als auch die durchschnittliche Positions- und Orientierungsabweichung ab 1% neuer Partikel immer weiter ansteigt. Also ist auch, wie schon bei der Lokalisierungszeit und beim Verlieren der Position, dieser Wert der sinnvollste Wert für die Anzahl neuer Partikel.

6.5.3 Abhängigkeit vom Sensor-Model

In Abbildung 6.12 sind Positionsabweichung und Orientierungsabweichung in Abhängigkeit von der Standardabweichung des Sensor-Models σ abhängig dargestellt.

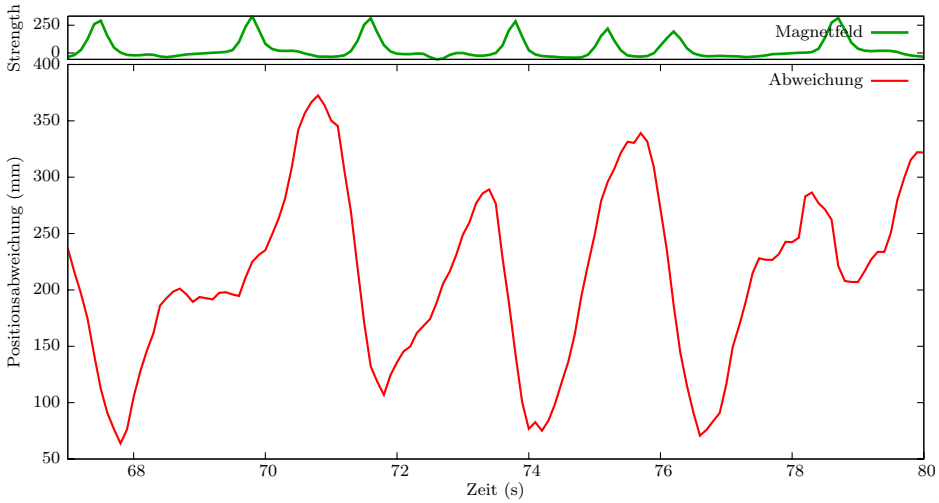


Abbildung 6.13: Positionsabweichung in einem beispielhaften Durchlauf mit 400 Partikeln. Es ist gut zu sehen, wie die durch die Odometrie hervorgerufene Positionsabweichung durch das Überfahren der Magnete korrigiert wird.

War zum Finden und Verfolgen noch ein recht großer Bereich von ca. 3000 bis 50000 zufriedenstellend, so sieht man hier, dass die geringste maximale und auch durchschnittliche Abweichung im Bereich zwischen 7000 und 9000 erreicht wird.

6.5.4 Abhängigkeit von der Odometriequalität

Bei der Betrachtung der vorherigen Tests fällt auf, dass die Positionsabweichung nur insoweit von den jeweiligen Parametern abhängig ist, als diese in einem gewissen Bereich liegen müssen, damit der Lokalisierungsalgorithmus überhaupt funktioniert. Hier haben also auch noch andere Parameter einen Einfluss.

Um nun zu sehen, welche Parameter dies sein könnten, ist in Abbildung 6.13 noch einmal ein Ausschnitt eines Durchlaufes des Lokalisierungsalgorithmus dargestellt. Es handelt sich hierbei um einen Durchlauf mit den realen Messdaten auf dem 24m^2 Feld bei einer Partikelanzahl von 400.

Es ist zu sehen, dass die Positionabweichung, wenn gerade ein Magnet überfahren wurde, mit ca. 70mm sehr klein ist. Fährt das Testfahrzeug nun weiter, steigt die Positionsabweichung wegen der Ungenauigkeit der Odometrie immer weiter, bis wieder ein Magnet überfahren wurde, um die Position zu korrigieren. Die durchschnittliche Positionsabweichung ergibt sich daher aus der Ungenauigkeit der Odometrie und dem Abstand der Magnete.

Um die Auswirkung einer ungenauen Odometrie zu erfassen, wurde ein weiterer Test durchgeführt, bei dem der von der Odometrie angenommene Radradius variiert wurde. In Abbildung 6.14 sind die Ergebnisse dargestellt. Es ist zu sehen, dass bei deutlich zu geringem oder deutlich zu hohem Radius der Lokalisierungsalgorith-

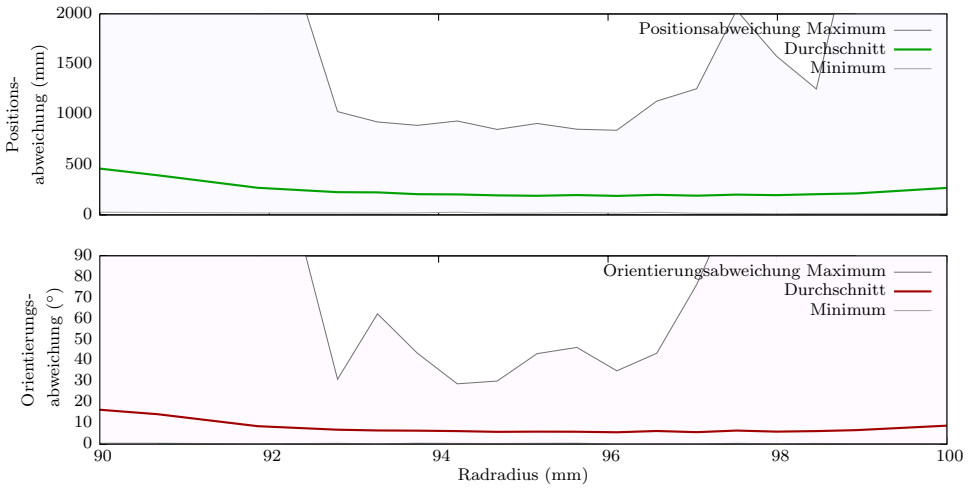


Abbildung 6.14: Positionsabweichung bei verschiedenen angenommenen Radradien.

mus nicht mehr vernünftig arbeitet. Dies ist dann der Fall, wenn die Partikel beim Control-Update viel zu wenig oder zu viel verschoben werden, so dass sie beim Überfahren des nächsten Magneten zu weit von der richtigen Position entfernt liegen, eine niedrige Gewichtung erhalten und beim Resampling herausfallen.

Bei einem Radius von 96mm ist die durchschnittliche Positions- und Orientierungsabweichung am geringsten. Das Rad scheint durch das Gewicht des Wagens oder falschen Luftdruck so eingedrückt zu sein, dass der Radius von seinem Nominalradius von 100mm um 4mm abweicht.

6.5.5 Abhängigkeit vom Magnetabstand

Um die Abhängigkeit der Positions- und Orientierungsabweichung vom Magnetabstand zeigen zu können, wurden mit dem Simulator Tests mit verschiedenen Magnetabständen durchgeführt. Dies ist zunächst schwierig, da die Magnetabstände nicht gleich sein dürfen, damit der Lokalisierungsalgorithmus funktioniert. Bedenkt man allerdings, dass die Magnetpositionen im Simulator zunächst mit einem regelmäßigen Gitter mit der Gitterweite g generiert und dann zufällig maximal $1/4\ g$ verschoben werden (Kapitel 6.1), so lassen sich die Abweichungen in Abhängigkeit dieser Gitterweite untersuchen.

Das Ergebnis dieses Tests ist in Abbildung 6.15 dargestellt. Es ist zu sehen, dass alle Abweichungen bei größerer Gitterweite ebenfalls größer werden. Dies ist auch klar, da im Motion-Update immer ein Fehler hinzukommt, der allerdings erst nach dem Überfahren des nächsten Magneten wieder korrigiert werden kann. Ist der Abstand zwischen den Magneten höher, so kann der Fehler häufiger hinzukommen und somit entstehen höhere Abweichungen.

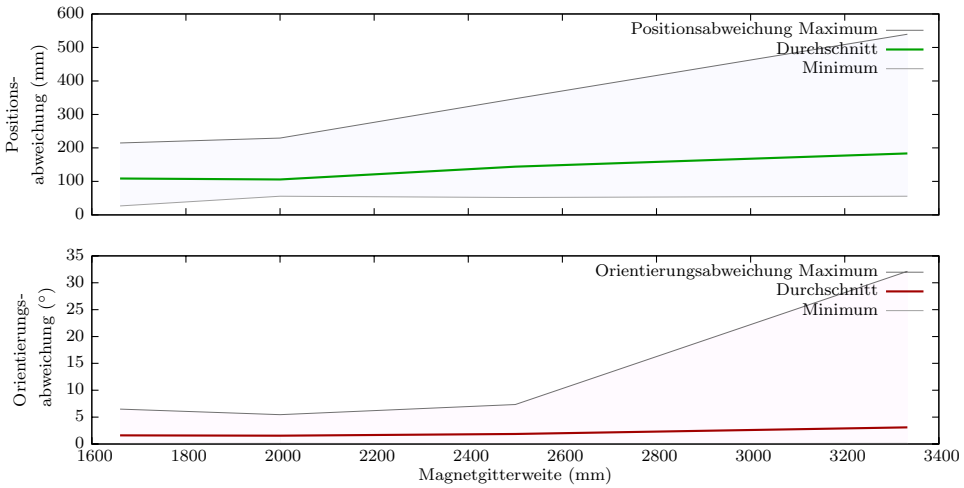


Abbildung 6.15: Positions- sowie Orientierungsabweichung bei unterschiedlichen Magnetabständen.

Zusammenfassend führen die schon zur Erstlokalisierung und zum Halten der Position benötigten Parameter von ca. 15 Partikeln/m^2 und 1% neuer Partikel zusammen mit einem $\sigma = 8000$ zu der besten Lokalisation. Die noch bestehenden Abweichungen lassen sich darüber hinaus nur noch über Verbesserung der Odometrie und über Verringerung des Magnetabstandes erreichen.

Kapitel 7

Zusammenfassung

In dieser Diplomarbeit wurde ein Algorithmus zur Lokalisierung eines Gabelstaplers geplant, implementiert und evaluiert. Dieser Algorithmus findet die Position des Gabelstaplers mit Hilfe von Magneten im Boden, die mittels eines speziellen Sensors, dem „Magnetlineal“, im Darüberfahren erkannt werden können und verfolgt die Position mit Hilfe von Odometriesensoren auch zwischen den Magneten. Diese Gabelstaplerlokalisierung soll später dazu dienen, die Position der mit dem Gabelstapler transportierten Ware in einem Lager nachhalten zu können.

Dieser Ansatz bietet gegenüber anderen Lokalisierungssystemen einen deutlichen Kostenvorteil. Durch die Verwendung von Magneten als Markierung sind keine teuren stationären Sender oder teure Sensoren, wie Laserscanner, auf dem Gabelstapler notwendig. Darüber hinaus bietet die Verwendung von Magneten den Vorteil, dass sie überall dort, wo der Gabelstapler fahren kann, auch immer erkannt werden können.

Zunächst wurden hierzu geeignete Odometriesensoren, d.h. Radsensoren an allen vier Rädern, ein Lenkwinkelsensor, ein Drehratensensor und ein Beschleunigungssensor ausgewählt und ein Testfahrzeug geplant, um diese Sensoren verwenden zu können. Nachdem dieses zur Verfügung stand, wurde es mit den Sensoren und einem Microcontroller-Board bestückt, das es ermöglicht, die Sensormesswerte auszulesen. Außerdem wurde das Testfahrzeug mit verschiedenen Referenzsystemen ausgestattet, um später die Qualität des Lokalisierugnsalgorithmus messen zu können.

Daraufhin wurde ein Software-Framwork entwickelt, welches so aufgebaut ist, dass das Microcontroller-Board auf dem Testfahrzeug nur die Messwerte der Sensoren ausliest und über Wireless-Lan auf den Entwicklungsrechner überträgt. Auf dem Entwicklungsrechner läuft das Kontroll- und Debug-Framework „Cargo-Control“, in das der eigentliche Lokalisierungsalgorithmus eingebettet ist. Cargo-Control bietet dabei die Möglichkeit, die Eingangsdaten, Zwischenergebnisse und Resultate des Lokalisierungsalgorithmus zu visualisieren und enthält einen Simulator, mit dem es möglich ist, das Testfahrzeug und die von ihm gesendeten Messwerte zu simulieren. Das Software-Framework bietet somit alles Nötige, um den Lokalisierungsalgorithmus implementieren und vor allem später testen zu können.

Als Lokalisierungsalgorithmus wurde ein Partikel-Filter ausgewählt. Mit diesem probabilistischen Filter ist es im Vergleich zu den untersuchten Alternativen, dem Kalman-Filter und dem Histogram-Filter, möglich, eine globale Lokalisierung mit vertretbarem Rechenaufwand durchzuführen.

Zur Implementierung des Partikelfilters wurden die benötigten probabilistischen Modelle für die Schritte des Partikelfilters aufgestellt. Für die Zustandsübergangswahrscheinlichkeit wurde hierzu ein Algortithmus zur Odometrieberechnung entwickelt und die Genauigkeit experiementell ermittelt. Dabei zeigte sich, dass der Beschleunigungssensor wegen seines starken Rauschens unbrauchbar war. Die Messwahrscheinlichkeit wurde mit der Differenz zwischen erwarteten und gemessenen Messwerten des Magnetlineals bestimmt. Zur Bestimmung des erwarteten Messwertes wurde eine Formel aufgestellt, deren Parameter mittels einer Messreihe approximiert wurden. Als Resampling-Verfahren wurde Low-Variance-Sampling wegen der geringeren Laufzeit und Sampling-Varinance verwendet.

Danach wurde die Qualität des Lokalisierungsalgorithmus abhängig von seinen wichtigsten Parametern untersucht. Es stellte sich heraus, dass der Lokalisierungsalgorithmus unbedingt Template-Partikel benötigt, um den Gabelstapler schnell, richtig und global lokalisieren zu können. Mit Template Partikeln ist es möglich, unter geeigneter Wahl der restlichen Parameter, mit ca. 15 Partikeln pro m^2 Hallenfläche eine globale Lokalisierung nach dem Überfahren von 3 bis 6 Magneten zu erreichen, die auch in mehr als 99,7% der Fälle nicht mehr verloren geht.

Die Positionsabweichung ist dabei hauptsächlich von der Genauigkeit der Odometrie und dem Abstand der Magnete abhängig und betrug im Durchschnitt ca. 120mm und maximal 800mm nach einer erfolgreichen Erstlokalisierung und Magnetabständen zwischen 1 und 3,5 Metern.

7.1 Ausblick

Obwohl der Lokalisierungsalgorithmus gut zur Gabelstaplerlokalisierung brauchbar ist, bieten sich Erweiterungs- und Ausbaumöglichkeiten.

Zunächst ist bei der Odometrie noch das Problem vorhanden, dass sich der Reifenradius durch Abrieb oder Druckverlust ändern kann. Dies führt dazu, dass alle Partikel um einen gewissen Mittelwert falsch verschoben wurden. Könnte dieser Mittelwert berechnet werden, so könnte der angenommene Radius entsprechend korrigiert werden, damit die Partikel näher an der richtigen Position liegen.

Dazu könnte beim Motion-Model das zufällige Rauschen nicht erst nach der Berechnung der zurückgelegten Strecke, sondern direkt auf den Radradius addiert und die Odometrie für jeden Partikel getrennt berechnet werden. Wird dieser addierte Wert nun für jeden Partikel gespeichert, so könnte eine Statistik darüber geführt werden, welcher Wert bei den Partikeln addiert wurde, die eine hohe Gewichtung erhalten haben und der Radius entsprechend angepasst werden.

Eine weitere Möglichkeit der Verbesserung bieten mehrdimensionale Magnetfeldsensoren. Im Augenblick wird nur die Tatsache zur Lokalisierung verwendet, dass ein Magnet überfahren wurde. Mittels eines 3D-Sensors lässt sich auch dessen Lage im Raum bestimmen, wodurch sich viel mehr Positionen bei der Lokalisierung ausschließen lassen würden.

Im Augenblick sind die Positionen der Magnete als Vektoren gespeichert. Würde eine hinreichend genaue Messapparatur gebaut, um das Magnetfeld in einer Halle zu vermessen, könnten die magnetischen Unregelmäßigkeiten im Boden ebenfalls zur Lokalisierung verwendet werden, und würden nicht mehr stören. Allerdings müsste dann eine andere Methode zum Generieren der Template-Partikel gefunden werden.

Literaturverzeichnis

[1] M. Hoppe. *Evaluation of Existing and Emerging Positioning Technologies*, 2004.

[2] J. Barnes, C. Rizos, J. Wang, D. Small, G. Voigt, N. Gambale. *Locata: A New Positioning Technology for High Precision Indoor and Outdoor Positioning*, 2003.

[3] P. Ibach, V. Stantchev, F. Lederer, A. Weiss. *WLan-Based Asset Tracking For Warehouse Management.* IADIS International Conference e-Commerce, Porto, Portugal, 2005.

[4] N. B. Priyantha, A. Chakraborty, H. Balakrishnan. *The Cricket Location-Support System.* Sixth Annual ACM International Conference on Mobile Computing and Networking (MOBICOM), 2000.

[5] A. Rossbacher. *Evaluierung von Bayes-Filtern für Lokalisierung und Tracking mit dem Cricket-System.* Diplomarbeit, Universität Dortmund, 2006.

[6] Sick AG. *Nav 200 Positionierungsystem zur Navigationsunterstützung - Technische Beschreibung*, 2004.

[7] H. Richert. *Entwicklung eines magnetischen 3-D-Monitoringsystems am Beispiel der nichtinvasiven Untersuchung des menschlichen Gastro-Intestinal-Traktes.* Dissertation, Friedrich Schiller Universität Jena, 2003.

[8] Robert Bosch GmbH. *Technical Customer Documentation Basic - TCD for DF11*, 2003.

[9] Robert Bosch GmbH. *Technical Customer Documentation - TCD - Steering Wheel Angle Sensor LWS4.0*, 2002.

[10] Robert Bosch GmbH. *Technische Kundenunterlage (TKU) Sensorcluster SC-MM3.22*, 2006.

[11] D. Deom, J. Hamerla, M. Hülsbusch, J. Kerdels, T. Kindler, H.-W. Koh, T. Lohmann, M. Neubach, C. Rink, A. Rossbacher, F. Rossdeutscher, B. Schmidt, C. Schumann, P. Serve. *Virtual Robot: Automatic Analysis of Situations and Management of Resources in a Team of Soccer Robots.* Technischer Bericht, Universität Dortmund, 2004.

[12] M. Broszeit, S. Czarnetzki, T. Diekmann, D. Fisseler, T. Kerkhof, M. Meyer, B. Schmitz, C. Rohde, X.-A. Tran, T. Wegner, J. Winter, C. Zarges. *Vierbeinige fußballspielende Roboter - Entwicklung eines dynamischen Weltmodells basierend auf Zuverlässigkeitsinformation und Schätzung.* Technischer Bericht, Uni Dortmund, 2005.

[13] Egnite Software GmbH. *Ethernut Version 1.3 Hardware Users Manual*, 1.8 Auflage, 2005.

[14] Microchip. *MCP2515 - Stand-Alone CAN Controller With SPI Interface*, 2003.

[15] E. Brookner. *Tracking and Kalman Filtering Made Easy.* John Wiley & Sons Inc, 1998. ISBN 0471184071.

[16] T. Röfer, T. Laue, D. Thomas. *Particle-filter-based self-localization using landmarks and directed lines.* In *RoboCup 2005: Robot Soccer World Cup IX* (A. Bredenfeld, A. Jacoff, I. Noda, Y. Takahashi, Herausgeber), Nummer 4020 in Lecture Notes in Artificial Intelligence. Springer, 2006 Seiten 608–615.

[17] T. Röfer, M. Jüngel. *Fast and robust edge-based localization in the Sony Four-Legged Robot League.* In *RoboCup 2003: Robot World Cup VII*, Nummer 3020 in Lecture Notes in Artificial Intelligence. Springer, 2004 Seiten 262–273.

[18] S. Thrun, W. Burgard, D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents).* The MIT Press, 2005. ISBN 0262201623.

[19] R. E. Kalman. *A New Approach to Linear Filtering and Prediction Problems.* Transactions of the ASME Journal of Basic Engineering, 1960.

[20] D. Fox, W. Burgard, S. Thrun. *Active markov localization for mobile robots.* Robotics and Autonomous Systems, 1998.

[21] C. Kwok, D. Fox, M. Meil. *Real-time particle filters.* Advances in Neural Information Processing Systems, 2002. 15.

[22] S. Lenser, M. Veloso. *Sensor Resetting Localization for Poorly Modelled Mobile Robots.* In *Proceedings of ICRA-2000, the International Conference on Robotics and Automation.* 2000 .

[23] T. Röfer, M. Jüngel. *Vision-Based Fast and Reactive Monte-Carlo Localization.* In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-2003).* 2003 Seiten 856 – 861.

[24] P. Ramachandrarao, G. V. S. Sastry, L. Pandey, A. Sinha. *A Novel Algorithm for a Quasiperiodic Plane Lattice with Fivefold Symmetry.* Foundations of Crystallography, 1991. 47.