technische universität
dortmund

PG 556

Endbericht

Teilnehmende

Kamen Avramov
Wolf Bublitz
Tony Demann
Katrin Erlinghagen
Alexander Görtz
Daniel Hegels
Jana Jost
Knut Krause
Sven Langkamp
Daniel Noack
Tobias Siebrecht
Sebastian Skibinski
Daniel Wulfert

Betreuer

Dr. Frank Weichert
Dipl.-Inf. Constantin Timm
Dipl.-Ing. Andreas Kamagaew
Dipl.-Logist. Christian Prasse
M. Sc. Jonas Stenzel

Lehrstuhl Informatik VII **Lehrstuhl Informatik XII** **Fraunhofer IML**

fakultät für
informatik

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Mathematische Notation

Einheiten

Einheiten werden gemäß dem internationalen Einheitensystem (SI) angegeben. Das heißt dass u. a. die Länge l in Metern ($[l]_{SI} = \mathrm{m}$) und die Zeit t in Sekunden ($[t]_{SI} = \mathrm{s}$) bemessen wird. Ferner können ebenfalls die SI-Präfixe für die Einheiten verwendet werden.

Vektoren

Vektoren werden mit Hilfe von **fett** gedruckten Kleinbuchstaben aus dem lateinischen Alphabet notiert: $\mathbf{a}, \mathbf{b}, \mathbf{c}, \ldots, \mathbf{z}$. Hierbei können wiederum Zeichen aus dem lateinischen Alphabet bzw. numerische Werte für die Indizes der Elemente des Vektors verwendet werden. Transponierte Vektoren werden mit einem hochgestellten großen T ausgezeichnet:

$$\mathbf{a} = \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} a_1 & \cdots & a_n \end{pmatrix}^T$$

Matrizen

Die verwendete Notation für Matrizen entspricht der für Vektoren. Der Unterschied besteht darin, dass für Matrizen ein großer, **fett** gedruckter Buchstabe aus dem lateinischen Alphabet ($\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots, \mathbf{Z}$) verwendet wird:

$$\mathbf{A} = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}.$$

Hierbei bezeichnet $1 \leq i \leq m$ die Zeile und $1 \leq j \leq n$ die Spalte für jeden Eintrag a_{ij} der $m \times n$-Matrix. Die transponierte Matrix zu $\mathbf{A}$ wird mit $\mathbf{A}^T$ angegeben.

Operatoren

Name	**Notation**
Skalarprodukt von $\mathbf{a}$ und $\mathbf{b}$	$\langle \mathbf{a}, \mathbf{b} \rangle$
Normale der Ebene E	$\mathbf{n}(E)$
Transponierte der Matrix A	A^T

1. Einleitung

Der vorliegende Endbericht der Projektgruppe *NetSensLog* (*Net*zwerkbasierte *Sens*orfusion für einen Fahrzeugschwarm in der Intra*log*istik) fasst die von der Projektgruppe erarbeiteten Ergebnisse zusammen. In diesem Kapitel soll zunächst eine allgemeine Einführung in die Thematik der Projektgruppe gegeben werden, bevor in den folgenden Kapiteln die einzelnen Teilaspekte der Projektgruppe vorgestellt werden.

In Abschnitt 1.1 wird zunächst die Motivation hinter der Aufgabenstellung der Projektgruppe beleuchtet, bevor Abschnitt 1.2 die konkrete Aufgabe der Projektgruppe erklärt. Anschließend wird in Abschnitt 1.3 der genaue Aufbau dieses Endberichts dargestellt.

1.1. Motivation

Der heutige Stand der Technik in der Intralogistik ist vielfach noch durch Stetigfördersysteme geprägt. Diese Systeme werden vor ihrem Aufbau hinsichtlich des zu erzielenden Durchsatzes geplant und dann entsprechend dimensioniert aufgestellt. Neben dem Hochregallager bestehen sie aus einer Vorzone, durch die Waren zum Hochregallager hin und aus dem Hochregallager hinaus transportiert werden. In der Vorzone befindet sich bei Stetigfördersystemen ein Netz aus verschiedenen Fördertechniken, die starr miteinander verbunden sind und nur eine endliche Zahl von Transportwegen ermöglichen. So werden die Waren beispielsweise mit Hilfe von Rollenförderern und Drehtischen transportiert.

Sollen die eingelagerten Waren direkt vor Ort kommissioniert werden, befinden sich in der Vorzone die hierzu notwendigen Kommissionierplätze. Jeder dieser Kommissionierplätze ist an die vorhandene Fördertechnik angebunden, damit die zu kommissionierenden Waren automatisch angeliefert werden können.

Der große Nachteil dieser Fördersysteme ist ihre Unflexibilität. Sie werden für einen bestimmten Durchsatz geplant und können später nur schwer an neue Anforderungen angepasst werden. Werden beispielsweise mehr Kommissionierplätze benötigt, so kann dies häufig nur durch aufwendige Umbauten an der bestehenden Anlage erreicht werden.

Hier setzt das neuartige Konzept der „Zellularen Fördertechnik“ an. In der Vorzone befinden sich nur die notwendigen Kommissionierplätze und in der Regel keine Stetigfördertechnik. Der Transport zwischen Hochregallager und Kommissionierplätzen wird durch ein Fahrerloses Transportsystem (FTS) realisiert, das aus mehreren Fahrerlosen Transportfahrzeugen (FTF) besteht.

Die FTF können sich in der Vorzone frei bewegen und agieren dabei vollständig autonom. Sie erhalten lediglich Aufträge vom Leitsystem, die sie anschließend eigenständig ausführen.

Hinter dem Konzept der „Zellularen Fördertechnik“ verbirgt sich die Idee, dass viele gleichartige autonome FTF in einem Fahrzeugschwarm agieren und von den Informationen anderer Fahrzeuge im Schwarm profitieren können. So können die Fahrzeuge beispielsweise ihre Positionsinformationen untereinander austauschen und entscheiden, welches Fahrzeug dem Abholpunkt eines Transportauftrages am nächsten ist.

Der große Vorteil dieser Systeme liegt in der vorhandenen Flexibilität. Wird ein höherer Durchsatz benötigt, können weitere Kommissionierplätze aufgebaut und zusätzliche Fahrzeuge in Betrieb genommen werden, ohne dass aufwendige Anpassungen, wie bei der Stetigfördertechnik, notwendig sind. Bei sinkendem Durchsatz können einzelne Fahrzeuge außer Betrieb genommen werden, sodass sich das Gesamtsystem jederzeit an die notwendige Systemleistung flexibel anpassen lässt.

Aufgrund der höheren Flexibilität gegenüber klassischer Fördertechnik ist die „Zellulare Fördertechnik“ Gegenstand aktueller Forschung. Da die Güte der Autonomie der eingesetzten Fahrzeuge primär von der Güte der zur Verfügung stehenden Sensorik, sowie der Software, die diese Sensorik nutzt, abhängt, ist es das Bestreben aktueller Forschung beides zu optimieren.

Aus diesem Forschungsinteresse heraus ergibt sich die Zielsetzung der Projektgruppe, in der eine aus Laserscanner, Odometriedaten und 3D-Tiefensensor (3D-Laserscanner, PMD-Kamera) bestehende Sensorgruppe zur nachhaltigen Detektion und Bewertung der zeitlich und örtlich variablen Umgebung entwickelt werden soll.

Gemäß der interdisziplinären Ausrichtung des Projektes erfolgte die Betreuung in Zusammenarbeit mit dem Lehrstuhl für Grafische Systeme (Informatik VII), dem Lehrstuhl für Eingebettete Systeme (Informatik XII) der TU Dortmund und dem Fraunhofer Institut für Materialfluss und Logistik (Fraunhofer IML).

1.2. Aufgabe der Projektgruppe

Alle von der Projektgruppe durchzuführenden Arbeiten werden am Multishuttle Move (s. Abschnitt 2), ein vom Fraunhofer IML entwickeltes Fahrerloses Transportfahrzeug, durchgeführt. Die bereits an diesem Fahrzeug vorhandene Sensorik wird um einen 3D-Tiefensensor erweitert und soll im Rahmen der Projektgruppe durch ein neu entwickeltes Vision-System ausgewertet werden.

Die Aufgabe der Projektgruppe umfasst den Entwurf und die Implementierung eines netzwerkbasierten Vision-Systems, das die neue 3D-Sensorik zur Kollisionsvermeidung verwendet und auf mehreren autonomen Fahrzeugen eingesetzt werden soll. Diese Aufgabe

stellt das zu erzielende Minimalziel dar. Neben diesem Minimalziel wurden eine Reihe weiterer Aufgaben definiert, die im Folgenden definiert werden.

Um die Messgüte der Sensorik zu steigern, soll eine Fusion der Bilddaten von einer 2D-CCD-Kamera und der neuen 3D-Sensorik durchgeführt werden. Hierzu werden die Fahrzeuge mit einem Industrie-PC ausgestattet, der mittels Grafikkarte für eine echtzeitfähige Bildverarbeitung erweitert wurde. Die fusionierten Daten sollen anschließend zur Erhöhung der Sicherheit bei der Kollisionsvermeidung und zur 3D-Lokalisation benutzt und anderen Fahrzeugen in geeigneter Form über das Netzwerk zur Verfügung gestellt werden.

Damit die Ergebnisse des entwickelten Vision-Systems verifiziert werden können, soll eine Visualisierung der detektierten Ergebnisse realisiert werden.

Ausgehend von den gestellten Aufgaben haben sich mehrere Arbeitspakete innerhalb der Projektgruppe ergeben, die in Abschnitt 1.3 mit Verweisen zu den jeweiligen Spezialkapiteln aufgelistet werden.

1.3. Aufbau des Endberichts

Kapitel 2 gibt eine kurze Einführung in Zellulare Transportsysteme und beschreibt die Arbeitsumgebung, in der die Projektgruppe durchgeführt wurde. Es wird eine kurze Erläuterung zum verwendeten Fahrzeug, der vorgesehen 3D-Sensorik und der Versuchshalle gegeben.

In Hinblick auf eine echtzeitfähige Bildverarbeitung wurden Versuchsreihen auf der Grafikkarte durchgeführt, um deren Eignung für die Beschleunigung der notwendigen Berechnungen zu überprüfen. Die entsprechenden Ausführungen sind in Abschnitt 4.2 zu finden.

Damit Sensordaten sinnvoll fusioniert und im Netzwerk bereitgestellte Informationen korrekt berücksichtigt werden können, ist eine globale Zeitbasis des Gesamtsystems notwendig. Erst wenn alle Fahrzeuge über eine synchronisierte Zeit verfügen, können erhaltene Informationen anhand ihres Zeitstempels sinnvoll ausgewertet werden. Kapitel 7 befasst sich mit den innerhalb der Projektgruppe getesteten Zeitsynchronisationsverfahren und beschreibt deren Ablauf und Funktionsweise.

Ausgehend von den Informatik-assoziierten und logistischen Herausforderungen beruht der Lösungsansatz des entwickelten Gesamtsystems der Projektgruppe NetSensLog auf nachfolgenden Teilaspekten:

- Akquisition der Sensordaten
- Klassifikation der Sensordaten im Rahmen der Grafische Datenverarbeitung
- Lokale Bahnplanung zur Kollisionsvermeidung

- Interaktion mit der Steuerungselektronik

Die zugrunde liegenden methodischen Konzepte lassen sich aus Sicht der Informatik in die Fachgebiete Grafische Systeme und Eingebettete Systeme einordnen. Algorithmen der digitalen Bildverarbeitung stellen die Grundlagen zur Extraktion salienter Strukturen und zur Klassifikation von Fahrzeugen und Umgebung zur Verfügung. Verfahren Eingebetteter Systeme erlauben es eine effiziente Kommunikation, sowie Steuerung der FTF zu gewährleisten.

In Kapitel 3 werden die Zusammenhänge und das Zusammenspiel zwischen den genannten Teilaspekten detaillierter aufgelöst.

In den jeweiligen Fachkapiteln werden die verwendeten Verfahren und die entwickelten Komponenten detailliert beschrieben. Kapitel 4 beschreibt detailliert den Aufbau der entwickelten Grafik-Pipeline, die sich um die Akquisition und Klassifikation der Sensordaten und um die Visualisierung der Ergebnisse kümmert.

Die Funktionsweise der lokalen Bahnplanung und die Realisierung der Kollisionsvermeidung werden in Kapitel 5 erklärt.

Das Kapitel 6 beschäftigt sich mit der zur Kommunikation notwendigen Middleware.

Im Anschluss an die detaillierten Ausführungen der einzelnen Bausteine des Gesamtsystems wird in Kapitel 8 der finale Prototyp beschrieben bevor in Kapitel 9 dessen Evaluationsergebnisse dargestellt werden.

Den Abschluss des Endberichts bildet Kapitel 10 mit einer Zusammenfassung der Ergebnisse und einem Ausblick auf mögliche Erweiterungen und Anpassungen des entwickelten Systems.

2. Zellulare Transportsysteme

In diesem Kapitel wird ein Überblick über die Arbeitsumgebung der PG gegeben. Hierzu werden in Abschnitt 2.2 das verwendete Fahrzeug, in Abschnitt 2.3 die Versuchsfläche und in Abschnitt 2.4 die getestete Sensorik vorgestellt.

2.1. Einleitung

Zellulare Transportsysteme basieren auf vielen gleichartigen fördertechnischen Einheiten. Bei den fördertechnischen Einheiten kann es sich beispielsweise wie im Falle der Projektgruppe um autonome Transportfahrzeuge handeln, die ihre Aufträge vollständig autonom durchführen. Zum Erreichen dieser Autonomie kommunizieren die verschiedenen Fahrzeuge untereinander, um beispielsweise ihre Position auszutauschen oder um Transportaufträge zu verhandeln.

Ein weiterer wichtiger Aspekt der Zellularen Transportsysteme ist die hohe Flexibilität der zugrunde liegenden Topologie. Bei steigendem Bedarf können so z. B. zusätzliche Fahrzeuge in Betrieb genommen und weitere Kommissionierstationen eingerichtet werden. Bei herkömmlicher Fördertechnik, wie beispielsweise Stetigförderern, wäre hierzu ein größerer Umbau der Anlage notwendig.

2.2. Fahrerlose Transportfahrzeuge

Bei einem Fahrerlosen Transportfahrzeug (FTF) handelt es sich um ein automatisch fahrendes Fahrzeug, das in vielen Formen eingesetzt werden kann. Je nach Anforderung kann es sich um eigens konstruierte Fahrzeuge oder für den autonomen Betrieb umgebaute Standardfahrzeuge, wie beispielsweise einen Gabelstapler, handeln.

Das vom Fraunhofer IML entwickelte und von der Firma Dematic gefertigte Multishuttle Move übernimmt innerhalb der Projektgruppe die Rolle der fahrerlosen Transportfahrzeuge.

Beim Multishuttle Move handelt es sich um ein Fahrzeug, dass die Vorteile von konventionellen Shuttle-Systemen, die sich in Zeilenregallagern bewegen, mit denen vom FTF vereint. Das Multishuttle Move kann sich nicht nur innerhalb eines Regals, sondern auch frei auf der Fläche bewegen. In Abbildung 2.1 ist ein Multishuttle Move bei der Ausfahrt aus dem Regal zu sehen.

Abbildung 2.1.: Multishuttle Move bei der Regalausfahrt (Quelle: Fraunhofer IML)

Abbildung 2.2.: Das Multishuttle Move nimmt mit seinem Teleskoparm einen Kleinteileladungsträger auf (Quelle: Fraunhofer IML)

Für die Bewegung innerhalb der Schiene besitzt das Multishuttle Move angetriebene Laufräder, die auf der Schiene aufliegen und so für Vortrieb sorgen können. In Abbildung 2.2 sind diese Laufräder an den Seiten des Fahrzeuges zu sehen.

Auf der Fläche bewegt sich das Multishuttle Move mit Hilfe eines Differentialantriebs. An der Vorderseite befinden sich zwei starr eingebaute Antriebsräder, die unabhängig voneinander angetrieben werden können und durch unterschiedliche Drehzahlen eine Kurvenfahrt oder Drehung auf der Stelle ermöglichen. Damit das Fahrzeug nicht auf dem Boden aufsetzt, besitzt es zudem ein weiteres Stützrad im Heck.

Für den Transport von Kleinteileladungsträgern (KLT) verfügt das Multishuttle Move über ein Lastaufnahmemittel (LAM), das mit Hilfe eines Teleskoparms arbeitet. Mit diesem LAM ist das Fahrzeug in der Lage KLT mit bis zu 40 kg Gewicht von beiden Seiten aufzunehmen bzw. auf beiden Seiten abzuladen. In Abbildung 2.2 ist ein Multishuttle beim Einladen eines KLT zu sehen.

2.3. Versuchsfläche

Die im Rahmen der Projektgruppe zur Verfügung stehende Versuchsfläche (ZFT-Halle) des Fraunhofer IML wird in der Abbildung 2.3 gezeigt. Die Versuchsfläche hat eine

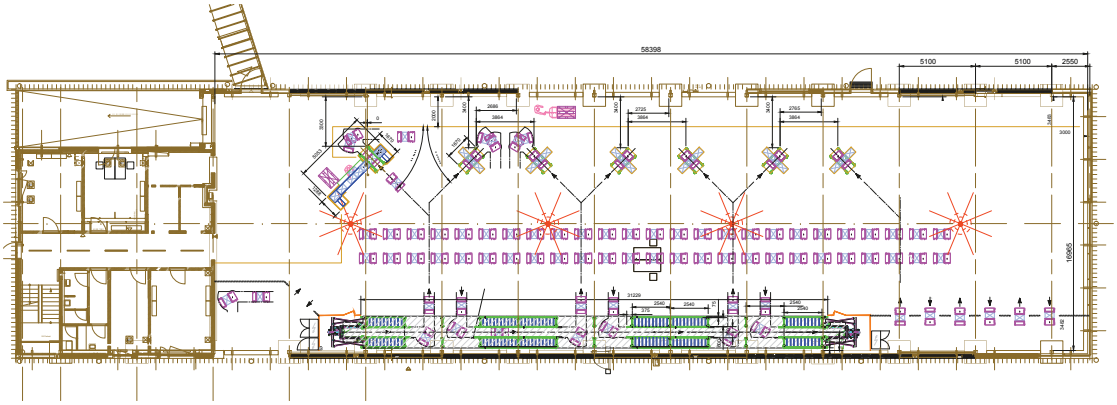


Abbildung 2.3.: Die Abbildung zeigt die ZFT-Halle in Draufsicht mit Bemaßungen, u. a. unter Berücksichtigung der Positionen der Kommissionierstationen. Die vier an der Decke angebrachten Deckenkameras (in der Abbildung • hervorgehoben) weisen sich partiell überlappende Sichtbereiche auf. Angedeutet sind ferner mehrere Fahrerlose Transportsysteme (• hervorgehoben), die u. a. beim Einfädelungsvorgang in die Kommissionierstationen gezeigt werden. (Quelle: Fraunhofer IML)

Fläche von ca. 58 m × 17 m und bietet Platz für ca. 50 Versuchsfahrzeuge. Für den Kommissionierungsvorgang sind sieben Stationen vorgesehen, die von den Fahrerlosen Transportfahrzeugen angefahren werden können. An der Decke der ZFT-Halle sind ferner vier Deckenkameras angebracht worden, über die sich u. a. im Rahmen von Tests die Fahrzeug-Positionen nachvollziehen lassen.

2.4. Sensorik

Das Ziel der Projektgruppe bestand darin, das in Kapitel 2.2 vorgestellte, multimodal ausgelegte Fahrerlose Transportsystem, um optische Sensoren für die 3D-Tiefenwahrnehmung[1] und neue Software zu erweitern, so dass Operationen außerhalb der Schienenumgebung mit hinreichender Zuverlässigkeit ohne den bisher verwendeten 2D-Laserscanner erfüllt werden können. Die Verwendung von 3D-Tiefenbildern bietet im Vergleich zu 2D-Tiefendaten viele Vorteile, da 2D-Tiefenbilder durch den auf eine Ebene beschränkten Sichtbereich im gegebenen Anwendungsgebiet sehr viel mehr Unsicherheiten implizieren. Aus diesem Grund erscheint die Verwendung eines 3D-Tiefensensors sehr erstrebenswert. Nicht zu unterschätzen ist, bei der um eine Raumdimension erweiterten Akquisition, das bei einer vergleichbaren Sampling- und Bildwiederhol-Rate wesentlich höhere Datenaufkommen, das von den verwendeten Algorithmen unter Annahme gleich

[1] Häufig wird an dieser Stelle auch der Begriff der 2,5D-Tiefenbilder verwendet, da diese im Allgemeinen keine Tiefeninformationen für verdeckte Objekte enthalten, wie dies z. B. bei einem CAD-Modell der Fall wäre.

bleibender Hardware effizienter verarbeitet werden muss, um einen vergleichbaren Durchsatz der 2D- bzw. 3D-Grafik-Pipelines zu ermöglichen. Da sich die vorhandene Hardware jedoch als der limitierende Faktor für die 3D-Grafik-Pipeline erwiesen hat, wurde diese durch ein leistungsfähigeres System auf Basis des Intel-Atom-Prozessors ausgetauscht (Details zum leistungsfähigeren System siehe Tabelle B.1).

2.4.1. Kategorisierung von Tiefenmessverfahren

Im Allgemeinen können Verfahren für die Tiefenbestimmung einer der drei folgenden Oberkategorien zugeordnet werden:

1. Stereoverfahren,
2. Auswertung monokularer Tiefenhinweise,
3. aktive Tiefengewinnung (s. [BJ95]).

Verfahren zur Bestimmung der Tiefeninformation aus einem monokularen Kamerasystem, d. h. einer einzelnen Kamera, versuchen die verlorene Tiefeninformation aus Oberflächeneigenschaften, wie Form oder Schattierung zu rekonstruieren. Diese Verfahren sind jedoch nicht in der Lage absolute Entfernungen zu bestimmen, sondern liefern nur relative Entfernungsangaben. Ferner stellen diese Verfahren im Allgemeinen explizite Anforderungen an die Aufnahmebedingungen. Sie sind somit für den praktischen Einsatz weniger geeignet, da keine zuverlässigen Aussagen bezüglich ihrer Güte getroffen werden können. Aus diesem Grund soll die Idee der Auswertung monokularer Tiefenhinweise an dieser Stelle zwar erwähnt, aber nicht weiter vertieft werden (weitere Informationen und Literaturempfehlungen siehe [BJ95]).

Stereoverfahren nutzen zwei herkömmliche Kameras, die – stark vereinfacht gesprochen – aus zwei unterschiedlichen Positionen die Szene aufnehmen. Es gilt jedoch die Annahme, dass die von den Kameras aufgenommenen Bildbereiche sich zumindest teilweise überlappen und somit nicht vollständig disjunkt sind. Aus den zu bestimmenden Punktkorrelationen in beiden Kamerabildern lassen sich analog zur menschlichen optischen Wahrnehmung die Tiefeninformation der aufgenommenen Szene rekonstruieren.

Verfahren zur aktiven Tiefengewinnung, d. h. unter Verwendung einer aktiven Energiequelle, stellen ein weites Gebiet dar und lassen sich grob in drei Unterkategorien aufteilen:

1. direkte Laufzeitverfahren,
2. indirekte Laufzeitverfahren (z. B. mit Hilfe der Phasenverschiebung) und Verfahren,
3. die strukturiertes Licht für die Tiefenbestimmung verwenden.

Laufzeitverfahren können mit Hilfe einer direkten oder indirekten Lichtlaufzeitmessung erfolgen (s. [BJ95]). Die direkte Lichtlaufzeitmessung misst die Zeit, die ein Lichtstrahl für den Hin- und Rückweg zum Objekt der gegebenen Szene benötigt. Bekannte Vertreter dieses Messverfahrens stellen die meisten 2D-/3D-Laserscanner dar. Problematisch ist bei diesem Ansatz die hohe Lichtgeschwindigkeit, die sehr hohe Anforderungen an die Präzision der Zeitmessung zwischen emittiertem und detektiertem Licht stellt. Die benötigte, hochpräzise Zeitmessvorrichtung verursacht einen signifikanten Kostenfaktor der direkten Lichtlaufzeitmessung. Dieses Problem versuchen jene Verfahren zu umgehen, die sich einer indirekten Lichtlaufzeitmessung bedienen und das emittierte Signal geeignet modulieren. Die Lichtmodulation kann über die Amplitude, Phase (siehe Abbildung 2.5), Frequenz oder eine Kombination aus den genannten Modulationsarten erfolgen. Einen Vertreter dieser Herangehensweise, der indirekten Lichtlaufzeitmessung, stellt die PMD-Kamera dar. Bei der PMD-Kamera erfolgt die Tiefenbestimmung über die Phasenverschiebung φ zwischen dem Sinus-förmig modulierten emittierten (elektrischen) und detektierten (optischen) Signal. Jedoch birgt die indirekte Lichtlaufzeitmessung auch Gefahren, die sich z. B. in einem beschränkten Eindeutigkeitsbereich widerspiegeln.

Die im Rahmen der Projektgruppe untersuchte Microsoft Kinect ist der letzten verbleibenden Unterkategorie „Tiefenbestimmung unter Verwendung strukturierten Lichts“ zuzuordnen. Hierbei wird ein bekanntes Muster auf die Szene projiziert, das anschließend für die Tiefenrekonstruktion genutzt wird. Aus den Differenzen zwischen dem projizierten und dem vom IR-CMOS registrierten Muster lassen sich mittels Triangulation die Tiefenunterschiede der Szene berechnen. Dieser Ansatz ist in der Fachliteratur seit längerem bekannt und wurde nun erstmalig für den Endverbraucher zugänglich gemacht, was sich vor allem im vergleichsweise niedrigen Preis der Microsoft Kinect äußert.

2.4.2. PMD-Kamera

Photonic-Mixer-Devices-Kameras (PMD-Kameras) (siehe Abbildung 2.4) stellen eine Alternative zu Stereokamerasystemen oder 2D-/3D-Laserscannern für die Bestimmung von Tiefenbildern dar. Ihr Funktionsprinzip basiert auf der indirekten Lichtlaufzeitmessung über die Bestimmung der Phasenverschiebung zwischen emittiertem (dem so genannten elektrischen) und detektiertem (dem so genannten optischen) Signal.

Die PMD-Technologie weist aktuell eine hohe Innovationsrate auf (siehe Abbildung 2.8). Sie wurde innerhalb der letzten Jahre kontinuierlich verbessert und die nahe Zukunft lässt noch weitere interessante Fortschritte auf diesem Gebiet erwarten. Im folgenden Verlauf soll diese noch relativ neue Technologie genauer betrachtet werden, die sich besonders durch simultane Akquisition von Tiefen- und Intensitätenbild mit Hilfe eines einzelnen Sensorchips hervorhebt.

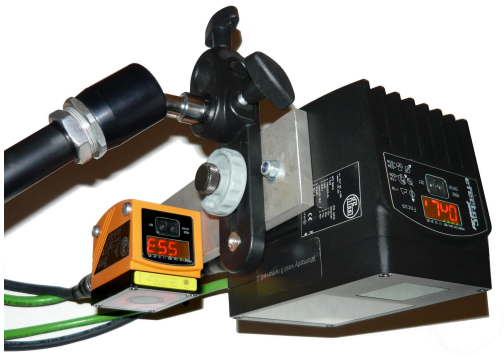


Abbildung 2.4.: Die Abbildung zeigt zwei an einem Stativ befestigte PMD-Kameras. Die linke PMD-Kamera, ein Vorserienmuster, (● orangenes Kameragehäuse) entspricht dem Typ O3D100AA. Die rechte PMD-Kamera (● schwarzes Kameragehäuse) des Typs O3D201AB stellt ein verbessertes Nachfolgermodell der O3D100AA dar. (Quelle: Fraunhofer IML)

Eigenschaften der O3D201AB (PMD-Kamera)

Die im Rahmen der Projektgruppe untersuchte PMD-Kamera O3D201AB von ifm weist 64×50-Messwerte bei einem Öffnungswinkel von $30° \times 40°$ auf. Die aktive Lichtquelle operiert im für den Menschen nicht sichtbaren Infrarotbereich, genauer bei 850 nm. Ferner hat die O3D201AB drei fest voreingestellte Modulationsfrequenzen (Erläuterung siehe Kapitel 2.4.3). Folglich können drei PMD-Kameras vom Typ O3D201AB simultan an einem Ort ohne Interferenzen verwendet werden. Die Verwendung von mehr als drei PMD-Kameras an einem Ort vom Typ O3D201AB erfordert somit die Synchronisation der Bildakquisition, genauer der Licht-Emission (einfache Realisierung mittels des globalen Zeitschlitzverfahrens Time Division Multiple Access (TDMA)). Im Rahmen der Projektgruppe wurden Untersuchungen bezüglich der Interferenzen zwischen mehreren baugleichen, auf die gleiche Modulationsfrequenz parametrisierten PMD-Kameras durchgeführt. Die Ergebnisse sind im Kapitel 9.5.3 zusammengefasst. Die Kommunikation zwischen dem Client und der O3D201AB erfolgt per Ethernet-Schnittstelle. Für die tabellarische Auflistung aller Eigenschaften siehe Tabelle 2.1.

2.4.3. Distanzberechnung aus der Phasenverschiebung φ

Die Berechnung der Distanz erfolgt bei PMD-Kameras über die indirekte Lichtlaufzeitmessung, d. h. der Auswertung der Phasenverschiebung zwischen emittiertem, Sinus-förmig moduliertem und dem detektierten Signal. Die Phasenverschiebung φ lässt sich rechnerisch wie folgt mit Hilfe von vier aufeinander folgenden Abtastwerten A_i bestimmen, die

Eigenschaft	Wert
Messwerte	64×50
Öffnungswinkel	$30° \times 40°$
aktive Lichtquelle	850 nm (Infrarot)
Modulationsfrequenz(en)	23 MHz, 20,6 MHz, 20,4 MHz
Schnittstelle(n)	Ethernet

Tabelle 2.1.: Die Tabelle zeigt eine Übersicht über die technischen bzw. optischen Eigenschaften der O3D201AB (PMD-Kamera) von ifm. [ifm10b]

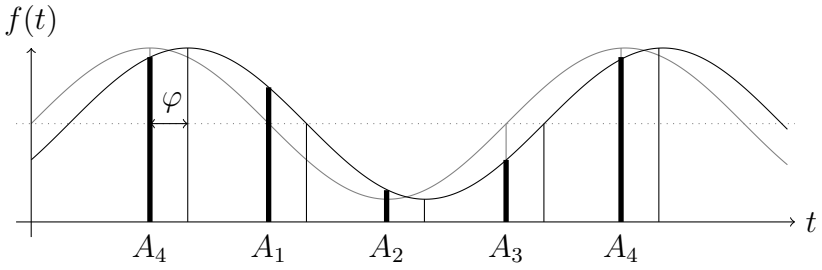


Abbildung 2.5.: Die Abbildung visualisiert die Korrelation zwischen emittiertem (• grauen) und detektiertem (• schwarzen) Signal. Hierbei entsprechen A_1 bis A_4 den Abtastpunkten, die jeweils einen konstanten Abstand von $\frac{\pi}{2}$ bzgl. des emittierten Signals aufweisen. An den Abtastpunkten wird der Funktionswert des detektierten Signals ausgewertet. Die Abbildung ist der Grafik aus [HR06] nachempfunden.

jeweils einen Abstand von $\frac{\pi}{2}$ haben:

$$\varphi = \tan^{-1}\left(\frac{A_1 - A_3}{A_2 - A_4}\right).$$

Die absolute Distanz d in Metern ergibt sich aus der Phasenverschiebung φ, der Lichtgeschwindigkeit c und der Modulationsfrequenz f_{mod} des emittierten Signals:

$$d = \frac{c \cdot \varphi}{4\pi \cdot f_{\text{mod}}}.$$

Der Zusammenhang zwischen emittiertem und detektiertem Signal wird in der Abbildung 2.5 gezeigt. Zusätzlich kann über die Offsets b die Intensität am jeweiligen Messpunkt bestimmt werden, so dass über eine Messung gleichzeitig ein Tiefen-, als auch ein Grauwertbild der Intensitäten erhalten werden kann (s. [HR06, HR07, Sch03]).

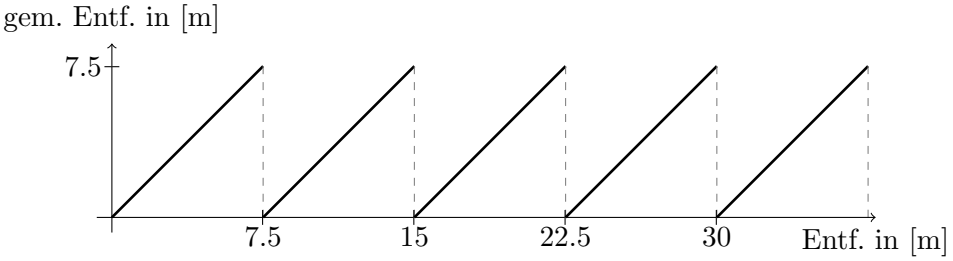


Abbildung 2.6.: Die Abbildung visualisiert den Eindeutigkeitsbereich bei der Distanzbestimmung mit Hilfe der Phasenverschiebung und der Modulations-Frequenz $f_{\text{mod}} = 20\,\text{MHz}$. Objekte, die sich außerhalb des Bereichs von 7,5 m befinden (die x-Achse entspricht der tatsächlichen Entfernung zum Objekt), werden in einer Entfernung (die y-Achse entspricht der gemessenen Entfernung) von $d \mod 7,5\,\text{m}$ wahrgenommen (d steht hierbei für die Distanz zum Objekt).

Eindeutigkeitsbereich der Distanzberechnung

Durch Bestimmung der Distanz über die Phasenverschiebung zwischen emittiertem und detektiertem Signal, ergibt sich ein bestimmter Eindeutigkeitsbereich für die Messung. Dies bedeutet, dass die Entfernung zu Objekten, die sich außerhalb dieses Bereichs befinden, nicht eindeutig bestimmt werden kann. Der Eindeutigkeitsbereich E ergibt sich aus der halben Wellenlänge λ:

$$E = \frac{\lambda}{2} = \frac{c}{2 \cdot f_{\text{mod}}}.$$

Für eine Modulations-Frequenz $f_{\text{mod}} = 20\,\text{MHz}$ ergibt sich somit der Eindeutigkeitsbereich von:

$$E = \frac{3 \cdot 10^8\,\text{m s}^{-1}}{2 \cdot 20 \cdot 10^6\,\text{s}^{-1}} = 7{,}5\,\text{m}.$$

Die Abbildung 2.6 visualisiert den Zusammenhang zwischen der tatsächlichen Entfernung von einem Objekt zur PMD-Kamera und der anhand der Phasenverschiebung bestimmten Distanz. Durch die Verwendung multipler Modulationsfrequenzen kann die Genauigkeit der Messung verbessert werden, sofern dieser Modus von der jeweiligen PMD-Kamera unterstützt wird (siehe z. B. [ifm10b, ifm10a]).

Integrationszeit(en) bei PMD-Kameras

Entscheidend für qualitative Messwerte (weder unter- noch überbelichtet) ist bei den PMD-Kameras die Wahl der richtigen Integrationszeiten.[2] Hierbei besteht der Unterschied zu

[2]Die Integrationszeit entspricht der Belichtungszeit einer herkömmlichen Kamera.

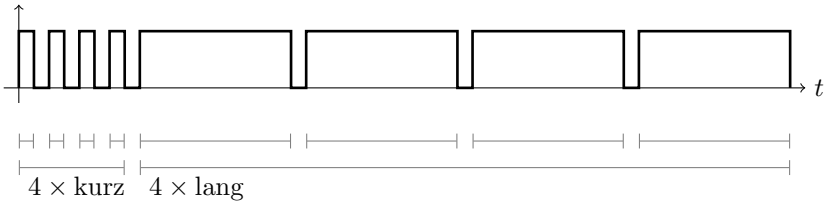


Abbildung 2.7.: Die Abbildung zeigt das standardmäßige, doppelte Integrationsschema, das u. a. von den beiden PMD-Kameras des Typs O3D100AA und O3D201AB unterstützt wird. Die Abbildung zeigt folglich die zu Beginn durchgeführten vier Messungen mit der kurzen Integrationszeit. Anschließend werden vier weitere Messungen mit der langen Integrationszeit durchgeführt. Die Notwendigkeit der vier Messungen pro Integrationszeit ist bedingt durch die Bestimmung der Phasenverschiebung.

einer normalen Kamera darin, dass normalerweise zwei unterschiedliche Integrationszeiten für die Gewinnung der Messdaten verwendet werden.[3] Der Grund für die Verwendung zweier unterschiedlicher Integrationszeiten liegt darin begründet, dass bei einer einzelnen langen Integrationszeit die Gefahr der Überbelichtung von nahe gelegenen bzw. stark reflektierenden Objekten besteht. Bei einer zu kurzen Belichtungszeit besteht wiederum die Gefahr, dass weiter entfernt gelegene oder stark Licht-absorbierende Objekte nicht detektiert werden können. Um folglich die Distanz von zu weit als auch von zu nahe gelegenen Objekten in einer Szene adäquat bestimmen zu können, wird ein doppeltes Integrationsschema verwendet (siehe Abbildung 2.7). Die unterstützten Integrationszeiten liegen zwischen 1 μs und 5 ms. Die Hersteller-Empfehlung für das Verhältnis zwischen kurzer und langer Integrationszeit entspricht 1 zu 10.[4] Die unterbelichteten Messwerte, die mit Hilfe der kurzen Integrationszeit gewonnen wurden, werden anschließend durch die Messwerte, die mit Hilfe der langen Integrationszeit gewonnen wurden, substituiert[5]. Die Abbildung 2.7 zeigt das doppelte Integrationsschema der O3D201AB beginnend mit der kurzen Integrationszeit. Zur Bestimmung der Phasenverschiebung φ sind jeweils vier Messungen pro Integrationszeit notwendig.

Entwicklung der Sensorauflösung bei PMD-Kameras

Bei den PMD-Kameras handelt es sich um noch eine relativ junge Technologie, an der aktuell intensiv geforscht und die somit kontinuierlich verbessert wird. Aktueller Stand der Technik sind PMD-Kameras mit 200 × 200-Messpunkten [PMD11]. Geplant für

[3] Beide vorgestellten PMD-Kameras können ebenfalls auf eine einzelne Integrationszeit umgestellt werden.
[4] Die O3D201AB unterstützt hardwareseitig einen undokumentierten Befehl zur automatischen Bestimmung der optimalen Integrationszeiten.
[5] Beim Vorgängermodell O3D100AA wird konträr mit der langen Integrationszeit begonnen. Somit gilt folglich, dass die überbelichteten Messwerte, die mit Hilfe einer langen Integrationszeit gewonnen wurden, durch Messwerte, die mit Hilfe einer kurzen Integrationszeit gewonnen wurden, substituiert werden.

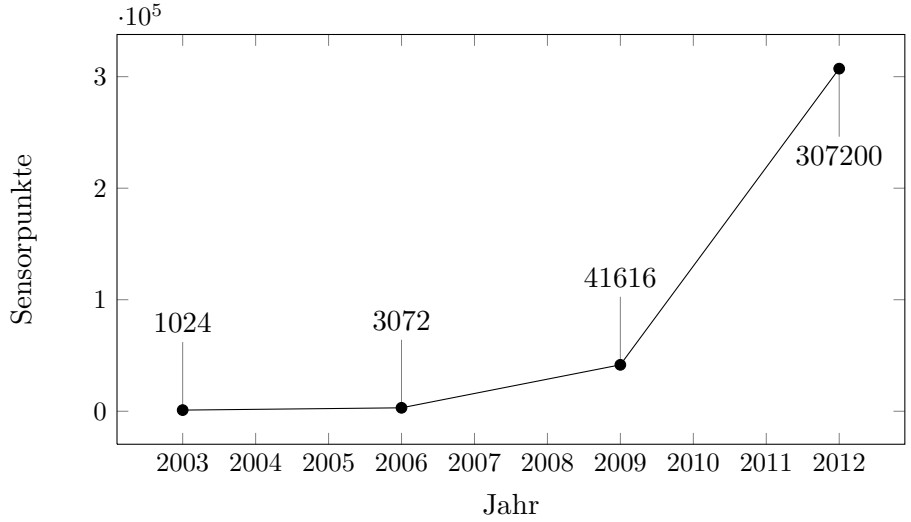


Abbildung 2.8.: Die Abbildung visualisiert die Steigerung der Anzahl der Sensorpunkte bei PMD-Kameras im Verlauf der Jahre. Die jeweiligen zu Grunde gelegten Sensorauflösungen lauten: $64 \times 16 = 1024$ (2003), $64 \times 48 = 3072$ (2006), $204 \times 204 = 41616$ (2009) und $640 \times 480 = 307200$ (~ 2012) Sensorpunkte. Daten nach [Fre08].

das Jahr 2012 sind jedoch Sensoren mit 640×480-Messpunkten. Im Automotive bzw. Industrieumfeld sind jedoch vorwiegend Sensoren mit einer wesentlich geringeren Anzahl von Messwerten präsent. Erklärbar ist dies u. a. dadurch, dass kleinere Sensormatrizen mittlerweile ausgereifter und robuster sind. Die Abbildung 2.8 zeigt die Entwicklung der Sensorpunkte seit 2003 und eine Schätzung der Sensorpunkte für das Jahr 2012.

Bildakquisition bei PMD-Kameras

Im Allgemeinen bieten PMD-Kameras entweder einen Ethernet- oder aber einen USB-Anschluss zur Parametrisierung und Bildakquisition. Die in diesem Kapitel beschriebenen PMD-Kameras weisen beide einen Ethernet-Anschluss auf, was wegen der größeren Maximalkabellänge von 100 m (USB hingegen „nur“ 5 m) ohne Verwendung von Repeatern für den Industrieeinsatz prinzipiell vorteilhafter ist. Die Kommunikation mit der PMD-Kamera über Ethernet erfolgt in der Regel über zwei separate TCP-Verbindungen. Über eine der Verbindungen wird die PMD-Kamera parametrisiert, über die andere erfolgt die Bildakquisition. Das für die Parametrisierung verwendete Protokoll entspricht XML-RPC (weitere Informationen siehe [Win99]), für die Bildakquisition bzw. Datenübertragung wird ein proprietäres Format verwendet. Zu Beginn wird ein Verbindungsaufbau mit Hilfe der XML-RPC Anweisung `MDAXMLConnectCP` initiiert. Anschließend erfolgt die Initialisierung des für die Bildakquisition zuständigen Servers der Kamera.

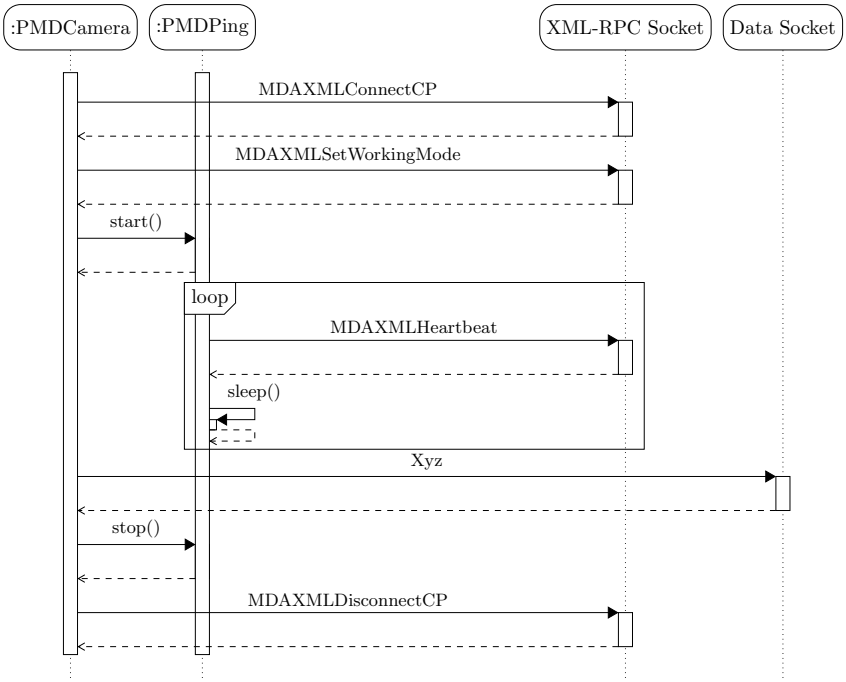


Abbildung 2.9.: Die Abbildung zeigt ein vereinfachtes Sequenzdiagramm, welches den Ablauf eines Verbindungsaufbaus per Ethernet zwischen dem Client und einer PMD-Kamera visualisiert.

Ferner wird eine Keepalive-Prozedur gestartet, die die Verbindung zur PMD-Kamera mit Hilfe des XML-XPC Befehls `MDAXMLHeartbeat` aufrecht erhält. Die Bildakquisition erfolgt über eine Anfrage per Datenverbindung. Eine Verbindungsterminierung kann clientseitig über den Befehl `MDAXMLDisconnectCP` initiiert werden. Der Ablauf wird in Abbildung 2.9 als vereinfachtes Sequenzdiagramm gezeigt. Nähere Informationen können der Entwickler-Referenz entnommen werden [ifm07, ifm10a].

Zwischenfazit

Bei den PMD-Kameras handelt es sich um eine Technologie, die sich in naher Zukunft nicht nur im Automotive- (z. B. zur Kollisionsvermeidung) sondern auch im Industriebereich (z. B. zur Kontrolle produzierter Güter oder für automatisierte Dekommissionierungs-Vorgänge) vollständig etablieren könnte. Da die Umrechnung der Sensordaten in Weltkoordinaten vollständig auf der Hardware der PMD-Kamera erfolgt und nicht u. U. rechenintensiv auf dem Client berechnet werden muss (wie z. B. bei der Microsoft Kinect), sind PMD-Kameras prinzipiell gut für den Einsatz in eingebetteten Systemen mit begrenzter Rechenleistung geeignet – so wie sie im Rahmen der Projektgruppe auf den Fahrzeugen zur Verfügung steht. Als nachteilig könnte die aktuell noch geringe

Anzahl von Messwerten aufgefasst werden, so dass sich die Erkennung von Objekten mittels PMD-Kamera generierter Tiefenbilder auf grobe Strukturen beschränken muss. Die geringe Anzahl von Messwerten der zur Verfügung stehenden PMD-Kameras war der Beweggrund für die Projektgruppe ihren Fokus hauptsächlich auf die höher auflösende Microsoft Kinect zu legen (siehe Kapitel 2.4.4). Ferner ist es zwingend erforderlich beim Einsatz multipler PMD-Kameras (sofern die Anzahl der möglichen Modulationsfrequenzen nicht ausreicht) die Bildakquisition, genauer die Licht-Emission zu synchronisieren, da es andernfalls zu Interferenzen und somit nachweislich zu Messfehlern kommt (siehe Kapitel 9.5.3).

2.4.4. Microsoft Kinect

Bei der Microsoft Kinect (siehe Abbildung 2.10) handelt es sich im Vergleich zu den anderen bisher vorgestellten, aktiven Tiefensensoren um Hardware, die speziell auf den Endverbrauchermarkt fokussiert ist. Das zuvor unter dem Namen „Natal“ bekannte Projekt, das in Kooperation zwischen der Firma PrimeSense und Microsoft gemeinsam entwickelt wurde (siehe [Mic10b, Pri10]), zielte darauf ab, die Interaktion mit der Microsoft Spielkonsole (XBOX 360) zu revolutionieren. Der, im Vergleich zu anderen Tiefensensoren aus dem industriellen Bereich, niedrige Preis hat dazu geführt, dass kurz nach der Einführung Ende 2010 OpenSource-Implementierungen für die verbreiteten Plattformen erstellt wurden (z. B. libfreenect). Diese Entwicklung wurde von Microsoft initial mit Missgefallen betrachtet, später zeigte man sich jedoch offener und initiierte ein eigenes offizielles, zudem offenes Framework (OpenNI). Anders als bei 2D-/3D-Laserscannern bzw. PMD-Kameras erfolgt die Tiefenbestimmung zwar unter Verwendung einer aktiven Lichtquelle, jedoch nicht über die (direkte oder indirekte) Lichtlaufzeitmessung, sondern über die Projektion und anschließende Auswertung eines bekannten Musters (siehe Abbildung 2.11).

Eigenschaften der Microsoft Kinect

Die Microsoft Kinect weist eine Tiefenauflösung von 640×480 Pixeln und eine Vollfarbauflösung von 1280×1024 Pixeln bei einem Öffnungswinkel von $58° \times 40°$ auf und übertrifft bzgl. der Sensorauflösung die PMD-Kamera deutlich.

Die Projektion des Musters erfolgt durch einen temperaturstabilisierten Laser und eine einfache Lochmaske. Die Stabilisierung erfolgt durch den Einsatz eines Peltier-Elements. Der Laser operiert im für den Menschen nicht sichtbaren Infrarotbereich, genauer bei 830 nm. Überlagerungen der von unterschiedlichen Kinect-Kameras projizierten Punktmuster und damit mit möglicherweise entstehenden Interferenzen sind im Use-Case der Projektgruppe anzunehmen. Die Untersuchungen hierzu werden in Kapitel 9.5.2 zusammengefasst. Die Projektion des Musters ist temporär abschaltbar, jedoch beträgt die hierdurch auftretende Verzögerung ca. (1000 ± 63) ms, aufgeteilt auf im Mittel 523 ms zum Einschalten, 13 ms für die Akquisition und 413 ms für das Ausschalten der Kinect (siehe

(a) MS-Kinect (geschlossen)

(b) MS-Kinect (offen)

Abbildung 2.10.: Die Abbildung 2.10(a) zeigt die Microsoft Kinect in geschlossenem Zustand, d. h. mit Gehäuse. In Abbildung 2.10(b) wurde das Gehäuse der Microsoft Kinect abgenommen, so dass der IR-Projektor, Color-CMOS und der IR-CMOS erkennbar sind (von links nach rechts). [iFi10]

Eigenschaft	Wert
Messwerte	640×480 (Tiefenbild), 1280×1024 (Vollfarbbild)
Messbereich	ca. 0,5 m bis 8 m
Öffnungswinkel	$58° \times 40°$
aktive Lichtquelle	830 nm (Infrarot Laser)
Schnittstelle(n)	USB

Tabelle 2.2.: Die Tabelle zeigt eine Übersicht über die technischen bzw. optischen Eigenschaften der Microsoft Kienct. [Mic10b, Pri10]

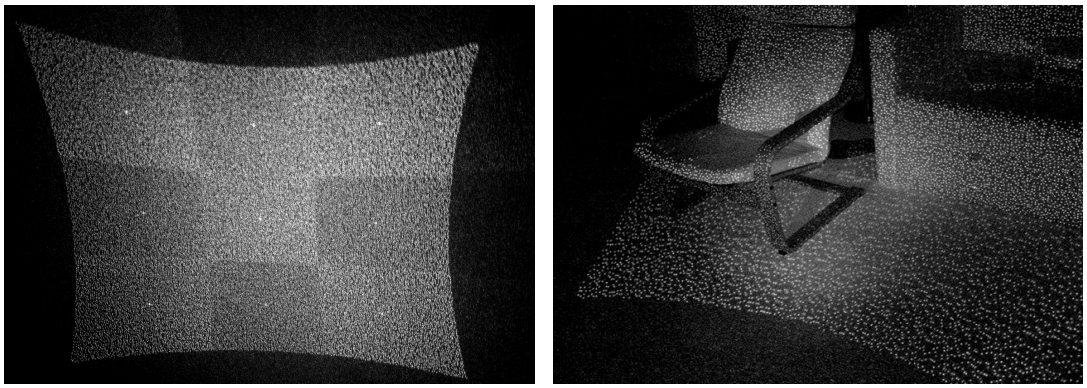
Kapitel 9.5.2). Somit entfällt die Möglichkeit des Einsatzes eines Zeitschlitzverfahrens (TDMA) mit der geforderten unteren Schranke für die Bildwiederholrate von 4 Hz. Als Anschlussmöglichkeit weist die Microsoft einen herkömmlichen USB-Anschluss auf. Für die tabellarische Auflistung aller Eigenschaften siehe Tabelle 2.2.

Tiefenbestimmung mittels strukturiertem Licht

Die Bestimmung des Tiefenbildes erfolgt bei der Microsoft Kinect mit Hilfe der Projektion und anschließenden Auswertung eines bekannten Punktemusters (siehe Abbildung 2.11). Interessant ist hierbei, dass die Aufbereitung der Daten vollständig in Hardware erfolgt (weitere Informationen siehe [Pri10]). Von offizieller Seite sind keine genaueren Informationen über die Funktionsweise bekannt. Die Patentschrift zum Sensor von PrimeSense ist allgemein gehalten, so dass Aussagen über die exakte Funktionsweise nur mutmaßlich anhand der vorgestellten Methoden getroffen werden können [ZSMG06]. Die Bestimmung von Tiefeninformationen aus der Projektion von strukturiertem Licht stellt jedoch keine neue Technologie dar. Aus der Literatur sind viele Ansätze bekannt, die sich dieses Verfahrens bedienen. Ein Beispiel für ein bereits 1990 vorgestelltes Verfahren für die Tiefenbestimmung aus strukturiertem Licht mit Hilfe einer binären Codierung nach P. Vuylsteke und A. Oosterlinck wird in Abbildung 2.12 gezeigt (für weitere Informationen wird an dieser Stelle auf [BJ95] verwiesen).

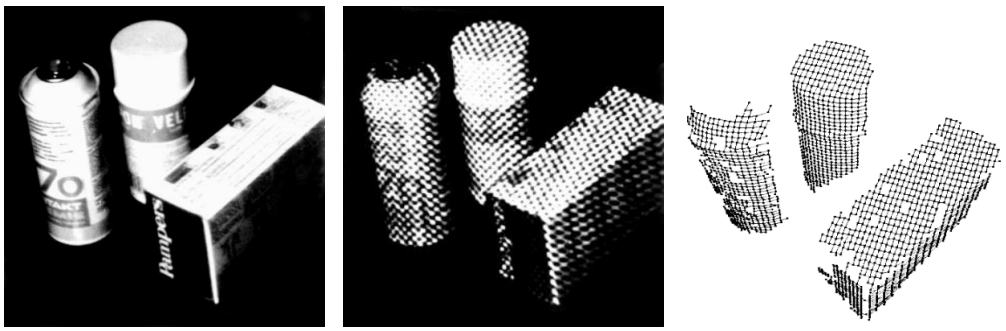
Bildakquisition bei der Microsoft Kinect

Da die Berechnung der Tiefendaten mittels Auswertung des projizierten Musters vollständig in Hardware erfolgt, besteht die Aufgabe für den Client einzig in der Anforderung der Bilddaten und der Umrechnung in absolute Distanzwerte (z. B. in Metern). Das Abbildungsverhalten der Microsoft Kinect ist nicht linear.



(a) (b)

Abbildung 2.11.: Die Abbildung 2.11(a) zeigt das von der Microsoft Kinect auf eine planare Fläche projizierte IR-Muster. Augenfällig ist die Schachbrettstruktur des projizierten Musters, mit jeweils sich durch eine hohe Helligkeit hervorhebenden Punkten in den Zentren der einzelnen Schachbrettfelder. Abbildung 2.10(b) zeigt die Projektion des Musters auf eine mögliche reale Szene. Angelehnt an [KR12].



(a) (b) (c)

Abbildung 2.12.: Die Abbildung 2.12(a) zeigt die ggbn. Szene als herkömmliches Intensitätenbild. Die Abbildung 2.12(b) zeigt die Projektion eines binären Musters auf die Objekte der Szene. Die Abbildung 2.12(c) zeigt die Rekonstruktion der Tiefeninformation aus der Projektion. [BJ95]

Zwischenfazit

Im Rahmen der Projektgruppe wurde die Entscheidung bezüglich des Tiefensensors zu Gunsten der Microsoft Kinect getroffen. Ausschlaggebende Gründe waren hierbei die der PMD-Kamera deutlich überlegene Auflösung und das neben dem Tiefenbild zusätzlich von der Microsoft Kinect bereitgestellte RGB-Bild.

3. Struktur der Lösung

Im Folgenden werden die Entscheidungsgründe für die gewählte Aufteilung sowie die Zusammenhänge der Module genauer beschrieben. In Abschnitt 3.2 wird zunächst erläutert welche Gründe zur Abstrahierung der Datenakquise geführt haben. In den folgenden Abschnitten 3.3 bis 3.6 wird ein Überblick über die einzelnen Komponenten, grafische Datenverarbeitung, Bahnplanung, Fahrzeugsteuerung und Middleware gegeben.

3.1. Einleitung

Aus den in Abschnitt 1 beschriebenen Gegebenheiten und Anforderungen an das aus dem Projekt hervorgehende System lassen sich verschiedene Teilbereiche ableiten, die sich in Modulen strukturieren lassen. Auf einer abstrakten Ebene betrachtet hat das Projekt die Zielstellung, die Fahrzeuge jeweils möglichst unabhängig voneinander zu befähigen, sich autonom anhand von externen Befehlsvorgaben in der intralogistischen Lagerhalle zu bewegen. Diese Bewegung ist unter der Berücksichtigung von gegebenenfalls vorhandenen Hindernissen durchzuführen, welche mit Hilfe der neu in das Fahrzeug integrierten optischen 3D-Sensorkomponenten erkannt werden. Konkret handelt es sich bei diesen Sensoren um die PMD- und Kinect-Kameras, welche in Kapitel 2 im Detail beschrieben werden.

Durch die Vielfältigkeit und Heterogenität der für das Gesamtkonzept zu berücksichtigenden Teilaspekte, kann eine Modularisierung als sinnvoll erachtet werden. Die Lösung wurde in grober Betrachtung in folgende Module aufgeteilt:

- Datenakquisition
- Grafische Datenverarbeitung
- Bahnplanung
- Lokalisierung und Steuerung

Die Module sind mit ihren jeweiligen Zusammenhängen in Abbildung 3.1 dargestellt. Die Pfeile stehen für den Übertragungsweg von Daten, wobei die Pfeilspitze jeweils die Übertragungsrichtung und die Beschriftung die übertragenen Daten angibt.

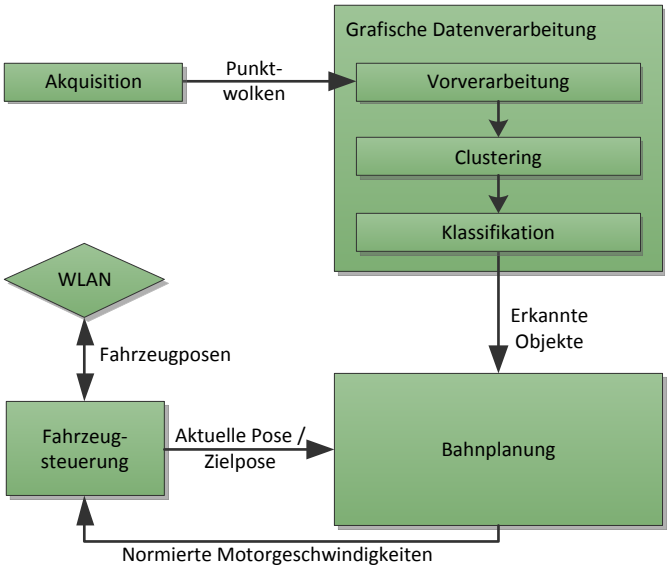


Abbildung 3.1.: Die Abbildung zeigt die verwendeten Module mit den jeweiligen Zusammenhängen. Die Pfeile stehen für den Übertragungsweg von Daten, wobei die Pfeilspitze jeweils die Übertragungsrichtung und die Beschriftung die übertragenen Daten angibt.

3.2. Datenakquisition

Auf Grund der Gegebenheit, dass potenziell verschiedene Sensoren eingesetzt werden, stellt die Datenakquisition ein eigenständiges Modul der Lösung dar. Die Aufgabe dieses Moduls besteht in der Verkapselung der jeweils unterschiedlichen Aufnahmefunktionalitäten der Sensoren sowie der Umwandlung der aufgenommenen Daten in ein für die verbundenen Module einheitliches Format. Mit dem Ziel einer möglichst geringen Einschränkung der prinzipiell in der Datenakquisition implementierbaren Sensoren, wurde eine 3D-Punktwolke als Ausgabeformat gewählt. Die entsprechende Vorgehensweise wird in Abschnitt 4.3 detailliert erläutert.

3.3. Grafische Datenverarbeitung

Um einen Nutzen aus den aufgenommenen Punktwolken ziehen zu können, ist eine weitere Verarbeitung und Auswertung notwendig, welche sich im Rahmen des modularen Konzeptes als grafische Datenverarbeitung bezeichnen lässt. Grundsätzlich besteht das Ziel dieses Moduls in der Detektion von Objekten der Umgebung anhand der eingehenden Punktwolken. Die zu detektierenden Objekte können unterschiedlicher Art sein. Es kann sich beispielsweise um statische Umgebungshindernisse, andere Fahrzeuge oder Stationen handeln. Zur typabhängigen Berücksichtigung der Hindernisse ist neben der Detektion und Posenbestimmung folglich auch eine Klassifikation notwendig. Zur Erfüllung dieser Zielstellung ist die grafische Datenverarbeitung als klassische Mustererkennungs-Pipeline aufgebaut, welche aus den ohne Rückkopplung aufeinander folgenden Schritten der Vorverarbeitung, Clustering und Klassifikation besteht [Fin11]. Der Aufbau der Pipeline ist in Abbildung 3.1 innerhalb des rechten Kastens dargestellt.

Ausschließlich die von der Datenakquisition ausgegebenen Punktwolken gehen in den ersten Schritt der grafischen Datenverarbeitung ein. Die Ausgabe des letzten Schrittes besteht in der Beschreibung der erkannten Objekte, welche die Position und Ausrichtung (Pose), den abgeschätzten Radius sowie den klassifizierten Typ beinhaltet. Diese Beschreibungen können von den nachfolgenden Modulen verwendet werden, um ihre jeweilige Funktionalität durchzuführen. Zum einen benötigt die Bahnplanung die Beschreibung von Hindernissen, um diese umfahren zu können, und zum anderen können die erkannten Station als globale Fixpunkte im Rahmen der Lokalisation verwendet werden.

3.4. Bahnplanung

Die Planung einer Trajektorie lässt sich prinzipiell ohne die Datenakquisition und ohne die grafische Datenverarbeitung durchführen, weswegen sie sich ebenfalls als eigenständiges Modul „Bahnplanung“ betrachten lässt. Zur Bestimmung einer Trajektorie gehen

verschiedene Informationen in das Modul ein. Zum einen wird die zum Planungszeitpunkt aktuelle Pose des Fahrzeugs sowie die Position und gegebenenfalls die Ausrichtung des Fahrziels benötigt. Zur Vermeidung von Hindernissen gehen zusätzlich die von der grafischen Datenverarbeitung ausgegeben Beschreibungen von umgebenden Objekten und die bekannten Posen anderer Fahrzeuge ein. Die Bahnplanung berechnet anhand dieser Daten zu jedem kontinuierlich wiederkehrend auftretenden Planungszeitpunkt für jeden Antrieb zeitlich lokale Steuerungsinformationen, welche an die Fahrzeugsteuerung weitergegeben werden.

3.5. Fahrzeugsteuerung

Das Modul der Fahrzeuglokalisation und -steuerung stellt das hardwarenächste Modul des Konzeptes mit zwei zentralen Aufgaben dar. Zum einen ist es für die Bestimmung der eigenen Fahrzeugpose anhand der von der grafischen Datenverarbeitung erkannten globalen Fixpunkte wie Stationen zuständig und zum anderen stellt es die Schnittstelle zwischen der Fahrzeugsteuerung und dem Bahnplanungsmodul. Zur Erfüllung der erstgenannten Aufgabe werden die Beschreibungen der erkannten Fixpunkte eingegeben. Um den Informationsaustausch mit der Bahnplanung zu gewährleisten, werden die eigene Fahrzeugpose und die Pose des anzufahrenden Zielpunktes ausgegeben und die Steuerungsinformationen für die Antriebe gehen in das Modul ein. Zusätzlich ist die Fahrzeuglokalisation dafür verantwortlich, der Bahnplanung von anderen Fahrzeugen die eigene Pose mitzuteilen.

3.6. Middleware

Um ein funktionsfähiges Konzept zu gewährleisten, muss eine Schnittstelle zwischen den verschiedenen Modulen hergestellt werden, welche die gesamte Kommunikation einheitlich abbildet. Die Anforderungen an diese Kommunikationsschnittstelle sind je nach kommunizierendem Modul unterschiedlich geartet. Grundsätzlich lässt sich die Übertragung von Daten lokal auf dem Fahrzeug von der Übertragung von Daten zwischen unterschiedlichen Fahrzeugen unterscheiden. Bei der fahrzeugübergreifenden Schnittstelle kann zusätzlich bezüglich der Toleranz von fehlgeschlagenen Daten eine Unterscheidung getroffen werden. Einige Daten müssen zwangsläufig vom empfangenden Fahrzeug erhalten werden und andere Daten dürfen zum Beispiel durch Störungen verloren gehen. Letzteres ist etwa bei der Übertragung der eigenen Pose an andere Fahrzeuge der Fall, da diese nach einem erfolglosen Übertragungsversuch in der Regel ohnehin veraltet ist. Während zur Erfüllung dieser heterogenen Anforderungen verschiedene Kommunikationskanäle verwendet werden, ist die Struktur der zu übertragenden Daten grundsätzlich redundant. Die Aufgabe der Middleware besteht in der einheitlichen Realisierung der geschilderten Kommunikation.

Die Middleware stellt die zentrale Schnittstelle zwischen den anderen Modulen dar und die Kommunikation findet über unterschiedliche Kanäle statt. Die von der Datenakquisition aufgenommenen Bilder werden lokal an die grafische Datenverarbeitung weitergeleitet und die erkannten Objekte werden ebenfalls lokal an die Bahnplanung übermittelt. Im Gegensatz dazu wird die eigene Pose an die anderen Fahrzeuge von der Fahrzeugsteuerung fahrzeugübergreifend übertragen. Letzteres ist in Abbildung 3.1 als rautenförmiges Symbol dargestellt.

4. Grafische Datenverarbeitung

Im Zuge der Grafischen Datenverarbeitung werden die Informationen des Tiefensensors, der ein Tiefenbild der vor dem Fahrzeug liegenden Objekte liefert, akquiriert (s. Abschnitt 4.3), aufbereitet und ausgewertet (s. Abbildung 3.1). Das Ergebnis ist die Anzahl und die Typen von Objekten, die erkannt wurden. Diese Daten werden anschließend an das Bahnplanungsmodul (s. Abschnitt 5) übermittelt. Somit stellt dieser Verarbeitungsschritt das Bindeglied zwischen der Datenakquisition und der Bahnplanung dar. Der Aufbau der Grafischen Datenverarbeitung kann grob in die vier aufeinander folgenden Verarbeitungsschritte untergliedert werden:

1. Akquisition (s. Abschnitt 4.3),
2. Boden- und Wandentfernung sowie Downsampling (s. Abschnitt 4.4.2, 4.4.1 und 4.4.3),
3. Clustering (s. Abschnitt 4.5.1) und
4. Klassifikation (s. Abschnitt 4.5.2).

Diese Pipeline wird von jeder akquirierten Punktwolke durchlaufen. Im Folgenden werden die Methodischen Hintergründe der einzelnen Pipelinestufen erläutert. Die konkrete Implementierung wird in Abschnitt 8 beschrieben.

4.1. Anforderungen und Ziele

Das wesentliche Ziel der Grafischen Datenverarbeitung besteht darin, die von dem Tiefensensor stammenden Daten zu benutzen, um Hindernisse, wie zum Beispiel Kisten, sowie Zielobjekte, wie eine Kommissionierstation, vor dem Fahrzeug zu erkennen und diese zu klassifizieren. Außerdem sollen Abstand, Winkel und Ausdehnung des Objekt möglichst exakt bestimmt werden.

Abbildung 4.1 zeigt beispielhaft, wie eine Situation aus Sicht eines Fahrzeuges aussehen könnte. Im Sichtfeld des Sensors des unteren Fahrzeuges (im Bild hellgrau dargestellt) befinden sich Teile eines zweiten Fahrzeuges sowie Teile einer Kommissionierstation. Die Aufgabe der Grafischen Datenverarbeitung ist es nun, diese vom Tiefensensor aufgenommenen Daten auszuwerten und die erkannten Objekte zu klassifizieren. Außerdem sollen die relativen Positionen in Bezug auf das Sensorkoordinatensystem berechnet werden.

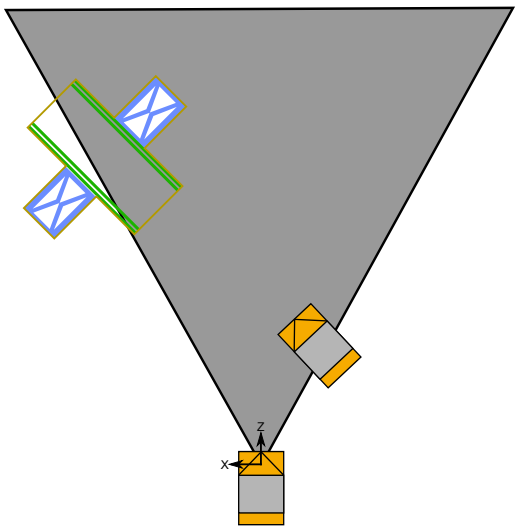


Abbildung 4.1.: Darstellung des Sichtbereiches eines Fahrzeuges mit Tiefensensor. Unten: Fahrzeug; Grau: Der Sichtkegel des Tiefensensors; Links oben: Kommissionierstation

Aus diesen können mit Hilfe der bekannten eigenen Position die globalen Positionen der erkannten Objekte berechnet werden.

Die Klassifikation soll minimal drei Klassen von Objekten unterscheiden können:

- Fahrzeuge,
- Stationen und
- Sonstiges.

Eine Unterscheidung der einzelnen Fahrzeuge soll dabei nicht stattfinden, da die Identität der Fahrzeuge nicht allein aus einem Tiefenbild rekonstruierbar ist. Es soll lediglich festgestellt werden, dass es sich bei dem erkannten Objekt um ein weiteres Fahrzeug handelt und es soll die Position sowie die Orientierung des Fahrzeuges aus den Daten errechnet werden. Zusätzlich kann der Beladungszustand des Fahrzeuges erkannt werden, wenn das Fahrzeug von der Seite gesehen wird.

Bei der Erkennung von Stationen soll neben der Position der Station zusätzlich ein salienter Punkt an der Station erkannt werden, so dass die genaue Position und Ausrichtung der Station in Bezug auf das Fahrzeug berechnet werden kann. Dadurch ist es möglich eine Bahn zu errechnen, die eine Einfahrt in die Station erlaubt.

Alle anderen Objekte sollen erkannt und deren Position bestimmt werden. Außerdem soll die Ausdehnung des Objektes bestimmt werden, so dass es gefahrlos umfahren werden kann.

Eine wichtige Anforderung der Grafischen Datenverarbeitung ist die Echtzeitfähigkeit des Systems. Daher muss die Auswertung der Tiefenbilder mindestens vier mal pro Sekunde stattfinden, so dass immer sichergestellt ist, dass genug Zeit für ein Brems- oder Ausweichmanöver zur Verfügung steht.

4.2. OpenCL

Auf Grund des hohen Rechenzeitbedarfs und den Echtzeitanforderungen an die Grafikpipeline ist auf den Fahrzeugen ein ION-Rechner eingebaut worden, auf dem die Berechnungen ablaufen können. Dieser Rechner besteht aus einem Intel Atom 330 Prozessor und einer GeForce 9400M Grafikkarte. Die Rechenleistung dieses Systems ist im Vergleich zu modernen Desktoprechnern sehr gering, jedoch ist der Einbau von schnelleren Komponenten mit höherem Stromverbrauch auf den batteriegetriebenen Fahrzeugen nicht sinnvoll. Die genauen Daten des ION-Systems sind im Anhang (s. Abschnitt B.1) zu finden.

Auf Grund der geringen Geschwindigkeit des Hauptprozessors ist getestet worden, ob mit Hilfe von OpenCL die komplette Grafikpipeline oder Teile davon auf der Grafikkarte ausgeführt werden können. Dabei sollte OpenCL genutzt werden, ein Framework, das es dem Programmierer ermöglicht, parallele Programme zu erstellen, die auf verschiedener Hardware, insbesondere auf der GPU, ausgeführt werden können. Dabei wird unterschieden zwischen dem Hostprogramm, das auf der CPU läuft und die Verwaltung aller Aufgaben übernimmt und den Kerneln, die die Programme enthalten, die zur parallelen Ausführung geschrieben wurden. Je nach dem gewählten Parallelisierungskonzept (Task- oder Datenparallelität) werden die Kernel unterschiedlich auf die parallel arbeitenden Einheiten der GPU (oder eines anderen OpenCL Device) ausgeführt. Dabei unterscheidet OpenCL konzeptionell zwischen einem Plattform-, einem Ausführungs-, einem Speicher- und einem Programmiermodell. Einzelheiten zum Konzept und zur Programmierung von OpenCL sind in [Mun08, NVI10, Mü11] zu finden.

Die Performancetests, die auf dem ION ausgeführt wurden, haben gezeigt, dass die GPU im Bereich der reinen Rechenleistung der CPU weit überlegen ist (s. Abschnitt 9.2.1). Schon bei der Ausführung der Tests hat sich jedoch herausgestellt, dass diese Überlegenheit der GPU nur dann voll zum tragen kommt, wenn alle Streaming-Prozessoren der GPU voll ausgelastet werden können, was in praxisrelevanten Algorithmen aber nicht immer garantiert werden kann. Wenn nicht parallelisierbare Berechnungen ausgeführt werden sollen, liegen sieben der acht Streaming-Prozessoren eines Streaming-Multiprozessors brach und die Rechenleistung reduziert sich entsprechend. Daher sind die Testergebnisse nur bedingt aussagekräftig, da diese immer darauf ausgelegt waren, alle Streaming-Prozessoren voll auszulasten. Weiterhin ist getestet worden, ob der Transfer der Daten auf die Grafikkarte und zurück in den Hauptspeicher schnell genug ablaufen, so dass die höhere Leistung der Grafikkarte nicht durch die Speichertransferzeiten wieder verloren gehen (s. Abschnitt 9.2.1). Dabei hat sich herausgestellt, dass die Transferzeiten nicht

vernachlässigt werden können. Nur, wenn schon bei der Datenakquisition eine Datenreduktion stattfindet, wird die Transferzeit der Daten zur Grafikkarte so stark reduziert, dass sich ein Zeitgewinn durch die Berechnung auf der GPU ergeben kann. Die Tests haben also gezeigt, dass eine Portierung der Grafikpipeline auf die Grafikkarte durchaus leichte Vorteile bringen kann. Allerdings ist der Aufwand für die Portierung der teils sehr komplexen Algorithmen auf die Grafikkarte für den zu erwartenden geringen Performancegewinn innerhalb der Projektgruppe nicht gerechtfertigt. Daher wurde der Ansatz nach den Tests nicht weiter verfolgt und die Grafikpipeline auf der CPU umgesetzt.

4.3. Akquisition der Punktwolke

Den initialen Schritt der Grafik-Pipeline bildet die Akquisition der Tiefen- bzw. RGB-Informationen. Hierbei werden die akquirierten Daten zunächst in eine vom Sensor unabhängige Punktwolkendarstellung transformiert, die die Basis für alle nachfolgenden Schritte der Grafik-Pipeline darstellt. Da die RGB-Informationen in den nachfolgenden Schritten der Pipeline nicht benötigt werden kann die akquirierte Punktwolke P vereinfacht wie folgt definiert werden:

$$P = \{(x, y, z) \mid x, y \in \mathbb{R}, z \in \mathbb{R}^+\} \tag{4.1}$$

Eine Punktwolke ist somit eine Menge von dreidimensionalen Punkten mit den kartesischen Koordinaten (x, y, z). Zusammen mit dem im Ursprung lokalisierten Zentrum des Tiefensensors kann ein Punkt aus P wie folgt interpretiert werden:

- x beschreibt die horizontale Verschiebung zum Sensorzentrum: Punkte mit $x > 0$ liegen rechts des Sensorzentrums, Punkte mit $x < 0$ liegen links.
- y beschreibt die vertikale Verschiebung zum Sensorzentrum: Punkte mit $y > 0$ liegen oberhalb des Sensorzentrums, Punkte mit $y < 0$ liegen unterhalb.
- z beschreibt die Tiefenverschiebung zum Sensorzentrum: Je größer y, desto weiter ist der Punkt vom Sensorzentrum entfernt.

4.4. Vorverarbeitung

Anschließend an den Akquisitionsschritt folgt der Vorverarbeitungsschritt. Im Vorverarbeitungsschritt werden die Quelldaten für die Objekterkennung aufbereitet. Hierzu erfolgt initial in Abhängigkeit vom verwendeten Sensor ein Clipping der Punktwolke. Dieses hat zur Aufgabe die von Tiefensensoren im Grenzbereich bestimmten, stark fehlerbehafteten Tiefendaten zu verwerfen. Anschließend folgen die Wand- (siehe Kapitel 4.4.1) und Bodenentfernung (siehe Kapitel 4.4.2). Durch diese beiden Vorverarbeitungsschritte entfallen im Allgemeinen alle für den Objekterkennungsschritt nicht notwendigen Punkte.

Ferner bewirken diese beiden vorgeschalteten Schritte einen höheren Durchsatz der Grafik-Pipeline, da in den nachfolgenden, rechenintensiven Schritten weniger Punkte betrachtet und verarbeitet werden müssen. Um die Dichte der Punktwolke zu homogenisieren erfolgt vor dem Objekterkennungsschritt noch eine Mittellung der einzelnen Punkte $\mathbf{p} \in P$ über einem dreidimensionalen Raster mit Quadern gleicher Größe (siehe Kapitel 4.4.3).

4.4.1. Wandentfernung

Da sich das Fahrzeug beim Einsatz innerhalb einer Halle bewegt, werden vom Tiefensensor auch Wände aufgenommen. Aufgrund der dabei entstehenden großen Anzahl von Punkten in der Punktwolke, ist eine Erkennung der Wände und Entfernung der entsprechenden Punkte notwendig.

Im Folgenden wird angenommen das Wände senkrecht auf dem Boden stehen, keine Fenster enthalten und nicht gekrümmt sind. Verstrebungen und Säulen werden als Rauschen betrachtet. Weiterhin kann davon ausgegangen werden, dass die Höhe der Wand größer ist als die Höhe aller anderen Objekt im Erfassungsbereich des Tiefensensors.

Für die Erkennung der Wände ist eine Segmentierung der Punktwolke notwendig. Als mögliche Verfahren wurden in [Rus09] die Segmentierung durch Region Growing sowie die Segmentierung durch modelbasiertes Fitting von Ebenen mittels RANSAC [FB81] beschrieben. Aufgrund des hohen Rechenzeitbedarfs ist eine Berechnung der Oberflächennormalen, wie sie für den Region Growing erforderlich ist, nicht praktikabel. Daher wurde im Rahmen der Projekgruppe der RANSAC-basierte Ansatz weiterverfolgt.

Random Sample Consensus

Der Random Sample Consensus (RANSAC) ist ein Verfahren mit der die Parameter eines Modells für eine gegebene Menge von Werten iterativ approximiert werden können. Bei dem im Rahmen der Projektgruppe verwendeten Verfahren wurde eine in [Rus09] beschriebe Variante der RANSAC-Algorithmus verwendet. Der Algorithmus arbeitet basierend auf [Rus09] in den folgenden Schritten:

Gegeben sei ein Modell sowie eine Menge P von Werten

1. Zufälliges Auswählen einer, zur Berechnung der Parameter ausreichenden, Anzahl von Werten
2. Berechnung der Modell-Parameter aus den gewählten Punkten
3. Bestimme eine neue Menge P^* von Werten, deren Abstand vom berechneten Modell innerhalb einer bestimmten Toleranz liegt. Speichere P^*
4. Die vorangegangen Schritte werden so lange wiederholt, bis eine vorgegebene Anzahl von Iteration erreicht wurde.

5. Die größte berechnete Menge P^* wird als Grundlage des bestimmten Modells ermittelt

Algorithmus

Der Wandentfernungs-Algorithmus berechnet für eine gegebene Eingabe-Punktwolke P, eine Ausgabe-Punktwolke die alle Punkte enthält, die nicht zu einer Wand gehören. Der Algorithmus arbeitet in drei Schritten: Zuerst wird aus der Eingabe-Punktwolke eine Menge von Punkten extrahiert die nur aus Wandpunkten besteht. Aus dieser Teilmenge werden die den Wänden entsprechenden Ebenen berechnet. Schließlich werden anhand der berechneten Ebenen alle Wandpunkte aus der Eingabe-Punktwolke entfernt.

Für die Berechnung einer Teilmenge $\overline{P}$ der Eingabe-Punktwolke P, die nur Wandpunkte enthält, kann ausgenutzt werden, dass Wände höher sind als alle anderen Objekte im Aufnahmebereich. Anhand eines empirisch ermittelten Schwellwertes $h_{\min}$ kann die reduzierte Punktwolke berechnet werden als: $\overline{P} = \{\mathbf{p} \mid \mathbf{p} \in P \wedge p_y > h_{\min}\}$

Um die Berechnung der Ebenen zu beschleunigen kann auf $\overline{P}$ optional ein Random Downsampling durchgeführt werden. Beim Random Downsampling wird die Menge $\overline{P}$ durch eine kleinere Menge von zufällig aus $\overline{P}$ gewählten Punkten ersetzt. Die Anzahl der gewählten Punkte hängt dabei von der Eingabe ab und muss empirisch bestimmt werden.

Kern des Algorithmus ist die Bestimmung von Wandebenen mittels RANSAC. Per RANSAC kann die Ebene innerhalb einer Punktwolke gefunden werden, die von den meisten Punkten gestützt wird. Als Modell wir eine Ebene in Hesse Normalform verwendet.

Wird bei der Ausführung des RANSAC-Algorithmus eine zu große Toleranz gewählt, kommt es wie in Abbildung 4.2 zu sehen, zu fehlerhaft erkannten Ebenen. Dieses Problem tritt hauptsächlich bei Ecken und weit entfernten Wänden auf. Es führt zu einer unvollständigen Entfernung der entsprechenden Punkte.

Wie in Abbildung 4.3 zu sehen, kann durch eine kleinere Toleranz eine bessere Anpassung der Ebene erreicht werden. Nach der Bestimmung der Wandebene kann das Rauschen eliminiert werden. Hierfür werden alle Punkte innerhalb eines vorgegebenen Anstands zur Wandebene entfernt. Sowohl die verwendete Toleranz als auch der Abstand in dem Punkte entfernt werden, müssen empirisch bestimmt werden.

Da in einer aufgenommen Punktwolke Punkte mehrere Wände auftreten können ist es notwendig, dass der Algorithmus mehrere Ebenen in der Punktwolke finden und die entsprechenden Punkte entfernen kann. Dies kann durch wiederholtes Entfernen einzelner Wandebenen erreicht werden.

Die berechneten Ebenen werden schließlich benutzt, um die Wandpunkte aus der Eingabe-Punktwolke zu entfernen. Hierfür wird für jeden Punkt die Entfernung zu allen Ebenen berechnet und bei zu geringer Entfernung entfernt.

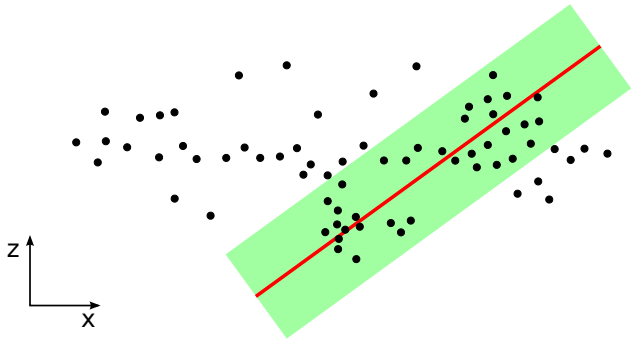


Abbildung 4.2.: Punktwolke einer Wand dargestellt. Aufgrund der sehr großen Toleranz (grün) bei der Anwendung RANSAC-Algorithmus, wird die Wandebene (rot) nicht korrekt erkannt.

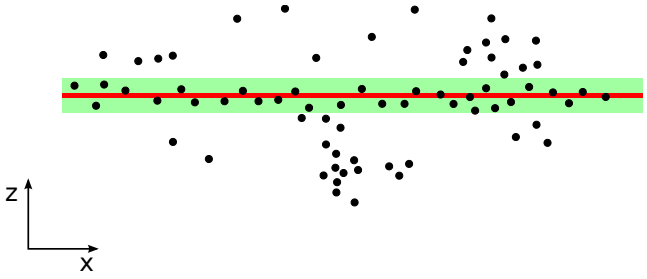


Abbildung 4.3.: Durch eine Verkleinerung der Toleranz wird die gefundene Wandebene verbessert.

4.4.2. Bodenentfernung

Bedingt durch die Montage des Tiefenbildsensors ist in vielen der akquirierten Punktwolken auch ein Teil des Boden auf dem sich das Fahrzeug bewegt erfasst worden. Diese Teilmenge der Punktwolke („Bodenpunkte“) ist für die Objekterkennung (s. Abschnitt 4.5) nicht relevant.

Diese Verarbeitungsstufe ist der zweite Abschnitt in der Pipeline und folgt direkt auf die „Wandentfernung“ (s. Abschnitt 4.4.1). Die übergebenen Punktwolke wurde somit bereits um Punkte reduziert, die durch die Aufnahme von Wänden erzeugt wurden.

Ziel dieser Pipeline-Verarbeitungsstufe ist aus der gegeben Punktwolke die „Bodenpunkte“ zu entfernen. Durch die Entfernung dieser Punkte wird die Größe der Punktwolke reduziert und damit der Aufwand für alle nachfolgenden Verarbeitungsschritte der Pipeline reduziert.

In diesem Rahmen wurden drei verschieden Ansätze zur Entfernung der Bodenpunkte entwickelt:

1. Bodenentfernung auf Basis des Random Sample Consensus (s. Abschnitt 4.4.2)
2. Bodenentfernung auf Basis der Principal Component Analysis (s. Abschnitt 4.4.2)
3. Bodenentfernung auf Basis eines selbst entwickelten Ansatzes (s. Abschnitt 4.4.2).

RANSAC Bodenentfernung

Mit der RANSAC Bodenentfernung wurde ein Ansatz verfolgt, der das in Kapitel 4.4.1 beschriebene Wandentfernungs-Verfahren für die Bodenentfernung adaptiert. Die RANSAC Bodenentfernung läuft dabei in vier Schritten ab:

1. Bestimmung einer Menge potenzieller Bodenpunkte
2. Segmentierung der Teilmenge mit RANSAC
3. Bestimmung der wahrscheinlichsten Bodenebene aus den segmentierten Ebenen
4. Entfernung aller Punkte deren Entfernung zur Ebene innerhalb eines Schwellwertes liegt

In ersten Schritt wird die Eingabe-Punktwolke auf eine kleinere Teilmenge potenzieller Bodenpunkte reduziert. Die Reduktion wird durch einen Passthrough-Filter erreicht, der alle Punkte oberhalb des Sensorzentrum sowie weit entfernt liegende Punkte entfernt. Um die Menge weiter zu reduzieren kann auf der bestimmten Punktmenge noch ein Random Downsampling durchgeführt werden (siehe Kapitel 4.4.1).

Aus der bestimmten Teilmenge wird mit dem in Kapitel 4.4.1 beschriebenen Algorithmus alle Ebenen E segmentiert. Eine Betrachtung aller möglichen Ebenen ist notwendig, da

es durch Objekte im Aufnahmebereich weitere Ebenen gefunden werden können. Um die Bodenebene zu bestimmen, werden alle gefunden Ebenen anhand der Ausrichtung zu xz-Ebene bewertet. Für die gefundene Bodenebene E_{Boden} gilt

$$|\langle \mathbf{n}(E_{\text{Boden}}), \mathbf{e}_y \rangle| = \min\{|\langle \mathbf{n}(E'), \mathbf{e}_y \rangle| \mid E' \in E\}$$

Die Entfernung der Bodenpunkte im vierten Schritt wird in Kapitel 4.4.2 beschrieben.

PCA Bodenentfernung

Ein Ansatz, um die Bodenebene zu finden besteht darin, eine Hauptkomponentenanalyse (PCA) durchzuführen. Die PCA erwartet als Eingabe eine Punktmenge eines m-dimensionalen Raumes und berechnet die Hauptausdehnungsrichtungen in absteigender Reihenfolge über diese Punktmenge. Die Idee des Verfahrens ist, die erhaltenen Ausdehnungsrichtungen zu untersuchen und zu bestimmen, durch welche dieser Richtungen die Ebene, die die Bodenpunkte enthält, aufgespannt wird. Die Güte des Verfahrens ist dementsprechend vor allem von den sichtbaren Boden- sowie nicht-Bodenpunkten abhängig.

Die PCA bekommt als Eingabe eine Menge von Punkten einer Dimension m, hier der Dimension $m = 3$. Gesucht ist die Ebene, so dass der aufsummierte Abstand aller Punkte von der Ebene minimal wird. Dies ist ein Optimierungsproblem, dass mit

$$\sum_{i=1}^{n} (\langle \mathbf{e}, \mathbf{a_i} \rangle - d)^2 \to \min$$

beschrieben werden kann. Dabei ist die Anzahl der Punkte mit n gegeben, die Punkte selber werden mit $\mathbf{a_i}$ bezeichnet. Der Koeffizientenvektor $\mathbf{e}$ und der Abstandswert d beschreiben die Ebene mit

$$\langle \mathbf{e}, \mathbf{x} \rangle = d$$

und der Nebenbedingung

$$\|\mathbf{e}\|_2^2 = 1. \tag{4.2}$$

Um die Rechnung zu vereinfachen werden diese Punkte zunächst in ihren Schwerpunkt verschoben, das heißt, mittelwertfrei gemacht. Damit vereinfacht sich die Formel, da nun $d = 0$ gilt. Um die Nebenbedingung 4.2 zu berücksichtigen werden Lagrange-Multiplikatoren verwendet. Damit kann die resultierende Gleichung mit

$$\min_{\mathbf{e},\lambda} \sum_{i=1}^{n} \langle \mathbf{e}, \mathbf{a_i} \rangle^2 - \lambda(\|\mathbf{e}\|_2^2 - 1)$$

beschrieben werden. Die optimalen Lösungen ergeben sich durch partielles Ableiten nach den Variablen $\mathbf{e}$ und λ und anschließendes Null setzen. Setzt man die Matrix $\mathbf{A}$

so, dass jede Spalte von $\mathbf{A}$ dem Spaltenvektor $\mathbf{a_i}$ entspricht, so ergeben sich die beiden Gleichungen:

$$\mathbf{A}\mathbf{A}^T\mathbf{e} = \lambda\mathbf{e} \tag{4.3}$$

sowie

$$\|\mathbf{e}\|_2^2 = 1.$$

Bei der Gleichung 4.3 handelt es sich um eine Eigenwertgleichung, die gelöst werden kann. Die Eigenvektoren der Lösungen geben in absteigender Reihenfolge die Hauptachsen an, die zugehörigen Eigenwerte die Summe der quadratischen Abstände der Punkte zu der Ebene. Weitere Einzelheiten zur Berechnung der PCA können [Mü11, Fin11] entnommen werden.

Das Auffinden der Bodenebene ausschließlich mit Hilfe der PCA auf der kompletten Punktwolke zeigt einige Schwächen, die dazu geführt haben, dass dieser Ansatz nicht weiter verfolgt wurde. Der Grundgedanke bei dieser Methode war es, dass die Bodenebene auf Grund der vielen sichtbaren Bodenpunkte so dominant in den Daten ist, dass sie die Hauptachsen, die von der PCA berechnet werden, bestimmen wird. Tests haben ergeben, dass dies ist in vielen Aufnahmen auch der Fall ist. Es kann jedoch auch der Fall eintreten, dass die Eingabepunktwolke nur sehr wenige oder sogar gar keine Punkte des Bodens beinhaltet. Dies ist immer dann der Fall, wenn ein Objekt sehr nahe an das Fahrzeug herankommt und dann viel oder den kompletten Sichtbereich des Sensors einnimmt. In diesem Fall werden die Hauptachsen die Bodenebene nicht repräsentieren. Des weiteren ist bei der Verwendung der gesamten Punktwolke zu beachten, dass die Hauptachsen auch bei guten Sichtbedingungen (es ist also viel vom Boden zu erkennen), nicht genau den Boden repräsentieren werden. Bedingt durch einige Punkte, die Objekten angehören, die auf dem Boden liegen (Fahrzeuge, Kisten oder Stationen) wird die berechnete Ebene einerseits zu hoch liegen und andererseits, was wesentlich bedeutender ist, um einen gewissen Winkel zu dem Boden rotiert sein. Es müssten also vor der Berechnung der PCA weitere Vorverarbeitungsschritte durchgeführt werden, die es garantieren, dass sehr viele Bodenpunkte in der Eingabe verbleiben, während nicht-Bodenpunkte vorab entfernt werden. Eine solche Vorverarbeitung ist nicht mehr getestet worden, da schon die vorher durchgeführten Tests gezeigt haben, dass die Berechnung der PCA relativ langsam ist und ein naiver Ansatz zur Bodenentfernung wesentlich schneller und leistungsfähiger erschien (s. Abschnitt 4.4.2).

Schnelle Bodenentfernung

Während die Ansätze zur Bodenentfernung auf Basis von RANSAC und PCA (s. Abschnitt 4.4.2) eine allgemeine Vorgehensweise zur Bestimmung von Ebenen in beliebigen Punktmengen darstellen, wird im Folgenden ein Algorithmus vorgestellt, der speziell für die in Abschnitt 4.3 akquirierten Punktwolken optimiert ist.

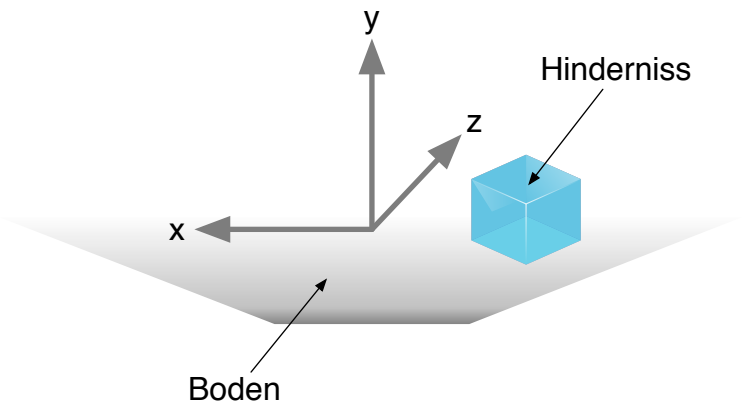


Abbildung 4.4.: Punktwolke aus der Perspektive des Tiefensensors (in Richtung der z-Achse).

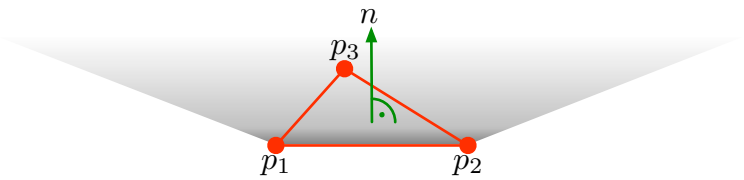


Abbildung 4.5.: Punktwolke mit dem Dreieck dass durch die Bodenpunkten $\mathbf{p}_1, \mathbf{p}_2$ und $\mathbf{p}_3$ aufgespannt wird.

Eigenschaften der Punktwolke Wie in Abbildung 4.4 zu sehen ist, bildet der aufgenommene Boden eine trapezförmige Teilmenge der Punktwolke. Diese Punkte werden im Folgende als „Bodenpunkte“ bezeichnet. Ziel ist es genau diese Teilmenge aus der Punktwolke zu entfernen.

Algorithmus Der Algorithmus zur Entfernung der Bodenpunkte ist in drei aufeinander folgende Schritte unterteilt:

1. Bestimmung von drei Bodenpunkten
2. Bestimmung der Ebene die diese drei Punkte enthält
3. mit Hilfe der berechneten Ebene wird die Teilmenge der Punktwolke die den Boden repräsentiert entfernt.

Bestimmung der Bodenpunkte Um eine Ebene im $\mathbb{R}^3$ eindeutig bestimmen zu können werden pro Punktwolke drei Punkte $\mathbf{p}_1, \mathbf{p}_2$ und $\mathbf{p}_3$ bestimmt. Dieses drei Punkte werden entsprechend Abbildung 4.5 aus der Menge der Bodenpunkte gewählt. Die Punkte $\mathbf{p}_1$ und $\mathbf{p}_2$ werden dabei so gewählt, dass sie jeweils in den dem Sensorzentrum möglichst nahen

Eckpunkten des Bodentrapezes liegen. Durch die relative Lage dieser Punkte zueinander kann besonders gut die Neigung der Bodenebene um die z-Achse erfasst werden. Ein dritter Punkt $\mathbf{p}_3$ wird mittig mit größerem Abstand zum Sensorzentrum gewählt, um die Neigung um die x-Achse zu bestimmen. Die Ebene die durch diese Punkte definiert wird approximiert die Lage des aufgenommen Bodens innerhalb der Punktwolke.

Berechnung der Bodenebene Auf Basis der drei Bodenpunkte $\mathbf{p}_1, \mathbf{p}_2$ und $\mathbf{p}_3$ kann die Normale $\mathbf{n}$ der Bodenebene bestimmt werden. Mit Hilfe dieser Normalen kann die Rotation der Ebene um die drei Achsen (x, y, z) berechnet werden. Die Normale wird mit

$$\mathbf{n} = \frac{(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)}{|(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)|} \tag{4.4}$$

berechnet. Die Normale $\mathbf{n}$ steht somit senkrecht auf den beiden Vektoren in der Ebene $\mathbf{p}_2 - \mathbf{p}_1$ und $\mathbf{p}_3 - \mathbf{p}_1$ und ist normiert. Zusammen mit dem Normalenvektor kann die Ebene wie folgt mit der Koordinatenform dargestellt werden:

$$\mathbf{x}^T \cdot \mathbf{n} = \begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \cdot \begin{pmatrix} n_1 \\ n_2 \\ n_3 \end{pmatrix} = \mathbf{b}. \tag{4.5}$$

Der Vektor $\mathbf{x}$ repräsentieren einen beliebigen Punkt auf der Ebene mit Normalenvektor $\mathbf{n}$. Für einen gegeben Vektor $\mathbf{n}$ und einen gegebenes $\mathbf{b}$ liegt ein Punkt $\mathbf{x}$ genau dann in der Ebene wenn (4.5) erfüllt ist.

Entfernung der Bodenpunkte Um die entsprechenden Bodenpunkte aus der Punktwolke entfernen zu können wird die berechnete Ebene durch Addition um den Vektor $(0, \varepsilon, 0)$ mit $\varepsilon \in \mathbb{R}^+$ verschoben. Durch diese Translation wird die Ebene leicht entlang der y-Achse angehoben. Im Idealfall wir die Ebene dabei knapp oberhalb einer möglichst großen Menge an Bodenpunkten gehoben. Anschließend werden alle Punkte unterhalb dieser Ebene aus der Punktwolke entfernt. Für die Implementierung des Prototyps wurde für ε ein Wert von 3 cm ($\varepsilon = 0,03$) ermittelt (s. Abschnitt 8.1.3).

Um den Abstand eines Punktes $\mathbf{x}$ zu einer Ebene bestimmen zu können wir die Bodenebene in (4.5) in die Hesse-Normalenform (HNF) umgewandelt wird. Die Darstellung in HNF

$$\frac{n_1 \cdot x_1 + n_2 \cdot x_2 + n_3 \cdot x_3 - b}{|\mathbf{n}|} = d \tag{4.6}$$

gewinnt man, indem (4.5) durch $|\mathbf{n}|$ normiert wird. Der Vorteil dieser Darstellung ist, dass man durch die Normierung der Ebene den Abstand d des Punktes $\mathbf{x}$ zur Ebene in Richtung

des Normalenvektors erhält. Unter der Voraussetzung, dass der Normalenvektor $\mathbf{n}$ immer in Richtung wachsender y-Koordinaten zeigt, gilt für d der folgende Zusammenhang:

$$\text{Lage von } \mathbf{x} \text{ relativ zur Ebene} = \begin{cases} \text{oberhalb}, & \text{wenn } d < 0 \\ \text{innerhalb}, & \text{wenn } d = 0 \\ \text{unterhalb}, & \text{wenn } d > 0 \end{cases}. \tag{4.7}$$

Daraus folgt, dass ein Punkt $\mathbf{x}$ genau dann unterhalb einer um 3 cm nach oben verschobenen Ebene liegt, wenn gilt $d > \varepsilon$. Alle Punkte die diese Bedingung erfüllen werden aus der Punktwolke entfernt.

Zusammenfassung Der hier vorgestellte Algorithmus stellt eine einfache Möglichkeit dar die Bodenpunkte aus der Punktwolke zu entfernen. Durch die starke Optimierung an die Eigenschaften der Punktwolke kann dieser Algorithmus einen Laufzeitvorteil gegenüber dem RANSAC-basierten Verfahren erzielen (s. Abschnitt 9.2.3). Dieser Algorithmus ist in der Pipeline, zusätzlich zur RANSAC-basierten Bodenentfernung, in Form einer alternativen Verarbeitungsstufe implementiert worden (s. Abschnitt 4.4.2).

4.4.3. Voxelgrid Downsampling

Zur Homogenisierung der Dichte wird die Punktwolke P zunächst durch ein dreidimensionales regelmäßiges Gitter G partitioniert (s. Abbildung 4.6). Alle Punkte in P werden durch das Gitter genau einer Zelle Z_i zugeordnet:

$$G = \bigcup Z_i = P. \tag{4.8}$$

Anschließend wird für jede Zelle Z_i ein repräsentativer Punkt $\tilde{\mathbf{z}}_i$ berechnet, indem die Koordinaten aller Punkte $\mathbf{p}$ der selben Zelle gemittelt werden:

$$\tilde{\mathbf{z}}_i = \frac{\sum_{\mathbf{p} \in Z_i} \mathbf{p}}{|Z_i|}. \tag{4.9}$$

Neben der Mittelwertbildung wäre auch die Auswahl des Medianwertes möglich. Diese hat jedoch gegenüber der Mittelwertbildung einen Laufzeitnachteil.

Auf diese Weise wird jede Zelle durch genau einen Punkt $\tilde{\mathbf{z}}_i$ repräsentiert, diese so gewonnen Punkte bilden zusammen die homogenisierte Punktwolke P':

$$P' = \{\tilde{\mathbf{z}}_0, \ldots, \tilde{\mathbf{z}}_i\}. \tag{4.10}$$

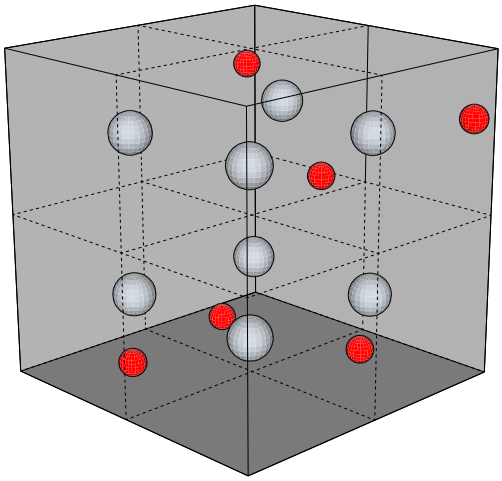


Abbildung 4.6.: Die Abbildung zeigt ein dreidimensionales Raster mit gleichförmigen Quadergrößen. In Rot dargestellt sind die beispielhaften Punkte der Punktwolke P. Diese würden während der Homogenisierung der Dichte für jede Quader-förmige Zelle gemittelt werden (symbolisch dargestellt durch die hellgrauen Kugeln in den Quader-Zentren).

Zusammenfassung Durch die Vorverarbeitung durch das Voxelgrids wird die Punktwolke durch lokale Mittelwertbildung in den Zellen des Gitters quantitativ reduziert, indem alle Punkte einer nicht leeren Zelle durch einen gemittelten Punkt ersetzt werden. Durch die Reduktion der Anzahl der Punkte ergibt sich neben einem Laufzeitvorteil für alle nachfolgenden Schritte der Pipeline, auch eine qualitative Verbesserung. Durch die Mittelwertbildung sind die Punkte in der resultierende Punktwolke gleichmäßiger Verteilt, was sich günstig auf den nachfolgenden Clustering-Schritt in der Pipeline (s. Abschnitt 4.5.1) auswirkt.

4.5. Objekterkennung

Nach der erfolgten Vorverarbeitung gilt es einzelne Objektstrukturen aus der Punktwolke zu isolieren und vorzugsweise zu klassifizieren. Den ersten Schritt bildet hierzu die Extraktion von so genannten Clustern unter Verwendung des euklidischen Abstandsmaßes. Anschließend werden für jedes Cluster die Normalenvektoren bezüglich des Sensorzentrums mit Hilfe adjazenter Punkte berechnet. Hierbei ist es wichtig den Radius der Umkugel zur Bestimmung der adjazenten Punkte geeignet zu wählen. Da die Dichte der Punkte der Punktwolke jedoch nicht konstant bezüglich der Tiefe ist, wurde der Radius der Umkugel mit Hilfe einer quadratischen Funktion $(0{,}015432 \cdot x^2)$ modelliert. Aus den Cluster-Normalen lässt sich für jedes Cluster eine Signatur berechnen. Über einen Nächster-Nachbar-Ansatz ist es möglich Cluster anhand ihrer Signaturen bekannten Signaturen zuzuordnen und sie auf diese Weise zu klassifizieren.

4.5.1. Clustering

Ziel des Clustering ist es jeden Punkt p aus einer Punktwolke P einem Cluster $C_i \subseteq P$ zuzuordnen, so dass im Idealfall jedes Objekt der Szene durch ein separates Cluster beschrieben wird und durch den anschließende Klassifikationsschritt (s. Abschnitt 4.5.2) separat klassifiziert werden kann.

Euklidisches Clustering Das hier verwendete euklidische Clustering von Rusu [Rus09] gehört zu der Klasse der hierarchisch agglomerativen Verfahren und nutzt den euklidischen Abstand

$$d(\mathbf{p}, \mathbf{q}) = |\mathbf{p} - \mathbf{q}| = \sqrt{(\mathbf{p}_x - \mathbf{q}_x)^2 + (\mathbf{p}_y - \mathbf{q}_y)^2 + (\mathbf{p}_z - \mathbf{q}_z)^2} \tag{4.11}$$

zweier Punkte $\mathbf{p}$ und $\mathbf{q}$ als Distanzmetrik. Mit dessen Hilfe wird nun für jeden Punkt p eine Menge von direkten Nachbarknoten $\mathbf{q}_i$ bestimmt für die gilt: $d(\mathbf{p}, \mathbf{q}_i) \leq r$, mit $r \in [0, \infty)$. In Abbildung 4.7(a) wird dies für einen Punkt p mit der Nachbarschaft $\mathbf{q}_0, \mathbf{q}_1, \mathbf{q}_2$ und $\mathbf{q}_3$ veranschaulicht. Für jeden Punkt in dieser Nachbarschaft wird nun jeweils erneut dessen Nachbarschaft bestimmt. Ein Cluster setzt sich dann, wie in Abbildung 4.7(a) gezeigt, aus der Vereinigung dieser zusammenhängenden Nachbarschaften zusammen.

Nach [Rus09] repräsentieren $C_i = \{\mathbf{p}_i \in P\}$ und $C_j = \{\mathbf{p}_j \in P\}$ bei einem gegebenen Radius r zwei separate Cluster falls gilt:

$$\forall ij : d(\mathbf{p}_i, \mathbf{q}_j) > r. \tag{4.12}$$

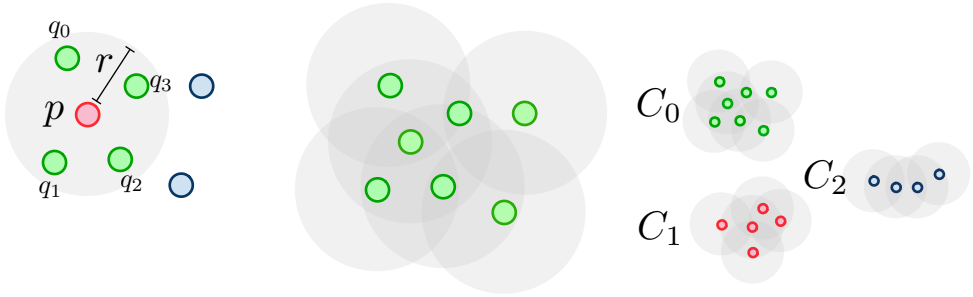
Die Anzahl der Cluster in die eine Punktwolke zerfällt ist somit maßgeblich vom Radius r abhängig. Für r gilt der folgende Zusammenhang:

$$\text{Anzahl Cluster } = \begin{cases} |P|\,, & \text{falls } r < \min(d(\mathbf{p}, \mathbf{q})) \\ x \in \{1, \ldots, |P|\}, & \text{falls } r \in [\min(d(\mathbf{p}, \mathbf{q})), \max(d(\mathbf{p}, \mathbf{q}))]\,. \\ 1, & \text{falls } r > \max(d(\mathbf{p}, \mathbf{q})) \end{cases} \tag{4.13}$$

Der erste sowie der dritte Fall bilden die Extremfälle. Im ersten Fall ist r so klein gewählt, dass jeder Punkt für sich einen Cluster bildet, da keine der Nachbarschaften einen weiteren Punkt enthält. Auf der Basis von einelementigen Clustern kann keine Klassifikation durchgeführt werden.

Im letzten Falls ist r so groß gewählt, dass die gesamte Punktwolke einen großen Cluster bildet. Die Klassifikation dieses großen Clusters entspricht der Klassifikation der Punktwolke als Ganzes. In diesem Fall wäre es z. B. nicht möglich zwei Fahrzeuge nebeneinander zu erkennen. Dies ist somit ebenfalls ein ungünstiges Szenario.

Wird r entsprechend der Bedingung in Fall zwei gewählt, ist es möglich dass die Punktwolke in mehrere Cluster zerfällt (s. Abbdilung 4.7(c)). Es gilt, je größer r desto größer die Cluster. Für die prototypische Implementierung wurde empirisch $r = 0{,}1\,\text{m}$ ermittelt (s. Abschnitt 4.5.1).



(a) Nachbarschaft (grün) eines Punktes **p** (rot) mit Radius r.

(b) Ein einzelnes Cluster.

(c) Nach dem Clustering zerfällt die Punktwolke in einzelne Cluster C_0, C_1 und C_2.

Abbildung 4.7.: Euklidisches Clustering nach [Rus09] (vereinfachte zweidimensionale Illustration)

Zusammenfassung Das euklidische Clustering nach [Rus09] entspricht einem hierarchisch agglomerativen Verfahren. Hierbei werden Cluster ausgehend von einem initialen Punkt schrittweise vergrößert. Als Distanzmetrik wird der euklidische Abstand verwendet, die Adaption bezüglich anderer Abstandsmetriken wären aber ebenfalls problemlos möglich. Ein Vorteil der gezeigten agglomerativen Vorgehensweise besteht darin, dass die Anzahl der vorhandenen Cluster nicht initial bekannt sein muss. Außerdem können sehr kleine Cluster bereits an dieser Stelle verworfen werden, was die Datenmenge für die nachfolgenden Pipelineschritte reduziert. Ein Nachteil dieser Vorgehensweise liegt jedoch in der Gier begründet, die statische, später nicht mehr revidierte Zuordnungen von Punkten **p** zu Clustern C_i trifft.

4.5.2. Klassifikation

Im Klassifikationsschritt erfolgt die Annotation der zuvor bestimmten Cluster. Hierzu wurden im Rahmen der Projektgruppe unterschiedliche Ansätze untersucht und bezüglich ihrer Einsatzfähigkeit in einer Echtzeitumgebung analysiert. Als sehr effizient und robust bezüglich verrauschter Messdaten hat sich hierbei eine Vorgehensweise nach Rusu [Rus09, S. 284] erwiesen. Hierbei erfolgt initial die Bestimmung von Cluster-Signaturen aus den jeweiligen Normalenvektoren. Eine Signatur wird hierbei durch einen 308-dimensionalen Vektor beschrieben (siehe Kapitel 4.5.2). Aus bekannten annotierten Signaturen kann eine Signatur-Datenbank aufgebaut werden. Zur Klassifikation einer Signatur genügt es den nächsten Nachbarn bezüglich der Signatur-Datenbank (z. B. in Form eines k-d-Baumes) zu bestimmen. Je nach Distanz zwischen der gegebenen Signatur und dem nächsten Nachbarn kann eine Klassifikation erfolgen oder aber auch das korrespondierende Cluster als unbekannt verworfen werden.

Viewpoint Feature Histogram

Bei Viewpoint Feature Histograms (VFH) handelt es sich um Deskriptoren für Punktwolken welche auf (Fast) Point Feature Histograms ((F)PFH) basieren. PFH Deskriptoren besitzen im Gegensatz zu den bekannten geometrischen Eigenschaften von Punktwolken wie Normalen und Krümmung einen wesentlich größeren Informationsgehalt. Dieser höhere Informationsgehalt wird erreicht indem die geometrischen Eigenschaften der umgebenden Punkte (k-Nachbarschaft) mit in den Deskriptor enkodiert werden.

Zu diesem Zweck wird für jedes Punktpaar $(\mathbf{p}_i, \mathbf{p}_j)$ und deren geschätzten Normalen $(\mathbf{n}_i, \mathbf{n}_j)$ die Abweichung der Roll-Nick-Gier-Winkel durch folgende Formeln abgeschätzt.

$$\begin{aligned}\alpha &= v \cdot \mathbf{n}_j \\ \phi &= u \cdot \frac{(\mathbf{p}_j - \mathbf{p}_i)}{d} \\ \theta &= \arctan(w \cdot \mathbf{n}_j, u \cdot \mathbf{n}_j)\end{aligned}$$

Wobei u, v, w ein Darboux-Frame-Koordinatensystem über $\mathbf{p}_i$ aufspannen.

$$\begin{aligned}u &= \mathbf{n}_i \\ v &= (\mathbf{p}_j - \mathbf{p}_i) \times u \\ w &= u \times v\end{aligned}$$

Das Histogramm berechnet sich dann aus den Werten von (α, ϕ, θ) für alle Punktpaare. (vgl. [RBB09]) Abbildung 4.8 zeigt exemplarisch den Einflussbereich eines PFH-Deskriptors für einen Ausgangspunkt $\mathbf{p}_q$, welcher in rot dargestellte ist. Dieser Punkt induziert einen Umkreis mit Radius r in dem die nächsten Nachbarn $\mathbf{p}_{k1}$ bis $\mathbf{p}_{k5}$ ermittelt werden. Der PFH-Deskriptor wird dann über dem, durch die Punkte gegebenen, vollständigen Graphen berechnet. In der Erweiterung auf ein dreidimensionalen Szenario wird der Umkreis einfach durch eine Umkugel mit Radius r ersetzt, das weitere vorgehen bleibt identisch. Für eine detaillierte Betrachtung der methodischen Grundlage des PFH-Deskriptors sei an dieser Stelle auf die entsprechenden Veröffentlichungen [Rusa] und [Rusb] verwiesen. Bei dem FPFH-Deskriptor handelt es sich methodisch um ein sehr ähnliches Verfahren, mit einem stark optimierten Berechnungskonzept, um den Einsatz in Echtzeitumgebungen zu ermöglichen, siehe dazu [RBB09].

Erweitert man diese zuvor dargelegten Verfahren um eine zusätzliche Komponente, dem sogenanten *Viewpoint*, erhält man die in der Projektgruppe verwendeten VFH Deskriptoren. Die in Abbildung 4.9 exemplarisch gegenüber gestellten Histogramme zeigen die Deskriptoren für einen Stapel von Kisten (links) und ein Fahrzeug (rechts). Die dargestellten Deskriptoren setzen sich Strukturell aus zwei Teilen zusammen. Die ersten 128 der 308 bins enkodieren die Viewpoint Informationen die restlichen 180 $(60 * 3)$ Bins ergeben sich aus den drei Komponenten (α, ϕ, θ) des FPFH Deskriptors welcher derart modifiziert wurde, dass sie nicht mehr die globale Abweichung bestimmen, sondern die relative Abweichung gegenüber der Normalen des Zentroiden der Punktwolke.

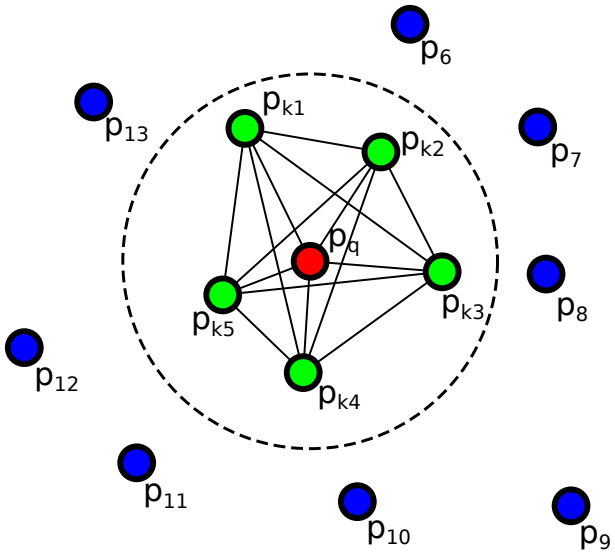
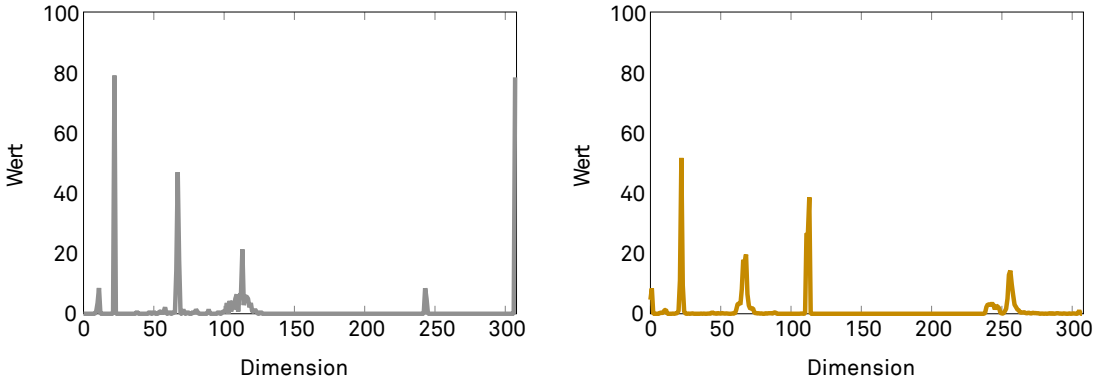


Abbildung 4.8.: Veranschaulichung des Einflussbereichs eines PFH-Deskriptors vgl. [RBB09]



(a) Signatur Kistenstapel

(b) Signatur Fahrzeug

Abbildung 4.9.: 308 Dimensionale VFH-Signaturen von zwei Objekten

Die Viewpoint Komponente ergibt sich dann aus den Winkeln zwischen der zentralen Viewpoint Richtung und den Normalen jedes Punktes.

Nächster Nachbar Klassifikation

Die nächster Nachbar Klassifikation bezüglich einer Signatur-Datenbank kann prinzipiell über eine gewöhnliche Suchdatenstruktur wie z. B. einen k-d-Baum erfolgen. Suchdatenstrukturen bieten gegenüber der linearen Suche im Allgemeinen einen signifikanten Geschwindigkeitsvorteil. Problematisch ist hierbei jedoch, dass Suchdatenstrukturen mit zunehmender Dimensionalität degenerieren und ihre Effizienz sogar schlechter als die von der linearen Suche sein kann – dieses Phänomen ist auch unter dem Begriff des Fluchs der Dimensionen bekannt. Da die VFH-Signaturen eine hohe Dimensionalität aufweisen aber dennoch eine lineare Suche aus Effizienz-Gründen nicht sinnvoll erscheint, wurde eine approximativer Ansatz für die Klassifikation gewählt. Hierbei wurde der Ansatz von Muja und Lowe [ML09] aufgegriffen. Hierbei werden je nach Testdatensatz entweder randomisierte kd-Bäume oder hierarchische k-Mittelwert-Bäume verwendet. Die Parameter zur gewählten Datenstruktur werden hierbei über einen Nelder-Mead Downhill Simplex Optimierungsalgorithmus verfeinert, so dass Datenstruktur und Parameter bezüglich des Testdatensatzes spezialisiert sind. Im Falle der Signatur-Datenbank entspricht der vollständige Signatur-Satz dem Testdatensatz. Eine Optimierung der Datenstruktur muss nur bei der Initialisierung erfolgen.

Erkennung salienter Stationsmerkmale

Im Rahmen der Verwendung in der Intralogistik muss das Fahrzeug in Stationen und Regale einfahren können. Aufgrund von Ungenauigkeiten in der Odometrie, ist es nicht möglich die Einfahrt allein mit den Odometrie-Daten anzufahren.

Nach Stand der Technik werden zur Verbesserung der Positioniergenauigkeit Laserscanner verwendet, die die Position des Fahrzeugs anhand angebrachter Reflexmarken bestimmen. Die Sensordaten auf Odometrie und Laserscanner werden mit einem Partikelfilter fusioniert.

Im Rahmen der Projektgruppe wurden zwei Ansätze entwickelt, um den Laserscanner durch die Verwendung eines Tiefensensors zu ersetzen. Ziel ist die erkannte Position des Fahrzeugs anhand eines, als Station klassifizierten, Clusters zu verbessern. Ein Ansatz basiert darauf, die Reflexmarken durch Stangen zu ersetzen, die vom Tiefensensor erkannt werden können und über Triangulation die Position der Fahrzeugs bestimmt werden kann. Als weiterer Ansatz wurde die Erkennung der Station durch Ellipsen-Fitting untersucht.

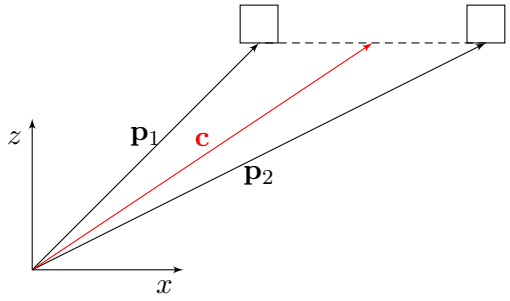


Abbildung 4.10.: Schematische Darstellung der berechneten Schwerpunkte $\mathbf{p}_1$ und $\mathbf{p}_2$ sowie des berechneten markanten Punkts $\mathbf{c}$

Erkennung anhand von salienten Strukturen Ein Ansatz die Position vor einer Station zu bestimmen besteht darin, zwei Stangen an beiden Seiten der Stationseinfahrt zu befestigen. Die Position dieser Stangen kann mittels Tiefensensor bestimmt werden und daraus die Position in Relation zur Station berechnet werden.

Für einen bereits klassifizierten Cluster, berechnet der Algorithmus einen markanten Punkt an der Station. Dieser befindet sich mittig zwischen den Schwerpunkten der beiden Stangen (siehe Abbildung 4.10). Der Algorithmus arbeitet dabei in den folgenden Schritten:

1. Extraktion aller Punkte, die zu den Stangen gehören
2. Segmentierung der Punkte durch Clustering
3. Berechnung der Cluster-Schwerpunkte
4. Berechnung des markanten Punkts aus den Schwerpunkten

Für die Extraktion aller zu Stangen gehören Punkte kann ausgenutzt werden, dass die Stangen über den oberen Rand der Station hinaus hervorstehen. Diese Punktmenge $\overline{P}$ kann durch Clipping aller Punkte unterhalb der Stationshöhe h_{min} sowie oberhalb einer Maximalhöhe h_{max} berechnet werden. Hierbei beziehen sich die h_{min} und h_{max} aus dem Abstand zur Bodenebene (siehe Kapitel 4.4.2). Für jeden Punkt in $\overline{P}$ mit Abstand d zum Boden, gilt $h_{\text{min}} < d < h_{\text{max}}$

Auf der Punktmenge $\overline{P}$ wird ein Clustering durchgeführt. Ergebnis dieses Schritts ist eine Segmentierung der Punktmenge bezüglich der Stangen. Waren im ursprünglich Cluster Punkte beider Stangen enthalten, so ergeben sich zwei neue Cluster. Ist dies nicht der Fall, bricht der Algorithmus ab.

Für die beiden bestimmten Cluster werden die Schwerpunkte $\mathbf{p}_1$ und $\mathbf{p}_2$ berechnet. Um Fehler bei den vorhergehenden Schritten auszuschließen, wird geprüft ob der Abstand der Schwerpunkte dem erwarteten Wert entspricht. Der markante Punkt $\mathbf{c}$ berechnet durch das arithmetische Mittel von $\mathbf{p}_1$ und $\mathbf{p}_2$.

Erkennung anhand von Ellipsen-Fitting Im Rahmen der Projektgruppe wurde zusätzlich zu der im vorherigen Abschnitt 4.5.2 erläuterten Methodik eine Erkennung von Stationen und deren Ausrichtung durch ihr elliptische Form in Betracht gezogen.

Zu diesem Zweck wurde zunächst die dreidimensionale Punktwolke auf eine zweidimensionale Punktwolke herunter projiziert. Bei diesem Vorgehen wurde aus Performancegründe die xz-Ebene als Projektionsebene genutzt da deren Abweichung zur tatsächlichen Bodenebene gering war. Auf die so erhaltenen Daten wurde ein Ellipsen Fitting nach Fitzgibbon und Pilu [FPF96] angewendet um die Form und somit die Ausrichtung der Station zu erkennen. Da dieses Verfahren ohne weitere Optimierung der Genauigkeit der schon bestehenden Methode unterlegen war wurde dieser Ansatz in der Projektgruppe nicht weiter evaluiert.

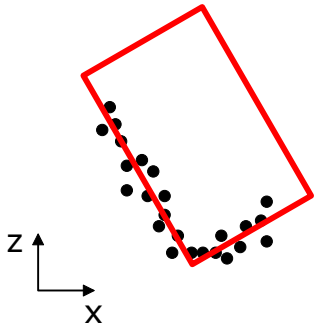
Verfeinerung der Initialpose

Sobald ein Cluster der Punktwolke als Fahrzeug klassifiziert wird, so ist es nötig, die Pose, welche von diesem Fahrzeug beschrieben wird, zu verfeinern. Zunächst sind im Cluster nicht das gesamte Fahrzeug, sondern nur maximal zwei Seiten abgebildet, weswegen die restlichen Größen wie beispielsweise der Fahrzeugmittelpunkt oder der Rotationswinkel berechnet werden sollen. Dazu führt der Algorithmus zur Verfeinerung der Initialpose folgende Schritte durch:

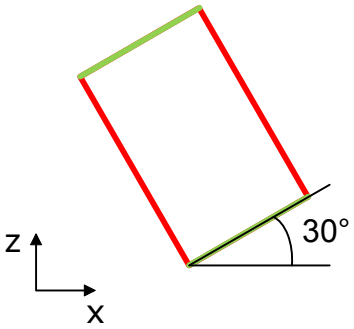
1. Projektion der Clusterpunkte auf die Bodenebene
2. RANSAC-Segmentierung
3. Berechnung eines Rechtecks um das Fahrzeug
4. Berechnung des Rotationswinkels des Fahrzeugs

Um die Ausrichtung des Fahrzeuges korrekt im Bezug auf die gefundene Bodenebene berechnen zu können, werden zunächst alle Punkte des Fahrzeug-Clusters auf diese projiziert.

Um unnötige Redundanz zu vermeiden und die weitere Berechnung zu beschleunigen, wird das Cluster im nächsten Schritt auf zwei oder vier Punkte reduziert (in Abhängigkeit der Anzahl der gefundenen Fahrzeugseiten), welche die erkannten Seiten des Fahrzeuges beschreiben. Dazu wird die Segmentierung solange pro Seite durchgeführt, bis die beiden Punkte mit maximalem Abstand gefunden werden, die übrigen Punkte werden entfernt. In Abhängigkeit der Anzahl der gefundenen Punkte lassen sich nun eine oder zwei Geraden aufstellen, welche die Seiten des erkannten Fahrzeuges annähern. Durch die begrenzte Reichweite des Tiefensensors besteht die Möglichkeit, dass die Seiten des Fahrzeuges nicht in voller Länge erkannt werden. Um diesem Problem entgegenzuwirken, werden die Seiten im nächsten Schritt auf die reale Seitenlänge des Fahrzeugs verlängert. Dazu wird die Länge der Geraden ermittelt und durch ein simples Thresholding-Verfahren entschieden, ob es sich um eine der Seiten des Fahrzeuges (die längere Seite) oder um



(a) Berechnetes Rechteck um die Clusterpunktwolke vom Fahrzeug.



(b) Berechnung der Rotationswinkel vom Fahrzeug.

Abbildung 4.11.: Poseverfeinerung von einem Fahrzeug anhand projezierter Punktwolke

die Front bzw. das Heck handelt (die kürzere Seite) und dementsprechend in z-Richtung verlängert (s. Abbdilung 4.11(a)).

Um die Pose des klassifizierten Fahrzeuges letztendlich berechnen zu können, genügt es, das Fahrzeug anhand der im vorherigen Schritt berechneten Seiten als Rechteck vereinfacht zu beschreiben. Daher besteht der nächste Schritt daraus, anhand der erhaltenen Seiten das zugehörige Rechteck zu berechnen, indem Senkrecht zu diesen neue Seiten mit den entsprechenden Fahrzeugmaßen konstruiert werden. Das so erhaltene Rechteck beschreibt nun, vereinfacht, das erkannte Fahrzeug. Dieses kann nun dazu verwendet werden um beispielsweise, anhand des Mittelpunktes, die Entfernung zur Kamera zu ermitteln.

Im letzten Schritt des Algorithmus wird noch der Rotationswinkel des Fahrzeuges bezüglich zur Kamera berechnet. Da die Klassifikation diesen bereits erkennt, allerdings meist sehr ungenau mit einer Abweichung von oftmals mehr als 20°, gilt es, diese Erkennung zu verbessern. Anhand der Lage des Rechtecks im Raum lässt sich auch der Rotationswinkel des Fahrzeuges berechnen, allerdings ist dieser nicht eindeutig. Bei dem Rechteck wird nicht zwischen Fahrzeugfront oder Heck unterschieden (s. Abbildung 4.11(b)) und somit gibt es bei einem Rotationswinkel von 30° zum Beispiel die Möglichkeit, dass das Fahrzeug in der Realität aber einen Rotationswinkel von 210° aufweist (Fahrzeug um 180° gedreht). Um dieses Problem zu lösen, werden beide Verfahren kombiniert, der gefundene Winkel der Klassifikation wird dazu benutzt, um den berechneten Rotationswinkel des Rechtecks eindeutig bestimmen zu können und somit die letztendliche Pose des Fahrzeuges zu erhalten.

5. Bahnplanung

Die Bahnplanung beschreibt die Wegfindung der einzelnen Fahrzeuge. Sie beinhaltet das Anfahren eines Zielpunktes ausgehend von einem Startpunkt und eine Kollisionsvermeidung mit anderen Fahrzeugen oder Objekten. Das Anfahren von einem Punkt A (z. B. ein Lager) zu einem Punkt B (z. B. eine Kommissionierstation) wird in der globalen Bahnplanung abgehandelt. Das Vermeiden von Kollisionen zwischen Fahrzeugen oder einem sich auf der Fahrbahn befindlichen Objekt wird von der lokalen Bahnplanung realisiert.

In diesem Kapitel wird zum einen ein Überblick über verschiedene Lösungsansätze in Abschnitt 5.2 gegeben und zum anderen wird detailliert auf die zwei, von der Projektgruppe realisierten, Ansätze Potentialfelder, Abschnitt 5.3, und Vector Filed Histogram, Abschnitt 5.4, eingegangen.

5.1. Problemstellung

Die Bahnplanung soll sich im Rahmen der Projektgruppe auf eine lokale Bahnplanung beschränken. Eine Kollisionsmöglichkeit mit sich in Fahrtrichtung befindlichen Objekten soll ab einem Abstand von 4 m zum Fahrzeug erkannt werden. Die lokale Bahnplanung soll in jedem Fall eine Kollision vermeiden und das Fahrzeug im Notfall stoppen. Die Erkennung einer bevorstehenden Kollision soll über die im Fahrzeug integrierte Sensorik erfolgen. Die Schwierigkeit hierbei ist vor allem die Wegfindung in dynamischen Umgebungen, wenn sich z. B. viele Fahrzeuge auf engem Raum bewegen. Neben der geforderten Kollisionsvermeidung sollten Ausweichmanöver im besten Fall wenig Zeit in Anspruch nehmen und der neu berechnete Weg sollte zudem möglichst kurz sein.

5.2. Lösungsmöglichkeiten

Für die Bahnplanung existiert eine Reihe an Lösungswege, welche sich hinsichtlich der gewählten Repräsentation der Umgebung des Fahrzeuges unterscheiden. So kann zwischen kartenbasierten, zellbasierten und Methoden ohne Kartierung der Umgebung differenziert werden.

5.2.1. Kartenbasierte Methoden

Der Ansatz hinter den kartenbasierten Verfahren besteht darin, den Teil der Umgebung, der keine Kollisionen mit dem Fahrzeug beinhaltet, als Graphen zu repräsentieren. Hierzu werden kollisionsfreie Punkte in der Umgebung als Knoten des Graphen dargestellt und eine Kante repräsentiert den kollisionsfreien Weg zwischen zwei Positionen. Den Kanten können jeweils noch Gewichte zugeordnet werden, die zum Beispiel den Abstand der beiden Positionen repräsentieren können [Sch01]. Definiert man nun den Start- und Endpunkt, so muss nur noch durch eine geeignete Graphensuche der kürzeste Weg oder der Weg mit den geringsten Kosten gefunden werden. Vertreter der kartenbasierten Methoden sind der Sichtbarkeitsgraph und die Voronoi Diagramme.

5.2.2. Zellbasierte Methoden

Die zellbasierten Verfahren unterteilen die Umgebung in verschiedene Teilbereiche, die Zellen genannt werden. Diese können als frei, gemischt oder voll bezeichnet werden. Hierbei steht frei für jene Zellen, die kein Hindernis enthalten. Zellen, die nur aus einem Hindernis bestehen, werden voll genannt und jene, die zum Teil Hindernisse und zum Teil Freiräume enthalten, werden als gemischt bezeichnet [Sch01]. Neben den Zellen werden zusätzliche Informationen über die Nachbarschaften der Zellen gespeichert. Unterteilen lassen sich die zellenbasierten Verfahren in exakte und angenäherte Verfahren. Die exakten zellenbasierten Verfahren verwenden Zellen, die nur frei oder voll sein können, variieren dafür aber in Größe und Form der Zellen [Reg08]. Hingegen wird bei den angenäherten zellenbasierten Verfahren zwischen freien, vollen und gemischten Zellen unterschieden. Auch kann die Größe der Zellen veränderlich sein. Lediglich die Form der Zellen ist gleich für alle Zellen in der Umgebung [Reg08]. Im Gegensatz zu den kartenbasierten Verfahren wird bei den zellenbasierten Verfahren das Problem der Bahnplanung für die gesamte Umgebung in Teilprobleme für die einzelnen Zellen zerlegt. Dies hat den Vorteil, dass der Graphenalgorithmus zunächst auf die Zellen und dann innerhalb der Zellen angewandt wird.

5.2.3. Methoden ohne Kartierung

Die kartenbasierten und zellbasierten Verfahren müssen zunächst eine Karte der Umgebung erstellen, um dann Suchalgorithmen für Graphen wie zum Beispiel Dijkstra oder A* zu verwenden. Eine solche Kartierung kostet Speicher und ist nicht in jeder Umgebung möglich. Da zunächst das Fahrzeug seine Umgebungskarte selbst erstellen muss. Zu den Methoden ohne Kartierung gehört die Potentialfeldmethode 5.3. Der Vorteil der Bahnplanung auf Basis eines Potentialfeldes besteht in der besseren Eignung bei dynamischen Änderungen. Andere Fahrzeuge und Hindernisse können ihre Position verändern, wodurch eine Neuberechnung des Weges erforderlich sein kann. Bei auf Kartierung oder Karten basierenden Methoden äußert sich dies in einer Neuberechnung der jeweiligen

Speicherstruktur und deren anschließende Auswertung. Dies kann bei häufig auftretenden Änderungen in der Umgebung des Fahrzeuges einen erhöhten Rechenaufwand bedeuten. Da bei der Potentialfeldmethode keine Kartierung oder Karte genutzt wird und stets nur Objekte bzw. Potentiale in lokaler Umgebung zur Berechnung genutzt werden, wird der Rechenaufwand durch Veränderungen im Potentialfeld nicht vergrößert und ist somit für dynamische Umgebungen besser geeignet. Ebenfalls eins für dynamische Umgebungen geeignete Verfahren stellt die Vector Field Histogram Methode 5.4 dar. Bei diesem Verfahren wird dynamisch die Dichte an Hindernissen in der Umgebung des Fahrzeuges gemessen und dynamisch ein Weg durch die Hindernisse bestimmt. Sowohl die Potentialfeld als auch die Vector Field Histogram Methode werden im folgenden beschrieben, da beide Ansätze zur Lösung des Bahnplanungsproblem im Rahmen der Projektgruppe betrachtet wurden.

5.3. Potentialfelder

Ein Potentialfeld ist ein Gradientenfeld, welches aus einem Vektorfeld erzeugt werden kann, indem dieses nach dem Ort abgeleitet wird. Die einzelnen Vektoren des Potentialfeldes zeigen somit in die Richtung in der das Potential am größten bzw. kleinsten ist. Darüber hinaus gibt es sowohl anziehende als auch abstoßende Potentiale. Diese Eigenschaft kann im Bereich der Bahnplanung eingesetzt werden, indem dem zu erreichenden Zielpunkt, eine definierte Position im Raum oder in der Fläche, beispielsweise das größte positive Potential zugewiesen wird. Das Fahrzeug, welches sich an einem beliebigen Startpunkt befindet, wird sich dem Zielpunkt nähern, wenn es sich entlang der Vektoren bewegt. Hindernisse können in diesem Fall durch negative Potentiale beschrieben werden. Je größer das negative Potential ist, desto mehr wird das Fahrzeug von ihnen abgestoßen und umso größer ist der Abstand mit dem das Fahrzeug das Hindernis umfährt. Es gilt demnach abzuwägen wie groß das Potential des Zielpunktes und die einzelnen Potentiale der Hindernisse gewählt werden müssen, damit das Fahrzeug am Zielpunkt ankommt und dabei weder zu große Umwege zurücklegt noch mit Hindernissen kollidiert.

Durch ein Potentialfeld können folglich sowohl die geforderte lokale Bahnplanung, das Umfahren von Hindernissen, als auch die globale Bahnplanung, das Anfahren eines Zielpunktes, ohne Mehraufwand durch dieses eine Verfahren realisiert werden. Dem im Rahmen der Projektgruppe umgesetzten Potentialfeld liegt die Veröffentlichung von Ge et al. [GC02] zugrunde. Ge et al. stellen dort ein Potentialfeld für Bahnplanungsszenarien vor, in welchem die anziehenden und abstoßenden Kräfte nicht nur auf Basis ihrer relativen Entfernungen zueinander, sondern auch basierend auch ihren Geschwindigkeiten berechnet werden können. Es werden demnach nicht nur statische, sondern auch dynamische Hindernisse und Ziele berücksichtigt. Das Anfahren eines bewegten Zieles hat für die Projektgruppe jedoch derzeit keine Relevanz, es kann jedoch bei Bedarf durch das Setzen der Geschwindigkeit des Zieles in der realisierten Bahnplanung aktiviert werden. Das anziehende Potential wurde in der Bahnplanung der Projektgruppe nach Ge et al. [GC02]

folgendermaßen definiert:

$$U_{\text{att}}(\mathbf{p}, \mathbf{v}) = \alpha_p \|\mathbf{p}_{\text{tar}}(t) - \mathbf{p}(t)\|^m + \alpha_v \|\mathbf{v}_{\text{tar}}(t) - \mathbf{v}(t)\|^n \tag{5.1}$$

$\mathbf{p}_{\text{tar}}(t)$ und $\mathbf{p}(t)$ definieren je die Position des Zieles bzw. des Fahrzeugs zu einem Zeitpunkt t und stellen im zweidimensionalen Fall je einen zweidimensionalen Vektor dar. $\mathbf{v}_{\text{tar}}(t)$ und $\mathbf{v}(t)$ stellen dahingegen die Geschwindigkeiten des Zieles und des Fahrzeugs zum Zeitpunkt t dar. $\|\mathbf{p}_{\text{tar}}(t) - \mathbf{p}(t)\|$ beschreibt die Euklidische Distanz und $\|\mathbf{v}_{\text{tar}}(t) - \mathbf{v}(t)\|$ die relative Geschwindigkeit jeweils zum Zeitpunkt t und zwischen Ziel und Fahrzeug. α_p und α_v sind skalare positive Parameter und m und n sind wählbare Konstanten. $U_{\text{att}}(\mathbf{p}, \mathbf{v})$ erreicht somit sein Minimum null, wenn sowohl die Euklidische Distanz als auch die relative Geschwindigkeit gleich null sind. Andererseits steigt das anziehende Potential, wenn die Euklidische Distanz und/oder die relative Geschwindigkeit zunimmt. Da im Rahmen der Projektgruppe derzeit lediglich statische Ziele definiert werden sollen, vereinfacht sich die Formel zu:

$$U_{\text{att}}(\mathbf{p}) = \alpha_p \|\mathbf{p}_{\text{tar}}(t) - \mathbf{p}(t)\|^m \tag{5.2}$$

Die aus dem negativen Gradienten des anziehenden Potentials $U_{\text{att}}(\mathbf{p})$ resultierende anziehende Kraft abhängig von der Position $\mathbf{p}$ lautet:

$$\mathbf{F}_{\text{att}}(\mathbf{p}) = -\nabla U_{\text{att}}(\mathbf{p}) = -\frac{\partial U_{\text{att}}(\mathbf{p})}{\partial \mathbf{p}} \tag{5.3}$$

Im Falle eines sich bewegendes Zieles würde sich die anziehende Kraft wie folgt ergeben:

$$\mathbf{F}_{\text{att}}(\mathbf{p}, \mathbf{v}) = -\nabla U_{\text{att}}(\mathbf{p}, \mathbf{v}) = -\nabla_{\mathbf{p}} U_{\text{att}}(\mathbf{p}, \mathbf{v}) - \nabla_{\mathbf{v}} U_{\text{att}}(\mathbf{p}, \mathbf{v}) \tag{5.4}$$

wobei

$$\nabla_p U_{\text{att}}(\mathbf{p}, \mathbf{v}) = \frac{\partial U_{\text{att}}(\mathbf{p}, \mathbf{v})}{\partial \mathbf{p}} \tag{5.5}$$

$$\nabla_v U_{\text{att}}(\mathbf{p}, \mathbf{v}) = \frac{\partial U_{\text{att}}(\mathbf{p}, \mathbf{v})}{\partial \mathbf{v}}. \tag{5.6}$$

Ist $\mathbf{p} \neq \mathbf{p}_{\text{tar}}$ und $\mathbf{v} \neq \mathbf{v}_{\text{tar}}$ ergibt sich durch Substitution von Formel 5.1 in Formel 5.4 die folgende Gleichung:

$$\mathbf{F}_{\text{att}}(\mathbf{p}, \mathbf{v}) = \mathbf{F}_{\text{att1}}(\mathbf{p}) + \mathbf{F}_{\text{att2}}(\mathbf{v}) \tag{5.7}$$

wobei

$$\mathbf{F}_{\text{att1}}(p) = m\alpha_{\text{p}} \|\mathbf{p}_{\text{tar}}(t) - \mathbf{p}(t)\|^{m-1} \mathbf{n}_{\text{RT}} \tag{5.8}$$

$$\mathbf{F}_{\text{att2}}(v) = n\alpha_v \|\mathbf{v}_{\text{tar}}(t) - \mathbf{v}(t)\|^{n-1} \mathbf{n}_{\text{VRT}} \tag{5.9}$$

$\mathbf{n}_{\text{RT}}$ beschreibt den vom Fahrzeug auf das Ziel zeigenden Einheitsvektor wohingegen $\mathbf{n}_{\text{VRT}}$ den Einheitsvektor der relativen Geschwindigkeitsrichtung des Zieles relativ zum Fahrzeug angibt.

Hindernisse und andere Fahrzeuge stellen abstoßende Potentiale und die daraus resultierenden abstoßenden Kräfte dar. Sie werden nach Ge et al. wie folgend definiert. Auch

im Falle der abstoßenden Potentiale können auch hier die Geschwindigkeiten von Hindernissen berücksichtigt werden, was besonders im Falle bei einer Kollisionsmöglichkeit zwischen zwei sich bewegenden Fahrzeugen hilfreich sein kann. Ge et al. nehmen hierzu ein punktförmiges Hindernis an. In diesem Fall ergibt sich die relative Geschwindigkeit zwischen dem Fahrzeug und dem anderen sich bewegendem Hindernis zu

$$\mathbf{v}_{RO}(t) = \left[\mathbf{v}(t) - \mathbf{v}_{\text{obs}}(t)\right]^T \mathbf{n}_{RO} \tag{5.10}$$

$\mathbf{n}_{RO}$ stellt den vom Fahrzeug zum Hindernis gerichteten Einheitsvektor dar. Ist $\mathbf{v}_{RO}(t) \leq 0$ bewegt sich das Fahrzeug vom Hindernis weg und eine Ausweichbewegung wird hinfällig. Im Folgenden wird die Annahme getroffen, das zu einem Zeitpunkt t die kürzeste Distanz zwischen einem Fahrzeug und einem Hindernis als

$$\rho_s(\mathbf{p}(t), \mathbf{p}_{\text{obs}}(t)) \tag{5.11}$$

gegeben ist. Angenommen ein Fahrzeug bewegt sich auf ein Hindernis zu und die maximale negative Beschleunigung ist a_{max}, dann ergibt sich der zurückgelegte Weg bis $\mathbf{v}_{RO}(t) = 0$ durch

$$\rho_m(\mathbf{v}_{RO}) = \frac{\mathbf{v}_{RO}^2(t)}{2a_{\max}} \tag{5.12}$$

Daraus ergibt sich das durch das Hindernis abstoßende Potential zu

$$U_{\text{rep}}(\mathbf{p}, \mathbf{v}) \begin{cases} 0 & \text{falls } \rho_s(\mathbf{p}, \mathbf{p}_{\text{obs}}) - \rho_m(\mathbf{v}_{RO}) \geq \rho_0 \text{ oder } \mathbf{v}_{RO} \leq 0 \\ U_{\text{eff}}(\mathbf{p}, \mathbf{v}) & \text{falls } 0 < \rho_s(\mathbf{p}, \mathbf{p}_{obs}) - \rho_m(\mathbf{v}_{RO}) < \rho_0 \text{ und } \mathbf{v}_{RO} > 0 \\ \text{nicht definiert} & \mathbf{v}_{RO} > 0 \text{ und } \rho_s(\mathbf{p}, \mathbf{p}_{\text{obs}}) < \rho_m(\mathbf{v}_{RO}) \end{cases} \tag{5.13}$$

mit

$$U_{\text{eff}}(\mathbf{p}, \mathbf{v}) = \eta \left(\frac{1}{\rho_s(\mathbf{p}, \mathbf{p}_{\text{obs}}) - \rho_m(\mathbf{v}_{RO})} - \frac{1}{\mathbf{v}} \right) \tag{5.14}$$

η und ρ_0 sind positive Konstanten, wobei ρ_0 den Einflussbereich des Hindernisses beschreibt. Bewegt sich das Fahrzeug vom Hindernis weg ($\mathbf{v}_{RO} \leq 0$) oder ist es zu weit von diesem entfernt($\rho_s(\mathbf{p}, \mathbf{p}_{obs}) - \rho_m(\mathbf{v}_{RO}) \geq \rho_0$), hat das abstoßende Potential keinen Einfluss und wird gleich null gesetzt. Befindet das Fahrzeug sich jedoch im Einflussbereich des Hindernisses ($\rho_s(\mathbf{p}, \mathbf{p}_{\text{obs}})$) und der Abstand zwischen ihnen erreicht $\rho_m(\mathbf{v}_{RO})$, läuft das abstoßende Potential gegen Unendlich und $\mathbf{v}_{RO}$ nimmt zu.

Aus dem in 5.13 beschriebenen abstoßenden Potential lässt sich, wie auch bei dem anziehenden Potential, die abstoßenden Kraft durch den negativen Gradienten erzeugen:

$$\mathbf{F}_{rep}(\mathbf{p}, \mathbf{v}) = -\nabla U_{\text{rep}}(\mathbf{p}, \mathbf{v}) = -\mathbf{p} U_{\text{rep}}(\mathbf{p}, \mathbf{v}) - \nabla_{\mathbf{v}} U_{\text{rep}}(\mathbf{p}, \mathbf{v}) \tag{5.15}$$

Die relative Geschwindigkeit des Fahrzeuges in Bezug zum Hindernis wird wie folgt definiert:

$$\mathbf{v}_{RO}(t) = (\mathbf{v}(t) - \mathbf{v}_{\text{obs}}(t))^T \mathbf{n}_{\text{RO}} = (\mathbf{v}(t) - \mathbf{v}_{\text{obs}}(t))^T \frac{\mathbf{p}_{\text{obs}}(t) - \mathbf{p}(t)}{\|\mathbf{p}_{\text{obs}}(t) - \mathbf{p}(t)\|} \tag{5.16}$$

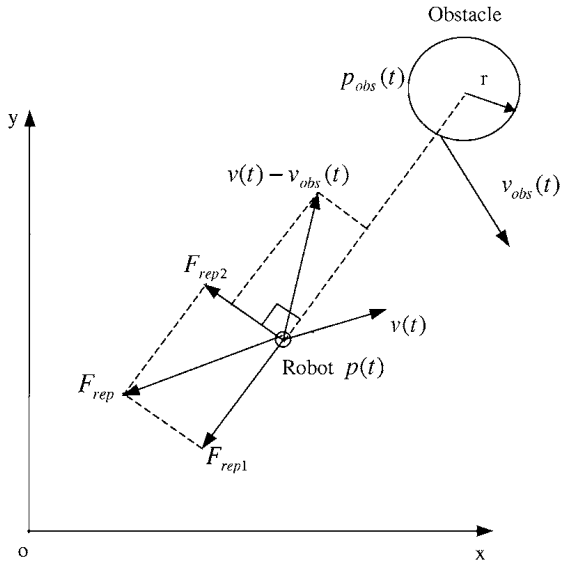


Abbildung 5.1.: Abstoßende Kraft im 2D-Fall [GC02]

Die Gradienten von $\mathbf{v}_{RO}(t)$ nach $\mathbf{v}$ bzw. $\mathbf{p}$ sind gegeben durch:

$$\nabla_{\mathbf{v}}\mathbf{v}_{RO}(t) = \mathbf{n}_{\mathrm{RO}}\nabla_{\mathbf{p}}\mathbf{v}_{RO}(t) = \frac{1}{\|\mathbf{p}_{\mathrm{obs}}(t) - \mathbf{p}(t)\|}\mathbf{v}_{RO}\mathbf{n}_{RO\perp} \tag{5.17}$$

$\mathbf{v}_{RO\perp}(t)\mathbf{n}_{RO\perp}$ beschreibt hierbei die Geschwindigkeitskomponente von $\mathbf{v}(t) - \mathbf{v}_{\mathrm{obs}}(t)$ in Richtung vom Fahrzeug zum Hindernis.

Die abstoßende Kraft ergibt sich folglich zu

$$\mathbf{F}_{\mathrm{rep}}(\mathbf{p},\mathbf{v}) \begin{cases} 0 & \text{falls } \rho_s(\mathbf{p},\mathbf{p}_{\mathrm{obs}}) - \rho_m(\mathbf{v}_{RO}) \geq \rho_0 \text{ oder } \mathbf{v}_{RO} \leq 0 \\ \mathbf{F}_{\mathrm{rep1}} + \mathbf{F}_{\mathrm{rep2}} & \text{falls } 0 < \rho_s(\mathbf{p},\mathbf{p}_{\mathrm{obs}}) - \rho_m(\mathbf{v}_{RO}) < \rho_0 \text{ und } \mathbf{v}_{RO} > 0 \\ \text{nicht definiert} & \mathbf{v}_{RO} > 0 \text{ und } \rho_s(\mathbf{p},\mathbf{p}_{\mathrm{obs}}) < \rho_m(\mathbf{v}_{RO}) \end{cases} \tag{5.18}$$

wobei

$$\mathbf{F}_{\mathrm{rep1}} = \frac{-\eta}{(\rho_s(\mathbf{p},\mathbf{p}_{\mathrm{obs}}) - \rho_m(\mathbf{v}_{RO}))^2}\left(1 + \frac{\mathbf{v}_{RO}}{a_{\max}}\right)\mathbf{n}_{RO} \tag{5.19}$$

und

$$\mathbf{F}_{\mathrm{rep2}} = \frac{\eta\mathbf{v}_{RO}\mathbf{v}_{RO\perp}}{\rho_s(\mathbf{p},\mathbf{p}_{\mathrm{obs}}a_{\max}(\rho_s(\mathbf{p},\mathbf{p}_{\mathrm{obs}}) - \rho_m(\mathbf{v}_{RO}))^2}\mathbf{n}_{RO\perp} \tag{5.20}$$

Der Zusammenhang zwischen $\mathbf{F}_{\mathrm{rep1}}$ und $\mathbf{F}_{\mathrm{rep2}}$ wird in Abbildung 5.1 dargestellt. Die das Fahrzeug vom Hindernis fernhaltende Kraft $\mathbf{F}_{\mathrm{rep1}}$ ist genau entgegengesetzt zu $\mathbf{v}_{RO}\mathbf{n}_{\mathrm{RO}}$, wohingegen $\mathbf{F}_{\mathrm{rep2}}$ in die gleiche Richtung wie $\mathbf{v}_{RO\perp}\mathbf{n}_{RO\perp}$zeigt und das Fahrzeug um das Hindernis zieht.

Sind mehrere Hindernisse vorhanden werden die einzelnen abstoßenden Kräfte zu einer abstoßenden Kraft aufaddiert:

$$\mathbf{F}_{\mathrm{rep}} = \sum_{i=1}^{n_{\mathrm{obs}}} \mathbf{F}_{\mathrm{rep}}i \tag{5.21}$$

Darüber hinaus sollte die Größe des Fahrzeuges mit einbezogen werden, indem das abstoßende Potential um ρ_{rob} erweitert wird.

$$U_{\text{rep}}(\mathbf{p},\mathbf{v}) \begin{cases} 0 & \text{falls } \rho_s(\mathbf{p},\mathbf{p}_{\text{obs}}) - \rho_{\text{rob}} - \rho_m(\mathbf{v}_{RO}) \geq \rho_0 \text{ oder } \mathbf{v}_{RO} \leq 0 \\ U_{\text{eff}}(\mathbf{p},\mathbf{v}) & \text{falls } 0 < \rho_s(\mathbf{p},\mathbf{p}_{\text{obs}}) - \rho_{\text{rob}} - \rho_m(\mathbf{v}_{RO}) < \rho_0 \text{ und } \mathbf{v}_{RO} > 0 \\ \text{nicht definiert} & \mathbf{v}_{RO} > 0 \text{ und } \rho_s(\mathbf{p},\mathbf{p}_{\text{obs}}) - \rho_{\text{rob}} < \rho_m(\mathbf{v}_{RO}) \end{cases} \tag{5.22}$$

mit

$$U_{\text{eff}}(\mathbf{p},\mathbf{v}) = \eta\left(\frac{1}{\rho_s(\mathbf{p},\mathbf{p}_{\text{obs}}) - \rho_{\text{rob}} - \rho_m(\mathbf{v}_{RO})} - \frac{1}{\mathbf{v}}\right) \tag{5.23}$$

$\rho_{\text{rob}}*$ gibt den Radius eines zylinderförmigen Fahrzeuges an und $\rho_{\text{rob}} \geq \rho_{\text{rob}}*$ stellt den künstlichen Sicherheitsabstand um das Fahrzeug herum dar. ρ_{rob} kann abhängig von der Qualität der Informationen, wie den Geschwindigkeits- und den Positionsinformationen, gewählt werden. Je ungenauer diese Informationen sind, desto größer sollte ρ_{rob} gewählt werden.

Letztlich kann die virtuelle Kraft $\mathbf{F}_{\text{total}}$ berechnet werden, indem die abstoßende und die anziehende Kraft addiert werden:

$$\mathbf{F}_{\text{total}} = \mathbf{F}_{\text{rep}} + \mathbf{F}_{\text{att}}. \tag{5.24}$$

Wie genau dieses Verfahren in der Projektgruppe umgesetzt wurde und welche Parameter für die erwähnten Konstanten gewählt wurden, wird in dem Kapitel Prototyp beschrieben.

5.4. Vector Field Histogram

Die im Folgenden beschriebene Vector Field Histogramm (VFH) Methode wurde 1991 von Borenstein und Koren entwickelt und unter [BK91] veröffentlicht. Es stellt ein echtzeitfähiges Verfahren zur Kollisionsvermeidung von Hindernissen dar, welches auf einem zweistufigen Verfahren zur Datenreduktion und einer dreistufigen Repräsentation der Daten basiert.

Die erste Stufe der Datenrepräsentation bildet ein zweidimensionales kartesisches Histogramm-Grid C. Dieses stellt die Umgebung des Fahrzeuges, welches Hindernissen ausweichen soll, dar und kann an jeder Koordinate entweder mit einem Hindernis belegt oder frei sein. C wird in regelmäßigen Abständen durch die, über die Sensoren des Fahrzeuges aufgenommenen, Umgebungsinformationen aktualisiert. Ein eindimensionales Histogramm H in Polarkoordinaten stellt die zweite Stufe der Datenrepräsentation dar. Es umfasst nicht, wie C, die komplette Umgebung des Fahrzeuges, sondern nur dessen momentanen aktiven Bereich [BK91]. H umfasst n Winkelsektoren mit einer Breite von α, wobei $\frac{360}{\alpha}$ eine ganze Zahl ergeben muss. Der durch H dargestellte Bereich wird auf den entsprechenden aktiven Bereich von C abgebildet. Dadurch wird jedem Sektor k ein

Wert h_k zugewiesen, welcher die polare Dichte an Hindernissen in Richtung des Sektors k wiedergibt [BK91]. Die letzte Stufe der Datenrepräsentation bilden die Ergebniswerte für die neue zu setzende Geschwindigkeit und Richtung des Fahrzeuges.

Die zuvor beschriebene Stufen der Datenrepräsentation werden nun in dem zweistufigen Verfahren zur Datenreduktion genutzt. Jede Zelle im aktiven Bereich von C wird im Folgenden als ein Hindernisvektor betrachtet, welcher sich zum einen aus einer Richtungskomponenten β und zum anderen aus einer berechneten Größe m zusammensetzt. Die Richtung β beschreibt die Strecke zwischen der aktiven Zelle und dem vehicle center point (VCP) und ist wie folgt definiert:

$$\beta_{i,j} = \tan^{-1} \frac{y_j - y_0}{x_i - x_0}. \tag{5.25}$$

Hierbei sind x_0 und y_0 die aktuellen Koordinaten des VCPs und x_i und y_j die Koordinaten der aktiven Zelle (i, j). Die Größe m ist gegeben durch:

$$m_{i,j} = (c_{i,j})^2(a - bd_{i,j}). \tag{5.26}$$

$c_{i,j}$ ist ein Sicherheitswert der aktiven Zelle (i, j). a und b stellen positive Konstanten und $d_{i,j}$ stellt die Entfernung zwischen der aktiven Zelle und dem VCP dar. Die Übereinstimmung zwischen $c_{i,j}$ und einem Sektor k ist gegeben durch [BK91]:

$$k = \text{INT}\left(\frac{\beta_{i,j}}{\alpha}\right). \tag{5.27}$$

Und die polare Dichte der Hindernisse h_k in einem Sektor k wird durch folgende Gleichung berechnet:

$$h_k = \sum_{i,j} m_{i,j}. \tag{5.28}$$

Durch die diskrete Struktur des Grids kann es zu Fehlern kommen, welche durch eine Glättung, angewandt auf H, reduziert werden können. Mit Hilfe des geglätteten Histogramms können nun die neue Geschwindigkeit und die Richtung des Fahrzeuges berechnet werden. Hierzu werden im geglätteten Histogramm H alle Täler bestimmt, welche unter einen zu wählenden Grenzwert fallen, siehe Abbildung 5.2. Die in Abbildung 5.2 mit A, B und C betitelten Hügel repräsentieren die erkannten Hindernisse, in diesem Fall drei Stück. Bei der Wahl des Grenzwertes muss darauf geachtet werden, dass dieser einerseits nicht zu groß gewählt wird, da sonst kleinere Hindernisse übersehen werden können und andererseits auch nicht zu klein gesetzt wird, da so schon feine Unebenheiten, beispielsweise des Bodens, als Hindernis erkannt werden können. Die Täler werden zu potentiellen Kandidaten in Hinblick auf die Berechnung der neuen Fahrtrichtung des Fahrzeugs. Es wird jenes Tal gewählt, welches am ehesten in Richtung des Ziels zeigt. Im Anschluss daran muss die genaue Fahrtrichtung des Fahrzeuges bestimmt werden, da Täler durchaus groß sein können und es somit mehrere Möglichkeiten gibt das Fahrzeug durch das Tal zu manövrieren. Hierzu wird zunächst die Größe des Tales berechnet, beispielsweise durch die zum Tal gehörende Anzahl an Sektoren. Die Täler können entsprechend ihrer Größe

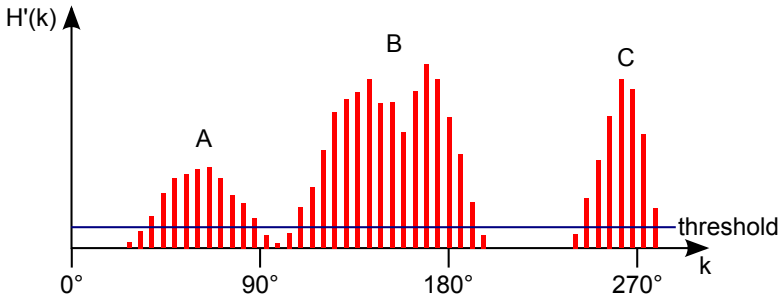


Abbildung 5.2.: Geglättetes Histogramm mit Grenzwert [BK91].

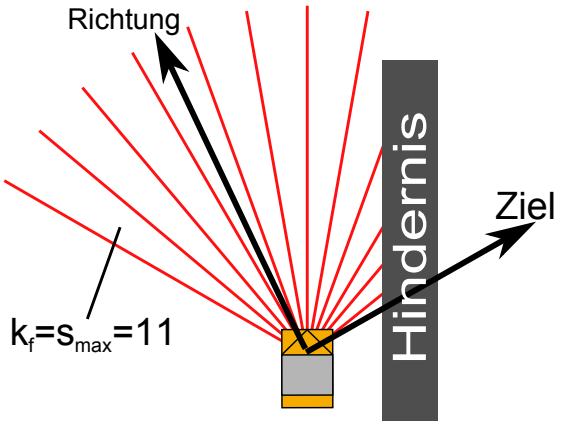
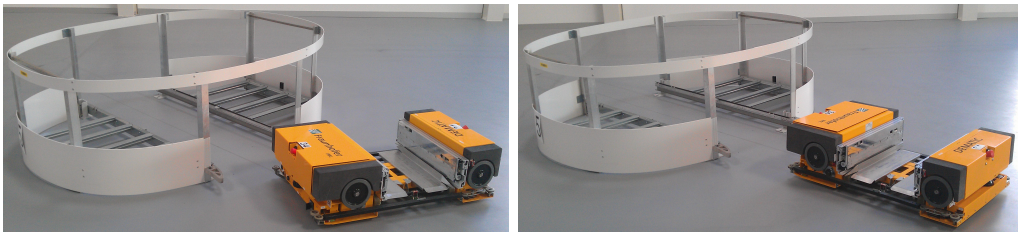


Abbildung 5.3.: Bestimmen der Zielrichtung. Vgl. [BK91]

anhand eines Grenzwertes $s_{\max}$ in kleine und große Täler aufgeteilt werden. Täler die aus weniger als $s_{\max}$ konsekutiven Sektoren bestehen werden als kleine Täler und welche mit mehr Sektoren als $s_{\max}$ werden als große Täler bezeichnet. Um die Fahrtrichtung des Fahrzeugs zu bestimmen wird im Falle eines großen Tales nicht das gesamte Tal zur Berechnung genutzt, sondern nur der interessante, dem Ziel zugewandte Teil des Tales. Hierzu wird der Sektor k_n bestimmt, welcher der Zielrichtung am nächsten liegt, wie in Abbildung 5.3 zu sehen ist. Von k_n aus werden nun $s_{\max}$ Sektoren entgegen der Zielrichtung vom Tal hinzugenommen, wodurch k_f bestimmt wird. Als neue Fahrtrichtung des Fahrzeuges wird die Mitte zwischen k_n und k_f gewählt:

$$\Theta = \frac{k_n + k_f}{2}. \tag{5.29}$$

Liegt jedoch ein kleines Tal mit einer Anzahl an Sektoren kleiner $s_{\max}$ vor, kann sodann das gesamte Tal zur Berechnung der Fahrtrichtung genutzt werden. Die Talgrenzen werden gleich k_n und k_f gesetzt und die Fahrtrichtung wird durch das Halbieren des Sektors bestimmt. Wie genau dieses Verfahren in der Projektgruppe umgesetzt wurde wird in den späteren Kapiteln beschrieben.



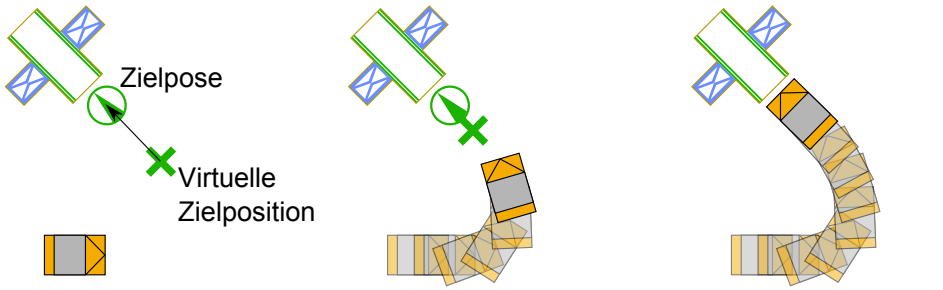
(a) Winkelversetzte Ausrichtung (b) Optimale Ausrichtung

Abbildung 5.4.: Ein Fahrzeug bei der Einfahrt in eine Kommissionierstation mit verschiedenen Ausrichtungen

5.5. Anfahren von Zielposen

Um beispielsweise beim Einfahren in eine Kommissionierstation oder ein Lagerregal eine Rotation des Fahrzeugs um das eigene Zentrum zu vermeiden ist es vorteilhaft eine anzufahrende Zielposition nicht nur lateral zu erreichen, sondern direkt in einem bestimmten Winkel zum Stehen zu kommen. Eine derartige Kombination aus Zielposition und zu erreichendem Ausrichtungswinkel wird als *Zielpose* bezeichnet. Abbildung 5.4 zeigt ein entsprechendes Szenario mit einem Fahrzeug vor einer Kommissionierstation mit verschiedenen Ausrichtungen. In dem in Unterabbildung 5.4(a) gezeigtem Zustand muss das Fahrzeug zunächst eine Rotationsbewegung vollziehen, wohingegen es in dem in Unterabbildung 5.4(b) gezeigtem Zustand direkt in die Kommissionierstation einfahren kann.

Die betrachteten Bahnplanungsverfahren der Potentialfelder und der VFH sind beide für das Anfahren einer Zielposition konzipiert. Um die Aufgabe eine Zielpose anzufahren mit diesen Verfahren lösen zu können, ist es vorteilhaft, diese auf das Anfahren einer Position zurückführen zu können. Zu diesem Zweck lässt sich der Begriff einer *virtuellen Zielpose* definieren. Hierbei handelt es sich um eine Position die relativ zur anzufahrenden Zielposition verschoben ist. Die Verschiebungsrichtung entspricht dem um 180° gedrehten Winkel der Zielausrichtung. Der Betrag der Verschiebung ist Abhängig vom Abstand des Fahrzeugs zur Zielposition. Wenn der Abstand des Fahrzeugs zur Zielposition kleiner wird, verringert sich gleichzeitig der Betrag der Verschiebung. Abbildung 5.5 zeigt den Zusammenhang der geschilderten Komponenten im Zeitverlauf.



(a) Ausgangszustand (b) Zwischenzustand (c) Endzustand

Abbildung 5.5.: Anfahren einer Zielpose unter Verwendung einer virtuellen Zielposition

6. Middleware

In diesem Kapitel wird in Abschnitt 6.2 besonders auf *OpenSplice* eingegangen. Insbesondere werden auch einige Nachteile beschrieben, welche die Sinnhaftigkeit der Implementierung einer eigenen Middlewarelösung implizieren. Die eigene Implementierung wird in Abschnitt 6.3 erläutert.

6.1. Einleitung

Die *Middleware* ist der Teil der Software, der einheitliche Schnittstellen zur Kommunikation der Prozesse anbietet. Ziel der Middleware ist es, die Komplexität der Kommunikation – auch über Fahrzeuggrenzen hinweg – vor den Prozessen der anderen Module zu verbergen.

Da die Middleware später auf eingebetteten Systemen läuft, ergeben sich spezielle Anforderungen. Zum einen soll die Middleware so ressourcenschonend wie möglich zu arbeiten, um die ohnehin knappen Leistungskapazitäten, welche zu großen Teilen von der Bildverarbeitung gebraucht werden, nicht unnötig zu verbrauchen. Zum anderen ist die Echtzeitfähigkeit des Systems von großer Bedeutung, da Aufgaben wie Kollisionserkennung bzw. -vermeidung darauf beruhen, den Status der anderen Fahrzeuge zu jeder Zeit möglichst genau und damit zeitnah zu kennen.

Fertige Middlewarelösungen existieren bereits und verfolgen unterschiedliche Lösungsansätze. In Betracht gezogen wurden u. a. *OpenSplice* und *Devices Profile for Web Services* (DPWS).

Beide beruhen auf einer anderen Architektur und genauer in Betracht gezogen wurde für diese Projektgruppe nur *OpenSplice*, da laut Dokumentation eine ressourcensparende Echtzeit-Middleware gegeben ist.

6.2. OpenSplice

Die erste Version der Middleware basierte auf *OpenSplice Data Distribution Service (DDS)*. Obgleich diese Version der Middleware nicht im Rahmen der finalen Middleware genutzt wird, wird im Folgenden dennoch auf ihre Implementierung eingegangen, da die entwickelte Lösung die geforderte Funktionalität bis auf einen Punkt erfüllte. Die Problematik bestand in einem technischen Fehler auf Seiten des *OpenSplice DDS*, welche

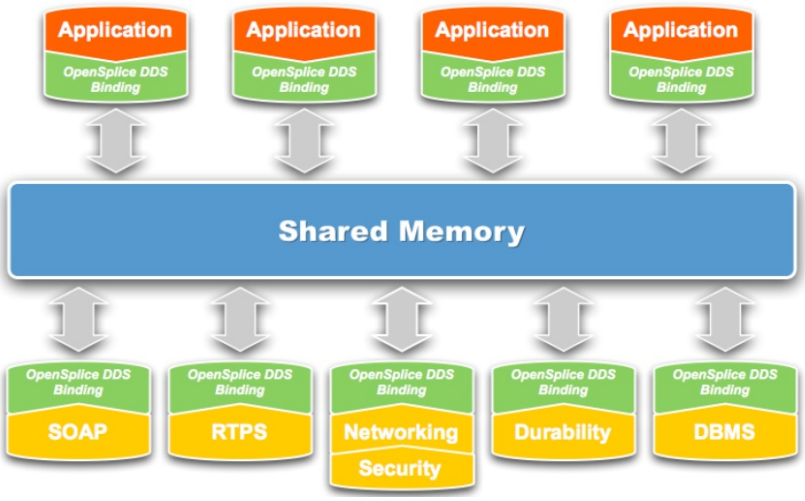


Abbildung 6.1.: Architektur von OpenSplice [Cor10]

durch die Projektgruppe nicht behoben werden konnte. In den folgenden Abschnitten zunächst auf allgemeine Eigenschaften und Strukturen von *OpenSplice DDS* eingegangen und im Anschluss daran die Umsetzung der Middleware mit *OpenSplice DDS* skizziert.

OpenSplice DDS ist eine von PrismTech entwickelte Implementierung des OMG[1] Data Distribution Service Standards für Echtzeitsysteme [Pri12]. Im Rahmen der Projektgruppe wurde die *OpenSplice DDS* Community Edition genutzt, lizenziert durch die GNU Lesser General Public Lizenz (LGPL). Der Unterschied zu der kostenpflichtigen Commercial Edition besteht in dem Support seitens PrismTech. Die Community Edition bietet als echtzeitfähige Infrastruktur einen auf UDP/IP basierenden Nachrichtenaustausch und Support für inhaltsbasierte Subscriptions, Datenpersistenz und transparentes Fehlermanagement an [Pri12]. Darüber hinaus realisiert *OpenSplice DDS* eine vollständige Implementierung des Data-Centric Publish-Subscribe (DCPS) Layers (Version 1.2) und des Interoperabilitäts-Wire-Protokolls spezifiziert in der OMG DDSI Version 2.1 [Obj09].

Die Besonderheit der Architektur von *OpenSplice DDS* ist der die Basis bildende Shared Memory wie in Abbildung 6.1 zu sehen. Durch dessen Nutzung wird die Latenz zwischen datenaustauschenden Komponenten minimiert und die Skalierbarkeit der Anzahl dieser Komponenten maximiert [Cor10]. Dieses Grundkonzept von *OpenSplice DDS* baut auf einen vollständig verteilten globalen Datenraum, den GDS (global data space), auf. Sogenannte Publisher und Subscriber können diesen zu jederzeit betreten und verlassen, siehe Abbildung 6.2. Publisher sind Komponenten welche Daten anbieten, wohingegen Subscriber Daten abonnieren. Beide können innerhalb des GDS veröffentlichen, welche Daten sie anbieten bzw. nutzen möchten. Die GDS wird wiederum einer Domain zugeteilt,

[1] Object Management Group. OMG ist seit 1989 ein internationales, gemeinnütziges, frei zugängliches Computer-Industrie-Konsortium [Obj12]

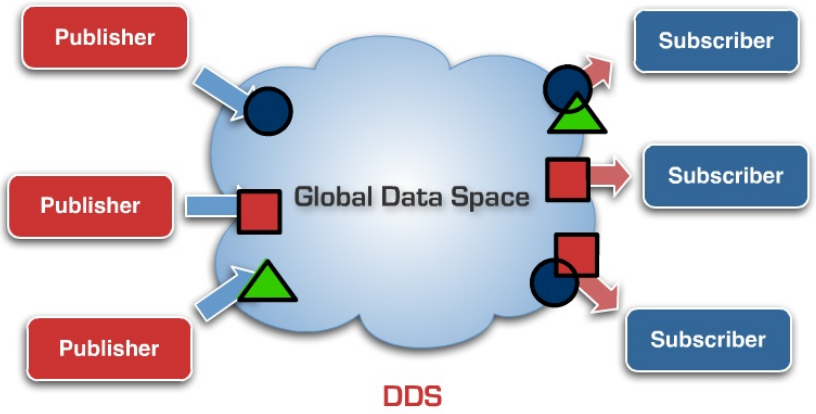


Abbildung 6.2.: Global Data Space [Cor10]

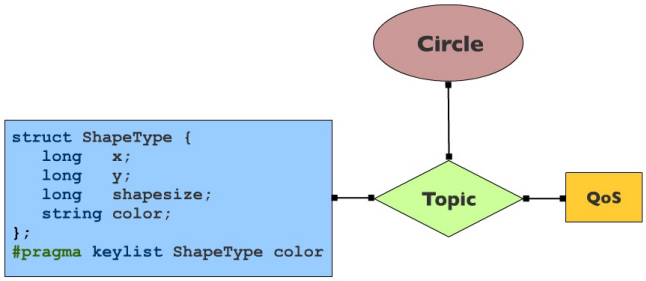


Abbildung 6.3.: Definition eines OpenSplice-Topics [Cor10]

welche abermals in verschiedene Partitionen unterteilt werden kann. Auf diese Weise können von der jeweiligen Aufgaben abhängige Partitionen wie z. B. eine Networking Partition erstellt werden, die beispielsweise an eine Liste von Unicast- oder Multicast-netzwerkadressen gebunden ist. Die Daten, welche von den Publishern und Subscribern innerhalb der Domain versendet werden, werden in *OpenSplice DDS* Topic genannt. Ein Topic setzt sich aus einem einzigartigen Namen, einem Typ und Quality of Service (QoS) Einstellungen zusammen. QoS Einstellungen können u. a. dazu genutzt werden die Lebensdauer oder die Zuverlässigkeit des Topics festzulegen. Der Typ beschreibt die Daten, welche mit einem oder mehreren Topics assoziiert werden, beispielhaft am Typ `ShapeType` in Abbildung 6.3 dargestellt. Ferner kann der Typ einen Key besitzen, welcher wiederum durch eine beliebige Anzahl an Attributen beschrieben werden kann. Auf Basis eines Topics können Topic Instanzen gebildet werden, deren Lebenszeit in *OpenSplice DDS* kontrolliert wird.

Oft ist es seitens Publisher oder Subscriber erwünscht nicht alle Instanzen eines Topics zu senden oder zu empfangen, dazu bietet *OpenSplice DDS* einen Filter namens ContentFilter

an. Mit Hilfe des ContentFilters kann der komplette Inhalt des Topic-Typs per SQL-Abfrage durchsucht und gefiltert werden.

Die Projektgruppe setzte diese Konzepte von *OpenSplice DDS* in einer in C++ programmierten Library um, welche die folgenden Anforderungen erfüllen sollte:

- Datenaustausch zwischen verschiedenen Fahrzeugen über WLAN
- Transfer den akquirierten Bilddaten hin zu der Bildverarbeitungspipeline auf einem Fahrzeug
- Transfer der Informationen aus den ausgewerteten Bilddaten hin zu dem Bahnplanungsprozess

Die Kommunikation auf und zwischen den Fahrzeugen findet hierzu innerhalb einer Domain statt, an welche sich die Fahrzeuge anmelden können, um als Publisher und Subscriber zu fungieren. Meldet sich ein neues Fahrzeug an, kann es fortan die Positionsdaten aller anderen angemeldeten Fahrzeuge über WLAN empfangen und zudem selbst welche verschicken. Auf dem Fahrzeug selbst können alle akquirierten Bilddaten intern an die Bildverarbeitungspipeline gesendet werden. Wie bereits erwähnt, findet eine Kommunikation nur innerhalb einer Domain im globalen Datenraum statt. Aufgrund dessen verfügt die, von der Projektgruppe entwickelte, *OpenSplice DDS* Library über eine Klasse `domain.cpp`, die den geforderten Rahmen für die Kommunikation erstellt. Zu diesem Rahmen gehören die folgenden Komponenten:

Domain Eine statische Instanz der Domain. Sie stellt die Basis der Kommunikation und die Funktion connectToDomain() bereit, welche wiederum die nun folgenden Komponenten in der Domain erzeugt.

DomainParticipant und DomainParticipantFactory Ein DomainParticipant repräsentiert die lokale Zugehörigkeit einer Applikation zu einer Domain und ermöglicht somit die Kommunikation innerhalb dieser Domain. In der DomainParticipantFactory werden die DomainParticipants erstellt. Die DomainParticipantFactory ist selbst ein Singleton und kann daher mit der Methode get_instance() erzeugt werden.

Publisher Über den DomainParticipant wird der Publisher erstellt, dabei kann zudem eine QoS Einstellung für den Publisher gesetzt werden.

Subscriber Über den DomainParticipant wird ebenfalls der Subscriber erstellt. Es kann ferner eine QoS Einstellung für den Subscriber gesetzt werden.

Registrierung der Topics Innerhalb der Methode connectToDomain() werden die Topics registriert, welche per *OpenSplice DDS* kommuniziert werden sollen, je ein Topic für msmpsoition und msmpointcloud. Die Funktion registerTopic wird jedoch an die gleichnamige Funktion in der jeweiligen Klasse `msmpositioninterface.cpp` bzw. msmpointcloudinterface weitergegeben.

Mit Hilfe der beiden Klassen `msmposition.cpp` und `msmpointcloud.cpp` können Instanzen zum Versenden und Empfangen der Positionsdaten und der Bilddaten erzeugt werden. Beide Klassen erben von der Klasse `opensplicedata.cpp`, welche den beiden Klassen zum einen eine ID und zum anderen einen Zeitstempel übergibt. Zusätzlich wird in der `msmposition.cpp` das Positionsdatum und in der `msmpointcloud.cpp` werden die Bilddaten in Form eines Arrays gehalten. Zudem beinhalten beide Klassen jeweils die Funktionen send() und readOneIfAvailable(), welche die jeweiligen gleichnamigen Funktionen in den jeweiligen Interfaces `msmpositioninterface.cpp` und msmpointcloudinterface aufrufen. Diese Interfaces beinhalten darüber hinaus die folgenden Komponenten und Funktionen:

`send()` Diese Funktion bereitet die Daten der msmposition- oder der msmpointcloud-Instanz zum Senden vor und verschickt sie über einen DataWriter.

`readOneIfAvailable()` Wird ein Datum, eine Position oder ein Bild empfangen, wird dies von einem DataReader entgegengenommen, der jeweilige Inhalt ausgelesen und als eine Instanz von entweder msmposition oder von msmpointcloud zurückgegeben.

`registerTopic(Topic QoS)` Voran registriert diese Funktion den Datentyp des Topics bei der Domain, danach kann das Topic erzeugt werden, welches den Namen MSM_POSITIONS bzw. MSM_POINTCLOUD trägt. Zusätzlich werden der DataWriter und der DataReader erzeugt, welche zum Schreiben und Lesen der Daten erforderlich sind.

Leider erwies sich der Ansatz mit *OpenSplice DDS* als nicht geeignet. Zum einen war die gemessene Latenz von 7 ms beim Einschreiben und Auslesen der Bilddaten in den Shared Memory zwischen der Datenakquirierung und der Grafikpipeline für die geforderte Echtzeitfähigkeit zu hoch, da Zeit für die nachfolgenden zeitaufwändigeren Berechnungen im Rahmen der Grafikpipeline berücksichtigt werden mussten. Zum anderen gab es seitens *OpenSplice DDS* einen Fehler, welcher es unmöglich machte variabel lange Datenstrukturen zu unterstützen und somit auch die Übertragung der Pointcloud via msmpointcloud. Seitens *OpenSplice DDS* war dieses Problem bereits zu diesem Zeitpunkt bekannt, es sollte laut *OpenSplice DDS* jedoch zeitnah keine Lösung angeboten werden. Daher entschied sich die Projektgruppe für die Entwicklung eines eigenen Middlewarekonzeptes, welches im folgenden Unterkapitel beschrieben wird.

6.3. Eigene Realisierung

Aufgrund der nicht zufriedenstellenden Ergebnisse aus Abschnitt 6.2 wurde eine eigene Implementierung der Middleware angefertigt. Der Vorteil bei dieser eigenen Lösung ist zu allererst, dass die Umsetzung genau unseren Ansprüchen entsprechend minimalistisch und effizient erfolgen konnte.

Es wurde außerdem stark darauf geachtet, dass die Anwendung der Middleware für die späteren Prozesse möglichst einfach ist. Dies wurde durch eine feste Definition der

vorhandenen Prozesstypen gelöst. Es ist bei der Initialisierung der Middleware erforderlich den jeweiligen aktuellen Prozesstyp anzugeben. Durch diese Angabe erfolgt eine Zuweisung dedizierter Systemressourcen zum jeweiligen Prozess. So ist eine leichte Adressierung der vorhandenen Middleware-Prozesse auf dem gleichen und auch auf anderen Systemen im Netzwerk möglich.

Um eine einheitliche Schnittstelle zu schaffen wurden sämtliche Datentypen die über die Middleware kommuniziert werden müssen modelliert. Abbildung 6.4 zeigt die Umsetzung der Model-Klassen.

Durch das Prinzip der Vererbung ist es möglich alle Datentypen zu kodieren, mit einer Anmerkung zu versehen und anschließend generisch über die Middleware zu verschicken. Der Empfänger kann nun den generischen Typ empfangen und mit Hilfe der Anmerkung gezielt auf alle eintreffenden Daten reagieren.

Wie das Klassendiagramm in Abbildung 6.4 gut erkennen lässt ist die Klasse `DataType` als globale Oberklasse so gestaltet worden, dass es möglich ist die davon erbenden Klassen innerhalb dieser zu serialisieren und somit generalisiert über das Netzwerk zu verschicken.

Die konkreten Subklassen `PositionAngle` und `Pointcloud` repräsentieren nun die Datentypen, die über die Middleware versendet werden. `PositionAngle` besteht dabei aus einer Position und einem Winkel und wird dazu benutzt verschiedene Posen zu kommunizieren. `Pointcloud` ist dazu in der Lage Daten einer Punktwolke über die Middleware zu versenden.

Da die Übertragungsgeschwindigkeit der Daten in einer Echtzeitumgebung sehr wichtig ist wurde für die lokale Kommunikation ein *shared memory* gewählt. Die Kommunikation im Netzwerk wird mit dem *User Datagram Protocol* (UDP) gelöst. Die Tatsache, dass diese Kommunikation nicht zuverlässig ist, ist für unseren Einsatzzweck belanglos, da sämtliche Daten so kurzlebig sind, dass eine erneute Übertragung eher durch die Übertragung von neuen Daten abgelöst werden kann.

Weiterhin wurde die Fähigkeit auf Befehle zu reagieren implementiert. Diese Befehle erwarten, anders als die kurzlebigen Nutzdaten, eine gesicherte Übertragung an den Empfänger. Die Kommunikation der Befehle über die Middleware wurde somit mit dem *Transmission Control Protocol* (TCP) gelöst.

Die Klasse `Command` führt das Schichten-Prinzip der Middleware fort und hat die Aufgabe eine generische Schnittstelle zu verschiedenen Kommandos zu bieten. Diese Lösung ist durch die Bedingungen nötig geworden, die das TCP mit sich bringt. Hiervon leiten nun wiederrum die konkreten Kommando-Subklassen ab. Im gezeigten Klassendiagramm ist dies die Klasse `PointCommand`. Aufgabe der Klasse `PointCommand` ist es eine Anfrage an einen Teilnehmer der Middleware zu versenden mit der Aufforderung die nächste ermittelte Punktwolke an den Versender des `PointCommand` zu übermitteln. Diese Funktionalität wird zur Visualisierung benötigt.

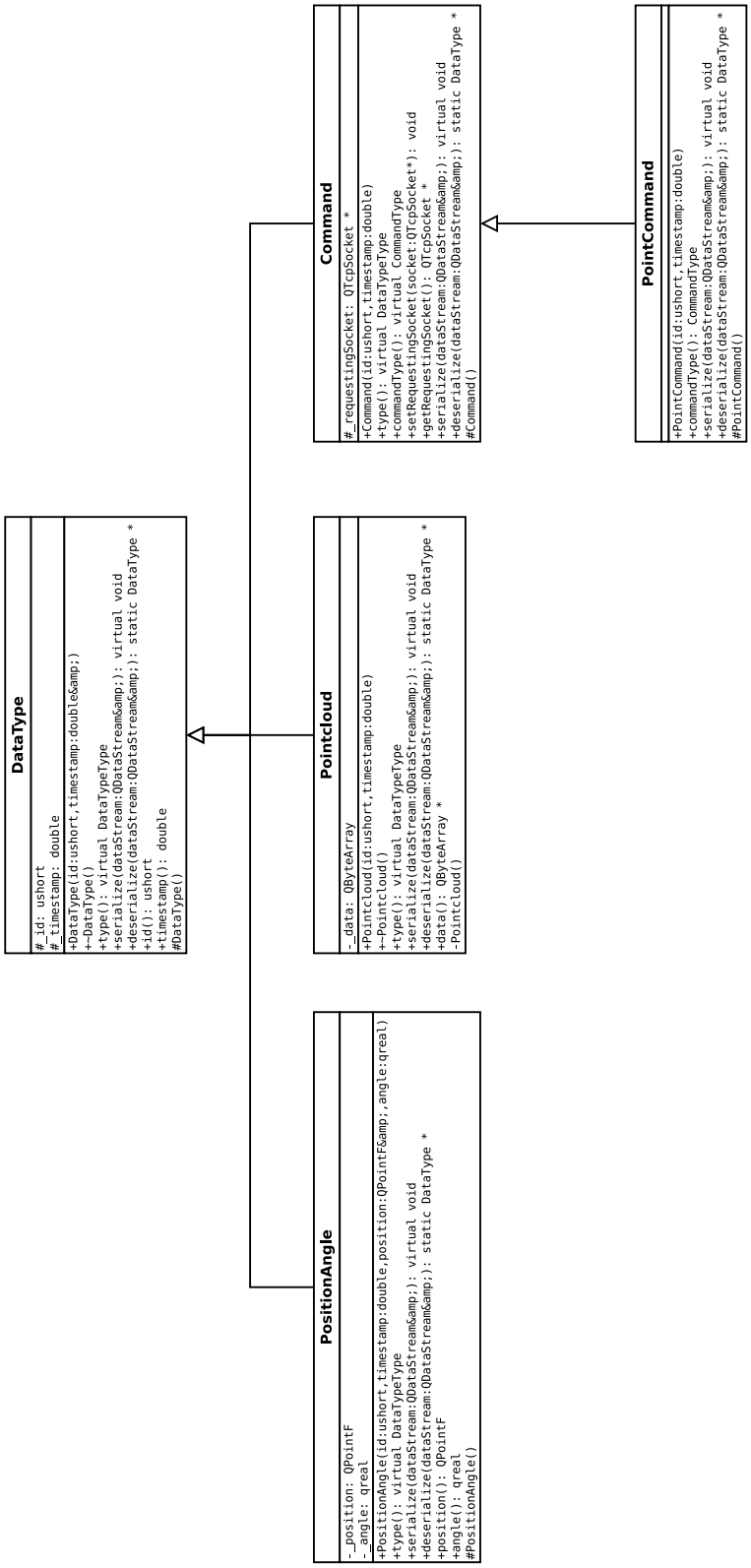


Abbildung 6.4.: Die Model-Klassen der Middleware

7. Zeitsynchronisation

Um im Netzwerk systemübergreifend Aufgaben zu planen, gemeinsame Datensätze mit Zeitstempeln zu annotieren oder zeitsensitive Protokolle zu implementieren, ist eine gemeinsame Zeitbasis notwendig. Da die Systemuhren verschiedener Rechner im Netzwerk üblicherweise nicht synchron laufen und nach einiger Zeit ohne Korrektur divergieren, ist eine Synchronisation dieser Uhren in regelmäßigen Zeitintervallen erforderlich.

In Hinblick auf eine mögliche Fusion verschiedener Sensordaten widmet sich dieses Kapitel einigen zur Zeitsynchronisation entwickelten Protokollen, die im Rahmen der Projektgruppe genauer betrachtet und für ihren speziellen Einsatz evaluiert wurden (siehe Abschnitt 9.4).

In Abschnitt 7.1 wird zunächst das Precision Time Protocol beschrieben, bevor in Abschnitt 7.2 auf das Network Time Protocol eingegangen wird. Das Kapitel endet mit einem kurzen Ausblick auf weitere in Erwägung gezogene Zeitprotokolle.

7.1. Precision Time Protocol

Beim Precision Time Protocol (PTP) handelt es sich um ein Zeitsynchronisationsprotokoll, mit dem verschiedene Uhren innerhalb eines kabelgebundenen Netzwerkes synchronisiert werden können. Innerhalb der Projektgruppe wird untersucht, ob PTP auch für den Einsatz über WLAN verwendet werden kann.

Im Folgenden wird ein kurzer Überblick über das PTP-Protokoll gegeben. Weiterführende Informationen sind dem PTP zugrunde liegenden Standard IEEE-1588 [IEE08] zu entnehmen.

7.1.1. Unterscheidung verschiedener Uhrentypen

Innerhalb des PTP-Standards werden mehrere Uhrentypen definiert, die verschiedene Aufgaben innerhalb eines Zeitnetzwerkes übernehmen. Die einzelnen Uhren können jeweils in Form von Hardware oder als reine Software-Uhren umgesetzt werden.

Ordinary Clock Bei der Ordinary Clock handelt es sich um eine einfache Uhr, die nur zur Synchronisation verwendet wird. Sie kann entweder als Master- oder als Slave-Clock arbeiten.

Boundary Clock Die Boundary Clock ist eine spezielle Uhr, die an Übergängen zwischen verschiedenen Netzwerken eingesetzt wird. Sie kann auf der einen Seite als Slave arbeiten und auf der anderen Seite als Master. Diese Uhr ermöglicht es, die auf der Slave-Seite empfangene Zeit in weitere angeschlossene Netzwerke zu propagieren.

Transparent Clock Die Transparent Clock dient dazu, empfangene PTP-Zeitstempel zu modifizieren und die angepassten Zeitstempel weiterzuleiten. In einem PTP-fähigen Switch kann somit beispielsweise die Verweildauer des PTP-Paketes innerhalb der Paket-Warteschlange berücksichtigt werden. Die ermittelte Verweildauer kann auf den im PTP-Paket enthaltenen Zeitstempel addiert und das so korrigierte Paket weitergeleitet werden. Hierdurch kann eine bessere Synchronisationsgenauigkeit erreicht werden.

Im Rahmen der Projektgruppe werden nur einfache Ordinary Clocks verwendet.

7.1.2. Synchronisationsverfahren

Für die Zeitsynchronisation wird innerhalb des Netzwerkes eine Uhr als spezielle Master-Clock gekennzeichnet. Diese Kennzeichnung kann entweder von Hand geschehen oder über das im PTP-Protokoll beschriebene Best-Master-Clock-Verfahren erfolgen, bei dem die verschiedenen Synchronisationsteilnehmer untereinander eine Master-Clock bestimmen. Die anderen Uhren im Netzwerk arbeiten als Slave-Clocks und synchronisieren sich auf die Zeit der Master-Clock.

Das PTP-Protokoll ist in der Lage die Verzögerungszeiten des Transportmediums bei symmetrischen Verbindungen vollständig herauszurechnen. Bei asymmetrischen Verbindungen, beispielsweise bei verschiedenen Routen für den Hin- und Rückweg eines Paketes nimmt die Synchronisationsgenauigkeit allerdings ab. In normalen kabelgebundenen Netzwerken von wenigen Metern Ausdehnung liegt diese Verschlechterung im niedrigen zweistelligen Nanosekunden-Bereich.

Weil innerhalb der Projektgruppe als Medium ein drahtloses Netzwerk eingesetzt wird, muss mit einem größeren Synchronisationsgenauigkeitsverlust gerechnet werden. Wie die Ergebnisse in Abschnitt 9.4.4 zeigen, liegt die Synchronisationsgenauigkeit allerdings noch innerhalb der gesetzten Anforderungen von 10 ms.

Um die Verzögerungszeiten des Transportmediums herauszurechnen, wird das Verfahren aus Abbildung 7.1 verwendet.

Die Master-Clock versendet zunächst eine Sync-Nachricht per Multicast an alle Slave-Clocks innerhalb der Multicast-Gruppe. Diese Nachricht enthält den aktuellen Zeitstempel der Master-Clock. Der Zeitpunkt t_1, an dem die Nachricht auf das Medium gelegt wird, wird zwischengespeichert und in der Follow_Up-Nachricht ebenfalls per Multicast an die Slave-Clocks übermittelt.

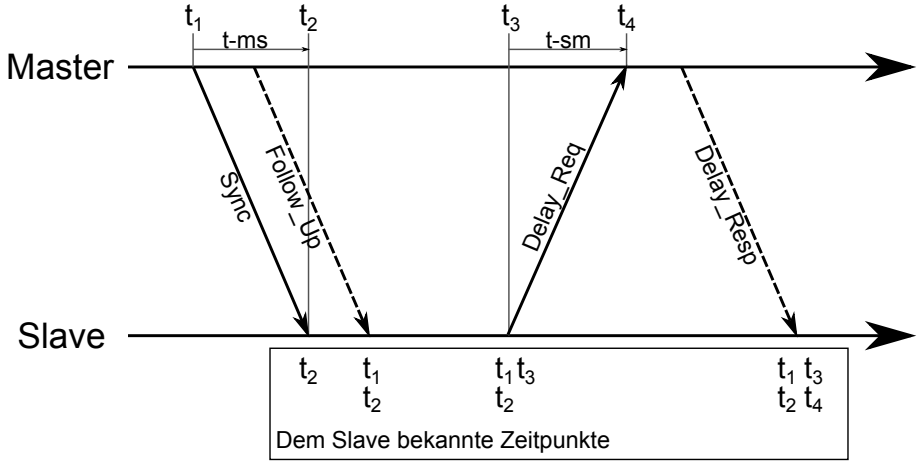


Abbildung 7.1.: PTP-Synchronisationsverfahren. Vgl. [IEE08]

Wenn eine Slave-Clock eine Sync-Nachricht empfängt, speichert sie den Zeitstempel, an dem die Nachricht empfangen wurde (t_2), zwischen. Mit der Zeit aus der Follow_Up-Nachricht kann die Master-to-Slave-Verzögerungszeit (t-ms) berechnet werden.

In bestimmten Intervallen versenden die Slave-Clocks Delay_Req-Nachrichten an die Master-Clock, um die Slave-to-Master-Verzögerungszeit zu berechnen. Dabei merkt sich die Slave-Clock den Zeitstempel t_3, an dem sie die Delay_Req-Nachricht verschickt hat. Die Master-Clock versendet in einer Delay_Resp-Nachricht den Zeitstempel t_4, der für den Empfangszeitpunkt der Delay_Req-Nachricht steht.

Mit diesen Informationen ist eine Slave-Clock in der Lage, die Master-to-Slave-Verzögerungszeit (t-ms) und die Slave-to-Master-Verzögerungszeit (t-sm) zu berechnen. Den Mittelwert dieser beiden Zeiten nutzt die Slave-Clock, um den erhaltenen Zeitstempel der Master-Clock zu korrigieren und als neue Zeit zu übernehmen.

7.1.3. Getestete Referenzimplementierungen

Für das PTP-Protokoll existieren verschiedene Implementierungen, die entweder in Hard- oder Software realisiert sind. Im Rahmen der durchgeführten Evaluation des PTP-Protokolls wurden die beiden folgenden Implementierungen verwendet.

PTPd

Bei *PTPd*[1] handelt es sich um eine Software-Implementierung des PTP-Protokolls, die unter Linux zur Zeitsynchronisation verwendet werden kann. *PTPd* kann neben der

[1] `http://ptpd.sourceforge.net`

Aufgabe als Slave-Clock auch die Aufgabe einer Master-Clock übernehmen und somit anderen im Netzwerk vorhandenen Geräten die aktuelle Zeit vorgeben.

Für die durchgeführten Messungen der Synchronisationsgenauigkeit mit Hilfe des PTP-Protokolls wurde *PTPd* als Zeitserver verwendet, um als Master-Clock für die Fahrzeuge zu fungieren. Der Zeitserver wurde dabei mit einem Synchronisations-Intervall von einer Sekunde betrieben.

Beckhoff PTP-Klemme EL6688

Für die Durchführung der Zeitmessungen mit PTP wurden zwei EtherCAT-Klemmen vom Typ *EL6688* [Bec10] von der Beckhoff Automation GmbH [Bec12] in zwei der vorhandenen Versuchsfahrzeuge der Projektgruppe eingebaut.

Die EtherCAT-Klemme *EL6688* kann direkt in das vorhandene Bussystem des Fahrzeuges eingebunden werden und erlaubt es der verwendeten SoftSPS sich mit einer außen anliegenden Weltzeit zu synchronisieren. Wenn die Klemmen im Slave-Modus konfiguriert sind, verwenden sie das an ihrem Ethernet-Anschluss anliegende PTP-Zeitsignal für die Zeitsynchronisation.

Anders als zu erwarten synchronisiert die PTP-Zeitklemme die Zeit innerhalb der SPS nicht direkt. Stattdessen berechnet sie einen Offset zwischen der Zeit innerhalb der SPS und der von außen anliegenden Weltzeit. Dieser Offset muss im ausgeführten SPS-Programm von Hand berücksichtigt werden.

7.2. Network Time Protocol

Für die Synchronisation der Fahrzeuge wurde neben dem Precision Time Protocol (PTP) auch das Network Time Protocol (NTP) getestet. Das Network Time Protocol (NTP) ist ein zur Synchronisation von Rechneruhren in Kommunikationsnetzen entwickeltes Protokoll. Im Vergleich zum PTP weist NTP eine verringerte Genauigkeit auf, hat allerdings den Vorteil dass es keine zusätzliche Hardware benötigt und somit kostengünstig implementiert werden kann.

7.2.1. Synchronisationsverfahren

NTP wurde seit 1982 unter der Leitung von David Mills an der University of Delaware entwickelt, und ist aktuell in Version v4 erhältlich. Ziel war es ein fehlertolerantes Protokoll zu entwickeln, dessen Genauigkeit auch über Netzwerke mit variabler Paketlaufzeit im Millisekundenbereich liegt. Die aktuelle Version v4 ist in RFC 5905 spezifiziert.

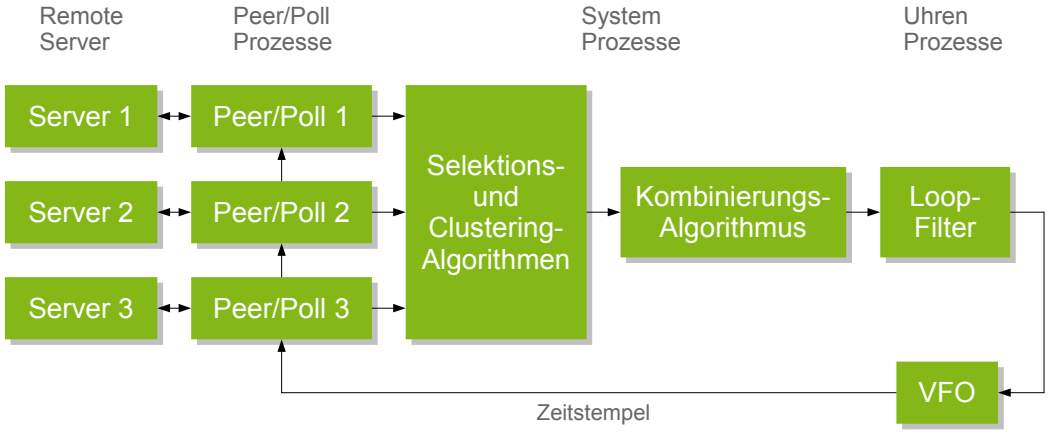


Abbildung 7.2.: NTP-Implementierungsmodell aus [Mil10]

Organisation

NTP verwendet zum Datentransport UDP auf Port 123 und ist hierarchisch aus mehreren Strata aufgebaut. Während die primären Zeitserver (Stratum 1) direkt an einen amtlichen Zeitstandard, wie z. B. eine Atomuhr angebunden sind, erfolgt die Synchronisation weiterer Knoten über ein hierarchisches, selbst organisierendes Netz, wobei Endanwender üblicherweise ein Stratumwert von vier besitzen. Durch redundante Referenzserver (Peers), kann der Ausfall von einem oder mehreren Knoten ausgeglichen werden.

Algorithmus

Die Architektur der aktuellen NTP Version lässt sich durch das Implementierungsmodell aus Abbildung 7.2 skizzieren. Für jeden entfernten Referenzserver sind dabei zwei Prozesse verantwortlich, ein sogenannter Peer-Prozess dessen Aufgabe der Empfang von Nachrichten des Servers oder einer Referenzuhr ist und ein Poll-Prozess der für den Versand von Nachrichten an den Referenzknoten zuständig ist. Ein Client sendet in regelmäßigen Abständen Nachrichten an den Server um seine Zeit zu aktualisieren. Dieses sogenannte Poll-Intervall liegt bei NTPv4 zwischen 4 s und 36 h.

Die Auswahlalgorithmen versuchen falsche Uhren, sogenannte „falsetickers“ zu erkennen und nur richtiggehende Uhren (truechimers) auszuwählen. Über die Offset-Werte aller ausgewählten Uhren wird anschließend ein gewichteter Mittelwert berechnet. Ältere NTP-Versionen betrachten nur die Offset-Werte der Uhr, die als genaueste klassifiziert wurde.

Die Schleife (Loop) und der variable Frequenzoszillator (VFO) bilden zusammen eine geregelte, lokale Uhr um den Jitter und Drift zu minimieren. Dabei wird sowohl die Zeit als auch die Frequenz der Systemuhr (hier durch den VFO repräsentiert) kontrolliert.

Der von NTP verwendete Zeitstempel hat eine Länge von 64 Bit und eine theoretische Auflösung von 233 ps, besitzt dafür allerdings eine Rollover-Periode von 136 Jahren. [HR05, Mil10]

7.2.2. Getestete Referenzimplementierungen

Implementierungen des Network Time Protocols sind für fast alle Rechnerplattformen verfügbar. Für UNIX-artige Betriebssysteme steht die Referenzimplementierung von NTP *ntpd* zur Verfügung. Eine Windows-Portierung dieser Implementierung stellt die Firma Meinberg bereit. Diese Version wurde für die Tests in Abschnitt 9.4 verwendet. Die NTP-Implementierung von Microsoft, die sich in Windows-Betriebssystemen befindet, wurde ebenfalls getestet. Weitere Implementierungen des Network Time Protocols fanden keine weitere Beachtung.

NTP-Varianten von Microsoft

Während Microsoft für Windows 2000 noch das Simple Network Time Protocol (SNTP) verwendete, welches eine vereinfachte Version von NTP darstellt, besitzen Betriebsysteme nach Windows 2000 eine eigene Version des Network Time Protocols, die als W32Time bekannt ist. Der W32Time-Dienst soll zwar mit der Referenzimplementierung von NTP weitgehend kompatibel sein, unterstützt jedoch nicht alle Funktionen von NTP und ist eigentlich für zeitkritische Applikationen ungeeignet, zu dem es nur für eine Synchronisationsgenauigkeit zwischen einer und zwei Sekunden ausgelegt ist. [Mic10a, Mic11]

Meinberg NTP

Die NTP Software der Firma Meinberg ist eine Windows-Portierung von *ntpd* [Mei12]. Sie verfügt zusätzlich über eine grafische Oberfläche zur Konfiguration (Time Server Monitor). Für die Evaluation wurde die Version NTP 4.2.4p8 verwendet und so konfiguriert, dass sie einen in der Versuchshalle vorhandenen Rechner als Zeitserver verwendet. Es wurde ein Synchronisationsintervall von 16 s verwendet.

7.3. Reference-Broadcast Synchronization und Flooding Time Synchronization Protocol

Neben PTP und NTP wurden auch die Reference-Broadcast Synchronization (RBS), welche die Broadcast-Pakete der Bitübertragungsschicht zur Synchronisation verwendet (vgl. [EGE02]) und das Flooding Time Synchronization Protocol (FTSP), ein weiteres hardwarenahes Protokoll, das mit Zeitstempeln auf der Sicherungsschicht arbeitet und für drahtlose Sensornetzwerke konzipiert wurde (vgl. [MKSL04]), in Betracht gezogen.

Da diese Protokolle allerdings einen Eingriff in die Hardware erforderlich gemacht hätten und PTP bzw. NTP die weitaus etablierteren Protokolle mit freien Implementierungen sind, wurden diese Synchronisationsprotokolle nicht weiter betrachtet.

8. Prototyp

Die in den vorhergehenden Abschnitten beschriebenen Module und die jeweiligen Verfahren wurden in Form eines Prototypen implementiert. Dieser Prototyp deckt die vollständige Vorgehensweise von der Aufnahme der Tiefendaten bis zur Ansteuerung eines realen Fahrzeugs unter Berücksichtigung der errechneten Daten ab.

In diesem Abschnitt ist die jeweilige Umsetzung aller notwendigen Einzelkomponenten beschrieben. In Abschnitt 8.1 ist die Implementierung der Grafikpipeline beschrieben, welche die Akquisition der Tiefenbilder, deren Umrechnung in Punktwolken, die Reduktion der Punkte auf die für die Erkennung von interessanten Objekten relevanten sowie die Erkennung von Objekten abdeckt. Die Informationen über die durch die Grafikpipeline erkannten Objekte wird von der Bahnplanung zum Umfahren von Hindernissen verwendet. Deren Implementierung ist in Abschnitt 8.2 beschrieben. Um letztendlich die Fahrzeuge in Abhängigkeit von den Ausgaben der Bahnplanung steuern zu können, wurden Entwicklungen im Bereich der Fahrzeugsteuerung getätigt, welche in Abschnitt 8.5 gezeigt werden.

Zur Entwicklung des Prototypen wurden mehrere Visualisierungswerkzeuge erstellt. Die Funktionsweise dieser Werkzeuge Abschnitt 8.3 erläutert.

Im Rahmen der Prototypenentwicklung wurde zusätzlich zu den anderen Komponenten auch die Zeitsynchronisation zwischen verschiedenen Fahrzeugen untersucht. Die Implementierung der hierbei verwendeten Verfahren ist in Abschnitt 8.4 beschrieben.

8.1. Grafik-Pipeline

Im Rahmen des Prototypen wurde für die Grafik-Pipeline eine Applikation mit dem Namen `MSM-Pipeline` entworfen. Die Applikation realisiert hierbei die Tiefendatenakquisition, die Verarbeitung und Interpretation der Daten und die anschließende Propagation der aus den Tiefendaten abgeleiteten Informationen. Hierbei wurde die Anwendung generisch gestaltet, so dass unterschiedliche Sensoren, aber auch unterschiedliche Verarbeitungspipelines getestet werden können. Die Propagation der Interpretations-Ergebnisse erfolgt hierbei über eine ebenfalls im Rahmen der Projektgruppe entwickelte und auf den Einsatzzweck optimierte Middleware (siehe Kapitel 6).

In Abbildung 8.2 wird das vereinfachte Klassendiagramm der `MSM-Pipeline` gezeigt. Die Applikation wurde hierbei als so genannte Einzelinstanz-Anwendung (engl. single

instance application) konzipiert. Das heißt dass auf einem Host prinzipiell nur eine einzige Instanz der Grafikpipeline läuft. Die an weitere Instanzen der Grafikpipeline übergebenen Kommandozeilen-Parameter werden jedoch an die Erstinstanz der `MSM-Pipeline` weitergereicht. Dieses Vorgehen ermöglicht die Initiierung, Steuerung und Terminierung der Grafikpipeline über die SoftSPS. Zu Entwicklungszwecken verfügt die `MSM-Pipeline` über eine `OpenGL`-basierte Visualisierung, die in der Klasse `PCLGLWidget` realisiert worden ist. Das Laden von 3D-Modellen erfolgt hierbei unter Verwendung der `VTK`-Bibliothek über die Klasse `WavefrontOBJLoader`. Von der abstrakten Klasse `AbstractFrameProcessor` lassen sich eigene `*FrameProcessor`en ableiten, über die die Verarbeitung der vom `*Player` gelieferten Rohdaten erfolgt. Jeder `*Player` ist hierbei ebenfalls von `AbstractPlayer` abzuleiten. Mit Hilfe dieses Ansatzes verfolgt die `MSM-Pipeline` das Konzept der einfachen Erweiterbarkeit, indem Basis-Klassen erweitert bzw. angepasst werden können. Auf diese Weise lassen sich weitere Sensor-Typen problemlos anbinden und Modifikationen der Grafik-Pipeline ausprobieren. Die Propagation der generierten Sekundärinformationen, die sich als Ergebnis der Grafik-Pipeline ergeben, erfolgt entweder über die Middleware oder direkt, sofern geringere Latenzen erforderlich sind, über die `SPS`. Zur Kommunikation mit der `SPS` dient hierzu die von der abstrakten Klasse `AbstractSPSGateway` abgeleitete Klasse `*SPSGateway`.

Die Abbildung 8.1 zeigt ein Bildschirmfoto der `MSM-Pipeline` Oberfläche. Die für Entwicklungszwecke gedachte Oberfläche ermöglicht die Datenakquisition über verschiedene Sensoren zu initiieren. Zusätzlich ermöglicht sie die komfortable Verwaltung bzw. Erstellung einer Signaturdatenbank. Detektierte Cluster können selektiert und ihre Signatur der aktuellen Datenbank hinzugefügt werden. Eine komfortable visuelle Überprüfung der Klassifikation wird über das Einpassen von CAD-Modellen ermöglicht. Angezeigt werden für jedes Cluster die jeweils drei nächsten Nachbarn aus der Signatur-Datenbank (sofern vorhanden) und die Distanz zu ihnen.

8.1.1. Verwendete Bibliotheken

Im Rahmen der Projektgruppe wurde aus Laufzeitgründen vorzugsweise objektorientiert mit Hilfe der Programmiersprache `C++` und dem Platform-abstrahierenden Framework `Qt` entwickelt.

OpenNI

Zur Parametrisierung und Tiefenbild-Akquisition der Microsoft Kinect bzw. der nahezu baugleichen Asus Xtion Pro wird im Rahmen der Projektgruppe das frei verfügbare `OpenNI`-Framework verwendet. Wegen der unter Windows XP im Vergleich zur `libfreenect` höheren Zuverlässigkeit wurde bezüglich des offiziellen Frameworks entschieden.

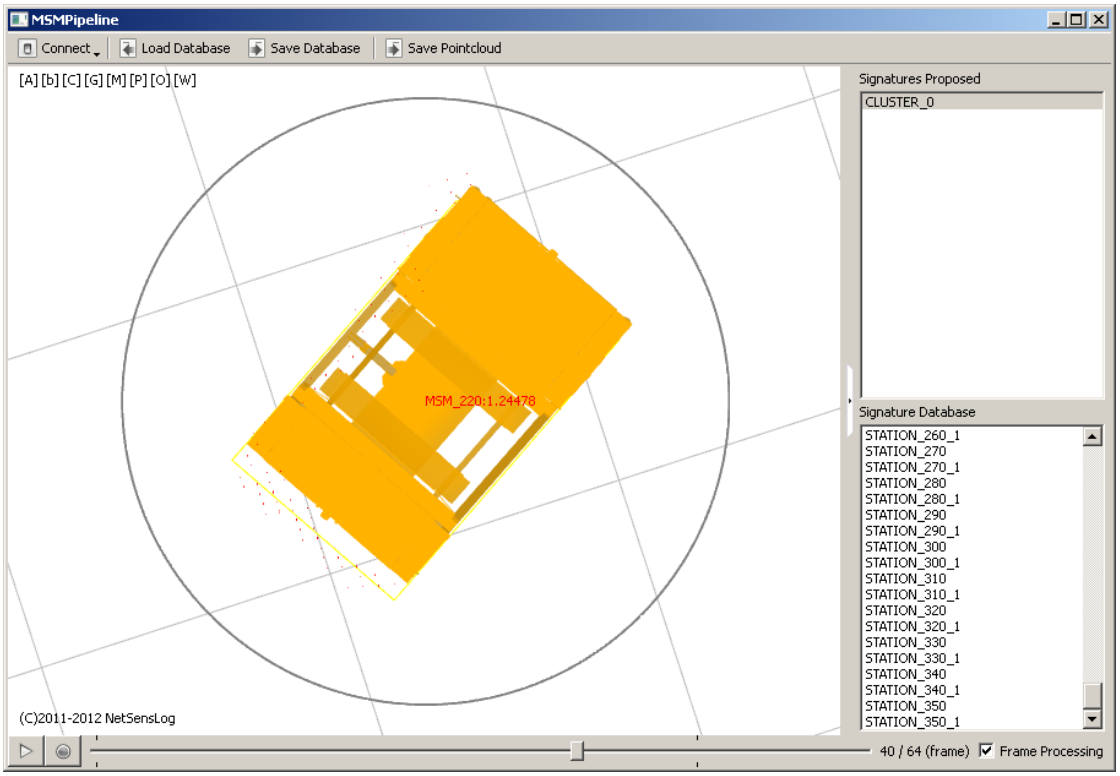


Abbildung 8.1.: Die Abbildung zeigt ein Bildschirmfoto der MSM-Pipeline Oberfläche.

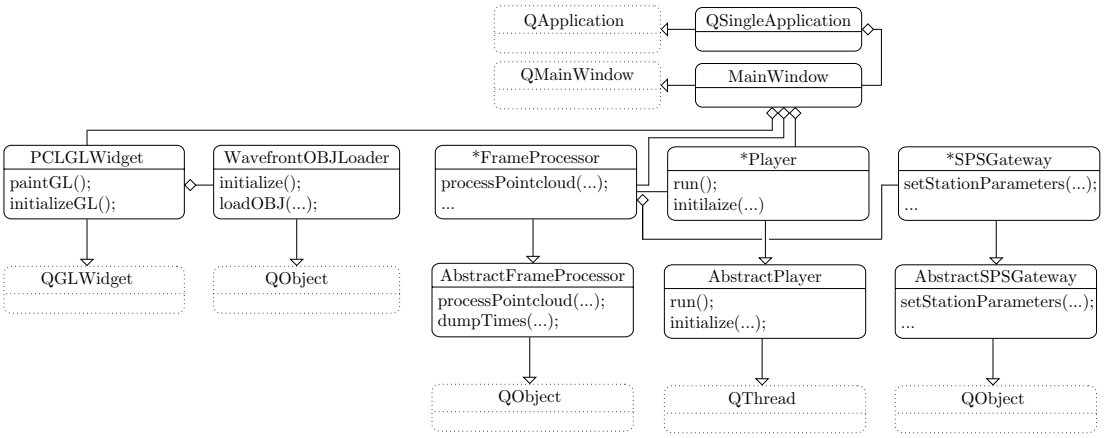


Abbildung 8.2.: Die Abbildung zeigt ein vereinfachtes Klassendiagramm der MSM-Pipeline.

Point Cloud Library

Zusätzlich wurde intensiv die `Point Cloud Library` (`PCL`) für die Verarbeitung und Interpretation der Tiefendaten verwendet. Die Point Cloud Library basiert wiederum auf folgenden Bibliotheken: `Boost`, `Eigen`, `Flann`, `QHull`, `VTK`.

Boost

Um eine nahtlose Integration zwischen `Qt` und der `Point Cloud Library` zu ermöglichen, wurde ebenfalls die Bibliothek `Boost` verwendet. Da es sich bei der `Point Cloud Library` und `Boost` um sehr Template-lastige Bibliotheken handelt, wurde zur Verkürzung der Kompilier- bzw. Link-Zeit ein Ansatz mit vorkompilierten Headern gewählt.

8.1.2. Akquisition der Punktwolken

Die Akquisition der 3D-Tiefendaten und die anschließende Umwandlung in ein Weltkoordinatensystem erfolgt durch eine zum verwendeten Sensor kompatible Instanz der Klasse `*Player`. Prinzipiell kann durch eine angepasste `*Player`-Klasse jeder beliebige Tiefensensor in die Grafik-Pipeline integriert werden. Die Daten werden anschließend in Form einer Punktwolke an die Instanz der Klasse `*FrameProcessor` weitergereicht. Diese übernimmt die Verarbeitung und Interpretation der Tiefendaten und propagiert die generierten Sekundärinformationen über die ihr zur Verfügung gestellten Kommunikationskanäle: z. B. Middleware, `SPS`. Ein schematischer Ablauf des im Rahmen der Projektgruppe implementierten `BasicFrameProcessor`s wird in Abbildung 8.3 gezeigt.

8.1.3. Vorverarbeitung

Den initialen Schritt der in Form des `BasicFrameProcessor`s realisierten Grafik-Pipeline bildet die Vorverarbeitung der generierten Tiefeninformationen. Hierzu werden zunächst die Rohdaten aufbereitet, bevor eine Extraktion der vorhandenen Cluster mit Hilfe des euklidischen Clusterings und die Klassifikation der gefundenen Cluster erfolgt.

Passthrough-Filter

Das Passthrough-Filtering (auch Clipping genannt) ermöglicht die Definition eines Ausschnitts bezüglich des gegebenen Weltkoordinatensystems der Punktwolke. Je nach Sensor sind die akquirierten Tiefendaten mit steigender Entfernung zu verrauscht für die weitere Verarbeitung, so dass ein Verwerfen dieser Daten notwendig ist. Bei der Microsoft Kinect wurde ein optimaler Operationsbereich von 0,5 m bis 7,0 m empirisch festgestellt,

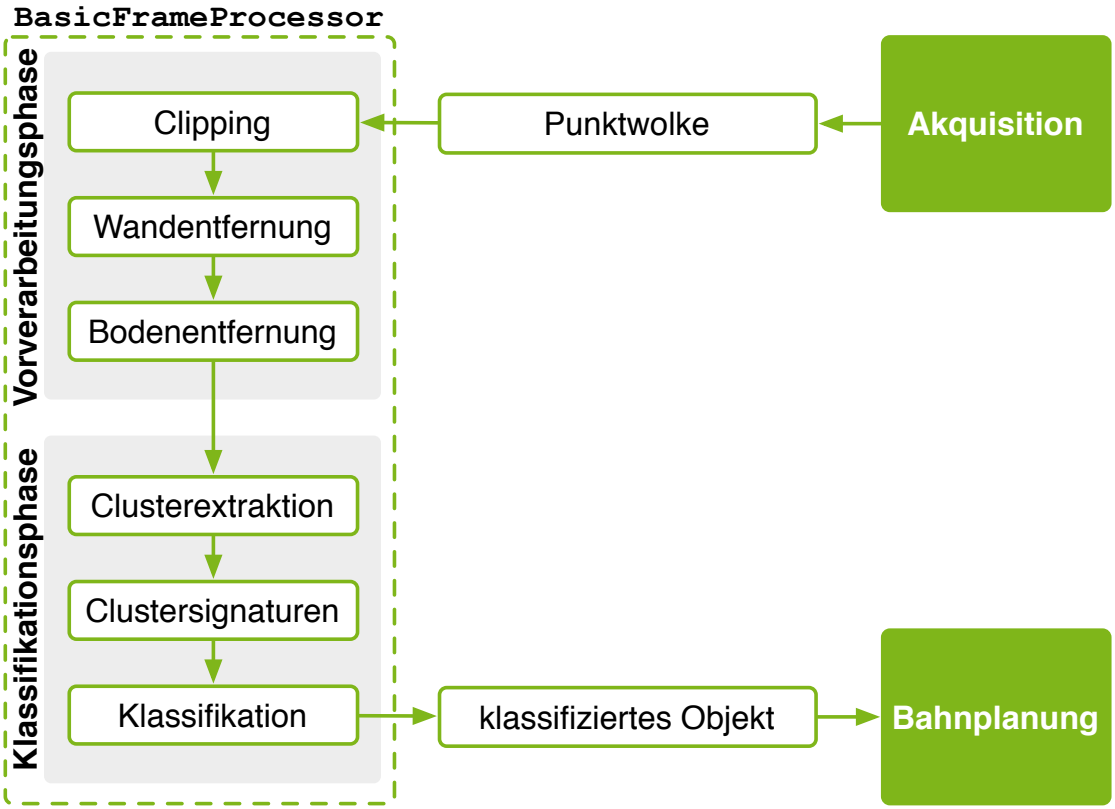


Abbildung 8.3.: Schematischer Aufbau der Grafik-Pipeline. Die Vorverarbeitungs- und Klassifikationsphase sind in Form des `BasicFrameProcessors` implementiert worden.

Listing 8.1: Algorithmus zum Passthrough-Filtering

```
procedure passthroughFiltering(pointcloud as PointCloud)
begin

    foreach point in pointcloud do

        if point.z < 0,5 or point.z > 7,0 then
            pointcloud.remove(point)
        end if

    end foreach

end
```

so dass für diesen Sensor das Clipping auf dieses Intervall festgelegt worden ist. Die Implementierung des Passthrough-Filters ist in Listing 8.1 zu sehen.

Es wurde ebenfalls die Entfernung von Ausreißern auf Basis statistischer Merkmale in Betracht gezogen. Da es sich hierbei jedoch um zumeist sehr aufwändige Operationen handelt, wurde im Hinblick auf die Laufzeiteffizienz auf eine Implementierung verzichtet. Sollte jedoch leistungsfähigere Hardware verwendet werden, so kann diese Option aktiviert werden. Die nächsten beiden Schritte der Grafik-Pipeline bilden die Wand- und Bodenentfernung. Ziel ist es hierbei einen Großteil unnötiger Punkte in Form von Wand- und Boden-Flächen zu verwerfen, so dass weniger Punkte in den nachfolgenden Schritten verarbeitet werden müssen. Die Erkennung von Wand- und Bodenflächen hat zusätzlich den Vorteil, dass die daraus gewonnen Informationen für die verbesserte Erkennung von Objekten genutzt werden können. Im Rahmen der Projektgruppe werden diese Informationen zur Verfeinerung der Objekt-Einpassung verwendet.

RANSAC Boden- und Wandentfernung

Zur Enfernung der Wände und des Bodens mit Hilfe des RANSAC-Algorithmus wurde der Algorithmus aus Kapitel 4.4.1. Die Implementierung besteht aus drei Teilen:

1. Reduktion der Punktwolke
2. Bestimmung der Ebenen
3. Entfernung der Punkte

Phase 1: Random Downsampling Um die Performance des Algorithmus zu erhöhen wird in der ersten Phase eine Reduktion der Punktwolke durchgeführt. Empirisch wurde ermittelt, dass eine Reduktion der Punktwolke um einen Faktor 10 ohne Einschränkung der Effektivität des Algorithmus erreicht werden kann.

Die Reduktion der Punktwolke erfolgt in zwei Schritten. Zunächst werden aus der Eingabepunkte P zufällig $0{,}1 \cdot |P|$ Punkte entnommen. Für jeden diese Punkt wird im folgenden Schritt entschieden, ob er im Suchbereich liegt. Der Suchbereich ist der Bereich im Raum in dem die nächsten Phase gesuchten Ebenen liegen. Der jeweilige Suchbereich für Wand- und Bodenentfernung wurde jeweils empirisch ermittelt. Für die Wandentfernung werden, wie in Listing 8.2 zu sehen, nur Punkte oberhalb einer Höhe von 1,5 m verwendet. Für die Bodenentfernung werden nur Punkte unterhalb des Sensorzentrums betrachtet.

Listing 8.2: Algorithmus zur RANSAC-Wandentfernung

```
procedure ransacWallRemoval(pointcloud as PointCloud)
begin
  var reduced_pointcloud as PointCloud = {}
  var walls as Ebenen = {}

  // Datenreduktion
  for i = 0 to |pointcloud|*0,1 do
      var point as Point3D := pointcloud.at(rand()*pointcloud.size)
      if point.y > 1,5 then
          reduced_pointcloud.add(point)
      end if
  end foreach

  // Bestimmung der Ebenen
  while |reduced_pointcloud| > w_min do
      var E as Ebene
      var inliers as PointCloud
      RANSACSegmentation(E, inliers)
      reduced_pointcloud := {x ∈ reduced_pointcloud | Distanz(x, E) < d_wall}
      walls.add(E)
  end while

  // Bodenpunkte entfernen
  foreach point in pointcloud do
    foreach wall in walls do
      if point.distanceToPlane(wall) < d_wall then
        pointcloud.remove(point)
      end if
  end foreach
end
```

Phase 2: Bestimmung der Ebenen Für die Erkennung der Wandebenen wurde der in Kapitel 4.4.1 beschriebene Algorithmus implementiert. Für die Bestimmung der jeweils am besten unterstützten Ebene wurde die RANSAC-Implementierung der Point Cloud Library verwendet. Für die unterstützenden Punkte wurde ein empirisch ermittelter Schwellwert von 0,1 m verwendet. Die RANSAC-Iterationen wurden auf 1000 begrenzt.

Nach dem Finden einer Ebene werden die Punkte die innerhalb einen Bereichs von 0,6 m zu beiden Seiten der gefundenen Ebene entfernt, um eventuelles Rauschen zu unterdrücken. Schließlich wird die Ebene gespeichert und der Vorgang wiederholt bis eine minimale Anzahl unterschritten wird. Diese wurde für die Kinect bei 50 Punkte festgelegt.

Der Algorithmus die Bodenentfernung läuft identisch ab. Allerdings wird in diesem Fall bereits nach der ersten gefundenen Ebene abgebrochen.

Phase 3: Entfernung der Punkte Im letzten Schritt werden alle Punkt in der Nähe der Ebenen aus der Original-Punktwolke entfernt. Hierzu wird über alle Punkte iteriert und die Entfernung jedes Punktes zu allen gefundenen Ebenen bestimmt. Punkte deren minimale Entfernung zu einer der Ebenen kleiner als 0,6 m ist, werden aus der Original-Punktwolke entfernt.

Bodenentfernung

Die Implementierung des Algorithmus (s. Listing 8.3) zur schnellen Bodenentfernung ist eine direkte Umsetzung des Ansatzes aus Abschnitt 4.4.2. Analog zum vorgestellten Methodik ist der Algorithmus ebenfalls ein drei Phasen unterteilt:

1. Bestimmung von drei Bodenpunkten
2. Berechnung der Normalen der Bodenfläche
3. Entfernung aller Bodenpunkte.

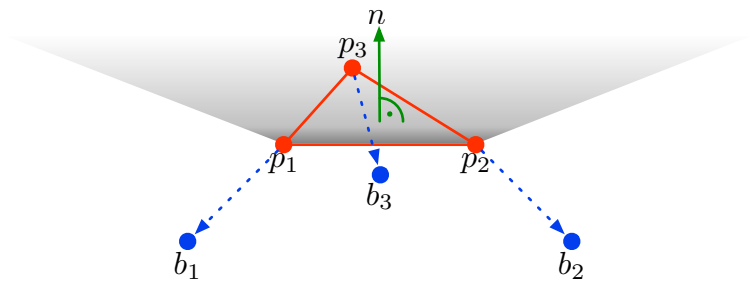


Abbildung 8.4.: Punktwolke mit dem Dreieck dass durch die Bodenpunkten $\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}$ aufgespannt wird.

Listing 8.3: Algorithmus zur schnellen Bodenentfernung

```
procedure fastGroundRemoval(pointcloud as PointCloud)
begin

    // Initialisierung
    var p_left as Point3D := (0, 10, 10)
    var p_back as Point3D := (0, 10, 10)
    var p_right as Point3D := (0, 10, 10)

    var b_left as Point3D := (1, −2, 0)
    var b_back as Point3D := (−1, −1, 0)
    var b_right as Point3D := (0, −2, 2)

    // Bodenpunkte bestimmen
    foreach point in pointcloud do

        if point.distanceTo(b_left) < p_left.distanceTo(b_left) then
            p_left := point
        end if

        if point.distanceTo(b_back) < p_back.distanceTo(b_back) then
            p_back := point
        end if

        if point.distanceTo(b_right) < p_right.distanceTo(b_right) then
            p_right := point
        end if

    end foreach

    // Normale bestimmen
    var n as Point3D = createNormal(p_left, p_back, p_right)

    if (n.y < 0) then
        n.toggleDirection
    end if

    // Bodenpunkte entfernen
    foreach point in pointcloud do

        if point.distanceToPlane(p_left, n) < 0,03 then
            pointcloud.remove(point)
        end if

    end foreach

end
```

Phase 1: Bestimmung von drei Bodenpunkten Für jede Punktwolke werden im ersten Schritt drei Bodenpunkte $\mathbf{p_1}, \mathbf{p_3}$ und $\mathbf{p_2}$ ermittelt. Durch diese drei Bodenpunkte wird eine Ebene definiert, die die tatsächliche Lage des Bodens approximiert. Zur Bestimmung dieser drei Punkt werden zunächst drei Hilfspunkte $\mathbf{b_1}, \mathbf{b_2}$ und $\mathbf{b_3}$ wie folgt definiert:

$$\mathbf{b_1} = \begin{pmatrix} 1 \\ -2 \\ 0 \end{pmatrix}, \mathbf{b_2} = \begin{pmatrix} -1 \\ -2 \\ 0 \end{pmatrix}, \mathbf{b_3} = \begin{pmatrix} 0 \\ -2 \\ 2 \end{pmatrix}. \tag{8.1}$$

Abhängig von diesen Hilfspunkten werden aus der Punktwolke die drei Punkte $\mathbf{p_1}, \mathbf{p_3}$ und $\mathbf{p_2}$ ausgewählt, so dass die folgenden Bedingungen erfüllt sind:

$$\mathbf{p_1} \in \{\mathbf{x} \in \text{Punktwolke} \mid \min(|\mathbf{b_1} - \mathbf{x}|)\} \tag{8.2}$$
$$\mathbf{p_2} \in \{\mathbf{x} \in \text{Punktwolke} \mid \min(|\mathbf{b_2} - \mathbf{x}|)\} \tag{8.3}$$
$$\mathbf{p_3} \in \{\mathbf{x} \in \text{Punktwolke} \mid \min(|\mathbf{b_3} - \mathbf{x}|)\}. \tag{8.4}$$

Entsprechend dieser Forderungen werden die Bodenpunkte so gewählt, dass sie einen minimalen Abstand zu einem der Hilfspunkte haben. Bedingung 8.2 definiert $\mathbf{p_1}$ als den Punkt der Punktwolke mit minimalem Abstand zu $\mathbf{b_1}$, $\mathbf{p_2}$ ist der Punkt mit minimalem Abstand zu $\mathbf{b_2}$ (s. Bedingung 8.3) und $\mathbf{p_3}$ der Punkt mit minimalem Abstand zu $\mathbf{b_3}$ (s. Bedingung 8.4).

Der Vorteil dieses Verfahrens wird in Abbildung 8.4 deutlich. Durch die Wahl der Hilfspunkte in 8.1 und den Bedingungen 8.2, 8.3 und 8.4 werden die Punkte $\mathbf{p_1}$ und $\mathbf{p_2}$ in die vorderen beiden Ecken des Bodentrapezes gezogen und $\mathbf{p_3}$ vom Ursprung weggezogen. Durch $\mathbf{p_1} - \mathbf{p_2}$ kann die Rotation des Bodens um die z-Achse erfasst werden, während von $\mathbf{p_3} - \mathbf{p_1}$ bzw. $\mathbf{p_3} - \mathbf{p_2}$ die Rotation um die x-Achse berücksichtigt. Zusätzlich wird auf diese Weise gewährleistet, dass bei leichten Rotationen um x und z, die drei Punkte $\mathbf{p_1}, \mathbf{p_3}$ und $\mathbf{p_2}$ nicht in einem Punkt zusammen fallen.

Phase 2: Berechnung der Normalen der Bodenfläche Durch $\mathbf{p_1}, \mathbf{p_3}$ und $\mathbf{p_2}$ wird eine Ebene definiert, die die Lage des Bodens innerhalb der Punktwolke approximiert. Um die Rotation der Bodenebene um alle drei Achsen (x, y, z) bestimmen zu können, wird die Normale n der Ebene berechnet. Der Normalenvektor wird wie in 4.4 definiert berechnet.

Ist n bestimmt worden, muss anschließend die Ausrichtung des Vektors überprüft werden. Zeigt n in Richtung fallender y-Werte, dann wird dieser durch Multiplikation mit -1 umgedreht. Der Vorteil der einheitlichen Ausrichtung von n wird in Abschnitt 8.1.3 genauer erläutert. Abschließend wird die Bodenebene durch einen Normalenvektor n repräsentiert, der senkrecht auf der Bodenebenen steht und in Richtung wachsender y-Werte zeigt.

Phase 3: Entfernung aller Bodenpunkte Im letzten Schritt werden alle Bodenpunkte aus der Punktwolke mit Hilfe der zuvor berechneten Bodenebene entfernt. Da die Bodenebene, die durch den Vektor n repräsentiert wird, potenziell innerhalb der Bodenpunkte in der Punktwolke liegt, wird diese zunächst leicht um einen Faktor ε Richtung des Normalenvektors n verschoben. Durch systematische Versuche wurde ein Verschiebungsfaktor von $\varepsilon = 0{,}03$ (3 cm) ermittelt. Ziel der Verschiebung ist es die Ebene oberhalb der Bodenpunkte zu positionieren und alle Punkte unterhalb der Ebenen aus der Punktwolke zu entfernen.

Abschließend muss nun Formel 4.6 für alle Punkte x der Punktwolke ausgewertet werden. Abhängig vom ermittelten Wert d (s. Formel 4.7) kann nun für jeden Punkt die Lage relativ zur Ebene bestimmt werden. Zusammen mit dem Verschiebungsfaktor, wird ein Punkt genau dann aus der Punktwolke Entfernt wenn gilt: $d < \varepsilon = 0{,}03$.

Voxelgrid-Downsampling

Nach erfolgter Entfernung von Wand- und Bodenflächen erfolgt ein Dichtenausgleich bzgl. der Punktwolke, das so genannte Voxelgrid-Downsampling. Dieser hat den Vorteil, dass übersättigte (nah beim Sensor liegende Flächen) ausgedünnt werden können. Die Verrasterung der Punktwolke und Mittelung der in die jeweiligen Rasterzellen fallenden Punkte trägt zusätzlich zur Verminderung von Akquisition-Artefakten bei, so dass den nachfolgenden Schritten der Grafik-Pipeline rauschärmere Daten zur Verfügung stehen. Über die Raster-Granularität lässt sich zudem der Grad der Reduktion der Punktwolke einstellen. Im Rahmen der Projektgruppe wurde eine geeignete Rastergröße von 2 cm × 2 cm × 2 cm empirisch bestimmt.

8.1.4. Objekterkennung

Die im Rahmen der Projektgruppe realisierte Objekterkennung basiert auf der Bestimmung von Clustern und ihrer Annotation. Als Extraktionsverfahren wurde hierzu das euklidische Clustering gewählt. Anschließend wird für jedes Cluster eine VFH-Signatur berechnet und die jeweils bzgl. des euklidischen Abstandes nächste Signatur aus der Datenbank bestimmt. Bezüglich der Distanz (auch Score genannt) lässt sich ein Schwellwert definieren, über den sich Klassifikationsentscheidungen annehmen oder aber auch verwerfen lassen.

Euklidisches Clustering

Für die Extraktion der euklidischen Cluster (s. Abschnitt 4.5.1) wurden empirisch geeignete Parameter r 0,1 m und die minimale Clustergröße von mindestens 50 Punkten bestimmt. Die Implementierung in Listing 8.4 kann in drei Phasen gegliedert werden:

1. Initialisierung

Listing 8.4: Algorithmus zum euklidischen Clustering

```
procedure euclidianClustering(pointcloud as PointCloud)
begin

    var r as integer
    r := 0.1 // Radius auf 0,1 Meter setzen

    var pointTree as kdTree(pointcloud) // Punktwolke wird in k−d−Baum verwaltet

    var clusterList as List // Liste aller gefundenen Cluster (initial leer)

    var q as Queue // Warteschlange zur Aufsammlung aller Clusterpunkte (initial leer)

    foreach point in pointcloud do

        q.add(point)

        foreach point' in q do

            // alle Punkte im Umkreis ermitteln
            var pointsInRange as PointList
            pointsInRange := pointTree.findNeighbours(point', r)

            q.add(pointsInRange)

        end foreach

        // alle Punkte der Warteschlange als Cluster zusammenfassen
        var cluster as Cluster(q)

        // Cluster in die Cluster−Liste eintragen
        var clusterList.add(cluster)

        // Warteschlange leeren
        q.clear()

    end foreach

    // zu kleine Cluster werden aus der Punktwolke entfernt
    foreach cluster in clusterList do
        if cluster.size() < 50 then
            pointcloud.remove(cluster.points())
        end if
    end foreach

end
```

2. Bestimmung der Cluster
3. Nachverarbeitung

Phase 1: Initialisierung Vor Beginn des eigentlichen Clusterings werden zunächst die folgenden Parameter und Datenstrukturen initialisiert:

- Der Radius `r` wird mit 0,1 m initialisiert.
- `pointTree` repräsentiert einen k-d-Baum, der mit den Punkten der Punktwolke initialisiert wird. Mit Hilfe des k-d-Baum können während der Clustering-Phase effizient die Nachbarn eines Punktes mit maximalem Abstand r ermittelt werden.
- In `clusterList` werden alle Cluster aufgesammelt.
- Mit Hilfe der Schlange `q` werden alle direkten Nachbarn eines Punktes aufgesammelt.

Phase 2: Bestimmung der Cluster Entsprechend der Methodik aus Abschnitt 4.5.1 werden nun agglomerativ die Cluster bestimmt. In Listing 8.4 ist dies mit Hilfe eines k-d-Baums `pointTree` und einer Warteschlange `q` realisiert worden. Die Äußere `foreach`-Schleife iteriert über alle Punkte der Punktwolke und fügt jeden Punkt zunächst in `q` ein. In der nachfolgenden `foreach`-Schleife werden nun für jeden Punkt in `q` die direkten Nachbarpunkte bestimmt. Dies kann effizient mit Hilfe des k-d-Baums vollzogen werden. Alle gefundenen Nachbarn werden ebenfalls in `q` eingefügt und in den nachfolgenden Iterationen der inneren Schleife nach dem selben Prinzip abgearbeitet. Diese Schleife läßt somit einen Cluster um den Punkt `point` wachsen. Nachdem `q` vollständig abgearbeitet wurde und somit kein Weitere Punkt diesem Cluster zugeordnet werden kann, werden diese Punkt als zusammenhängender Cluster in der Liste `clusterList` abgelegt und `q` wird geleert. Anschließend wird in der äußeren Schleife mit dem nächsten Punkt fortgefahren. Schließlich sind nach dem Durchlauf beider Schleifen alle berechneten Cluster in `clusterList` abgelegt worden.

Phase 3: Nachverarbeitung Abschließend werden alle agglomerativ berechneten Cluster in `clusterList` auf eine Mindestgröße von 50 Punkten überprüft. Alle Punkte die zu Clustern gehören, die unterhalb dieser Schranke liegen, werden aus der Punktwolke entfernt. Durch diese quantitative Vorauswahl wird abhängig von der akquirierten Punktwolke der Aufwand für die nachfolgende Klassifikation verringert.

Viewpoint-Feature-Histogramm und Nächster-Nachbar-Klassifikation

Für jede bereits geclusterte Punktwolke wird mit Hilfe des in Kapitel 4.5.2 erläuterten Algorithmus eine Signatur mit 308 Dimensionen berechnet welche als Eingabe für die Nächster-Nachbar-Klassifikation dient. Zur effizienten Bestimmung des nächsten Nachbarn einer noch nicht annotierten Signatur bzgl. einer gegebenen Signaturdatenbank

empfiehlt sich die Verwendung einer Suchdatenstruktur. Problematisch ist hierbei die große Dimensionalität der VFH-Signatur. Denn Anfragen bzgl. herkömmlicher Suchdatenstrukturen degenerieren mit steigender Dimension zur linearen Suche (das Phänomen ist auch bekannt als der Fluch der Dimensionen). Eine Alternative bieten hierbei approximative Ansätze. Im Rahmen der Projektgruppe wurde hierzu ein Ansatz von Lowe auf Basis der `FLANN`-Bibliothek gewählt [ML09, Muj11].

8.2. Lokale Bahnplanung

Die lokale Bahnplanung ist, genau wie die Implementierung der Grafikpipeline, als Einzelinstanz-Anwendung realisiert. Allgemein gehen in den Bahnplanungsprozess die Daten der von der Grafikpipeline klassifizierten Hindernisse sowie die von der SoftSPS ermittelten Posen- und Zielposen-Daten ein. Die Ausgabe des Bahnplanungsprozesses besteht aus zwei normierten Motorgeschwindigkeiten im Wertebereich $[-1; 1]$, die der Fahrzeugsteuerung übergeben werden.

Zur Realisierung der Eingabe der klassifizierten Hindernisdaten ist die Bahnplanung über die eigens entwickelte Middleware mit der Grafikpipeline verbunden, welche Datenpakete bestehend aus Position, Winkel und Radius eines Hindernisses über einen Shared-Memory überträgt. Die Kommunikation zwischen der Fahrzeugsteuerung und dem Bahnplanungsprozess findet entsprechend Abschnitt 8.5.2 statt.

8.2.1. Hauptmethode

Kern der Bahnplanungsprozess-Implementierung stellt eine Methode dar, deren Ausführung von der Fahrzeugsteuerung zyklisch in einer Frequenz von 100 Hz getriggert wird. Dies ist zum Erreichen einer Echtzeitfähigkeit, welche durch die Implementierung der Fahrzeugsteuerung auf einer SoftSPS gewährleistet werden soll, vorteilhaft.

Auf Grund der Begebenheit, dass die Ausführungsfrequenz der Hauptmethode der Bahnplanung mit 100 Hz deutlich höher als die Hindernis-Erkennungsrate der Grafikpipeline von etwa 4 Hz ist, werden erkannte Hindernisse im Bahnplanungsprozess zwischengespeichert.

Listing 8.5 zeigt den prinzipiellen Ablauf der Hauptmethode. Zunächst werden die Eingabeparameter aus der Fahrzeugsteuerung gelesen und eine virtuelle Zielposition berechnet sowie die global gespeicherten Informationen über Hindernisse aktualisiert, sofern von der Grafikpipeline neue Datenpakete vorliegen. Mit diesen Informationen wird anschließend über das eigentliche Bahnplanungsverfahren die einzuschlagende Fahrtrichtung berechnet und auf Grundlage dieser die normierten Motorgeschwindigkeiten ermittelt und an die Fahrzeugsteuerung übergeben. In den folgenden Abschnitten werden diese Einzelschritte detaillierter beschrieben.

Listing 8.5: Hauptmethode der Bahnplanung

```
procedure PerformMotionplanning;

var
    p, pTarget : Pose;
    virtualTarget : Position;
    targetAngle : Real;
    targetDistance : Real;

begin
    p := GetCurrentPose();
    pTarget := GetCurrentTargetPose();

    UpdateObstacles();

    virtualTarget := CalculateVirtualTarget(p, pTarget);
    targetDistance := CalculateEuclideanDistance(p, virtualTarget);
    targetAngle := CalculateTargetAngle(p, targetDistance, virtualTarget);

    UpdateMotorSpeeds(targetDistance, targetAngle);
end;
```

8.2.2. Berechnung der virtuellen Zielposition

Die Berechnung der virtuellen Zielposition erfolgt nach dem in Abschnitt 5.5 beschriebenem Vorgehen. Der Betrag der Verschiebung wird dabei nach oben auf einen Maximalwert von drei Metern beschränkt und linear in Abhängigkeit vom Abstand des Fahrzeugs zum Ziel berechnet.

8.2.3. Bahnplanungsverfahren

Auf Grund der Gegebenheit, dass das eigentliche Bahnplanungsverfahren lediglich auf Grundlage der aktuellen Fahrzeugposition, der zu erreichenden Zielposition sowie dem Abstand zu dieser Zielposition einen einzuschlagenden Winkel berechnen muss, lässt sich diese Aufgabe prinzipiell von verschiedenen Verfahren ohne Änderung der Rahmenbedingungen lösen. Da das Verfahren der künstlichen Potentialfelder eine sehr starke Abhängigkeit von den gewählten Parametern aufweist wird im Prototyp das VFH-Verfahren verwendet.

8.3. Visualisierung

Im Rahmen der Entwicklung des Prototypen sind zwei verschiedene Visualisierungskomponenten entstanden. Zum einen handelt es sich dabei um eine visuelle Simulationssoftware

und zum anderen um eine Echtzeitvisualisierung der Zustände der Prototypenfahrzeuge während sie sich das Fahrzeug in der Versuchshalle bewegt.

8.3.1. FTS-Simulator

Zur Entwicklung und zum Testen der gewählten Methoden für die lokale Bahnplanung und die Vermeidung von Kollisionen mit Hindernissen wurde eine Simulationssoftware (FTS-Simulator) implementiert. Dieses Werkzeug ist in der Lage, eine maßstabsgetreue schematische Abbildung der Versuchshalle darzustellen und sowohl Fahrzeuge als auch Hindernisse, wie etwa Kisten, zu simulieren. Diese Objekte werden an den entsprechenden Koordinaten in die schematische Halle eingezeichnet. Die simulierten Fahrzeuge werden auf der Fläche in Abhängigkeit von einstellbaren Motorgeschwindigkeiten bewegt.

Simulation des Fahrzeugverhaltens

Um die Auswirkungen der Verfahren auf das Verhalten der simulierten Fahrzeuge möglichst ohne Abweichung auf die realen Fahrzeuge übertragen zu können, wurden die Verfahren nicht im Rahmen des Simulators neu implementiert. In dem in Abschnitt 8.2 näher beschriebenen Bahnplanungsprozess sind alle Schnittstellen zu externen Komponenten wie der Motoransteuerung oder der Grafikpipeline in Form von abstrakten Klassen realisiert. Für diese abstrakten Klassen existiert eine Implementierung als Schnittstelle zum FTS-Simulator. Der Bahnplanungsprozess kann somit lokal auf einem Simulationsrechner zusammen mit dem Simulator ausgeführt werden und sein Verhalten simuliert werden.

8.3.2. Echtzeitvisualisierung

Zur Überwachung der Fahrzeugparameter während der Tests und zur nachträglichen Auswertung der Testfahrten, wurde innerhalb der Projektgruppe das Echtzeitvisualisierungstool *Vehicle Monitoring Software* (VMS) erstellt. Die Anforderungen die an dieses Tool gestellt wurden, beinhalteten eine 2D-Visualisierung der Position aller in der Versuchshalle befindlichen, für die Projektgruppe relevanten Fahrzeuge, eine Ausgabe von zusätzlichen, relevanten Fahrzeuginformationen für ein gewähltes Fahrzeug und eine Aufnahme- und Abspielfunktion für die nachträgliche Auswertung.

Akquisition der Fahrzeugzustände

Um die Zustände der einzelnen Fahrzeuge zu akquirieren, sendet jedes am Fahrbetrieb teilnehmende Fahrzeug mehrmals in der Sekunde seine Zustandsinformationen über die Middleware (siehe Abschnitt 6.3) an das Netzwerk. An jeder beliebigen Stelle können nun diese Daten passiv abgefragt werden und wie von der *Vehicle Monitoring Software* weiterverarbeitet werden. Diese Konstruktion ermöglicht, dass die VMS an beliebigen

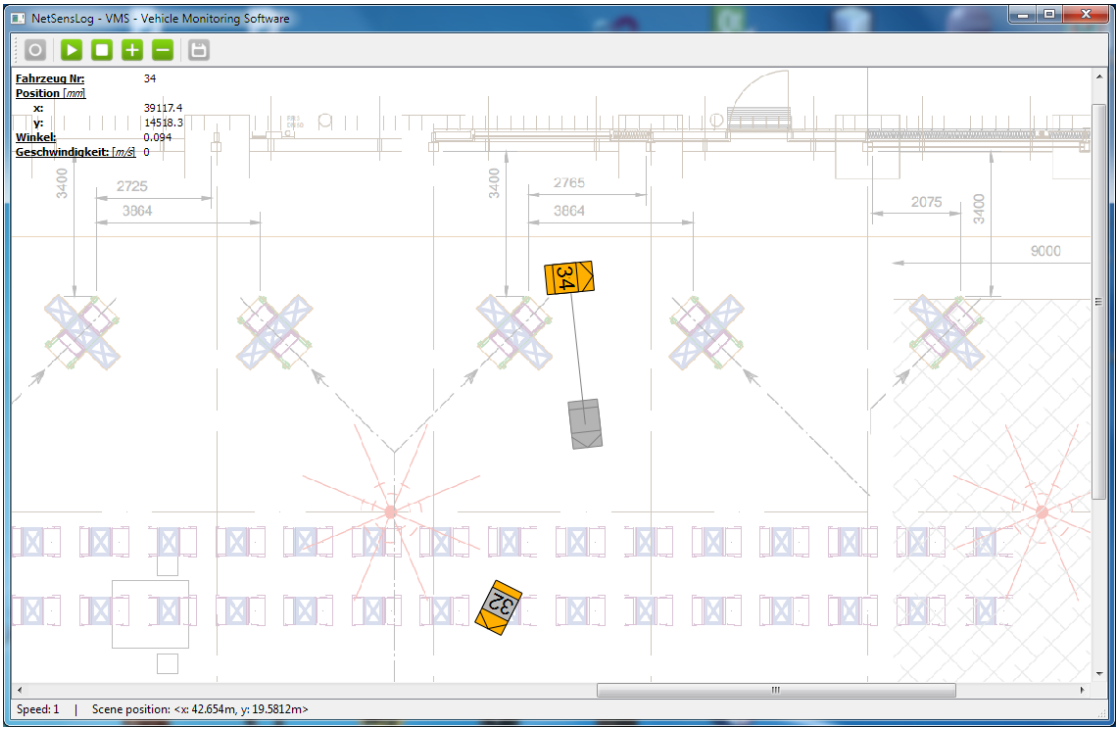


Abbildung 8.5.: Die Vehicle Monitoring Software (VMS) mit zwei visualisierten Fahrzeugen

Stellen im Netzwerk ohne Konfiguration der Sende- oder Empfangsstelle ausgeführt werden kann und zudem Ausfälle einzelner Sender oder neue Teilnehmer am Fahrbetrieb unkompliziert und akkurat detektiert werden.

Einbindung erweiterter Informationen

Neben der Visualisierung der Fahrzeugposition in der Halle und der Unterscheidung der einzelnen Fahrzeuge durch die Anzeige der Fahrzeugnummer ist bei jedem angezeigten Fahrzeug der Beladungszustand, also ob das Fahrzeug einen Transportbehälter mitführt oder nicht, durch Farbinformationen in der Darstellung ersichtlich.

Wird ein Fahrzeug in der VMS markiert, werden für das ausgewählte Fahrzeug neben den genauen Positionskoordinaten außerdem der aktuelle Winkel und die Geschwindigkeit in Echtzeit im oberen linken Fensterrand angezeigt und die Zielpose des Fahrzeuges dargestellt.

Aufnahme- und Abspielfunktion

Um Daten von Versuchsfahrten zu akquirieren und für spätere Analysen zu speichern, verfügt die Software über eine Aufnahmefunktion mit der die kompletten von VMS empfangenen Daten in einer Datei gespeichert werden. Bei der Datei handelt es sich um eine einfache CSV-Datei, so dass die Daten auch leicht von anderen Programmen verwendet werden können. Außerdem können die gespeicherten Daten von der *Vehicle Monitoring Software* geladen und in variabler Geschwindigkeit abgespielt werden. Das Programm wechselt dabei jeweils automatisch zwischen einem Live-Modus und einem Abspiel-Modus.

8.4. Zeitsynchronisation

Für die Zeitsynchronisation wird die NTP-Implementierung der Firma Meinberg [Mei12] verwendet. Hierzu wird die Software auf jedem Fahrzeug installiert und für die Synchronisation mit einem Zeitserver konfiguriert. Die Konfiguration wird dabei so gewählt, dass jedes Fahrzeug alle 16 s den eingetragenen Zeitserver abfragt.

8.5. Anpassung der Fahrzeugsteuerung

Damit die im Rahmen der Projektgruppe erstellte Bahnplanung und Grafik-Pipeline mit der Fahrzeugsteuerung auf dem Fahrzeug automatisch gestartet werden und mit der

Steuerung kommunizieren können, mussten ein paar Anpassungen an der Fahrzeugsteuerung vorgenommen werden. Die vorgenommenen Anpassungen werden in den folgenden Abschnitten beschrieben.

8.5.1. Starten der Grafik-Pipeline

Damit die Grafik-Pipeline bei der Inbetriebnahme des Multishuttle Move automatisch gestartet wird, wurde in der Fahrzeugsteuerung eine entsprechende Routine eingebaut. Diese Routine startet die Grafik-Pipeline immer dann, wenn auch die Fahrzeugsteuerung gestartet wurde. Also beim Einschalten des Fahrzeugs oder auch nach dem Zurücksetzen der Fahrzeugsteuerung.

8.5.2. Ausgliedern der Bahnplanung

Die ursprüngliche Bahnplanung des Multishuttle Move war direkter Bestandteil der Fahrzeugsteuerung.

Um die von der Projektgruppe erstellte Bahnplanung flexibler gestalten zu können und dabei auf die entwickelte Middleware und Informationen der Grafik-Pipeline zurückgreifen zu können, wurde die Bahnplanung aus der Fahrzeugsteuerung ausgegliedert.

Innerhalb der Fahrzeugsteuerung ist eine Schnittstelle entstanden, die der Bahnplanung Informationen über den anzufahrenden Punkt bereitstellt. Zudem wertet sie die von der Bahnplanung übermittelten Steuerungsinformationen aus und setzt sie auf dem Fahrzeug um.

Genau wie die Grafik-Pipeline wird auch die Bahnplanung zusammen mit der Fahrzeugsteuerung automatisch gestartet.

8.5.3. Interne Anpassungen der Fahrzeugsteuerung

Neben der Ausgliederung der Bahnplanung und dem Starten der benötigten Programme sind weitere interne Anpassungen vorgenommen worden.

Alle vorhandenen Fahrbefehle wurden so modifiziert, dass sie die neue Bahnplanung verwenden. Damit ein Absturz der Bahnplanung nicht dazu führt, dass das Fahrzeug unkontrolliert weiterfährt, wurde zudem eine Art Totmannschalter integriert. Wenn die Bahnplanung nicht regelmäßig eine bestimmte Variable ändert, löst das Fahrzeug automatisch eine Bremsung aus und bleibt solange stehen, bis die Bahnplanung wieder korrekt arbeitet.

9. Evaluation

In diesem Kapitel werden die Evaluationsergebnisse der einzelnen Komponenten des Systems vorgestellt. Dazu werden in den folgenden Unterkapiteln die einzelnen, in der Projektgruppe entworfenen Komponenten untersucht. Kapitel 9.2 evaluiert die einzelnen Verarbeitungsschritte der Grafikpipeline sowie deren Eignung für die Objekterkennung und Klassifikation. Im Kapitel 9.3 werden verschiedene Testszenarien der Bahnplanung beschrieben und die bei diesen Tests ermittelten Ergebnisse erläutert. Das Kapitel 9.4 beschäftigt sich mit der Zeitsynchronisation und evaluiert verschiedene Synchronisationsverfahren und deren Eignung in der Projektgruppe. Zuletzt wird in Kapitel 9.5 auf die Problematik der Beeinflussung mehrerer Sensoren auf kleinem Raum eingegangen und das Ausmaß der gegenseitigen Störung der Sensoren beschrieben.

Die in diesem Kapitel beschriebenen Tests wurden durchgeführt, um eine Aussage treffen zu können, wie Erfolgreich der Prototyp die verschiedenen Ziele der Projektgruppe erreicht hat. Dazu wurden umfassende Testszenarien entwickelt, die die einzelnen Komponenten des Prototyps sowohl in sehr typischen Situationen, als auch unter Extrembedingungen untersuchen. Dies ermöglicht es, Schwachstellen zu erkennen und gut funktionierende Teile des Systems zu identifizieren.

9.1. Allgemeiner Versuchsaufbau

Für die folgende Evaluation der Einzelkomponenten und des Gesamtsystems wurde die Zellulare Fördertechnik Halle des Fraunhofer IML genutzt. Eine schematische Darstellung der Halle wird in Abbildung 9.1 gezeigt, weitere Details zur Versuchshalle finden sich in Kapitel 2.3. Alle zur Evaluation genutzten Aufnahmen wurden von den mit zusätzlichen Sensoren ausgestatteten Multishuttle Move Fahrzeugen aufgenommen. Um die Fahrzeuge mit einer Microsoft Kinect als zusätzlichen Sensor und einem ION-Rechner aus zu statten wurde eine speziell für diesen Zweck angefertigte Fahrzeugfront genutzt welche es dem Fahrzeug weiterhin Ermöglicht in das Lagerregal ein zu fahren. Außerdem musste die Stromversorgung der Fahrzeuge angepasst werden um Platz für den neuen Rechner und die Kinect zu schaffen sowie den Sensor mit Strom zu versorgen. Als weiterer Sensor wurde eine ASUS Xtion in ein Versuchsfahrzeug integriert welche auf Grund ihrer wesentlich kompakteren Bauform vollständig in eine dafür angefertigte Fahrzeugfront integriert werden konnte ohne über das Fahrzeug hinaus zu ragen. Weitere Informationen zur Grundausstattung der Fahrzeuge findet sich in Kapitel 2.2. Detaillierte Beschreibungen

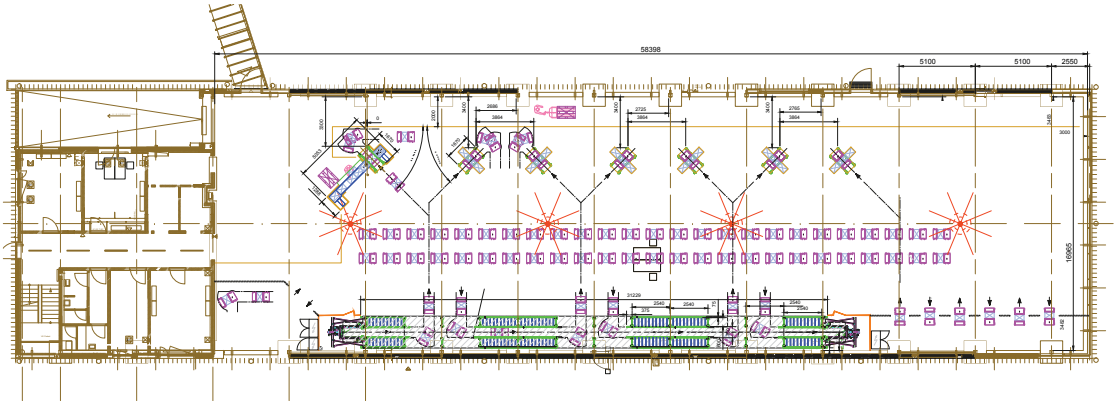


Abbildung 9.1.: Die Abbildung zeigt die ZFT-Halle in Draufsicht mit Bemaßungen, u. a. unter Berücksichtigung der Positionen der Kommissionierstationen. Die vier an der Decke angebrachten Deckenkameras (in der Abbildung • hervorgehoben) weisen sich partiell überlappende Sichtbereiche auf. Angedeutet sind ferner mehrere Fahrerlose Transportsysteme (• hervorgehoben), die u. a. beim Einfädelungsvorgang in die Kommissionierstationen gezeigt werden.

des Versuchsaufbaus der einzelnen evaluierten Komponenten finden sich in den folgenden Unterkapiteln.

9.2. Grafik-Pipeline

In den folgenden Kapiteln werden die einzelnen Komponenten der Grafik-Pipeline evaluiert. Zunächst wird dazu in Kapitel 9.2.1 auf die Möglichkeit der Auslagerung der Grafikpipeline auf die Grafikkarte eingegangen. Danach werden nacheinander die in der Grafikpipeline vorhandenen Verarbeitungsschritte untersucht. Das Kapitel 9.2.2 geht auf die Genauigkeit der Wanderkennung und -entfernung unter verschiedenen realitätsnahen Bedingungen ein. In Kapitel 9.2.3 werden die beiden zur Verfügung stehenden Bodenentfernungsmethoden, die RANSAC- und die schnelle Bodenentfernung gegenübergestellt und die Leistungsfähigkeit der Verfahren verglichen. Die Qualität des Clusterings und vor allem das Problem des Zerfalls von zusammenhängenden Clustern in ungünstigen Situationen wird in Kapitel 9.2.4 evaluiert. Das Kapitel 9.2.5 geht auf den letzten Verarbeitungsschritt, die Klassifikation, ein und zeigt die Ergebnisse der Klassifikation und somit die des gesamten Verfahrens auf.

9.2.1. OpenCL

Auf Grund der knappen Resourcen auf den Fahrzeugen ist getestet worden, ob die Grafikpipeline mit Hilfe von OpenCL anstatt auf dem Hauptprozessor auf der Grafikkarte ausgeführt werden kann. Die größten Vorteile der Auslagerung liegen darin, dass der Hauptprozessor entlastet und mehr Rechenzeit für andere Programme wie die Middelware, die Bahnplanung etc. zur Verfügung stehen würde. Außerdem könnte die große parallele Rechenleistung der Grafikkarte für die komplexen Berechnungen der Grafikpipeline ausgenutzt werden. Die Pipeline ist auf Grund des großen Portierungsaufwandes der nicht parallelen Algorithmen letztendlich nicht auf der Grafikkarte umgesetzt worden, jedoch sind einige Tests durchgeführt worden, ob sich die gegebene Hardware für eine Portierung eignen würde.

Versuchsaufbau

Die Fahrzeuge wurden jeweils mit einem ION-Chipsatz nachgerüstet. Dieser besitzt einem Intel Atom 330 Prozessor, welcher selbst wiederum aus zwei Kernen besteht die mit 1,6 GHZ getaktet ist. Diesem Prozessor steht ein GeForce9400M Grafikchip von NVIDIA zur Seite, der aus zwei Streaming Multiprozessoren mit je acht Streaming Prozessoren besteht. Damit kommt dieser Grafikchip auf eine theoretische Rechenleistung von 54 Gfps. Der Stromverbrauch des Atom und des GeForce Kerns ist mit 8 bzw. 12 W TDP angegeben und damit für den mobilen Bereich einigermaßen gut geeignet. Weitere technische Daten zu der verwendeten Hardware kann dem Anhang B.1 entnommen werden. [Int11, NVI11]

OpenCL-Tests auf dem ION

Um zu entscheiden, ob es einen Geschwindigkeitsvorteil bringt bei der vorhandenen Hardware die Grafikpipeline auf der Grafikkarte zu implementieren sind verschiedene Testreihen durchgeführt worden. Zum einen sollte herausgefunden werden, wie schnell einzelne Operationen im Parallelbetrieb auf der Grafikkarte berechnet werden können und zum Anderen sollte überprüft werden, wie lange der Transfer von Daten im Hauptspeicher auf die Grafikkarte sowie von der Grafikkarte zurück in den Hauptspeicher dauert. Nur wenn sowohl die Rechengeschwindigkeit als auch die Transferraten hoch genug sind, ist es möglich, die geforderte Anzahl von mindestens vier zu verarbeitenden Bildern die Sekunde zu erreichen.

Das Messen des Zeitbedarfs eines Kernels, der auf der Grafikkarte ausgeführt wird, ist nicht ganz einfach, da der Kernel parallel zum Hauptprogramm ausgeführt wird. Daher müssen entweder spezielle Profilingprogramme genutzt werden oder es werden Befehle benutzt, die das Hauptprogramm so lange blockieren, bis der Kernel auf der Grafikkarte vollständig abgearbeitet worden ist. Tests haben ergeben, dass beide Methoden in etwa

die gleichen Ergebnisse liefern. Daher wurde auf Grund der einfacheren Handhabbarkeit auf die Methode mit dem blockierenden Hostprogramm zurückgegriffen.

Die selben Testreihen sind sowohl auf dem ION, also dem Rechner, der in den Fahrzeugen verbaut ist, durchgeführt worden, sowie auf einem Desktoprechner, so dass ein Vergleich zwischen den Rechenleistungen zur Verfügung steht.

Basisoperationstests

Bei diesem Testfall sollte ermittelt werden, wie schnell verschiedene elementare Operationen auf der Grafikkarte ausgefürht werden können verglichen mit der Ausführungszeit der selben Operationen auf dem Hauptprozessor. Dabei muss beachtet werden, dass im Gegensatz zu den Berechnugen auf dem Hauptprozessor die Grafikkarte darauf ausgelegt ist, mehrere gleichartige Operationen parallel auszuführen, so dass dies explizit bei den Tests berücksichtigt wird. Zu unterscheiden ist außerdem, auf welche Art von Daten die Berechnungen angewendet werden. So werden Operationen auf Integerdaten auf anderen Ausführungseinheiten der Grafikkarte ausgeführt als Fließkommadaten. Daher werden die Tests je zweifach durchgeführt, um beide Fälle abzudecken.

Zunächst wurden Kernel geschrieben, die ausschließlich eine Art von elemtaren Operation ausführen sollten, wie zum Beispiel Additionen, Subtraktionen und so weiter sowie für Grafikanwendungen häufig benötigte trigonometrische Funktionen wie Sinus- oder Wurzelberechnungen. Die Daten, die für die Berechnungen genutzt wurden, sind randomisierte Zahlen, die zu Anfang des Tests auf die Grafikkarte transferiert wurden. Um möglichst nur die reine Rechenzeit des Kernels zu messen und dieses Ergebnis nicht durch Speichertransferzeiten von dem Grafikspeicher zum Streamingprozessor zu überdecken, wird nur ein kleiner Datensatz geladen und dann viele Operationen des gleichen Typs ausgeführt. Auf diese Weise sollte die Zeit zum Ausführen der Berechnung einen wesentlich höheren Anteil an der Gesamtausführungszeit des Kernels haben als die Speicheroperationen.

Zunächst sollen die Ergebnisse der Tests auf dem ION aufgeführt werden. Die Tabellen 9.1 und 9.2 zeigen diese Ergebnisse. In der Spalte „CL GPU“ werden die Zeiten für die Berechnungen auf der Grafikkarte aufgelistet. Die Spalte „Host CPU“ gibt die Werte an, die für die äquivalenten Tests auf dem Hauptprozessor gemessen wurden. Zuerst wurde gemessen, wie lange es dauert, einen Kernel zu kompilieren und auf die Grafikkarte zu transferieren. Da dieser Prozess nur einmal während der Initialisierung durchgeführt werden muss, spielt der Wert für den laufenden Betrieb keine große Rolle. Die Werte der Zeile „Write Buffer“ geben an, wie lange der Transfer der Zufallszahlen auf die Grafikkarte dauert. An dieser Stelle war eine Blockierung des Hostprogramms nicht möglich, so dass die Werte eventuell nicht korrekt sind. Die Zeitmessung durch Blockieren des Hauptprogramms für das Lesen eines Puffers ist jedoch möglich, so dass die Transferzeit an diesen Werten abgelesen werden kann. Insgesamt sind die Datenmengen in diesem Test jedoch zu klein, um aussagekräftige Werte im Bereich der Messgenauigkeit (im Millisekundenbereich) zu liefern. In den nächsten Zeilen sind die Zeiten für die Operationen

Device:	CL GPU	Host CPU
Kernel laden	11	–
Buffer schreiben	0*	–
10^6 Additionen	0**	38
10^6 Subtraktionen	0**	39
10^6 Multiplikationen	126	115
10^6 Divisionen	546	1241
10^6 Sinusoperationen	245	2691
10^6 Wurzeloperationen	107	3044
Buffer lesen	0	–
Schreiben+Kernel+Lesen	1024	7168

Tabelle 9.1.: Zeitbedarf für verschiedene Integer-Operationen auf dem ION in Millisekunden. *: Nicht messbar. **: Vom Compiler optimiert.

Device:	CL GPU	Host CPU
Kernel laden	6090	–
Buffer schreiben	0*	–
10^6 Additionen	71	669
10^6 Subtraktionen	71	669
10^6 Multiplikationen	71	662
10^6 Divisionen	107	853
10^6 Sinusoperationen	223	5853
10^6 Wurzeloperationen	68	2832
Buffer lesen	0	–
Schreiben+Kernel+Lesen	611	11538

Tabelle 9.2.: Zeitbedarf für verschiedene Fließkomma-Operationen auf dem ION in Millisekunden. *: Nicht messbar.

angegeben, die jeweils eine Millionen mal durchgeführt wurden. Die letzte Zeile gibt die Gesamtlaufzeit für das Schreiben der Daten in den Speicher, das Ausführen aller Operationen und das Auslesen der Ergebnisse wieder.

Die Zahlen, die mit einem ** markiert sind scheinen fehlerhaft zu sein. Es ist an diesen Stellen wahrscheinlich, dass die Operationen nicht durchgeführt wurden, da der Compiler einfache Anweisungsfolgen optimiert und die Schleifen, die in den Kernel verwendet werden, vereinfacht.

Es ist deutlich zu erkennen, dass die Operationen auf der Grafikkarte wesentlich schneller berechnet werden können, als es auf dem Atom-Prozessor möglich ist. Vor allem im Bereich der trigonometrischen Funktionen ist die Grafikkarte stark im Vorteil. Außerdem ist es kein großer Unterschied, ob auf Integer- oder auf Fließkommawerten gearbeitet wird. Erstaunlicherweise sind die Berechnungen von Fließkommazahlen hier sogar leicht

Device:	CL GPU	Host CPU
Kernel laden	244	–
Buffer schreiben	0*	–
10^6 Additionen	0**	21
10^6 Subtraktionen	0**	18
10^6 Multiplikationen	53	27
10^6 Divisionen	119	183
10^6 Sinusoperationen	172	415
10^6 Wurzeloperationen	114	340
Buffer lesen	0	–
Schreiben+Kernel+Lesen	458	1004

Tabelle 9.3.: Zeitbedarf für verschiedene Integer-Operationen auf einem modernen Desktoprechner in Millisekunden. *: Nicht messbar. **: Vom Compiler optimiert.

schneller. Es muss jedoch beachtet werden, dass die Tests darauf ausgelegt sind, alle 16 Streamingprozessoren der Grafikkarte durchweg voll auszulasten. Dies ist in so einem einfachen Testfall mit einer einfachen Befehlsabfolge leicht zu realisieren. In praxisorientierten Anwendungen ist es manchmal nicht so einfach, alle Streamingprozessoren immer voll auszulasten, da nur gleiche Anweisungen in verschiedenen parallel ausgeführten Threads auch tatsächlich parallel ausgeführt werden können. Daher kann sich die hier angegebene Berechnungsgeschwindigkeit bei nicht gut parallelisierbaren Algorithmen stark verlangsamen.

Als Vergleich zu den auf dem ION erreichten Zeiten ist der selbe Test auf einem Desktoprechner mit einem Intel Core i7 2600K Prozessor mit einer GTX 570 Grafikkarte ausgeführt worden. Die entsprechend erreichten Zeiten sind in den Tabellen 9.3 und 9.4 aufgeführt. Man kann deutlich den großen Leistungsunterschied erkennen, den heutige Desktoprechner gegenüber dem ION aufweisen können.

Insgesamt lässt sich für diese Testreihe also festhalten, dass die reine Rechenleistung der auf den Fahrzeugen verfügbaren Grafikkarte schon deutlich höher ist, als die des Hauptprozessors. Es muss jedoch stets beachtet werden, dass eine sorgfältigen Programmierung vorliegt, die es gewährleistet, dass die Streamingprozessoren der Grafikkarte immer voll ausgelastet sind.

Speichertransfertests

Die reine Rechenleistung der Grafikkarte ist nicht allein ausschlaggebend, ob sich eine Implementierung der Pipeline auf die Grafikkarte lohnt. Ein weiterer wichtiger Faktor ist, dass die Daten, die von der Datenakquisition geliefert werden auf die Grafikkarte transferiert werden müssen und die Ergebnisse auch wieder von der Grafikkarte zurück

Device:	CL GPU	Host CPU
Kernel laden	150	–
Buffer schreiben	0*	–
10^6 Additionen	54	105
10^6 Subtraktionen	54	105
10^6 Multiplikationen	54	109
10^6 Divisionen	671	156
10^6 Sinusoperationen	162	833
10^6 Wurzeloperationen	84	246
Buffer lesen	0	–
Schreiben+Kernel+Lesen	1079	1554

Tabelle 9.4.: Zeitbedarf für verschiedene Fließkomma-Operationen auf einem modernen Desktoprechner in Millisekunden. *: Nicht messbar.

Device:	CL GPU	Host CPU
Write Buffer	53*	0
Read Buffer	479	0

Tabelle 9.5.: Profilingergebnisse der Speichertransfertests für den ION. Die mit * markierten Zeilen geben Werte an, die über die angewendete Testmethode nicht korrekt ermittelt werden konnten.

in den Hauptspeicher geschrieben werden müssen. Nur bei geringen Transferzeiten ist es sinnvoll, die Berechnungen auszulagern.

In der Tabelle 9.5 sind die Zeiten für den Transfer der Daten von und zu der Grafikkarte für jeweils 40 Megabyte Daten angegeben. Dies ist zwar etwas mehr, als die von der Akquisition zu erwartenden Menge, gibt aber einen guten Überblick über die zu erwartenden Zeiten. Auch hier ist zu beachten, dass das die Zeiten für das Schreiben eines Puffers auf der Grafikkarte über das Hostprogramm nicht richtig gemessen werden kann. Daher sind auch hier die Daten für das Lesen eines Puffers wesentlich aussagekräftiger.

Man erkennt deutlich, dass die Zeit für den Speichertransfer von 40 Megabyte alles andere als zu vernachlässigen ist. Auch bei stark reduzierten Datenmengen können immer noch leicht einige 100 Millisekunden vergehen, ehe die Daten im Speicher der Grafikkarte zur Verfügung stehen. Daher ist der große Vorteil der schnelleren Berechnung auf der Grafikkarte zum Teil wieder relativiert, da diese Zeit nicht mehr für die eigentlichen Berechnungen zur Verfügung steht.

Zum Vergleich sollen auch hier noch einmal die Daten für das System mit dem Core i7 2600K Prozessor und der GTX 570 Grafikkarte angegeben werden. Die genauen Werte sind in Tabelle 9.6 zu finden.

Device:	CL GPU	Host CPU
Write Buffer	10*	0
Read Buffer	22	0

Tabelle 9.6.: Profilingergebnisse der Speichertransfertests auf einem Core i7 2600K und einer GTX 570 Grafikkarte. Die mit * markeirten Zeilen geben Werte an, die über die angewendete Testmethode nicht korrekt ermittelt werden konnten.

Zusammenfassend lässt sich also sagen, dass die Speichertransferraten gerade auf dem ION extrem schlecht sind. Es kostet viel Zeit, die Daten zu transferieren, so dass diese Zeit für die Berechnungen nicht mehr zur Verfügung steht. Vor allem die Möglichkeit nur Teile der Pipeline auf der Grafikkarte auszuführen und damit Daten häufig zwischen Hostsystem und Grafikkarte zu transferieren kann also keinesfalls umgesetzt werden.

Zusammenfassung und Fazit

Die Implementierung der Pipeline auf der Grafikkarte ist nicht umgesetzt worden. Auch wenn die reine Rechengeschwindigkeit der Grafikkarte für die Umsetzung spricht, gibt es mehrere Gründe, die dagegen sprechen. Der Hauptgrund für die Nichtumsetzung ist der immense Arbeitsaufwand, der mit der Implementierung zusammenhängt. Das Schreiben eines Rahmensystems hat sich als relativ einfach herausgestellt, jedoch ist die Implementierung der eigentlichen Kernel das Hauptproblem. Es können nicht einfach Algorithmen, die auf dem Hauptprozessor gut funktionieren eins zu eins übernommen werden. Die Notwendigkeit auf die hochparallelisierte Architektur einzugehen erfordert das Schreiben von Kerneln, die exakt auf diese Architektur ausgelegt sind. Die Grafikkarte ist nur dann schneller als der Hauptprozessor, wenn es gelingt, die vorhandenen Streaming Prozessoren immer voll auszulasten. Bei nicht für die parallele Ausführung geeigneten Algorithmen werden viele der Streamingprozessoren nicht genutzt und viel der Rechenleistung geht verloren. In dem Fall ist der Hauptprozessor mit großer Wahrscheinlichkeit sogar schneller.

9.2.2. Wandentfernung

Bei der Wandentfernung wird ein RANSAC-Algorithmus verwendet, um die Punkte, die eine der Wände darstellen, in der Szene zu identifizieren und zu entfernen. Dabei errechnet der Algorithmus die wahrscheinlichsten Parameter der Wandebene und entfernt die Punkte, die nahe dieser Ebene liegen. Während der Evaluation ist getestet worden, wie gut diese Parameter mit den realen Werten übereinstimmen.

Versuchsaufbau

Um zu evaluieren, wie genau die Wanderkennung den Abstand und die Ausrichtung der Wand in Bezug auf das Sensorzentrum erkennt, wurde das Fahrzeug in einem vorgegebenen Abstand vor eine Wand gestellt und eine längere Aufnahme gestartet, in der die Ergebnisse der Wandentfernung aufgezeichnet worden sind. Anschließend sind die Ergebnisse ausgewertet worden.

Ergebnisse der Wandentfernung

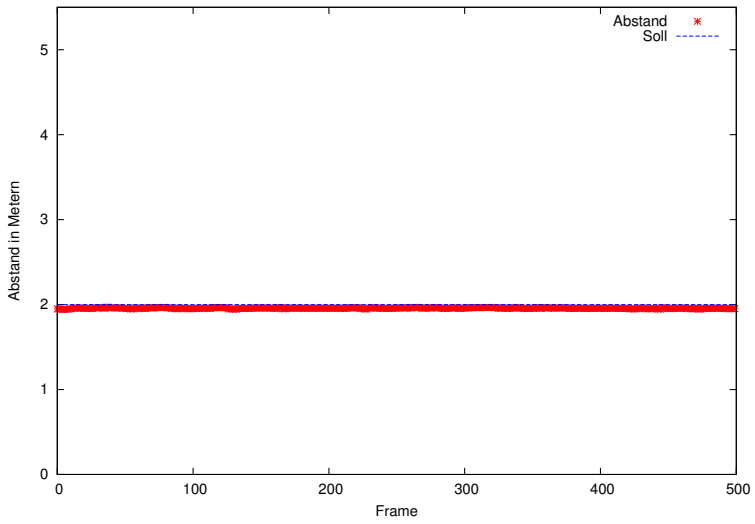
Für die Auswertung der Testergebnisse soll je eine Wand in 2 m und in 4.5 m Entfernung vom Sensorzentrum analysiert werden. Zunächst werden die Ergebnisse der Entfernungsmessung beschrieben und anschließend die Ergebnisse bei der Erkennung der Ausrichtung der Wand.

Entfernungsmessung In den Abbildungen 9.2(a) und 9.2(b) sind die gemessenen Entfernungen der Wand in jedem der aufgenommenen Frames dargestellt. Man kann leicht erkennen, dass die ermittelte Entfernung sehr gut mit der tatsächlich gemessenen Entfernung übereinstimmt. Bei der Messung in 2 Meter Entfernung wurde im Mittel eine Entfernung von 1.957 m gemessen, was dem tatsächlichen Wert sehr nahe kommt. Auch die Messung bei 4,5 Meter Entfernung ist mit im Mittel 4,461 m sehr genau. Bei der Messung in 2 Meter Entfernung kann jedoch eine wesentlich geringere Varianz von nur $2,5 \cdot 10^{-5}$ m festgestellt werden als bei der Messung in 4,5 Meter mit $5,4 \cdot 10^{-4}$ m. Dies liegt an den im allgemeinen stärker verrauschten Daten, die vom Kinect-Sensor geliefert werden, wenn die Punkte weiter vom Sensorzentrum entfernt sind.

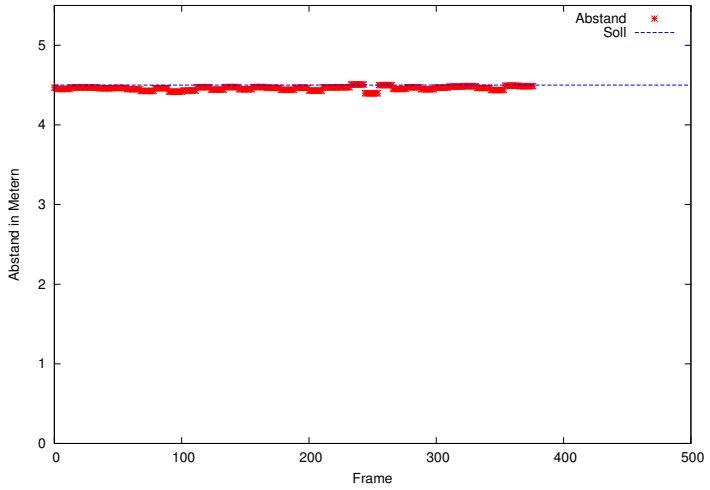
Winkelmessung Die Abbildungen 9.3(a) und 9.3(b) zeigen die z-Komponente der errechneten Wandnormale. Diese sollte bei der gewählten Testkonfiguration genau den Wert -1 haben. Man erkennt deutlich die geringen Abweichungen vom Idealwert. Im Mittel ist bei einer Entfernung von 2 Metern ein Wert von -1 mit einer Varianz von $1,739 \cdot 10^{-9}$ errechnet worden. Auch bei einer größeren Entfernung von 4,5 Metern ist im Mittel ein Wert von -1 mit einer Varianz von nur $2{,}207 \cdot 10^{-7}$ ermittelt worden. Auch hier lässt sich die schlechtere Varianz mit dem größeren Rauschen bei der Aufnahme erklären.

Zusammenfassung und Fazit

Die oben beschriebenen Tests zeigen sehr deutlich, dass die erkannten Wandparameter sehr exakt mit den tatsächlichen Wänden übereinstimmen. Daher kann das Verfahren sehr gut benutzt werden um die Punkte, die die Wand darstellen zu löschen und eine polygonbasierte Repräsentation der Wand zu berechnen.

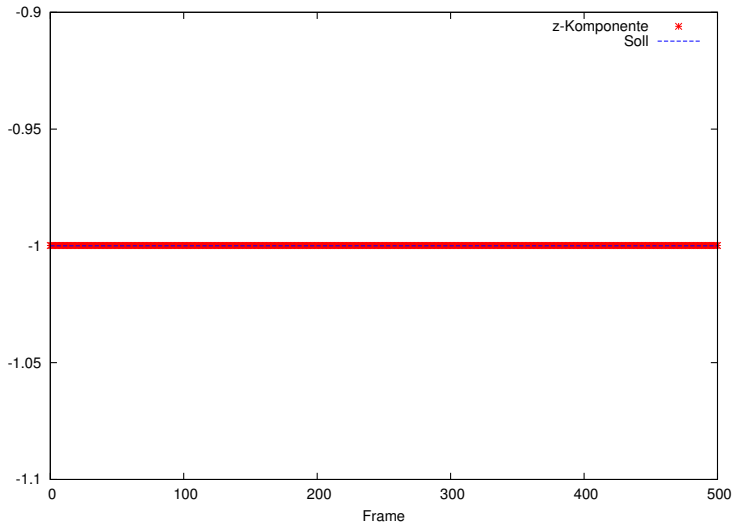


(a) Entfernungsmessung in 2 Metern Abstand

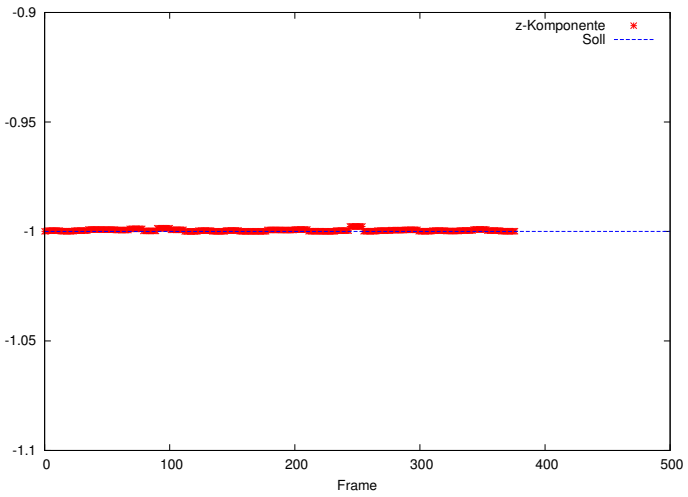


(b) Entfernungsmessung in 4,5 Metern Abstand

Abbildung 9.2.: Entfernungsmessung zur Wand in verschiedenen Abständen. Rot: Die berechnete Entfernung der Wand vom Sensorzentrum in mehreren Frames. Blau: die gemessene Entfernung



(a) z-Komponente der Normalen bei 2 Metern Entfernung



(b) z-Komponente der Normalen bei 2 Metern Entfernung

Abbildung 9.3.: Dargestellt ist die z-Komponente der Normalen der Wandebene bei einer Aufnahme mit 500 Frames. Die blaue Linie gibt die tatsächliche z-Komponente an.

9.2.3. Bodenentfernung

Die Leistungsfähigkeit der Bodenentfernung kann durch einige Testszenarien evaluiert werden. Zum einen soll dabei untersucht werden, wie gut die Parameter der Bodenebene von den beiden implementierten Verfahren gefunden werden können und zum anderen wie schnell die Verfahren sind. Als Testgrundlage dienen aufgezeichnete Daten einer realen Fahrt eines Fahrzeuges.

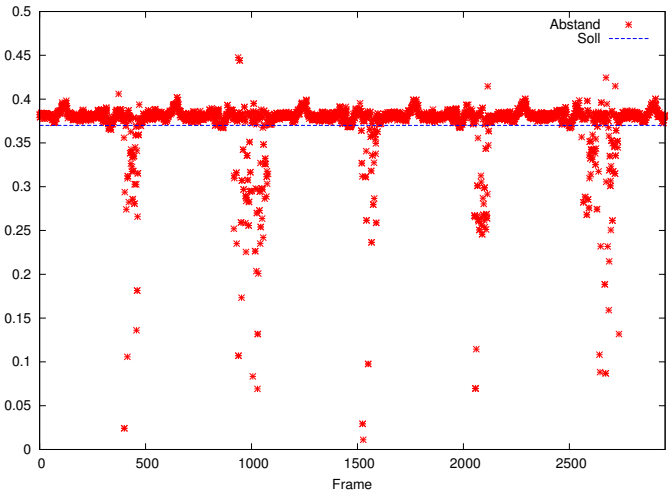
Versuchsaufbau

Zur Erhebung von Testdaten wird das Fahrzeug mit fertig montierter Kinect über einen Testparkour geschickt, der einer realen Fahrstrecke nahe kommt. Der Parkour besteht aus der Ausfahrt aus einer Station, dem Fahren einer kurzen Schleife auf offener Strecke und anschließender Einfahrt in die Station, aus der das Fahrzeug gekommen ist. Diese Runde wird mehrfach wiederholt. Die Bilder, die von der Kinect während der Fahrt aufgenommen werden, werden zunächst gespeichert. Diese Bilder werden dann auf der Zielplattform ausgewertet. Die Ergebnisse werden in den folgenden Abschnitten dargestellt.

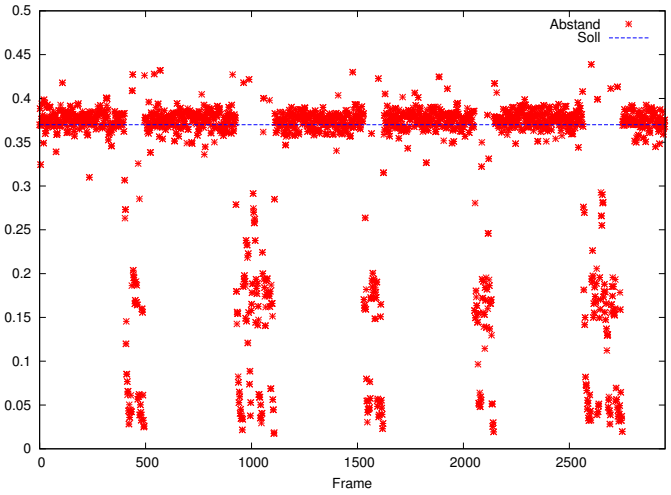
Ergebnisse der Bodenentfernung

In diesem Abschnitt werden die Ergebnisse der Parameterfindung der Bodenentfernung aufgeführt. Dabei werden zunächst die Ergebnisse der RANSAC-Bodenentfernung (s. Abschnitt 4.4.2) und im Anschluss die Ergebnisse mit dem schnellen Algorithmus (s. Abschnitt 4.4.2) zur Bodenentfernung geschildert. Die Bodenentfernung sucht die Bodenebene anhand der Punkte in der gegebenen Punktwolke und ermittelt die Normale der Ebene sowie den Abstand der Ebene vom Nullpunkt. Diese Werte ermöglichen die Beschreibung der Ebene, die den Boden repräsentiert. Ein Gütekriterium ist die korrekte Entfernung der gefundenen Ebene vom Nullpunkt, ein weiteres Kriterium die y-Komponente der Normalen der Ebene. Auch wenn beide während der Fahrt auf Grund von Bodenwellen und Erschütterungen des Sensors nicht völlig konstant sein können, sollten die Parameter nur innerhalb eines gewissen Rahmens um den tatsächlichen Wert schwanken. Die Ergebnisse der beiden Methoden sind in den folgenden Abschnitten beschrieben.

RANSAC-Bodenentfernung Die RANSAC Methode (s. Abschnitt 4.4.2) berechnet für die Höhe des Sensorzentrums über dem Boden einen relativ konstanten Wert von durchschnittlich 37,148 cm. Dies stimmt mit dem per Maßband ermittelten Wert von 37,0 cm sehr gut überein. Die minimale Höhe in dieser Messung beträgt -2,085 cm, was einen klaren Fehler darstellt und nur in einem einzigen Frame vorkommt. Der maximale Wert dieser Messung beträgt 44,753 cm und die Varianz 0,148 cm. Die Messwerte sind in Abbildung 9.4(a) dargestellt. Man kann gut erkennen, dass es fünf Zeitpunkte gibt, in denen die Höhe fälschlicherweise zu niedrig bestimmt wird. Dies geschieht immer bei

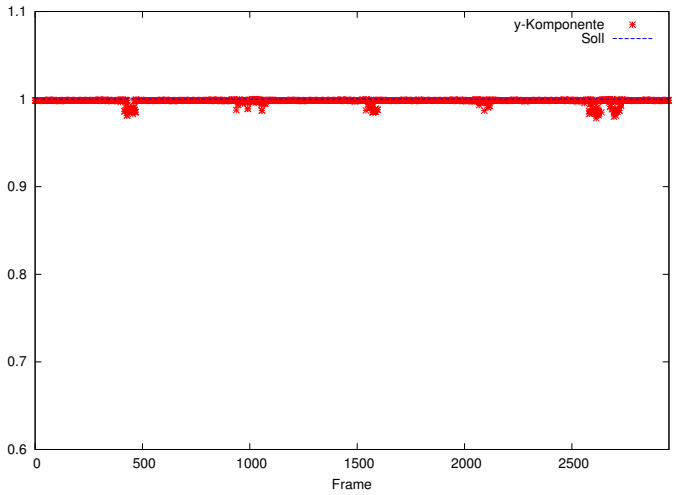


(a) Berechnete Höhe des Bodens bei der RANSAC-Bodenentfernung

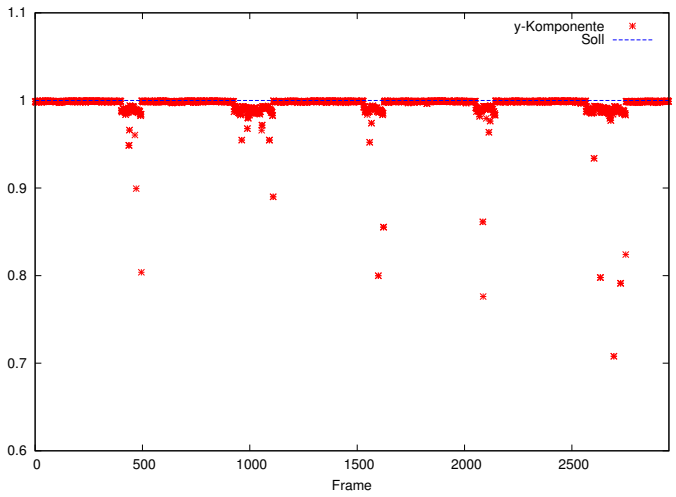


(b) Berechnete Höhe des Bodens bei der schnellen Bodenentfernung

Abbildung 9.4.: Dargestellt ist die Höhe der Kamera über dem Boden, berechnet mit der Bodenentfernung bei einer Aufnahme mit 2952 Frames. Die blaue Linie gibt die tatsächliche Höhe an.



(a) Berechnete y-Komponente bei der RANSAC-Bodenentfernung



(b) Berechnete y-Komponente bei der schnellen Bodenentfernung

Abbildung 9.5.: Dargestellt ist die y-Komponente der Normalen der Bodenebene, berechnet mit der Bodenentfernung bei einer Aufnahme mit 2952 Frames. Die blaue Linie gibt die tatsächliche y-Komponente an.

der Einfahrt in die Station, also in Frames, in denen viele Punkte der Punktwolke zur Station gehören und wenige dem Boden angehören.

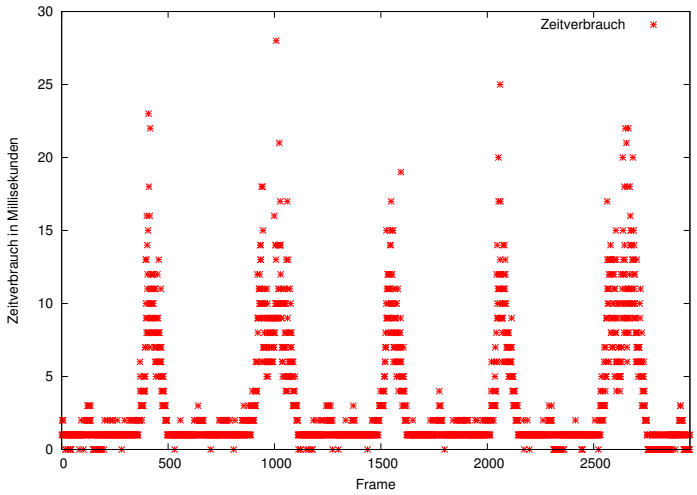
Auch die y-Komponente der Normalen der Bodenebene kann Aufschluss über die Güte der gewonnenen Ebene geben. So sollte die Normale immer in y-Richtung zeigen, also einen Wert von 1 in der entsprechenden Komponente aufweisen. Durch Bodenwellen und sonstige Unebenheiten kann dieser Wert leicht schwanken. Große Abweichungen sollten jedoch nicht entstehen. Der Mittelwert der gemessenen Daten beträgt 0,998. Der minimale und maximale Wert der Messung ist 0,978 und 1,000. Die Varianz beträgt $6 \cdot 10^{-6}$. Die ermittelten Werte sind in Abbildung 9.5(a) angegeben. Vor allem die sehr kleine Varianz der Daten zeigt, dass die Bodenebene immer sehr gut mit der xz-Ebene übereinstimmt, was dem Sollergebnis entspricht. Große Abweichungen gibt es nicht.

Schnelle Bodenentfernung Die Werte für die Höhe des Sensorzentrums über dem Boden, die mittels der schnellen Bodenentfernung ermittelt werden, bieten ein nicht ganz so gutes Bild wie bei der RANSAC-Methode. Die Werte sind weniger Konstant sondern streuen wesentlich mehr. Der Mittelwert der Daten liegt bei 32,451 cm. Der mittels Maßband gemessene Wert liefert jedoch 37,0 cm. Damit ist eine systematische Abweichung nach unten festzustellen. Der minimale Wert liegt bei 17,393 cm und der maximale Wert bei 43,887 cm. Die Varianz ergibt sich bei dieser Messung zu 1,126 cm. Die genauen Messergebnisse sind in Abbildung 9.4(b) dargestellt. Es ist gut zu erkennen, dass ähnlich wie bei dem RANSAC-Ansatz auch hier die Stationseinfahrten die meisten Probleme verursachen, da auch hier die Messwerte an diesen Stellen stark vom Soll abweichen. Aber im Gegensatz zur RANSAC-Methode ist die Streuung der Daten auch in Abschnitten ohne ein Hindernis im Bild wesentlich größer, wie zum Beispiel in dem Bereich um die Frames 600 bis 900.

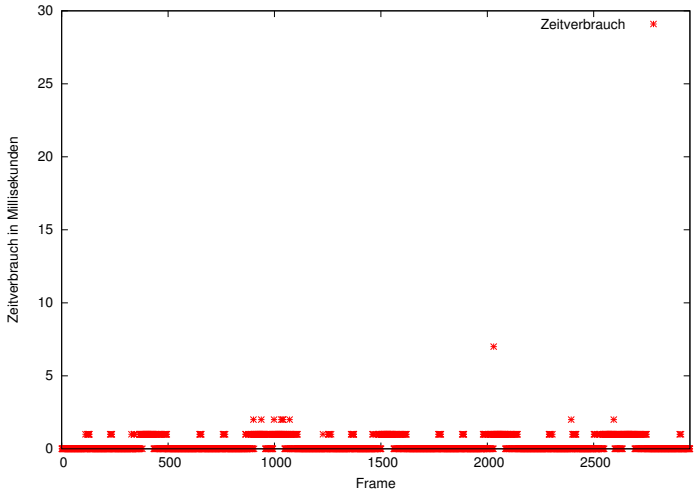
Bei der Betrachtung der y-Komponente der Normalen in der gleichen Aufnahme ergibt sich ein sehr ähnliches Bild wie bei der RANSAC-Methode. Die Werte verändern sich nur sehr wenig und schwanken leicht um den Idealwert von 1. Der Mittelwert ist hier 0,995, was nur minimal schlechter ist, als bei der RANSAC-Methode. Der minimale und der maximale Wert ist 0,708 sowie 1,000. Damit ist die Abweichung nach unten wesentlich größer als bei der RANSAC-Methode. Jedoch sind dies nur sehr wenige Ausreißer, wie an der Varianz von 0,0003 gut zu erkennen ist. Insgesamt ist also auch hier eine gute Übereinstimmung der Bodenebene mit der xz-Ebene auszumachen.

Geschwindigkeitstests

In diesem Abschnitt sollen die Geschwindigkeiten der beiden Methoden zur Bodenentfernung wiedergegeben werden. Dabei wird der Zeitverbrauch der jeweiligen Methode während einer langen Fahrt ermittelt.



(a) Zeitverbrauch bei der RANSAC-Bodenentfernung



(b) Zeitverbrauch bei der schnellen Bodenentfernung

Abbildung 9.6.: Dargestellt ist der Zeitbedarf der Bodenentfernung in Millisekunden bei einer Aufnahme mit 2952 Frames. Die blaue Linie gibt den Mittelwert der Messdaten an.

RANSAC-Bodenentfernung Die RANSAC Methode ist sehr schnell bei der Ermittlung der Parameter der Bodenebene und dem Entfernen der entsprechenden Punkte. Bei insgesamt 2952 aufgezeichneten Bildern mit unterschiedlicher Komplexität der Szene werden im Mittel nur 2,852 ms benötigt. Minimal werden in dieser Aufnahme 0,0 ms und maximal 28,0 ms für die Berechnungen verbraucht. Die Varianz liegt bei 13,099 ms. Die genauen Messdaten sind in Abbildung 9.6(a) dargestellt. Der geringe Zeitverbrauch von ca. 0 ms wird immer dann erreicht, wenn in dem aufgenommenen Bild außer dem Boden keine weiteren Punkte auftreten. Die maximalen Zeiten werden bei Aufnahmen benötigt, in denen viele Punkte, vor allem von verschiedenen Objekten, vorhanden sind. In der Abbildung sind daher die 5 Stationseinfahrten zu erkennen, wobei der hohe Rechenzeitbedarf immer bei voller Sicht auf die Station entsteht.

Schnelle Bodenentfernung Die schnelle Bodenentfernung (s. Abschnitt 4.4.2) basiert auf einem schnellen, naiven Berechnungsverfahren und braucht dementsprechend weniger Zeit als die RANSAC basierte Methode. Bei der gleichen Aufnahme wie bei der RANSAC-Auswertung im letzten Abschnitt ergibt sich im Mittel ein Zeitverbrauch von nur 0,239 ms. Der minimale Zeitverbrauch liegt bei 0,0 ms und der maximale bei 7,0 ms. Dabei ist die Varianz 0,202 ms. Die genauen Messdaten sind in Abbildung 9.6(b) dargestellt. Es ist gut zu erkennen, dass fast durchgängig nur ein bis zwei Millisekunden Berechnungszeit nötig sind. Es entsteht auch keine direkte Abhängigkeit mehr von der Komplexität der Szene wie es bei der RANSAC-Methode der Fall ist.

Zusammenfassung und Fazit

Zusammenfassend lässt sich festhalten, dass beide Algorithmen zur Bodenentfernung ein sehr gutes Ergebnis erzielen. Die RANSAC-Methode ist etwas besser, was die Ergebnisse betrifft. So sind bei dieser Methode die Varianzen von der ermittelten Höhe des Sensors sowie der y-Komponente der Normalen geringer als bei der schnellen Variante. Dennoch zeigt sich, dass die schnelle Methode ebenfalls verwendet werden kann, ohne wesentlich schlechtere Ergebnisse in Kauf nehmen zu müssen. Außerdem benötigt diese im Vergleich zur RANSAC-Methode weniger Zeit für die Berechnung der Ebene. Im Vergleich zu den Zeiten, die von den restlichen Schritten in der Verarbeitungspipeline benötigt werden, können aber beide Methoden bedenkenlos verwendet werden.

9.2.4. Clustering

Zur Evaluation des Clusterings wurden mit einer empirisch ermittelten Parametrisierung verschiedene Posen und Beladungszustände eines Fahrzeugs untersucht. Es wurde besonderes Augenmerk darauf gelegt, dass ein Fahrzeug möglichst durch einen zusammenhängenden Cluster repräsentiert wird.

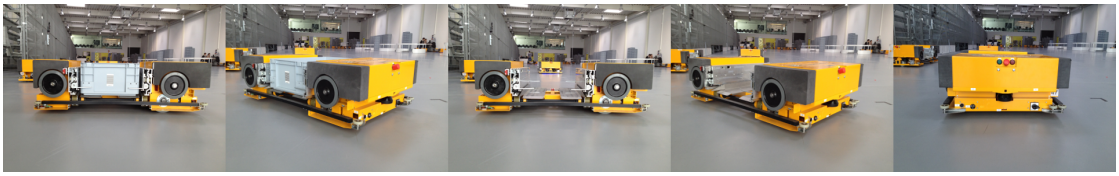


Abbildung 9.7.: Die Abbildung zeigt die im Rahmen der Evaluation der Cluster-Splitrate und Klassifikationsrate verwendeten Fahrzeugposen mit und ohne Beladung. Von links nach rechts: Fahrzeug von der Seite mit Kiste, schräg mit Kiste, von der Seite ohne Kiste, schräg ohne Kiste und frontal.

Versuchsaufbau

Für das zur Clusterextraktion verwendete euklidische Clustering wurde der Radius r der Umkugel auf $0,1\,\mathrm{m}$ und die untere Schranke für die minimale Anzahl von Punkten pro Cluster auf 50 Punkte festgesetzt. Diese Werte wurden im Rahmen einer empirischen Studie bestimmt. Die Werte ermöglichen es, dass zwei Fahrzeuge mit einem Mindestabstand von $10\,\mathrm{cm}$ bis zu einer Maximaldistanz von $4\,\mathrm{m}$ segmentiert werden können.

Für die Evaluation des verwendeten euklidischen Clustering-Algorithmus wurden fünf charakteristische Testfälle bestimmt (siehe Abbildung 9.7) und mit den oben beschriebenen Parametern untersucht:

1. Ein Fahrzeug lateral mit Kiste
2. ein Fahrzeug frontolateral mit Kiste,
3. ein Fahrzeug lateral ohne Kiste,
4. ein Fahrzeug frontolateral ohne Kiste
5. und ein Fahrzeug frontal.

Zusammenfassung und Fazit

Als problematisch haben sich Fahrzeuge ohne Kiste erwiesen. Durch das sehr schmale Verbindungsstück zwischen Front und Heck des Fahrzeugs neigen die korrespondierenden Cluster in mindestens zwei Cluster zu zerfallen, so dass sie im nachfolgenden Klassifikationsschritt nicht mehr eindeutig klassifiziert werden können. Der Zerfall in multiple Cluster (Cluster-Split) bei seitlich betrachteten Fahrzeugen ohne Kiste, ist ab einer Entfernung von über $2{,}3\,\mathrm{m}$ zu beobachten (siehe Abbildung 9.8). Im Rahmen der Projektgruppe konnte aus Zeitgründen kein Verfahren bestimmt werden, bei dem es nicht (oder nur in wenigen Ausnahmefällen) zu einem Zerfall der Cluster kommt. Es ist jedoch anzunehmen, dass ein Dichte-basiertes Clustering zu einer geringeren Zerfallsquote von Clustern führen könnte.

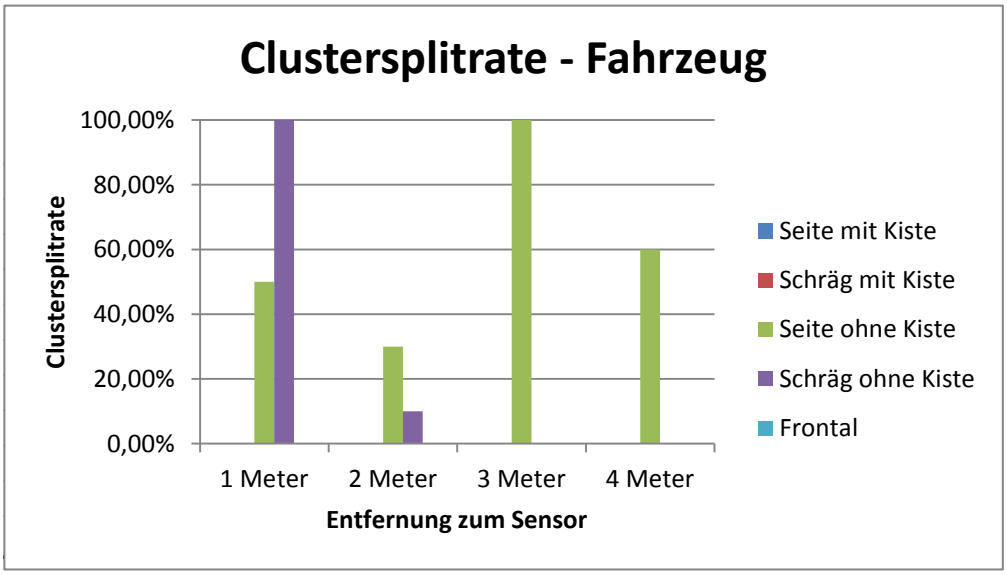


Abbildung 9.8.: Die Abbildung zeigt die Evaluation der Cluster-Splitrate anhand der fünf zuvor definierten Fahrzeug Posen (mit und ohne Kiste).

9.2.5. Klassifikation

Die Evaluation der Klassifikation soll zeigen, mit welcher Güte Fahrzeuge in verschiedenen Ausrichtungen, aus verschiedenen Abständen erkannt werden.

Versuchsaufbau

Zur Bestimmung der Erkennungsrate werden die in Abschnitt 9.2.4 beschriebenen fünf Testszenarien jeweils aus 1 m, 2 m, 3 m und 4 m untersucht. Die Distanzen geben den Abstand zum Sensorzentrum an. Die daraus resultierenden 20 Testfällen sind in Tabelle 9.7 zu sehen. Für jeden dieser Fälle wird im Folgenden überprüft, ob ein Fahrzeug in korrekter Ausrichtung erkannt wird.

Zusammenfassung und Fazit

Die Erkennungsrate für Fahrzeuge des vorgestellten Klassifikationsalgorithmus wird in Abbildung 9.9 gezeigt. Erkennbar ist, dass Fahrzeuge bis zu einer Distanz von 2 m unabhängig von ihrer Beladungssituation zu 100 % erkannt werden können. Ab einer Distanz über 2 m lässt sich eine sinkende Erkennungsrate feststellen, die zudem durch die mit der Distanz zunehmende Cluster-Splitrate bei Fahrzeugen ohne Kiste verstärkt wird. Das Problem der Zerfalls von Clustern wurde im Rahmen der Evaluation als

Abstand	mit Kiste	Ausrichtung
1 m	ja	lateral
1 m	ja	frontolateral
1 m	nein	lateral
1 m	nein	frontolateral
1 m		frontal
2 m	ja	lateral
2 m	ja	frontolateral
2 m	nein	lateral
2 m	nein	frontolateral
2 m		frontal
3 m	ja	lateral
3 m	ja	frontolateral
3 m	nein	lateral
3 m	nein	frontolateral
3 m		frontal
4 m	ja	lateral
4 m	ja	frontolateral
4 m	nein	lateral
4 m	nein	frontolateral
4 m		frontal

Tabelle 9.7.: Testfälle zur Bestimmung der Erkennungsrate. Bei den vier frontalen Testfällen ist es nicht relevant, ob das Fahrzeug eine Kiste geladen hat, da diese durch die Front des Fahrzeugs verdeckt wird.

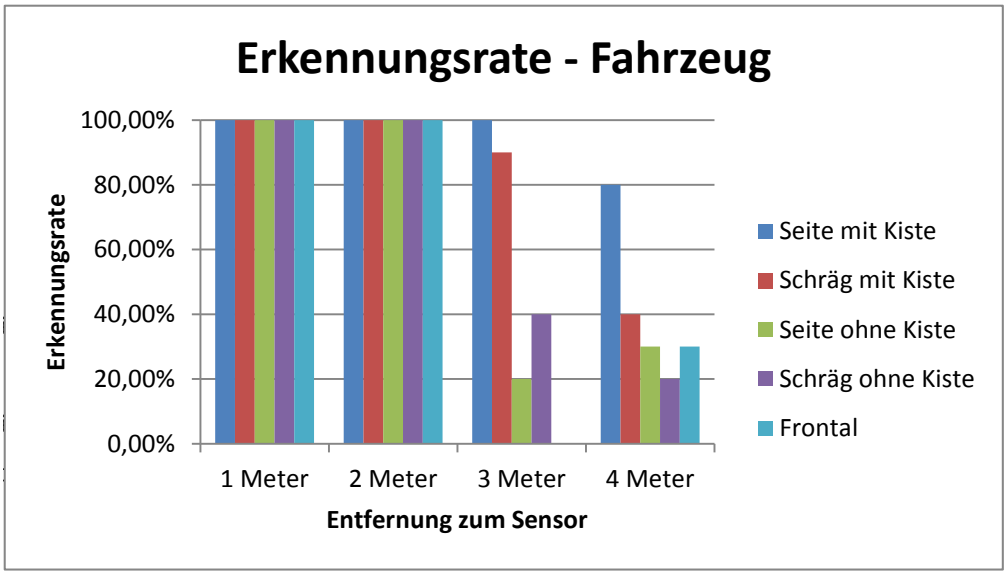


Abbildung 9.9.: Die Abbildung zeigt die Evaluation der Klassifikationsrate anhand der fünf zuvor definierten Fahrzeug Posen (mit und ohne Kiste).

wichtiges noch zu lösendes Problem erkannt, so dass durch die Pipeline unabhängig von der Beladungssituation eine gleichbleibend hohe Erkennungsrate gewährleistet wird.

Im weiteren Verlauf der Evaluation der Analyse-Pipeline wurde die Genauigkeit der Posenerkennung untersucht. Hierbei wurde die tatsächliche Pose in Grad des Fahrzeugs mit der durch die Analyse-Pipeline bestimmten Pose unter Berücksichtigung einer Toleranz von $\pm 10\,^\circ$ verglichen. Bei einer Distanz von 1,5 m wurde einer Posenerkennungsrate von 78 %, bei 2 m von 69 % und bei 3,3 m von nur noch 28 % bestimmt. Signifikant ist die mit der Distanz abfallenden Posenerkennungsrate, die größtenteils durch die mit der Distanz stark abnehmende Anzahl von Punkten einhergeht, welche zu einer wenig robusten Bestimmung der Normalen führt.

9.2.6. Diskussion

In den letzten Kapiteln sind die einzelnen Komponenten der Grafikpipeline auf ihre Leistungsfähigkeit und ihre Eignung für die Projektgruppe untersucht worden. Dabei lässt sich festhalten, dass sowohl die Boden- als auch die Wandentfernung mit den eingesetzten Methoden sehr gute Ergebnisse liefert. Bei der Wandentfernung konnten die Wände in allen getesteten Fällen identifiziert und deren Position und Orientierung sehr genau ermittelt werden. Bei der Bodenentfernung wird der Boden in den meisten Fällen korrekt erkannt und aus den Eingabedaten entfernt. Schwächen in dem Verfahren zeigen sich vor allem in komplexen Szenen, in denen wenig Boden, aber viele andere Objekte im Sensorbereich liegen. Der Zeitverbrauch bei den Berechnungen der Bodenentfernung

[Pose] =°	1,5 m	2,0 m	3,3 m
10	10	10	200
20	20	20	200
30	30	30	150
40	40	40	190
50	50	50	40
60	60	60	140
70	70	70	200
80	80	80	190
90	90	90	350
100	100	100	100
110	110	110	280
120	120	120	200
130	130	130	130
140	140	140	40
150	150	50	150
160	160	160	190
170	250	170	170
180	180	180	180
190	190	170	190
200	230	200	200
210	210	50	320
220	220	220	220
230	320	300	230
240	280	200	200
250	250	250	340
260	260	170	170
270	250	260	170
280	340	160	340
290	300	240	150
300	230	230	310
310	220	220	150
320	320	320	150
330	330	210	200
340	340	340	190
350	350	350	190
360	360	360	110
Erkennungsrate:	78 %	69 %	28 %

Tabelle 9.8.: Die Tabelle zeigt die Posenerkennungsrate der vorgestellten Pipeline bei einem Fahrzeug mit Kiste. Die erlaubte Abweichung bzgl. der Posenerkennung wurde auf ±10 Grad festgelegt.

liegt mit wenigen Millisekunden im Rahmen. Der Vergleich zwischen den getesteten Algorithmen hat gezeigt, dass auch ein naiver Ansatz mit nur geringem Qualitätsverlust angewendet werden kann. Beim Clustering konnten ebenfalls gute Ergebnisse erzielt werden. Der größte Schwachpunkt ist hier der Zerfall eines Objekts in zwei oder mehr Cluster, was vor allem bei seitlich betrachteten Fahrzeugen ohne Beladung geschieht. Die Güte der Klassifikation ist vor allem von den Ergebnissen des Clusterings und dem Abstand der Objekte abhängig. Die Erkennungsrate ist bei Entfernungen der Objekte von unter zwei Metern sehr gut, verschlechtert sich jedoch mit der Entfernung zusehens.

Auf Grund des immensen Arbeitsaufwandes und dem nur geringen zu erwartenden Leistungszuwachses ist die Grafikpipeline nicht auf die Grafikkarte portiert worden.

Die gesamte Pipeline funktioniert in den meisten getesteten Situationen gut. Die Vorverarbeitung, bestehend aus der Wand- und Bodenentfernung, eliminiert die für die Klassifikation nicht relevanten Daten sehr zuverlässig aus der Eingabepunktwolke. Die Klassifikationsphase, bestehend aus der Bildung der Cluster sowie der Klassifikation eben dieser, funktioniert auf kleine Entfernung sehr gut. Die Leistung der Clusterextraktion und damit auch der Klassifikation nimmt jedoch mit der Entfernung stark ab. Hier wäre es sinnvoll, nach alternativen Algorithmen zu suchen, die den Zerfall von Objekten in mehr als ein Cluster verhindert. Es ist anzunehmen, dass mit Verminderung der Clusterzerfallsrate auch die Klassifikationsleistung steigt.

Insgesamt sind die von der Grafikpipeline erkannten und klassifizierten Objekte so gut, dass sie für die Bahnplanung gut verwendet werden können.

9.3. Lokale Bahnplanung

Im Folgenden werden die Kriterien sowie Testszenarien zur Evaluation der Bahnplanung definiert und anschließend die so ermittelten Versuchsergebnisse ausgewertet.

9.3.1. Kriterien

Zur Beurteilung der Bahnplanung müssen zunächst Kriterien definiert werden. Diese werden dann in verschiedenen Testszenarien überprüft. Ein Gütekriterium für die Bahnplanung stellt das Erreichen der Zielposition dar. Hierzu wird der euklidische Abstand zwischen Fahrzeugposition und Zielposition berechnet. Nimmt der euklidische Abstand einen Wert kleiner als ε an, so gilt dieses Kriterium als erfüllt. Je größer dieser als ε ist, desto schlechter ist die Bahnplanung. Hierbei dient ε als Toleranzmaß und ist auf 2 cm festgelegt.

Da die Bahnplanung auch zum Anfahren von Posen verwendet wird, wird auch hierfür ein Gütekriterium definiert. Ebenfalls darf der euklidische Abstand zwischen Fahrzeugpose und Zielpose den Toleranzwert ε nicht überschreiten. Zudem darf der Winkel vom Fahrzeug nur um den Wert δ von dem Winkel der Zielpose abweichen, andernfalls gilt

Fahrbefehl	Zielposition		Zielpose	
Zielkriterium	$\|\mathbf{p} - \mathbf{t}\|$	$\leq \varepsilon$	$\|\mathbf{p} - \mathbf{t}\|$	$\leq \varepsilon$
			$\|\theta_\mathbf{p} - \theta_\mathbf{t}\|$	$\leq \delta$
Hinderniskriterium	$\|\mathbf{p} - \mathbf{o}\|$	$> \alpha$	$\|\mathbf{p} - \mathbf{o}\|$	$> \alpha$

Tabelle 9.9.: Gütekriterien für die Fahrbefehle Zielposition/-pose anfahren: Fahrzeugposition $\mathbf{p}$, Hindernis $\mathbf{o}$, Zielposition $\mathbf{t}$, $\varepsilon = 2\,\text{cm}$, $\delta = 3°$

das Kriterium als nicht erfüllt. Die Abweichung δ wird auf 3° festgelegt. Die Höhe der Abweichung gibt Aufschluss über die Güte der Bahnplanung.

Neben dem Erreichen des Ziels stellt auch das Umfahren von Hindernissen ein Bewertungskriterium dar. Eine Kollision mit einem Hindernis unabhängig vom Ausmaß wird als Nichterfüllen des Kriteriums bewertet. Somit gilt es, dass ein Hindernis mit einem gewissen Mindestabstand umfahren werden muss.

Zusammenfassend lassen sich pro Fahrbefehl (Zielposition oder -pose anfahren) zwei Kriterien aufstellen. Diese sind in Tabelle 9.9 aufgelistet.

Neben diesen Mindestkriterien lassen sich noch weitere Kriterien für die Evaluierung der Bahnplanung mit VFH aufstellen. So ist die Länge des gefahrenen Weges bei gleichem Start- und Zielpunkt von Interesse, da diese Aufschluss über die Effizienz der gewählten Route gibt. Ein weiteres zu untersuchendes Kriterium besteht in der durchschnittlichen Winkeländerung. Durch häufiges Abbiegen und somit durch eine häufige Winkeländerung wächst der Odometriefehler und das FTF wird mehr von der Zielpose abweichen. Daher ist ein kurzer zurückgelegter Weg mit wenigen Winkeländerungen wünschenswert.

9.3.2. Testszenarien

Die Tests werden in der Fahrzeughalle durchgeführt. Der Nachteil besteht darin, dass Fehler, die durch das Zusammenspiel der einzelnen Projektkomponenten, wie der Grafikpipeline oder der Middleware, entstehen und sich aufsummieren, und Odometriefehler auftreten können. So kann das Verfahren zur lokalen Bahnplanung nicht als unabhängige Komponenten getestet und beurteilt werden.

Alle zuvor erwähnten Kriterien werden für die Bahnplanung mittels VFH verwendet. Zum Testen des Verfahrens werden die in den Abbildungen 9.10, 9.11 und 9.12 gezeigten Testszenarien benutzt.

Die Testszenarien zeigen einen schematischen Teilausschnitt der Versuchshalle sowie das verwendete Testfahrzeug mit der Nummer 5 und wurden mit Hilfe der Vehicle-Monitoring-Software generiert. Die vom Fahrzeug anzufahrenden Flächenpositionen sind als schwarze Punkte dargestellt und entsprechend der abzufahrenden Reihenfolge verbunden. Die

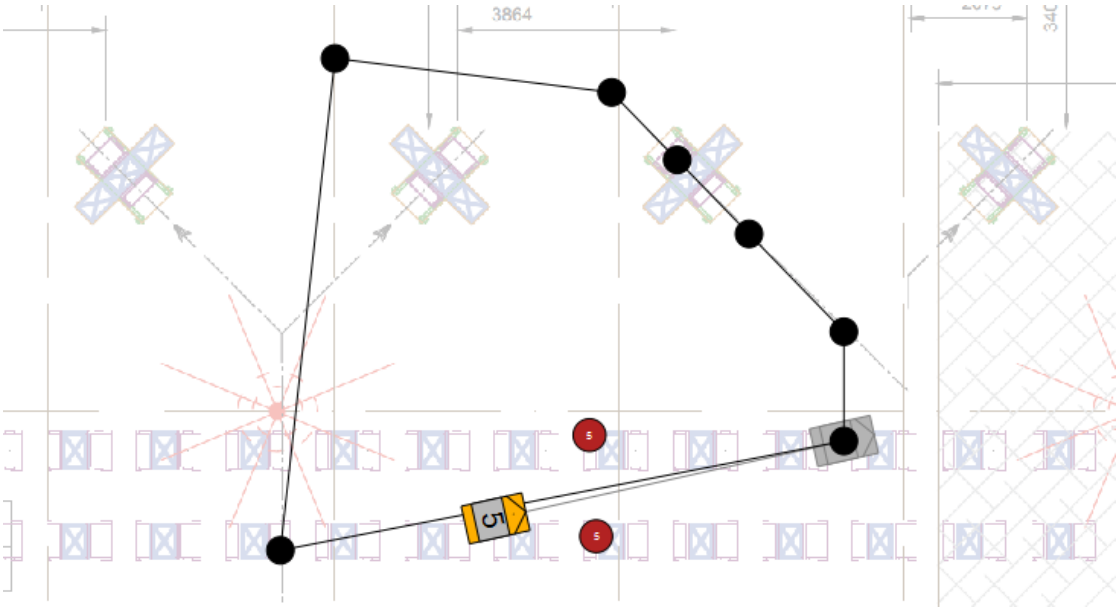


Abbildung 9.10.: Testszenario 1 zur Überprüfung der Ziel- und Hinderniskriterien

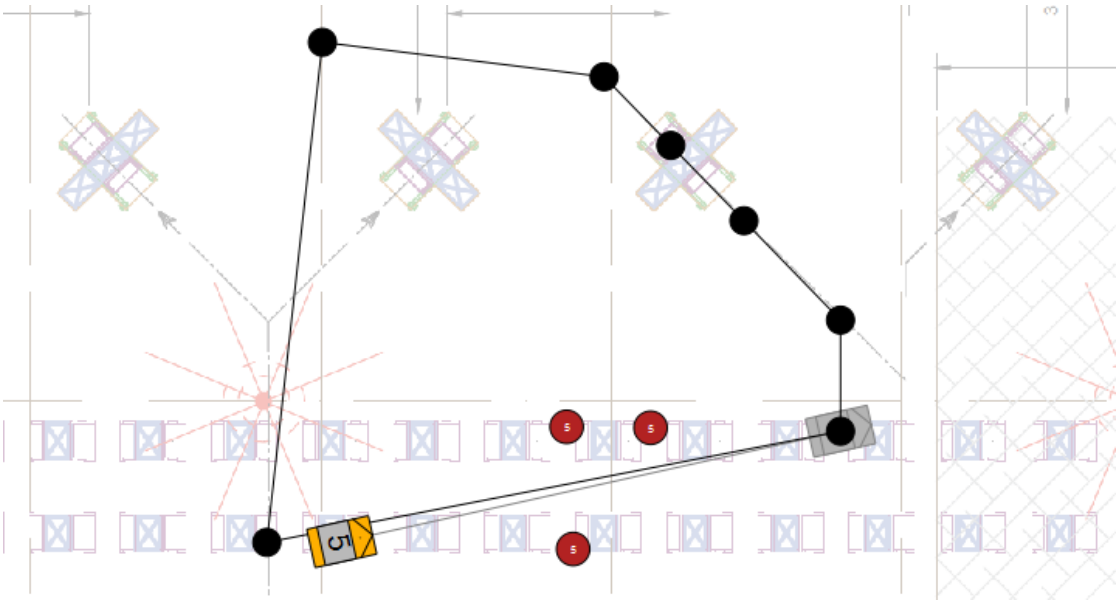


Abbildung 9.11.: Testszenario 2 zur Überprüfung der Ziel- und Hinderniskriterien

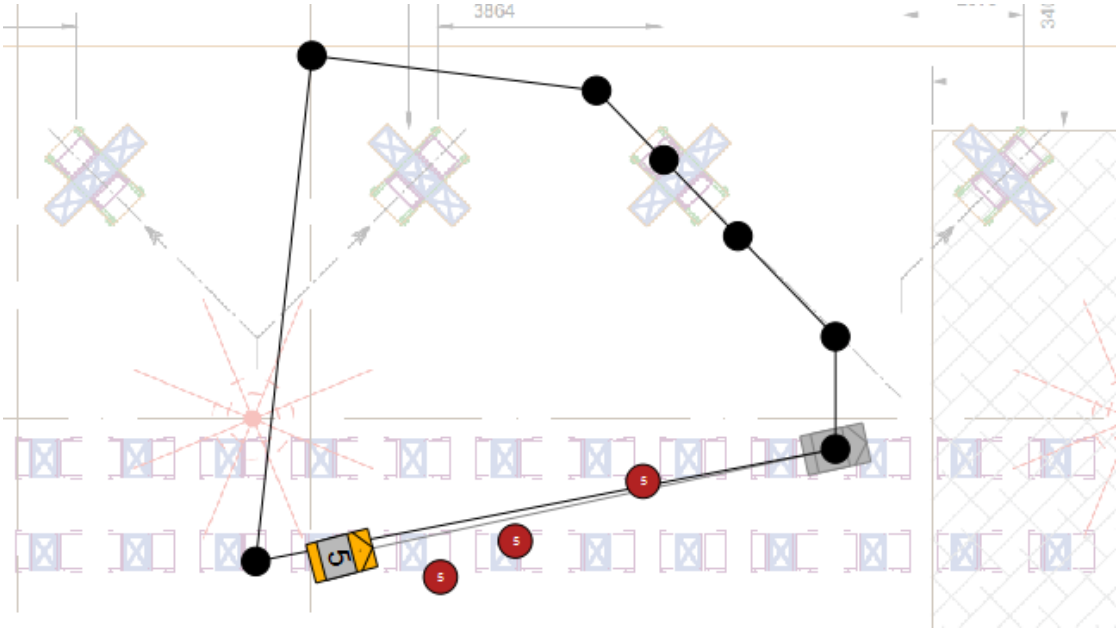


Abbildung 9.12.: Testszenario 3 zur Überprüfung der Ziel- und Hinderniskriterien

Kriterium	Testszenario 1	Testszenario 2	Testszenario 3
[⌀ Zielpositionsabw.] = mm	3,5	21,1	7,9
[⌀ Zielwinkelabw.] = °	0,75	0,25	0,44
[Gefahrene Strecke] = m	33	34	37
[Summierte Winkeländ.] = °	444,21	449,69	688,39

Tabelle 9.10.: Evaluation der definierten Kriterien anhand der Testszenarien

dabei erkannten Hindernisse, bei denen es sich um die Fläche gestellte Transportbehälter handelt, sind als rote Kreise zu sehen.

9.3.3. Versuchsergebnisse

Die geschilderten Testszenarien wurden in der Versuchshalle mit dem Testfahrzeug abgefahren und die dabei aus der Bahnplanung resultierende Trajektorien wurden unter Verwendung der Vehicle-Monitoring-Software aufgezeichnet und gespeichert.

Auf diese Art und Weise konnten im Nachhinein die definierten Kriterien anhand der aufgezeichneten Daten evaluiert werden. Die Ergebnisse dieser Evaluation sind in Tabelle 9.10 dargestellt.

Die aufgezeichneten Daten enthalten die Zustände des Testfahrzeugs zu aufeinanderfolgenden diskreten Zeitpunkten. Aus diesen Zuständen wurde die durchschnittliche

Zielpositionsabweichung als Durchschnitt der kleinsten euklidischen Abstände zu den jeweils anzufahrenden Zielpositionen ermittelt. Auf gleiche Art und Weise wurde die durchschnittliche Zielwinkelabweichung bestimmt. Bei der gefahrenen Strecke handelt es sich um den vom Fahrzeug im jeweiligen Testszenario in Summe zurückgelegten Weg. Die summierte Winkeländerung wurde nach Formel 9.1 bestimmt.

$$\text{Summierte Winkeländ.} = \int_t \left| \frac{\partial \text{Fahrzeugwinkel}}{\partial t} \right| \tag{9.1}$$

9.3.4. Diskussion

Die Auswertung der errechneten Messwerte lässt sich in zwei Teilbereiche aufteilen. Einerseits lassen sich die durchschnittliche Zielpositions- bzw. Zielwinkelabweichung als Qualitätsmerkmal für das möglichst exakte Erreichen der vorgegebenen Zielposen interpretieren. Hierbei ist auffällig, dass die Abweichung der Position im zweiten Testszenario erheblich größer als in den anderen Fällen ist. Dies lässt sich dadurch erklären, dass der zweite Testfall direkt – und ohne Zurücksetzen der Odometrie – nach dem ersten durchgeführt wurde. Dies hat zur Folge, dass die durch die Odometriefehler induzierte Abweichung der Positionsdaten im zweiten Fall deutlich größer ist. Die durchschnittliche Zielwinkelabweichung weist in allen Testszenarien in etwa den gleichen, hinreichend kleinen, Betrag auf.

Andererseits lassen die gefahrene Strecke und die summierte Winkeländerung einen Rückschluss auf den Einfluss der im Testszenario platzierten Hindernisse auf die von der Bahnplanung errechneten Trajektorien zu. In den ersten beiden Szenarien hat das Fahrzeug den Zwischenraum passiert, wobei es im zweiten Szenario auf Grund der zusätzlichen dritten Kiste einen zusätzlichen Bogen fahren musste. Dies schlägt sich in den Messwerten insofern nieder, dass beide Kriterien einen leicht höheren Wert aufweisen. Im dritten Szenario wurden die Hindernisse durch ein größeren Ausweichbogen umfahren, wodurch zum einen ein größerer Weg zurückgelegt wurde und zum anderen deutlich mehr Rotationsbewegungen notwendig waren.

Abgesehen von der beschrieben Abweichung im zweiten Testszenario, konnte die zu erzielenden Zielpositions- und Winkelabweichungen von 20 mm und 3° unterschritten werden.

Da die Teststrecke ein geschlossenes konvexes Polygon bildet, muss das Fahrzeug in Summe Rotationsbewegungen von mindestens 360° ausführen. Um Hindernisse umfahren zu können müssen zusätzliche Rotationen stattfinden. Dass die real auftretenden summierten Winkeländerungen nur 84,21°, 89,69° und 328,39° höher liegen, zeigt, dass ein verhältnismäßig rationaler und effizienter Weg gewählt wurde.

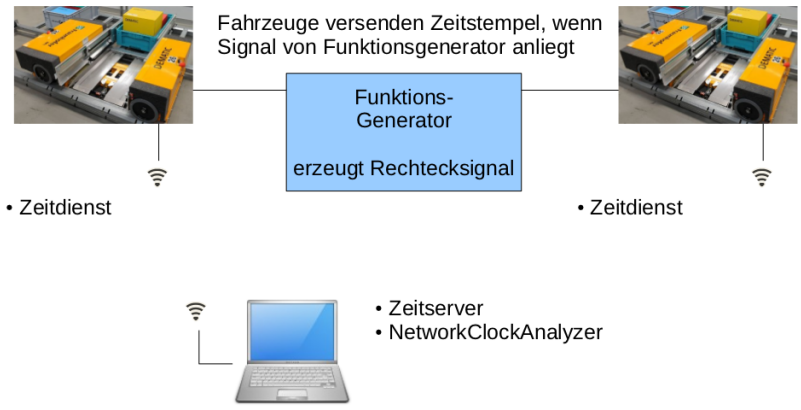


Abbildung 9.13.: Versuchsaufbau für die durchgeführten Zeitmessungen. Die Fahrzeuge synchronisieren ihre Uhrzeiten mit der Zeit des im Netzwerk vorhandenen Zeitservers.

9.4. Zeitsynchronisation

In diesem Abschnitt werden der Versuchsaufbau für die durchgeführten Zeitmessungen und die dort erhaltenen Ergebnisse beschrieben. Ziel dieser Messreihen sollte sein, die ausgewählten Protokolle auf ihre Eignung, hinsichtlich einer Verwendung im Rahmen der Projektgruppe, zu untersuchen. Die Abweichungen, der durch die Protokolle synchronisierten Uhren der Fahrzeuge, sollte die geforderte Zeitschranke von 10 ms nicht überschreiten und möglichst nahe an die gewünschte Zeitschranke von 1 ms heran reichen. Eine Besonderheit dabei war, das keines der Protokolle für den Einsatz im WLAN konzipiert wurde, wodurch sich einige Schwierigkeiten ergaben.

9.4.1. Generischer Versuchsaufbau

Um aussagekräftige Ergebnisse über die Genauigkeit der Zeitsynchronisation treffen zu können, musste ein geeigneter Versuchsaufbau entwickelt werden.

Hardwareaufbau

Um die variablen Verzögerungszeiten der Datenpakete im WLAN nicht mit in die Zeitmessung zu integrieren, wird den Fahrzeugen mithilfe eines Funktionsgenerators mitgeteilt, wann sie ihren aktuellen Zeitstempel verschicken sollen. Der Funktionsgenerator erzeugt ein Rechtecksignal, welches an einen Digital-Eingang der Fahrzeuge geführt wird. Weil der Funktionsgenerator an alle Fahrzeuge direkt angeschlossen ist, erhalten diese alle zum selben Zeitpunkt das Signal zum Verschicken ihres Zeitstempels.

Insgesamt ergibt sich der Versuchsaufbau aus Abbildung 9.13. Durch den Funktionsgenerator wird das Versenden der aktuellen Zeit auf den Fahrzeugen getriggert. Die gesamte Kommunikation erfolgt über das in der Halle vorhandene WLAN.

Messprogramm auf dem Fahrzeug

Das Messprogramm auf dem Fahrzeug ist relativ einfach gehalten und verschickt seine Daten über das UDP-Protokoll.

Während der gesamten Laufzeit des Programms wird in jedem Task-Zyklus der Messeingang, an dem das Rechtecksignal anliegt, ausgewertet. Wird eine steigende Flanke detektiert, wird ein Flag gesetzt, welches das Senden einer Nachricht an den NetworkClockAnalyzer anstößt.

Die Nachricht enthält verschiedene Informationen, die in einem von der Projektgruppe festgelegten Format versendet werden. Als Erstes wird die IP-Adresse des Fahrzeuges eingetragen, welche später zur Identifikation und Unterscheidung der Mess-Nachrichten verschiedener Fahrzeuge dient. Als Nächstes schließt sich eine Nachrichten-Nummer an, mit deren Hilfe festgestellt werden kann, ob eventuell zu einem bestimmten Zeitpunkt nur von einem Fahrzeug Werte vorliegen. Diese müssen bei der Auswertung dann verworfen werden. Als Letztes werden High- und Low-Part der Zeit, durch ein # getrennt, in die Nachricht eingetragen. Das Zusammensetzten zu einem Zeitstempel übernimmt der NetworkClockAnalyzer.

Die so gebaute Nachricht wird über UDP an den NetworkClockAnalyzer verschickt.

Programm zum Aufzeichnen der übermittelten Zeitstempel

Der NetworkClockAnalyzer ist ein selbst geschriebenes Qt-Programm. Diese Software ist darauf ausgelegt, UDP-Pakete mit Zeitstempeln zu empfangen und in eine CSV-Datei zu speichern. Beim Starten des Programms werden die entsprechenden CSV-Dateien mit dem aktuellen Datum und der aktuellen Zeit im Namen angelegt.

Dem NetworkClockAnalyzer ist es egal, um welche Zeitquelle es sich handelt, solange die empfangenen UDP-Nachrichten dem vorgesehenen Format (siehe Listing 9.1) entsprechen. Dies bedeutet, dass er für alle durchgeführten Zeitmessungen mit PTP und NTP verwendet werden kann. Sollten im Anschluss an die Projektgruppe weitere Zeitsynchronisationsverfahren getestet werden, kann der NetworkClockAnalyzer weiter verwendet werden.

Die empfangenen IP-Adressen der Fahrzeuge werden auf das letzte Segment zusammengekürzt, weil dieses zur Unterscheidung der Fahrzeuge ausreicht. Außerdem werden der High- und Low-Part des empfangenen Zeitstempels in einen 64-Bit-Zeitstempel umgerechnet.

Listing 9.1: Nachrichtenformat einer UDP-Nachricht für den NetworkClockAnalyzer

```
client_nr:192.168.100.34;message_nr:10;time_stamp:83979570#590779080;

client_nr:
IP−Adresse des Fahrzeugs

message_nr:
Fortlaufende Nummer der Messung

time_stamp:
High−Part des Zeitstempels
# Trennzeichen
Low−Part des Zeitstempels
```

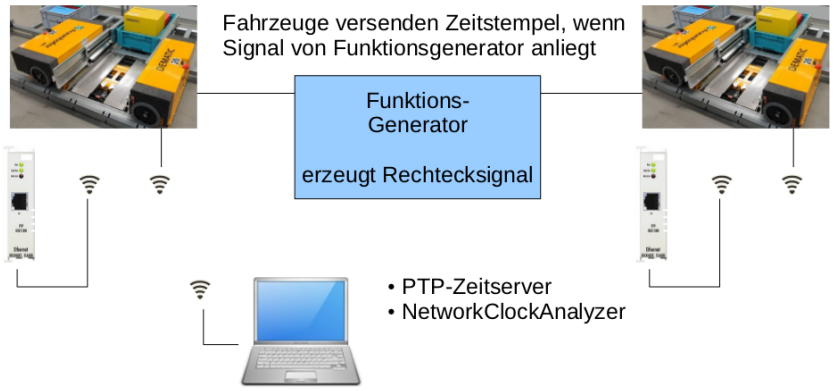


Abbildung 9.14.: Versuchsaufbau für die PTP Zeitmessungen. Die Fahrzeuge synchronisieren ihre Uhrzeiten mit der Zeit, des auf dem Notebook ausgeführten PTPd.

Auswerteskript für die Aufzeichnungen

Zur Auswertung der Messergebnisse wurde ein Octave-Skript geschrieben, das aus den Messdaten verschiedene Diagramme erzeugt und eine Reihe von Informationen, wie beispielsweise die durchschnittliche Zeitabweichung zwischen den Fahrzeugen, ausgibt. Mit diesem Skript wurde die Auswertung der Ergebnisse aus Abschnitt 9.4.4 durchgeführt.

Das Octave-Skript wertet dabei nur Zeitstempel aus, die von allen Fahrzeugen vorliegen. Dies wird anhand der Nachrichtennummer sichergestellt.

9.4.2. Versuchsaufbau für Precision Time Protocol

Gegenüber dem generischen Aufbau aus Abschnitt 9.4.1 müssen für die Auswertung der PTP-Zeitsynchronisation ein paar Änderungen durchgeführt werden.

Weil die PTP-Klemme die Zeitsignale an ihrem eigenen Ethernet-Anschluss erwartet, musste eine Möglichkeit gefunden werden, wie die Pakete aus dem WLAN zu diesem Ethernet-Anschluss gelangen können. Am Ende der Evaluierungsphase wurde der Aufbau aus Abbildung 9.14 verwendet. Hier wurde an jeder der PTP-Klemmen eine eigene WLAN-Bridge angeschlossen, sodass sie direkt über das WLAN mit ihrem Zeitserver kommunizieren können.

Die Aufgabe des Zeitservers übernimmt ein Notebook mit dem in Abschnitt 7.1.3 beschriebenem PTP-Zeitserver PTPd.

9.4.3. Versuchsaufbau für Network Time Protocol

Für die Auswertung der NTP-Zeitsynchronisation kann der generische Aufbau aus Abschnitt 9.4.1 unverändert verwendet werden.

Die Aufgabe des Zeitservers auf dem Notebook, sowie des Zeitdienstes auf den Fahrzeugen übernimmt die NTP-Implementierung der Firma Meinberg, die in Abschnitt 7.2.2 bereits vorgestellt wurde.

9.4.4. Auswertung

In diesem Abschnitt werden die Ergebnisse aus den durchgeführten Zeitmessungen mit den Protokollen NTP und PTP vorgestellt.

Precision Time Protocol

Für die Evaluierung des PTP-Zeitsynchronisationsprotokolls wurden verschiedene Messreihen mit unterschiedlichen Werten für das Aktualisierungsintervall und der Taskzykluszeit durchgeführt.

In Tabelle 9.11 sind die Ergebnisse der beiden besten Zeitmessungen aufgelistet. Bei der PTP-Kurzzeitmessung handelt es sich um eine relativ kurze Messung, die sich über ca. 20 Minuten erstreckt hat. Die PTP-Langzeitmessung erstreckte sich über insgesamt zwei Stunden und 20 Minuten. Wie den zugehörigen Spalten in der Tabelle 9.11 zu entnehmen ist, lag die durchschnittliche Zeitabweichung zwischen beiden Fahrzeugen bei 0,756334 ms bzw. 0,596729 ms. Diese Werte liegen unterhalb der im Pflichtenheft geforderten Zeitgenauigkeit von 10 ms und unterschreiten sogar die optionale Schranke von 1 ms. Wie dem logarithmischen Fehlerdiagramm der PTP-Langzeitmessung (Abbildung 9.15) zu entnehmen ist, wird die gesetzte Zeitschranke aufgrund des nichtdeterministischen Mediums

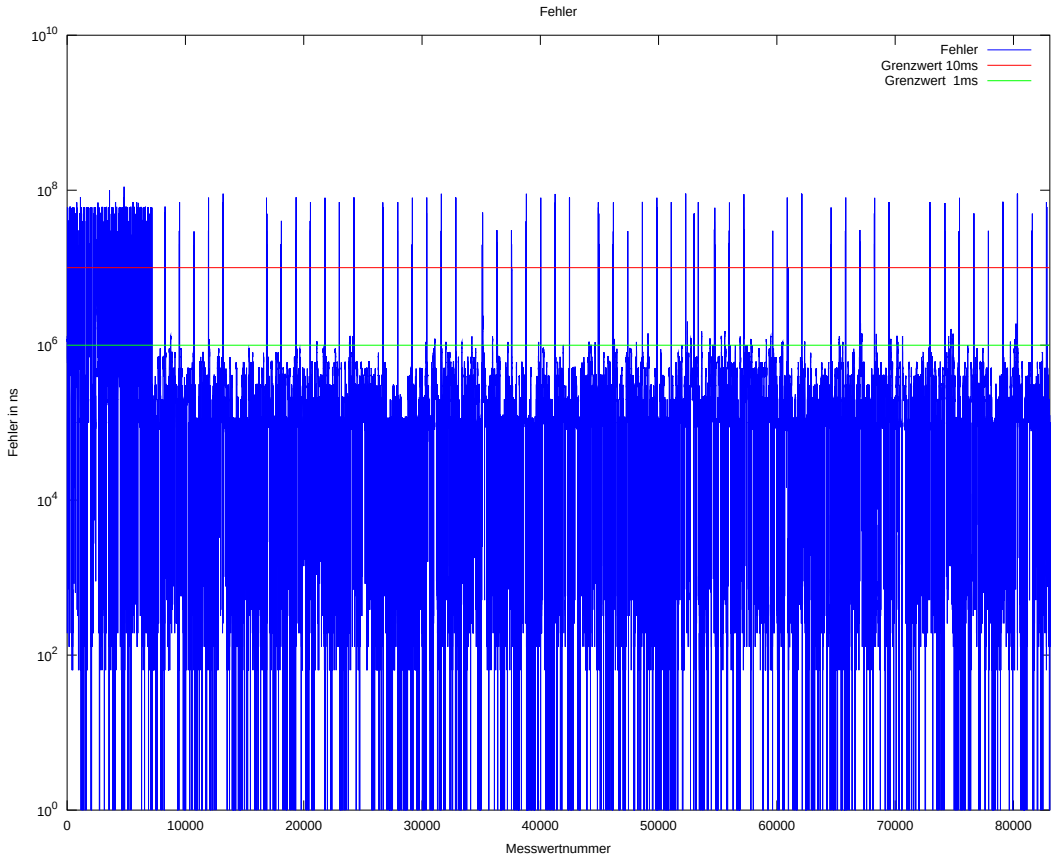


Abbildung 9.15.: Der logarithmische Fehlergraph der PTP-Langzeitmessung mit den eingezeichneten Grenzwerten von 1 ms und 10 ms. Die Einregelzeit am Anfang der Messung ist deutlich zu erkennen.

Messung	Kurzzeitmessung	Langzeitmessung	gefilterte Messung
Protokollversion	v2	v2	v2
Aktualisierungs Intervall	1 s	1 s	1 s
Taskzykluszeit	100 μs	100 μs	100 μs
Messwerte	13519	83813	83813
Abtastrate	10 Hz	10 Hz	10 Hz
Paketverlustrate	1,472 %	0,884 %	0,884 %
max. Fehler	81,800 ms	111,100 ms	7,945 ms
min. Fehler	0,000 ms	0,000 ms	0,000 ms
arithm. Mittel	0,756 ms	0,597 ms	0,285 ms
Standardabweichung	4,216 ms	4,337 ms	0,542 ms
$3 - \sigma$	12,648 ms	13,100 ms	1,629 ms
Median	0,200 ms	0,199 ms	0,105 ms
Tiefpassfilter	-	-	exp. Glättung
Glättungsquotient	-	-	$\alpha = 0{,}1$

Tabelle 9.11.: Ergebnisse der Zeitmessungen mit dem PTP-Protokoll, bestehend aus einer Kurzzeitmessung, einer Langzeitmessung und den gefilterten Ergebnissen der Langzeitmessung.

WLAN nicht in jedem Fall eingehalten. Wenn die UDP-Pakete zur Zeitsynchronisation durch Paketkollisionen verloren gehen, kann die Synchronisationsgenauigkeit kurzzeitig abnehmen.

Ein einfacher Ansatz, der diese Problematik etwas abmildern kann, ist die Verwendung eines Tiefpassfilters, durch den die kurzen Peaks herausgefiltert werden können. Auf die Messwerte der PTP-Langzeitmessung haben wir exemplarisch eine exponentielle Glättung erster Ordnung

$$y_t^* = \alpha \cdot y_t + (1 - \alpha) \cdot y_{t-1}^* \tag{9.2}$$

mit einem Alpha-Wert von 0,01 angewendet und sie in Tabelle 9.11 als gefilterte PTP-Langzeitmessung dargestellt.

Wie der entsprechenden Spalte zu entnehmen ist, hat sich der maximale Fehler auf unter 8 ms reduziert und die Standardabweichung ist ebenfalls deutlich zurückgegangen. Mit einem nachgeschalteten Tiefpassfilter lassen sich somit die Ausreißer verringern und die Synchronisationsgenauigkeit verbessern.

Network Time Protocol

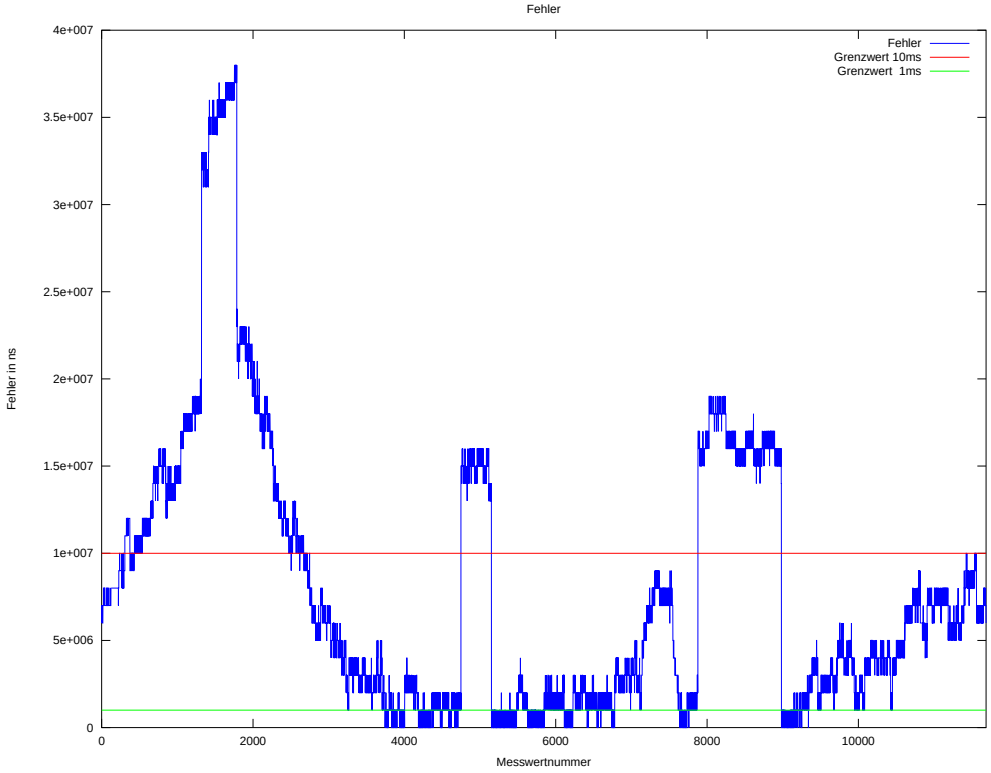
In einer ersten Testreihe wurde die Eignung der beiden NTP-Implementierungen von Microsoft und Meinberg für die Verwendung in der Projektgruppe getestet. Dabei sollten

die Implementierungen vor allem zeigen, wie nah sie an die vorgegebene Zeitschranke von 10 ms herankommen. Während die Software von Meinberg eine Portierung von ntpd, also der Referenzimplementierung von NTPv4 darstellt, die für unsere Tests in Version NTP 4.2.4p8 [Mei12] vorlag, nutzt Microsoft zwar ein zu NTP kompatibles Paketformat und auch einige Algorithmen aus dem NTP-Standard, implementiert aber sonst auf eigene Weise (vgl. [Mic04]). Die NTP-Implementierung von Microsoft verfehlte in den Tests, mit einem eingestellten Poll-Intervall von 1 s, die vorgegebene Zeitschranke im Mittel um den Faktor 100, während die Meinberg-Implementierung, mit einem konfigureirten Poll-Intervall von 16 s, im Mittel unter der geforderten Zeitschranke blieb, weswegen wir uns entschlossen die weiteren Tests ausschließlich mit dieser Implementierung durchzuführen und die NTP-Implementierung von Microsoft für eine mögliche Verwendung in der Projektgruppe ausschlossen.

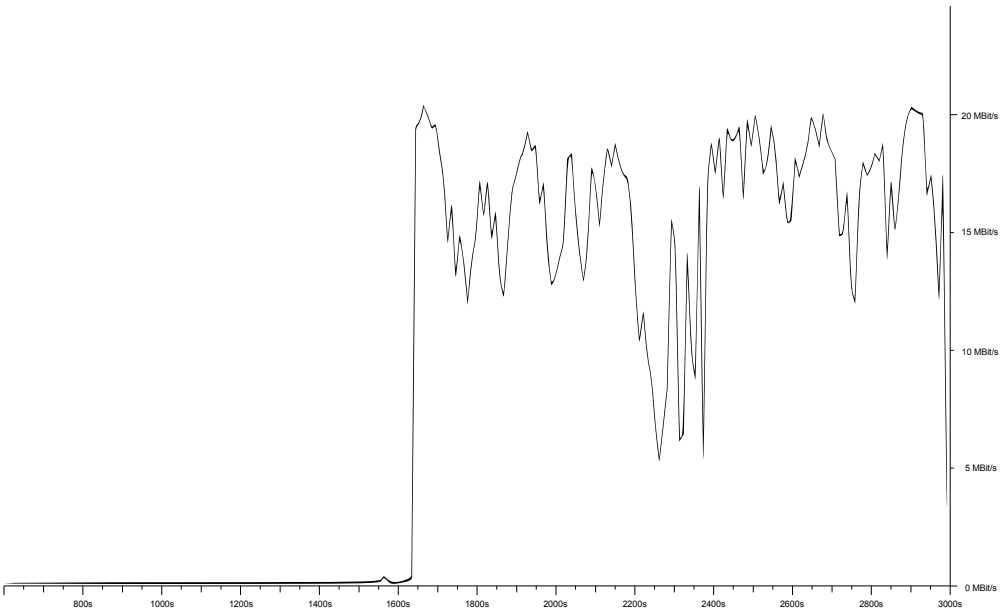
Im weiteren Verlauf wurden Tests bezüglich der Robustheit der NTP-Implementierung auf erhöhte Netzwerklast, sowie die Messung der Einregelzeit durchgeführt. Während NTP-Messung wurde das Übertragungsmedium, ein WLAN-Funknetz nach 802.11a mit einer Bruttoübertragungsrate von 54 $^{\text{Mbit}}/_{\text{s}}$ und einer gemessenen maximalen Nettoübertragungsrate von ca. 21,5 $^{\text{Mbit}}/_{\text{s}}$, mit unregelmäßigen Störübertragungen wie in Abbildung 9.16(b) zu sehen, nahezu vollständig ausgelastet. Die Abweichung zu weiteren Langzeitmessungen ohne erhöhte Netzwerklast liegen im Mittel etwa 2,5 ms höher, während der Median gleich geblieben und die Standardabweichung etwa doppelt so hoch ist. Dieses zeigt, dass das Network Time Protocol robust gegenüber Störungen ist und so auch unter realen, schlechten Bedingungen weiterhin gute Resultate liefert. Die Einregelzeit, d. h. die Zeit die das Protokoll braucht bis es stabile Werte liefert, lag bei allen Messungen bei etwa 15 Minuten bei einem Poll-Inervall von 1 s.

In einer weiteren Testreihe zu NTP wurde versucht, die gewonnenen Messdaten mithilfe eines Tiefpasses weiter zu verbessern und vor allem große Schwankungen und Peaks im Zeitverlauf herauszufiltern. Da die untersuchten NTP-Implementierungen die Systemzeit direkt beeinflussen und in ihren Regelkreis einbinden, müssten für einen praktikablen Einsatz eines solchen Filters die gefilterten Zeitwerte in einer gesonderten Variablen abgespeichert und von Programmen, die diese Zeit nutzen wollen, auch wieder aus dieser ausgelesen werden. Außerdem müsste eine Implementierung einer zweiten Uhr entweder in Hardware oder in Software erfolgen, die in regelmäßigen Zeitabständen die gefilterte Systemzeit erhält.

Für die gefilterten NTP-Messungen wurde hier, wie schon bei PTP, exponentielle Glättung erster Ordnung exemplarisch als Filter verwendet. Für die Messung Tiefpass 1 wurde ein α von 0,01, bei Messung Tiefpass 2 ein sehr kleiner α-Wert von 0,001 verwendet. Wie in Tabelle 9.12 zu sehen ist, ist mit einem schon recht klein gewähltem α von 0,01 die Verbesserung gegenüber den ungefilterten Messdaten nur marginal. Der Grund dafür ist, dass Abweichungen bei NTP nicht in kurzen Peaks wie bei PTP auftreten, sondern in zeitlich viel länger anhaltenden, teilweise in den Messungen bis zu acht Minuten andauernden, sehr breiten Spitzen auftreten.



(a) Fehlergraph von der NTP-Messung



(b) Nettoauslastung des Netzwerkes während der NTP-Messung

Abbildung 9.16.: Fehlergraph und Netzwerklast während der NTP-Messung

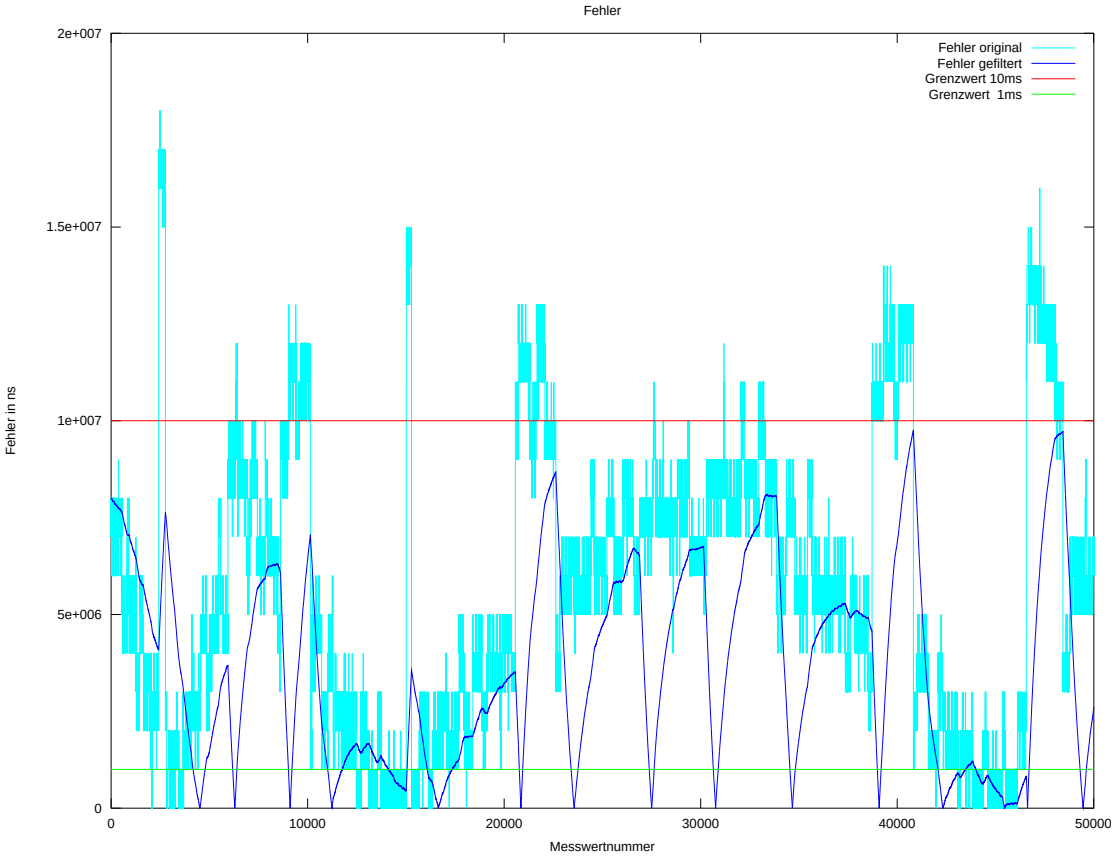


Abbildung 9.17.: Fehlergraph der ungefilterten Daten und der mit Hilfe exponentieller Glättung 1. Ordnung gefilterten Messdaten mit einem α-Wert von 0,001

Messung	ungefiltert	Tiefpass 1	Tiefpass 2
Filter	-	exp. Glättung	exp. Glättung
Glättungsquotient	-	$\alpha = 0{,}01$	$\alpha = 0{,}001$
max. Fehler	18,000000 ms	15,879808 ms	9,756480 ms
min. Fehler	0,000000 ms	0,000128 ms	0,000256 ms
arithm. Mittel	5,697454 ms	5,480008 ms	3,669888 ms
Standardabweichung	3,675134 ms	3,501343 ms	2,567015 ms
$3 - \sigma$	11,025401 ms	10,504030 ms	7,701044 ms
Median	6,000000 ms	5,618240 ms	3,348416 ms

Tabelle 9.12.: Ergebnisse der NTP-Messungen.Zunächst die ungefilterten Daten und dann bei Einsatz eines Tiefpassfilters mit verschiedenen Glättungsquotienten.

Um diese Abweichungen mit Hilfe eines Tiefpasses herauszufiltern bräuchte man einen sehr starken Glättungsquotienten, welcher die Reaktionsfähigkeit verringern, sowie die Einregelzeit deutlich verlängern würde. In Abbildung 9.17 sieht man, dass man mit einem α-Wert von 0,001 bei der Langzeitmessung auch im maximalen Fehler unter 10 ms bleiben würde. Auch wenn ein so hoher Glättungsquotient für das hier gewählte Verfahren eher unüblich ist, so ist in Anbetracht dessen, dass Zeit ohnehin einen streng linearen Verlauf aufweist und nur selten größere Veränderungen im Zeitverlauf, wie zum Beispiel der Umstieg auf Sommerzeit, zu erwarten sind, ein hoher α-Wert durchaus praktikabel. Alternativ kommen natürlich auch andere Filterverfahren zur Stabilisierung der NTP-Zeitwerte in Frage.

9.4.5. Diskussion

Sowohl mit der in Abschnitt 9.4.4 getesteten Version des Precision Time Protocols, die zum Teil in Hardware implementiert war, als auch mit der Software-Implementierung des Network Time Protocols in Abschnitt 9.4.4, ist eine Einhaltung der gegebenen Zeitschranke von 10 ms möglich. Auch wenn der PTP-Versuchsaufbau ein leicht besseres und stabileres Ergebniss auch ohne den Einsatz eines zusätzlichen Tiefpasses aufwies, so ist die NTP-Variante einfacher und kostengünstiger einzurichten.

In Hinblick auf den Einsatzzweck und eine größere Anzahl an Fahrzeugen bleibt zu überlegen, ob für diesen Fall ein Zeitsynchronisationsprotokoll bei dem eine Master-Clock Synchronisationsinformationen mit Hilfe von Broadcast, an eine beliebige Anzahl passiv agierender Slave-Clocks sendet nicht besser geeignet wäre, als ein Protokolltyp wie NTP oder PTP, bei dem jedes zu synchronisierende Gerät bei einer zentralen Instanz Zeitinformationen abfragt. Da eine effiziente und offene Implementierung eines solchen Protokolls nicht zur Verfügung stand und der Traffic bei PTP und NTP auch bei einer größeren Anzahl an Fahrzeugen verhältnismäßig gering wäre, blieb dies eine rein theoretische Überlegung.

9.5. Koexistenz-Untersuchung

Im Rahmen der Koexistenz-Untersuchung haben wir uns mit der Frage beschäftigt, inwieweit sich die zur Verfügung gestellten Sensoren, respektive Microsoft Kinect sowie PMD-Kamera, sich gegenseitig beeinflussen bzw. stören. Vor allem interessierte uns das Ausmaß dieser Störungen, da kleine, sporadisch auftretende Bildfehler durch die Vorverarbeitung der Grafischen Datenverarbeitung (s. Abschnitt 4.4) beseitigt werden können und selbst ein Verlust einzelner Bilder im Kontext der kontinuierlichen Akquisition, keine merklichen Auswirkungen auf den Gesamtbetrieb des Fahrzeuges hätte. Falls diese Störungen allerdings zu groß sind, d. h. sich auf mehrere hintereinander folgende Bilder auswirken, so müssten weitere Maßnahmen zur Synchronisation der Bildakquisition getroffen werden.

Falls eine Evaluation der Koexistenz ergeben würde, dass eine Synchronisation der Bildakquisition notwendig wäre, gab es bereits weitreichende Überlegungen dieses mit einem Zeitschlitzverfahren zu realisieren, was bei der Microsoft Kinect allerdings konstruktionsbedingt nicht ohne Weiteres möglich gewesen wäre.

9.5.1. Versuchsaufbau

Bei den folgenden Untersuchungen wurde das Hauptaugenmerk vor allem auf zwei Konstellationen gelegt, bei denen mit Störungen zu rechnen ist. Zum einen wurde bei beiden Sensortypen erwartet, dass es bei einer Anordnung vis-à-vis Störungen bei der Erfassung geben könnte, zum anderen war damit zu rechnen, dass Probleme, durch Interferenzen bei der PMD-Kamera (s. Abschnitt 2.4.2) bzw. Überschneidung der projizierten Muster bei der Kinect (s. Abschnitt 2.4.4), auftreten können falls zwei Sensoren dasselbe Objekt analog erfassen.

9.5.2. Microsoft Kinect

Da bei ersten Tests mit der Microsoft Kinect bereits durch eine erste visuelle Auswertung ersichtlich wurde, dass durch den Parallelbetrieb zweier Sensoren so gut wie keine Störungen auftraten, erfolgte die weitere Auswertung dieser auch lediglich durch visuelle Kontrolle der erstellten Aufnahmen.

Ergebnisse Kinect

Bei unseren Tests hat sich gezeigt, dass zwei Kinect, sofern sie sich nicht gegenseitig angucken gut nebeneinander betrieben werden können. Beim Betrachten desselben Objektes waren in den Aufnahmen allenfalls kleine Artefakte zu sehen, die auch beim Betrieb von nur einer Kinect in Form des Rauschens auftreten können.

Problematischer ist der Fall, wenn zwei Kinect sich gegenseitig sehen können. Im dabei schlimmsten Fall trifft der IR-Projektor der einen Kinect direkt die IR-Kamera der zweiten Kinect. In diesem Fall wird die zweite Kinect geblendet und es stehen keine ausreichenden Bildinformationen mehr zur Verfügung. In den Aufzeichnungen hat sich dies in Form eines fast vollständig leeren Bildes dargestellt.

Wenn diese Probleme vollständig vermieden werden sollen, müssen die Kinect miteinander synchronisiert werden, damit zu jedem Zeitpunkt nur eine Kinect ein Bild erstellt. Messungen der Ein- und Ausschaltzeiten des IR-Projektors haben ergeben, dass diese ungefähr in der Größenordnung von jeweils 500 ms liegen und eine solche Synchronisation deshalb für unseren Einsatzzweck nicht praktikabel ist. Eine Synchronisation über eine Art Zeitschlitzverfahren ließe sich demnach nur durch zusätzlich anzubringende Mechanik realisieren. Hier könnten z. B. Blenden verwendet werden, die mit Hilfe von kleinen Motoren vor den IR-Projektor geklappt werden.

In dem, von uns erstellten Prototypen findet keine Synchronisation der verschiedenen Kinect statt, weil die Beeinflussung nur in sehr seltenen Fällen groß ist und wir in Kauf nehmen, dass ggf. ein Bild nicht ausgewertet werden kann.

Wie sich die Koexistenz beim Betrieb von mehr als zwei Kinect verhält, wurde von uns nicht evaluiert, weil zum Zeitpunkt der Tests nur zwei Kinect verfügbar waren.

9.5.3. PMD-Kamera

Da die PMD-Kamera viel mehr Möglichkeiten bei der Konfiguration bietet und schon bei einer ersten visuellen Kontrolle Bildfehler entdeckt wurden, wurde bei diesem Sensortyp eine genauere Untersuchung, bezüglich der Koexistenz mehrerer PMD-Kameras, durchgeführt.

Spezieller Versuchsaufbau

Um zu gewährleisten, dass beide PMD-Kameras zum selben Zeitpunkt ihre Aufnahmen machen, wurden die Geräte im Trigger-Modus verwendet und von einem externen Funktionsgenerator mit frei einstellbarer Rechteckfrequenz ausgelöst. Getestet wurde die gegenseitige Störung und Beeinflussung der Messergebnisse der PMD-Kameras bei verschiedenen Blickwinkeln zueinander, sowie verschiedenen Modulationsfrequenzen und Integrationszeiten der Sensoren selber.

In einem ersten Test, der die Beeinflussung der Kameras bei direkter, gegenseitiger Anstrahlung evaluieren sollte, wurden die zum Testen gewählten PMD-Kameras auf einer planen Ebene gegenüber liegend montiert. Bei einem weiteren Test, der dazu diente die Störeinflüsse der PMD-Kameras, bei paralleler Aufnahme eines Objektes zu ermitteln, wurden die beiden Kameras wie in Abbildung 9.18 auf jeweils ein Fahrzeug montiert und im Abstand von 3m zum Referenzobjekt aufgestellt. Der Winkel zwischen den beiden

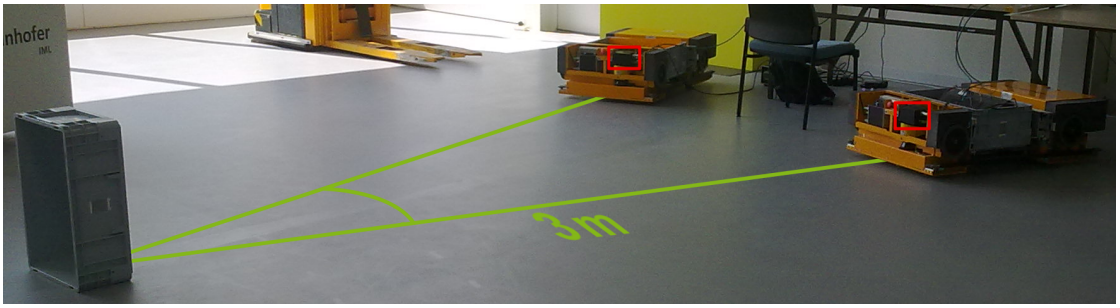


Abbildung 9.18.: Versuchsaufbau mit zwei Fahrzeugen mit montierten PMD-Kameras (siehe rote Umrahmung) und eines Referenzobjekts (graue Kiste links im Bild) im Abstand von 3m

Fahrzeugen betrug in etwa 20°, wobei jedes Fahrzeug um 10° jeweils links beziehungsweise rechts der Referenzobjektnormalen positioniert war.

Auswertungsprogramm für Koexistenz-Messungen

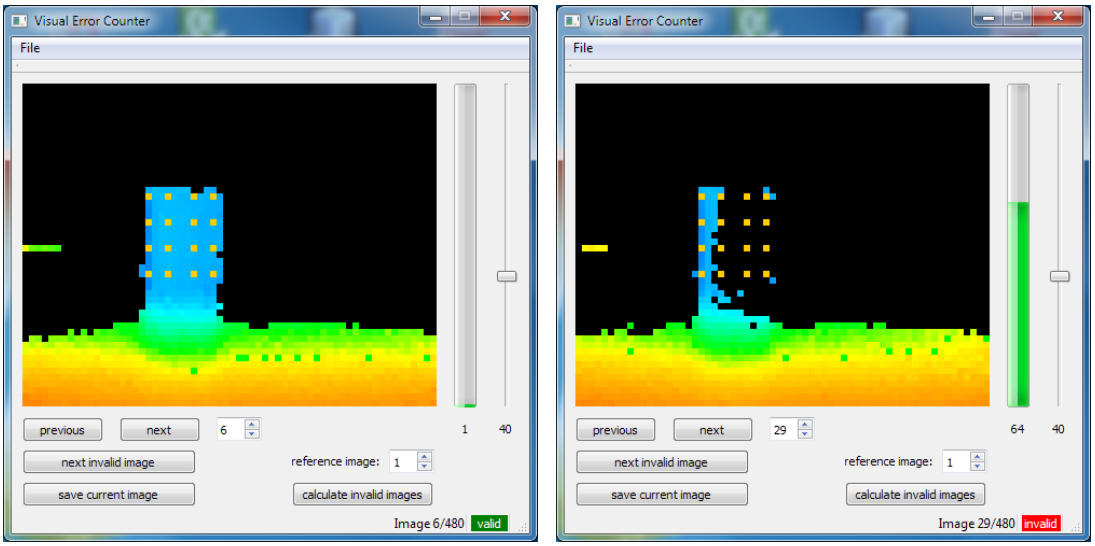
Der VisualErrorCounter ist ein kleines Programm, das im Rahmen der Koexistenz-Analyse der PMD-Kamera geschrieben wurde. Mit ihm können die Aufnahmen der Kamera geladen und ausgewertet werden.

Zur Auswertung werden eine Reihe markanter Punkte in einem gewählten Referenzbild von Hand festgelegt, die anschließend in allen Bildern der Aufzeichnung betrachtet werden, um die Abweichung in der gemessenen Tiefe gegenüber dem Referenzbild zu berechnen. Es kann ein Schwellwert für die erlaubte Tiefenabweichung festgelegt werden, bei dessen Überschreitung ein Bild als fehlerhaft angesehen wird. Der VisualErrorCounter berechnet dann den Anteil fehlerhafter Bilder der gesamten Aufzeichnung sowie die Anzahl an hintereinander folgenden Bildfehlern, die für unsere Einsatzzwecke entscheidend sind, anhand der gewählten Punkte und des Schwellwerts. In Abbildung 9.19 ist der VisualErrorCounter zu sehen.

Zu beachten ist, dass diese Art der Auswertung eine zentrale, subjektive Komponente besitzt, da die Punkte von Hand vorgegeben werden. Da allerdings zum Zeitpunkt der Evaluation der Koexistenz, eine Entscheidung über erkennbare und nicht erkennbare Objekte durch die Grafische Datenverarbeitung noch nicht möglich war und es substanziell darum ging einen Eindruck der möglichen Störeinflüsse zu erhalten, wurde diese Art der Auswertung, mittels des VisualErrorCounters, als ausreichend erachtet.

Ergebnisse PMD-Kamera

Für die Testreihen mit den PMD-Kameras wurden die in Tabelle 9.13 aufgeführten, allgemeinen Versuchsparameter gewählt. Mithilfe eines Funktionsgenerators wurden über



(a) fehlerfreies Bild (b) fehlerhaftes Bild

Abbildung 9.19.: VisualErrorCounter: Ein als gültig deklariertes Bild in 9.19(a) und ein als fehlerhaft deklariertes Bild in 9.19(b), bei dem die Abweichung gegenüber dem Referenzbild größer als der eingestellte Schwellwert ist.

Versuchsparameter	Wert
Aufnahmen pro Sekunde	4
Dauer pro Messung	120 s
Schwellwert im VisualErrorCounter	25 %

Tabelle 9.13.: Gewählte, allgemeine Versuchsparameter für die Testreihen zur Untersuchung der Koexistenz der PMD-Kamera.

f_1	f_2	Integrationszeit	fehlerhafte Bilder
23 MHz (Modus 0)	-	7,9 ms	0 %
23 MHz (Modus 0)	23 MHz (Modus 0)	7,9 ms	2,71 %
23 MHz (Modus 0)	23 MHz (Modus 0)	9,9 ms	17,7 %
20,4 MHz (Modus 1)	23 MHz (Modus 0)	8,5 ms	0 %
20,6 MHz (Modus 2)	23 MHz (Modus 0)	15,7 ms	0 %

Tabelle 9.14.: Ergebnisse der Auswertung von Aufnahmen eines Objektes mit unterschiedlicher Modulationsfrequenzen und Intergrationszeiten zweier PMD-Kameras.

den Trigger-Eingang der PMD-Kameras vier Bilder pro Sekunde aufgenommen. Diese Bilder wurden über einen Zeitraum von zwei Minuten aufgezeichnet und später mithilfe des VisualErrorCounters ausgewertet.

Für die Auswertung innerhalb des VisualErrorCounters wurde ein Schwellwert von 25 % verwendet. Dies bedeutet, dass alle Bilder als fehlerhaft angesehen werden, deren Tiefe anhand der gesetzten Punkte eine Abweichung von mehr als 25 % besitzen.

In Tabelle 9.14 sind die Parameter der durchgeführten Messungen dargestellt. Bei allen Messungen wurde die Integrationszeit von Hand so eingestellt, dass die PMD-Kamera einen ähnlichen Erkennungsbereich wie die Microsoft Kinect aufwies. Bei einer höheren Integrationszeit können weiter entfernte Objekte erkannt werden. Dies geht allerdings zu Lasten der Erkennung von nahen Objekten, die dann häufig überstrahlt sind und nicht mehr erkannt werden.

In der letzten Spalte sind die Ergebnisse der Auswertung mit dem VisualErrorCounter dargestellt. Es fällt direkt auf, dass keinerlei Störungen auftreten, wenn eine einzelne PMD-Kamera betrieben wird (siehe erste Zeile) oder wenn die beiden betriebenen PMD-Kameras mit unterschiedlichen Modulationsfrequenzen arbeiten (siehe vierte und fünfte Zeile). Weil das verwendete PMD-Kamera-Modell insgesamt drei verschiedene Modulationsfrequenzen besitzt, können drei PMD-Kameras gleichzeitig ohne gegenseitige Beeinflussung betrieben werden.

Die Messungen aus Zeile zwei und drei wurden mit identischer Modulationsfrequenz durchgeführt, wobei lediglich die Integrationszeit etwas variiert wurde, um den Erkennungsbereich zu beeinflussen. Wie an den Ergebnissen zu erkennen ist, stören sich auf gleicher Modulationsfrequenz arbeitende PMD-Kameras gegenseitig, wobei die negative Beeinflussung mit steigender Integrationszeit weiter zunimmt.

Aufgrund der Tatsache, dass in den durchgeführten Messungen nur relativ geringe Störungen aufgetreten sind und die weitere Arbeit der Projektgruppe sich auf die Microsoft Kinect konzentriert hat, wurde kein Koexistenz-Verfahren für die PMD-Kameras entwickelt. Für den Betrieb auf den Projektgruppen-Fahrzeugen wäre so ein Verfahren ohnehin

nicht notwendig gewesen, weil ausreichend verschiedene Modulationsfrequenzen für den gleichzeitigen Betrieb zur Verfügung standen.

Eine grundsätzliche Idee für ein Koexistenz-Verfahren ging in die Richtung, dass die PMD-Kameras auf die verfügbaren Modulationsfrequenzen gleichmäßig verteilt werden und zudem ein Zeitschlitzverfahren implementiert wird, welches über den Trigger-Mechanismus der PMD-Kameras den Zeitpunkt der Aufnahmen von Kameras auf derselben Modulationsfrequenz steuern kann. Hierzu müssten jedoch weitere Messungen durchgeführt werden, um zu ermitteln wie viel Zeit das Auslösen einer Aufnahme benötigt.

9.5.4. Diskussion

In den durchgeführten Experimenten hat sich gezeigt, dass mehrere Kameras gleichen Typs sich gegenseitig negativ beeinflussen können. Die entstehenden Beeinträchtigungen reichen dabei von geringen Artefakten, die ähnliche Auswirkungen wie das ohnehin vorhandene Rauschen besitzen, bis hin zum völligen Auslöschen einzelner Aufnahmen.

Im Versuchsbetrieb mit den zwei zur Verfügung stehenden Fahrzeugen haben sich nur geringe Beeinflussungen bei Verwendung der Microsoft Kinect gezeigt, die durch die Bewegung der Fahrzeuge in der Regel maximal ein Bild vollständig unbrauchbar machen. Bei Verwendung der PMD-Kameras kommt es bei zwei Fahrzeugen zu keinerlei Störungen, weil die beiden Kameras mit unterschiedlichen Modulationsfrequenzen und somit ohne gegenseitige Beeinflussung betrieben werden können.

Als Konsequenz der durchgeführten Experimente hat die Projektgruppe auf eine Synchronisation der eingesetzten Kameras verzichtet, weil sich im Versuchsbetrieb mit den zur Verfügung stehenden Fahrzeugen keinerlei negative Auswirkungen gezeigt haben.

Beim Einsatz auf einer größeren Menge von Fahrzeugen müsste die Problematik der Koexistenz allerdings neu evaluiert werden, weil hier unter Umständen größere Störungen auftreten können.

10. Zusammenfassung und Ausblick

Zum Abschluss des Endberichts wird an dieser Stelle eine kurze Zusammenfassung der durchgeführten Arbeiten, Vorgehensweisen und der dabei gesammelten Erkenntnisse gegeben. Ausgehend von diesen Erkenntnissen resultieren offene Aspekte und Erweiterungsmöglichkeiten, die etwa aus technischen Gründen nicht realisiert wurden und aufbauend auf die Arbeiten der Projektgruppe entwickelt werden könnten. Diese Punkte werden in Abschnitt 10.2 gezeigt.

10.1. Zusammenfassung

Ausgehen von der in Kapitel 1.2 thematisierten Zielsetzung der Projektgruppe, bestand diese in der Konzeptionierung und Realisierung eines netzwerkbasierten Sensorsystems, das neuartige 3D-Sensorik (Microsoft Kinect, PMD-Kamera, ASUS Xtion) zur Kollisionsvermeidung verwendet und auf mehreren autonomen Fahrzeugen eingesetzt werden soll. Im weiteren Verlauf der Projektgruppe ließ sich diese Aufgabe in die Teilaufgaben Klassifikation der Sensordaten, Kommunikation der einzelnen Prozesse und Fahrzeuge untereinander, Interaktion mit der Steuerungselektronik sowie einer lokalen Bahnplanung zur Kollisionsvermeidung aufteilen.

Ausgehend von den gesammelten Erkenntnissen der Evaluation (s. Kapitel 9) lässt sich feststellen, dass die genannten Teilaufgaben erfolgreich umgesetzt wurden und die Minimalziele der Projektgruppe sind erreicht worden.

Im Detail betrachtet lässt sich die Lösung der gegebenen Aufgaben in die Bereiche der Sensordatenakquisition, der grafischen Datenverarbeitung, der Middleware und der Bahnplanung unterteilen, auf welche im Folgenden etwas näher eingegangen wird.

Im Bereich der Sensordatenakquisition wurde der Microsoft Kinect als Sensor für die Aufnahme der Fahrzeugumgebung in Form von Punktwolken der Vorzug gegenüber der PMD-Kamera gegeben, da sie über eine höhere Auflösung sowie über ein zusätzlich zum Tiefenbild vorhandenes RGB-Bild verfügt. Mit Hilfe der gewählten Microsoft Kinect wurde eine Akquisition der Sensordaten mit einer Bereinigung und Datenreduktion realisiert. Aus den Daten werden im Zuge der Bereinigung verrauschte Punkte sowie alle zu Boden und Wänden gehörenden Punkte entfernt.

In diesen reduzierten Punktwolken werden verschiedene Objekte durch ein euklidisches Clustering unterschieden und im Rahmen der Klassifikation identifiziert. Hierbei werden Viewpoint-Feature-Histogramme als Klassifikationsmerkmale verwendet und die

Erkennung des Objekttyps mit einer Nächster-Nachbar-Klassifikation realisiert. Für die Weiterverarbeitung der Informationen in der Bahnplanung werden Position und Größe der Objekte an den Bahnplanungsprozess übergeben. Zusätzlich übertragen andere aktive Fahrzeuge ihre Position und Geschwindigkeit über das Netzwerk. Insbesondere im Rahmen der Clusterings tritt die Problematik des Clusterzerfalls auf, wodurch die Qualität der nachfolgenden Klassifikationsschritte negativ beeinflusst wird.

Die Bahnplanung ist in der Lage, das Fahrzeug anzusteuern und Punkte auf der Fläche zu erreichen. Dabei ist zusätzlich das Anfahren eines Punktes in einer vorgegebenen Ausrichtung möglich. Die Wahl der Trajektorie erfolgt mit Hilfe einer Vector-Field-Histogram-Implementierung unter Berücksichtigung der erkannten Hindernisse und der bekannten Positionen anderer Fahrzeuge. Eine Kollision mit der Umgebung wird somit vermieden. Zusätzlich zu der Implementierung der Vector-Field-Histogramme wurde die Anwendbarkeit eines Potentialfeld-basierten Bahnplanungsverfahrens untersucht. Dabei hat sich dieser Ansatz auf Grund der notwendigen Parameteroptimierungen als weniger praktikabel herausgestellt. Zu Entwicklungs- und Demonstrationszwecken wurden eine Simulationssoftware für die Bahnplanungsalgorithmen sowie eine Echtzeitvisualisierung der Fahrzeugpositionen entwickelt.

Zur Realisierung der Kommunikation zwischen verschiedenen Fahrzeugen und zwischen den Prozessen der entwickelten Module wurde eine stark an die gegebene Problemstellung angepasste Middleware erstellt. Diese nutzt die Netzwerkinfrastruktur zum Versenden von Daten zwischen Fahrzeugen und einen Shared Memory zur Interprozesskommunikation.

Auf Grund der Gegebenheit, dass die Software auf einem eingebetteten System läuft, ergeben sich Einschränkungen der verfügbaren Ressourcen und Rechenleistung. Um insbesondere die Algorithmen der Grafikpipeline performant umsetzen zu können, wurde eine Portierung auf die Grafikkarte mit OpenCL untersucht. In idealen Testszenarien ließ sich ein Performancegewinn gegenüber der CPU erzielen, im Ganzen betrachtet war der Gewinn allerdings nicht groß genug, um den Portierungsaufwand aller Verfahren zu rechtfertigen.

Zusätzlich zu den beschriebenen Aspekten wurden Möglichkeiten zur Zeitsynchronisation der Fahrzeuge untersucht. Diese Zeitsynchronisation dient als Grundlage für eine potentielle Sensordatenfusion zwischen verschiedenen Fahrzeugen sowie der zeitlichen Synchronisation im Sinne der Koexistenz von Sensoren mit aktiver Beleuchtung der Umgebung. Die Koexistenz der im Rahmen des Projektes in Betracht gezogenen Sensoren wurde hinsichtlich der gegenseitigen Beeinflussung untersucht. Besonders die PMD-Kameras wiesen dabei einen stärkeren negativen Effekt auf als die Microsoft Kinect oder die ASUS Xtion.

10.2. Ausblick

Aus der Arbeit der Projektgruppe ergaben sich sowohl methodische als auch konzeptionelle Ansätze für mögliche Erweiterungen und alternative Ansätze zur differenzierten Untersuchung – auf diese sei im Folgenden eingegangen.

Ein erster Ansatz besteht in der Erweiterung der entwickelten Sensorplattform um eine zusätzliche Fusion multipler Sensoren. Die gewonnen Sensordaten der Einzelfahrzeuge könnten über ein Netzwerk verbunden werden um ein Gesamtbild der Fahrzeugumgebung zu erzeugen. Hier könnte das System durch Deckenkameras und ein Positionsbestimmungssystem, wie z. B. NanoLoc, ergänzt werden. Aus einer derartigen Erweiterung folgt jedoch unweigerlich die Notwendigkeit einer – auf wenige Millisekunden genauen – zeitlichen Synchronisierung der Sensoren, wie sie theoretisch in Kapitel 7 eingeführt wurde.

Als Ergänzung im Bereich der Sensorik ist der mögliche Austausch der Hauptsensorkomponente durch genauere oder robustere Sensoren wie z. B. 3D-Laserscanner oder PMD-Kameras denkbar. Bei diesen Sensoren handelt es sich um industrieerprobte Hardware.

Auch im Themenkomplex der grafischen Datenverarbeitung ergaben sich weitergehende Ansätze für zukünftige Entwicklungsmöglichkeiten und Betrachtungsansätze. Zur Verminderung der Clustersplit-Rate ist eine Untersuchung weiterer Clusteringverfahren denkbar. Zur Objekterkennung wurde das VFH-Verfahren genutzt, was sich jedoch bei der Erkennung von komplizierten Strukturen (z. B. Menschen und Stationen) als ungünstig erwiesen hat. An dieser Stelle müssten alternative Verfahren getestet und evaluiert werden. Außerdem könnte ein Trackingverfahren zum Einsatz kommen, um Bewegungen von Objekten erfassen zu können.

Da sich die Entwicklungsplattform für den Einsatz von OpenCL als nicht leistungsfähig genug herausgestellt hat, wurde der Einsatz verworfen. Auf einer Plattform mit leistungsfähigerer Hardware könnte der Einsatz von OpenCL die Performance der Grafikpipeline verbessern. Auch der Einsatz von FPGAs wäre an dieser Stelle zur weiteren Leistungssteigerung möglich.

Für die Bahnplanung bietet sich eine Verbesserung der Potentialfeldmethode an. Hierbei wäre u. a. eine automatische Parameteroptimierung denkbar. Außerdem könnte eine alternative Potentialfunktion basierend auf einer Gaußfunktion implementiert werden, um das Potentialfeld kontrollierbarer zur gestalten. Für den Einsatz mehrerer simultan fahrender Fahrzeuge wäre eine globale Bahnplanung von Vorteil, wodurch lokale Minima und mögliche Deadlock-Situationen vermieden werden können.

A. Pflichtenheft

A.1. Zielbestimmung

In großen Logistikzentren kommen zunehmend „Fahrerlose Transportsysteme“ zum Einsatz, die mit Hilfe von „Fahrerlosen Transportfahrzeugen“ Lasten aus einem Lager zu verschiedenen Kommissionierstationen transportieren. Weil häufig mehrere Fahrzeuge gleichzeitig eingesetzt werden, müssen diese so koordiniert werden, dass drohende Kollisionen frühzeitig erkannt und vermieden werden.

Im Rahmen der Projektgruppe NetSensLog soll eine Steuerung für ein dezentrales Transportsystem entwickelt werden, das durch autonom agierende Fahrzeuge flexibel auf die Anforderungen der Warenwirtschaft reagieren kann.

Durch den Verzicht auf eine zentrale Koordination, muss jedes der Fahrzeuge in die Lage versetzt werden, autonom mit seinem Umfeld zu interagieren. Zur Realisierung dieses Ziels werden die Fahrzeuge zusätzlich jeweils mit einer RGB-Kamera sowie einer Kamera zur Tiefenbilderzeugung ausgerüstet. Das Tiefenbild wird entweder mit strukturiertem Licht (Microsoft Kinect) oder mit Hilfe von indirekter Lichtlaufzeitmessung (klassische PMD-Kamera) erzeugt. Aufbauend auf einer echtzeitfähigen Bildverarbeitung soll jedes Fahrzeug sein Umfeld wahrnehmen und Hindernisse erkennen können.

Die auf den einzelnen Fahrzeugen ausgewerteten Sensor-Informationen sollen soweit sinnvoll in Form eines verteilten Sensornetzwerkes mit anderen Fahrzeugen geteilt werden. Konkret betrifft dies zunächst die Informationen über die Postion und Ausrichtung eines Fahrzeuges innerhalb der Versuchshalle (Abbildung A.3), die es ermöglichen sollen, die Qualität der Lokalisation des entwickelten Vision-Systems zu verbessern.

Damit ausgetauschte Sensorinformationen korrekt ausgewertet werden können, muss ein globales Zeitsynchronisationsverfahren eingesetzt werden, mit dessen Hilfe die zeitliche Abweichung zwischen den verschiedenen Fahrzeugen auf ein akzeptables Maß von weniger als 10 ms reduziert werden kann. Als Kommunikationsmedium für dieses Sensornetzwerk wird auf `WLAN` im 5 GHz-Bereich zurückgegriffen

A.1.1. Synchronisation

In einer Testphase soll ein geeignetes Zeitsynchronisationsverfahren evaluiert werden, mit dem die geforderte Genauigkeit von weniger als 10 ms über `WLAN` oder alternativ `nanoLOC` realisiert werden kann. Erste Tests konzentrieren sich dabei auf das Zeitprotokoll `NTP`.

Ist NTP nicht geeignet, muss ein alternatives Synchronisations-Protokoll gefunden werden. Alternativen stellen PTP, RBS und FTSP dar, die noch auf ihre Verwendbarkeit unter Microsoft Windows XP (Embedded Edition) überprüft werden müssen.

Ist ein geeignetes Zeitsynchronisationsverfahren gefunden worden, muss dieses auf den Fahrzeugen umgesetzt werden

Musskriterien:

- Synchronisation auf weniger als 10 ms genau
- Evaluieren eines geeigneten Zeitsynchronisationsverfahrens
- Umsetzung des gewählten Zeitsynchronisationsverfahrens

Wunschkriterien:

- Synchronisation auf weniger als 1 ms genau

Abgrenzungskriterien :

- Eine Zeitsynchronisation auf 10 ms genau ist laut Projektvorgabe ausreichend

A.1.2. Koexistenz von Sensorsystemen

Auf allen Fahrzeugen sollen Sensoren des selben Typs eingesetzt werden (z. B. PMD-Kameras, Microsoft Kinect). Je nach verwendetem Sensortyp können sich die einzelnen Sensoren gegenseitig beeinflussen (aktive Sensoren). Zur Gewährleistung der Koexistenz ist ein `Zeitschlitzverfahren` vorgesehen.

Musskriterien:

- Untersuchung der gegenseitigen Beeinflussung des verwendeten aktiven Sensors
- globales `Zeitschlitzverfahren` (grobgranular)

Wunschkriterien:

- lokales `Zeitschlitzverfahren` (feingranular), d. h. nur Beachtung anderer Sensoren im Beeinflussungsradius

Abgrenzungskriterien :

- keine

A.1.3. Kommunikation

Es soll eine dezentrale Kommunikation zwischen den Fahrzeugen realisiert werden, in deren Rahmen die Statusinformationen der Fahrzeuge kommuniziert werden. Als Kommunikationsmedium soll hierbei `WLAN` verwendet werden (wenn möglich unter Verwendung von Broadcast- oder Multicast-Techniken). Die Kommunikation wird über die bestehende Netzwerkinfrastruktur, bestehend aus vier in der Halle angebrachten WLAN-Access-Points (siehe A.3), die über Ethernet mit dem zentralen Server verbunden sind und zu den Zugangspunkten kompatiblen WLAN-Modulen auf jedem Fahrzeug realisiert.

Musskriterien:

- Bestimmung eines geeigneten Kommunikationsverfahrens
- dezentrale Kommunikation über `WLAN`

Wunschkriterien:

- räumliche Gruppenbildung

Abgrenzungskriterien :

- Es soll die vorhandene Infrastruktur genutzt werden

A.1.4. Bahnplanung

Im Rahmen der *globalen* Bahnplanung soll ein Fahrzeug von einem Punkt A (z. B. ein Lager) zu einem Punkt B (z. B. eine Kommissionierstation) fahren. Hierzu wird die Route vorberechnet. Neben dieser im Rahmen der Projektgruppe *nicht* zu implementierenden, globalen Bahnplanung, soll jedoch eine *lokale* Bahnplanung realisiert werden. Besteht eine Kollisionsmöglichkeit zwischen Fahrzeugen oder einem sich auf der Fahrbahn befindlichen Objekt, so sollen die Fahrzeuge in jedem Fall eine Kollision vermeiden. Die Erkennung einer bevorstehenden Kollision soll über die im Fahrzeug integrierte Sensorik erfolgen. Das systembedingt eingeschränkte Sichtfeld der bildgebenden Sensoren wird hierbei nach Möglichkeit durch die Abstandsinformationen der NanoLOC-Module unterstützt werden.

Eine Kollisionssituation liegt vor, wenn ein Hindernis im Abstand von 4 m von dem Fahrzeug auf seiner geplanten Route erkannt wird.

Bei einer potenziellen Kollison zwischen Fahrzeugen soll über ein Prioritätsverfahren ermittelt werden, welche Fahrzeuge anhalten müssen und welches seine Route vor allen anderen Fahrzeugen fortsetzen darf. Um die Effektivität des Verfahrens zu verbessern, wird den Fahrzeugen mit kleinerer Priorität der Weg des Fahrzeuges mit höchster Priorität mitgeteilt, so dass jene ihre Route neu berechnen können, um so nicht lange warten zu müssen.

Musskriterien:

- lokale Bahnplanung:
  - Kollisionssituationen frühzeitig erkennen
  - Fahrzeug stoppen
  - Prioritätsverfahren (z. B. kleinste IP-Adresse hat höchste Priorität) im Falle einer Kollision mehrerer Fahrzeuge

Wunschkriterien:

- feingranulareres Prioritätsverfahren
- Mitteilen der Fahrzeugroute an andere Fahrzeuge für eine neue Bahnplanung
- Umfahren der Hindernisse, wenn sinnvoll (bei nicht autonom beweglichen, dynamischen Hindernissen)

Abgrenzungskriterien :

- globale Bahnplanung: die globale Bahnplanung umfasst die Berechnung eines günstigen Weges von Punkt A zu Punkt B.

A.1.5. Bewegungsplanung/Lokalisation

Für die Lokalisation (d. h. die Bestimmung der eigenen Position) sollen das Funkortungssystem `nanoLOC` und die Sensordaten einer PMD Kamera oder Microsoft Kinect benutzt werden.

Musskriterien:

- Lokalisation in der Halle auf mindestens 1 m genau (mit Hilfe von `nanoLOC` sowie einer PMD Kamera oder Microsoft Kinect)
- Bestimmung des Gierwinkel mit einer Genauigkeit von $\pm 3°$

Wunschkriterien:

- Genauere Positionsbestimmung durch die Auswertung der Odometrie
- Bestimmung des Gierwinkel mit einer Genauigkeit von $\pm 2°$

Abgrenzungskriterien:

- Kollisionserkennung und Kollisionsvermeidung wird nicht durchgeführt
- Initiale Bestimmung des Gierwinkels erfolgt nicht durch Odometriedaten

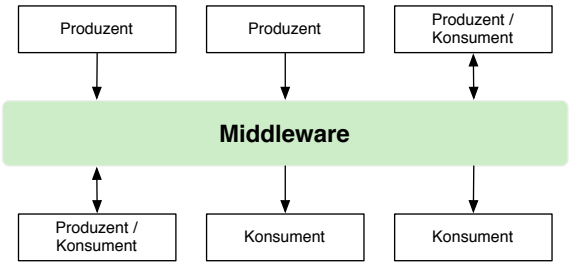


Abbildung A.1.: Zentrale kommunikation über die Middleware

A.1.6. Middleware

Innerhalb der Fahrzeugsoftware gibt es zwei Klassen von Komponenten: Produzenten und Konsumenten. Die Produzenten (z. B. Kameras) erzeugen Daten, die von den Konsumenten (z. B. Bildverarbeitung) verarbeitet werden können (s. Abbildung A.1.6). Konsumenten können Daten bei Produzenten abonnieren. Diese Daten stellen fahrzeugintern Sensordaten, Statusinformationen etc. dar, wohingegen Daten, welche fahrzeugextern über WLAN übertragen werden, reine Statusinformationen. Optional können über WLAN auch Sensordaten, beispielsweise Bilddaten, von einem Fahrzeug (Produzent) zu einem Server (Konsument) übertragen werden. Die Middleware stellt somit die zentrale, vermittelnde Instanz zwischen Produzenten und Konsumenten dar.

Musskriterien:

- Realisierung einer Middleware als Schnittstelle zwischen verschiedenen Produzenten und Konsumenten
  - Produzenten können Daten an Konsumenten senden
  - Konsumenten können Daten von Produzenten empfangen
  - hinsichtlich des Datenaustausches über WLAN werden Statusinformationen übertragen

Wunschkriterien:

- die gesammelten Daten werden über die bestehende Netzwerkinfrastruktur auch anderen Fahrzeugen/Systemen zur Verfügung gestellt
  - möglichst in Echtzeit
  - Übertragung von ausgewählten Bilddaten über WLAN zu einem Server

Abgrenzungskriterien:

- innerhalb der Middleware werden keine datenverändernden Schritte realisiert (die Middleware dient auschließlich als Vermittlungsschicht zwischen Produzenten und Konsumenten)

A.1.7. Sensor-Kalibrierung

Die Sensorik sollte vor ihrer Verwendung kalibriert werden, so dass die akquirierten Messdaten in den nachfolgenden Verarbeitungsschritten direkt genutzt werden können. Die Kalibrierung stellt somit den ersten Schritt der Bildverarbeitungspipeline dar (siehe Abbildung A.1.8).

Musskriterien:

- Überprüfung der verwendeten Sensoren bzgl. ihres Abbildungsverhaltens
- Falls notwendig Kalibrierung der verwendeten Sensoren (PMD-Kamera, Microsoft Kinect, Deckenkameras

Wunschkriterien:

- Fusion der einzelnen Kamerabilder zu einem Gesamtbild

Abgrenzungskriterien:

- Keine Berücksichtigung von thermischen oder optischen (inhomogener Beleuchtung) Einflüssen bei der Kalibrierung

A.1.8. Datenakquirierung

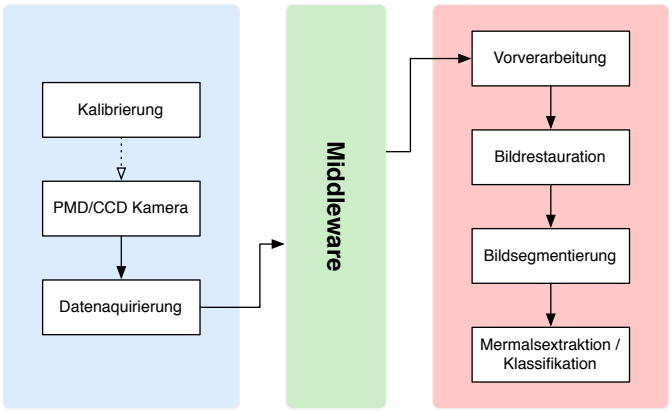


Abbildung A.2.: Datenakquisition und Bildverarbeitung

Im Datenakquirierungsschritt (s. Abbildung A.1.8) werden die Sensordaten von den unterschiedlichen Sensoren angefordert (z. B. von einer RGB- und Tiefenkamera) und anschließend über die Middleware bereitgestellt.

Musskriterien:

- Simultane Datenakquirierung mit min. 10 Hz von einem Tiefensensor und einer RGB-Kamera (anfallende Datenraten siehe Tabelle A.4)

- die Bildinformationen werden über die Middleware bereitgestellt

Wunschkriterien:

- Projektion des Tiefenbildes auf das RGB-Bild

Abgrenzungskriterien:

- im Akquisitonsschritt erfolgen keine weiteren Operationen, wie z. B. Vorverarbeitung, Merkmalsextraktion oder Klassifikation

A.1.9. Bildverarbeitung auf den Fahrzeugen

Die Bildverarbeitungspipeline (rot hervorgehoben in Abbildung A.1.8) realisiert die Schritte: Bildvorverarbeitung, Bildrestauration, Bildsegmentierung, Merkmalsextraktion und Klassifikation. Als Eingabe setzt die Bildverarbeitungspipeline Farb- und Tiefenwerte (siehe Kapitel A.1.8) voraus. Als Ausgabe liefert die Bildverarbeitungspipeline eine entsprechend den Musterklassen annotierte Punktwolke.

Musskriterien:

- Differenzierung zwischen mindestens zwei Musterklassen (Fahrzeuge und Sonstiges)
- die Bildverarbeitungspipeline soll auf der Zielhardware mit mindestens 4 Hz laufen

Wunschkriterien:

- Ausweitung der Musterklassen (z. B. auf Personen, Stationen)
- Unterscheidung zwischen *dynamischen* und *statischen* Objekten

Abgrenzungskriterien:

- die Erkennung der Fahrzeug-IDs ist nicht vorgesehen

A.1.10. Sensordatenfusion

Dem Fahrzeug stehen Sensordaten aus mehreren Quellen zur Verfügung, wie zum Beispiel aus den unterschiedlichen Kameratypen (RGB-, Tiefenkamera), ungefähre Positionsinformationen über `nanoLOC` sowie der Odometrie. Diese sollen für eine genauere Positionsbestimmung fusioniert werden.

Musskriterien:

- die ungefähren Positionsdaten *aller* Fahrzeuge, bestimmt durch `nanoLOC` (auf ca. 2 m genau), und die vom jeweiligen Tiefenmessverfahren akquirierten Tiefendaten sollen gemeinsam genutzt (d. h. fusioniert) werden

Wunschkriterien:

- Steigerung der Effizienz durch Beschränkung der über `nanoLOC` gewonnen Positionsdaten auf einen spezifizierten Radius um die eigene Fahrzeugposition
- unter Umständen können für die Sensordatenfusion folgende Verfahren hinzugenommen werden (falls notwendig): Intensitätsbild (herkömmliche RGB-Kamera, Fischaugen-Deckenkamera), Odometriedaten

Abgrenzungskriterien:

- es sollen keine Bild- oder Tiefendaten zwischen den Fahrzeugen kommuniziert werden (Datenratenbeschränkung des Übertragungsmediums)

A.1.11. Benutzungsoberfläche

Es ist nicht erforderlich, eine Benutzeroberfläche auf den Fahrzeugen oder auf einem zentralen Rechner zur Verfügung zu stellen. Wünschenswert ist dagegen eine Visualisierung auf einem zentralen Rechner. Diese soll eine zweidimensionale Übersicht über den Status der Fahrzeuge geben, indem folgende Punkte realisiert werden:

Musskriterien:

- keine

Wunschkriterien:

- Darstellung der Halle
- Darstellung der Fahrzeugpositionen
- Darstellung der Fahrzeugorientierung
- Darstellung der Sichtfelder der Fahrzeuge
- Darstellung der Sensorbilder einzelner Fahrzeuge

Abgrenzungskriterien:

- keine Visualisierung oder GUI auf den Fahrzeugen

A.2. Produkteinsatz

A.2.1. Anwendungsbereiche

Das während der PG NetSensLog entwickelte System ist für den Einsatz in Distributionszentren und Lagereinrichtungen innerhalb von Hallen vorgesehen. Mit diesem System wird Ware aus Zeilenregallagern zu verschiedenen Kommissionierstationen transportiert.

Für die einzelnen Transportaufgaben werden mehrere autonome Fahrzeuge eingesetzt, die sich sowohl im Regal, wie auch frei auf der Fläche bewegen können. Diese Fahrzeuge verfügen über eine umfangreiche Sensorik, die im innerhalb der PG entwickelten Vision-System dazu genutzt wird, um drohende Kollisionen mit Hindernissen oder anderen Fahrzeugen zu erkennen und entsprechende Maßnahmen zu ergreifen, mit denen diese Kollisionen verhindert werden. Es muss in jedem Fall sichergestellt sein, dass keine Kollisionen stattfinden, da sich auch Personen auf der Fläche bewegen können und diese nicht verletzt werden dürfen.

A.2.2. Zielgruppen

Die während der PG NetSensLog entwickelte Lösung richtet sich an Logistikunternehmen, wie beispielsweise Reichelt oder Amazon, bei denen die Kommissionierung durch Mitarbeiter direkt im Lager oder mit Hilfe von starren Fördertechniken wie etwa Rollenförderern durchgeführt wird.

Durch den Einsatz des neuen Systems können die Arbeitsbedingungen für die Mitarbeiter verbessert werden, weil sie an einer flexiblen Kommissionierstationen während einer Schicht ortsfest arbeiten und die benötigte Ware direkt zu ihnen gebracht wird. Zudem bleibt die Flexibilität innerhalb der Kommissionierung erhalten, weil keine starre Fördertechnik verwendet wird, die sich nur schwer an neue Randbedingungen, wie beispielsweise zusätzliche Kommissionierstationen im Weihnachtsgeschäft anpassen lässt.

A.2.3. Betriebsbedingungen

Für den reibungslosen Betrieb des Systems muss die in Abbildung A.3 dargestellte Halle oder eine äquivalente Umgebung zur Verfügung stehen. Des Weiteren müssen mindestens fünf modifizierte Multishuttle Move betriebsbereit, aufgeladen und mit der für die PG benötigte Sensorik ausgerüstet sein.

A.3. Produktübersicht

Die Spezifikation sieht den Einsatz von mindestens zwei Multishuttle Move Fahrzeugen vor, die autonom ihnen übertragene Aufgaben erfüllen können, ohne dabei miteinander zu kollidieren. Das Gesamtsystem gliedert sich in die folgende drei Komponenten:

Server-Anwendung Ein zentraler Server, dessen Aufgabe es ist, alle rechenintensiven Vorgänge, die nicht von den Multishuttel Move Fahrzeugen geleistet werden können, zu übernehmen. Zusätzlich wird dieser Server zur Visualisierung der einzelnen Fahrzeuge in ihrer Umgebung und der Darstellung von Fahrzeugparameter oder Momentaufnahmen der Fahrzeugsensoren genutzt, soweit diese Funktionen realisiert werden. Eine weitere Aufgabe der Server-Anwendung ist die Auswertung der Deckenkameras zur Lokalisation und Identifikation der Fahrzeuge.

Fahrzeugsoftware auf der ION Plattform Auf ION Plattform werden sowohl alle zur Objekterkennung notwendige Softwarekomponenten implementiert als auch all jene Komponenten welche nicht zwingend auf der im Folgenden beschriebenen *Speicherprogrammierbaren Steuerung* realisiert werden müssen.

Fahrzeugsoftware auf der TwinCat SPS Auf der TwinCat SPS wird all jene Software realisiert, die Echtzeitbedingungen erfüllen muss, welche sich nicht auf der Nvidia ION Systemplattform erreichen lassen. Außerdem wird die Software, welche auf bereits bestehende, in der SoftSPS realisierte, Anwendung basiert, ebenfalls in der SPS umgesetzt.

A.4. Produktfunktionen

Die folgende Auflistung liefert eine Übersicht der bereitgestellten Funktionen. Funktionen sind mit /F Funktionsnummer eindeutig gekennzeichnet. Ein nachgestelltes W kennzeichnet ein Wunschkriterium:

Eingebette Systeme

/F100/Synchronisation: Zeitsynchronisation auf 10 ms genau

/F101W/ Synchronisation: Zeitsynchronisation auf 1 ms genau

/F102/ Synchronisation: Evaluieren eines geeigneten Zeitsynchronisationsverfahrens

/F103/ Synchronisation: Umsetzung des gewählten Zeitsynchronisationsverfahrens

/F104/ Koexistenz von Sensorsystemen: Koexistenz zwischen den verschiedenen Sensoren (eines Fahrzeugs oder mehrere) ermöglichen

/F105/ Koexistenz von Sensorsystemen: globales Zeitschlitzverfahren nutzen

/F106W/ Koexistenz von Sensorsystemen: lokales Zeitschlitzverfahren nutzen

/F107/ Kommunikation zwischen den Fahrzeugen: Auswahl eines geeigneten Kommunikationsverfahren

/F108/ Kommunikation zwischen den Fahrzeugen: dezentrale Kommunikation über WLAN nutzen

/F109W/ Kommunikation zwischen den Fahrzeugen: Gruppenbildung mittels Kommunikation realisieren

Grafische Datenverarbeitung

/F200/ Datenakquirierung: Akquirieren von Tiefendaten und RGB-Kameradaten zur Weiterverarbeitung

/F201/ Datenakquirierung: Bildinformationen geeignet an Middleware übergeben

/F202W/ Datenakquirierung: Projektion des Tiefenbildes auf das RGB-Bild

/F203/ Sensor-Kalibrierung: verwendete Sensoren bzgl. ihres Abbildungsverhalten überprüfen

/F204/ Sensor-Kalibrierung: Wenn nötig Kalibrierung der Sensoren

/F205/ Sensordatenfusion: Fusionierung der jeweiligen Tiefendaten und Positionsdaten aller Fahrzeuge vom *nanoLOC*

/F206W/ Sensordatenfusion: zusätzliche Verwendung von Intensitätsbildern oder Odometriedaten

/F207W/ Sensordatenfusion: Effizienzsteigerung durch die Verwendung nur Positionsdaten von nahen Fahrzeugen

/F208W/ Sensordatenfusion: Fusionierung der einzelnen Kamerabilder zu einem Gesamtbild

/F209W/ Bildverarbeitung: Bildrestauration

/F210W/ Bildverarbeitung: Bildsegmentierung

/F211/ Bildverarbeitung: Merkmalsextraktion/-klassifikation, so dass andere Fahrzeuge von übrigen Objekten unterschieden werden können

/F212W/ Bildverarbeitung: Reduktion der Datenmenge

/F213W/ Bildverarbeitung: Senkung des Rechenaufwands der nachfolgenden Pipeline

/F214W/ Bildverarbeitung: Klassifikation in Fahrzeuge, Sonstiges und optional Stationen

/F215W/ Bildverarbeitung: weitere Unterteilung der zweiten Klasse in dynamische & statische Objekte

/F216/ Lokalisation der Fahrzeuge: Lokalisation des Fahrzeugs auf mind. 1 m genau

/F217/ Lokalisation der Fahrzeuge: Positions- und Gierwinkelbestimmung, Identifikation der Fahrzeuge über Deckenkameras

/F218/ Lokalisation der Fahrzeuge: Gierwinkel mit Genauigkeit von $\pm 3°$ bestimmen

/F219W/ Lokalisation der Fahrzeuge: genauere Positionsbestimmung durch Odometrie

/F220W/ Lokalisation der Fahrzeuge: Gierwinkel mit Genauigkeit von $\pm 2°$ bestimmen

/F221/ Bahnplanung: lokale Bahnplanung

/F222W/ Bahnplanung: feingranulares Prioritätsverfahren

/F223W/ Bahnplanung: Fahrzeugroute an andere Fahrzeuge zur Bahnplanung mitteilen

/F224W/ Bahnplanung: Hindernisse überfahren, falls sinnvoll

/F225/ Middleware: Realisierung als Schnittstelle zwischen Produzenten und Konsumenten

/F226W/ Middleware: gesammelte Daten über die bestehende Netzwerkinfrastruktur auch anderen Fahrzeugen/Systemen zur Verfügung stellen

/F227W/ Benutzungsoberfläche: Darstellung der Halle

/F228W/ Benutzungsoberfläche: Darstellung der Fahrzeugsposition

/F229W/ Benutzungsoberfläche: Darstellung der Fahrzeugorientierung

/F230W/ Benutzungsoberfläche: Darstellung der Sichtfelder der Fahrzeuge

/F231W/ Benutzungsoberfläche: Darstellung der Sensorbilder einzelner Fahrzeuge

A.5. Produktdaten

Aus Benutzersicht werden während des Betriebs keine Daten erzeugt, die längerfristig gespeichert werden müssen.

Auf den Fahrzeugen werden Daten über die Sensorik generiert und es gehen Nachrichten über die Netzwerkverbindung ein. Spätestens wenn das Fahrzeug seinen Zielort erreicht hat, können Daten, die für das Erreichen des Ziels gespeichert wurden, gelöscht werden.

A.6. Produktleistungen

Es werden folgende Leistungen von dem System erbracht:

Musskriterien:

- Es werden mindestens zwei Fahrzeuge unterstützt
- Synchronisation auf 10 ms genau
- Lokalisation in der Halle auf mindestens 1 m genau
- Bestimmung des Gierwinkel mit einer Genauigkeit von $\pm 3°$
- Autonome Fahrzeugsteuerung im Sinne der in Abschnitt A.1.4 beschriebenen Vorgehensweise
- Detektion von Hindernissen mit Hilfe von Tiefenbildern

Wunschkriterien:

- Es werden mindestens fünf Fahrzeuge unterstützt.
- Identifikation der Fahrzeuge
- Synchronisation auf 1 ms genau
- Genauere Positionierung auf < 1 m genau
- Bestimmung des Gierwinkel mit einer Genauigkeit von $\pm 2°$
- Visualisierung

Abgrenzungskriterien:

- Die Koordinierung zur Anfahrt einer der Kommissionierstationen bzw. des Hochregallager, so wie die Fahrt innerhalb dieser Vorrichtungen sind bereits im Vorfeld seitens des Fraunhofer-Institut für Materialfluss und Logistik implementiert worden. Diese können somit vernachlässigt werden.

A.7. Benutzungsoberfläche

Eine Benutzeroberfläche für das System ist nicht erforderlich. Wünschenswert ist eine Visualisierung auf einem zentralen Rechner. Diese soll eine zweidimensionale Übersicht über den Status der Fahrzeuge geben, indem folgende Punkte realisiert werden:

Musskriterien:

- keine

Wunschkriterien:

- Darstellung der Halle
- Darstellung der Fahrzeugpositionen
- Darstellung der Fahrzeugorientierung
- Darstellung der Sichtfelder der Fahrzeuge
- Darstellung der Sensorbilder einzelner Fahrzeuge

Abgrenzungskriterien:

- keine Visualisierung oder GUI auf den Fahrzeugen

A.8. Nichtfunktionale Anforderungen

Fehlertoleranz und Sicherheitsanforderungen

Die Fahrzeuge bestehen aus mehreren voneinander abhängigen Elementen, deren Ausfall unterschiedlich kritische Folgen haben kann. Ein möglicher Ausfall der Antriebseinheit, der dazugehörigen fahrzeuginternen Kommunikationsinfrastruktur und der dazugehörigen Softwarekomponenten wird explizit nicht betrachtet, da diese Systeme bereits gegeben sind und nicht im Rahmen der Projektgruppe entwickelt werden.

Die bereits in den FTF verbauten Laserscanner verbleiben im Fahrzeug und werden direkt mit der Antriebseinheit verbunden. Sie sollen dabei als eine Art Bumper dienen, der sicherstellt, dass das Fahrzeug auch beim Ausfall der im Rahmen der PG entwickelten Software Kollisionen mit der Umgebung zuverlässig verhindern kann. Dabei werden die Laserscanner nicht innerhalb des von der PG entwickelten Systems zur Kollisionsvermeidung verwendet und stellen nur ein zusätzliches Notfallsystem für Probleme während der Testphase dar. Prinzipiell soll das FTF Kollisionen ohne die Laserscanner vermeiden können.

Ein Ausfall der Kameras soll nach Möglichkeit auf der Ebene der Datenakquirierung erkannt werden und mit einem Stillstand der Fahrzeuge quittiert werden.

Leistung und Effizienz

Die Performance der entwickelten Softwarekomponenten zur Akquise der Sensordaten, Verarbeitung der Sensordaten und Lokalisierung wird für die Einhaltung weicher Echtzeitbedingungen in dem Sinne konzipiert, dass die für eine Kollisionsvermeidung notwendigen Daten mit einer in einem Zeitraum von 3 Sekunden durchschnittlich hinreichend großen Frequenz und hinreichend kleinen Latenz vorliegen. Hinreichend bedeutet hierbei, dass eine Kollision in den durchgeführten Testfällen bei den im Anhang angegebenen maximalen Geschwindigkeiten und Beschleunigungen der Fahrzeuge nicht auftritt.

Skalierbarkeit

Es soll grundsätzlich möglich sein, das System mit zusätzlichen FTF zu erweitern. Die Software wird hierbei so ausgelegt, dass eine Verwaltung von bis zu 50 Fahrzeugen prinzipiell durch Wahl geeigneter Datenstrukturen möglich ist. Die Überprüfung der Skalierbarkeit beschränkt sich allerdings auf die Durchführung von Tests mit 2 Fahrzeugen.

Rechtliche Rahmenbedingungen

Da es sich um ein wissenschaftliches Projekt handelt werden rechtliche Rahmenbedingungen für den Betrieb der FTF nicht betrachtet.

A.9. Technische Produktumgebung

Software:

- Microsoft Windows XP Embedded (Multishuttle Move [MOVE]), Windows 7 Enterprise (Server [SUPERVISOR])
- Soft-SPS

Hardware:

- 5 × Multishuttle Move (siehe A.5)
- 5 × NVIDIA ION basierte EEE-PCs [MOVE] (siehe A.6)
- 5 × PMD-Kamera (siehe A.1), Microsoft Kinect (siehe A.2)
- 5 × nanoLOC
- 4 × nanoLOC Referenzknoten
- 4 × MOBOTIX Deckenkamera (siehe A.3 und A.3)
- 4 × WLAN-Zugangspunkte (siehe A.3)
- 1 × Server (CPU: Intel i7-870, Grafik: NVIDIA Quadro 600, Arbeitsspeicher: 8 GB, HDD: 2 TB (RAID 0)) [SUPERVISOR]

A.10. Spezielle Anforderungen an die Entwicklungsumgebung

Software:

- Microsoft Visual Studio 2010
- OpenCL Version 1.1
- Nokia Qt $\geq$ 4.7.2
- Matlab Version 7.12

Hardware:

- siehe Spezifikation in Abschnitt A.12

A.11. Schlussbemerkung

Im Laufe der Programmentwicklung ist es möglich, die in diesem Pflichtenheft aufgeführten Anforderungen durch im Sinne des Arbeitszeitaufwands gleichwertige Anforderungen zu ersetzen oder einzuschränken, falls hierfür eine erhebliche Notwendigkeit entsteht. Gründe können neue wissenschaftliche Erkenntnisse sein oder Programmteile, deren Implementierung sich als nicht umsetzbar herausgestellt haben, sowie unvorhergesehene, hardwareseitige Einschränkungen.

A.12. Anhang

Eigenschaft	Wert
Messwerte	64×50
Öffnungswinkel	$30° \times 40°$
aktive Lichtquelle	850 nm (Infrarot)
Modulationsfrequenz(en)	23 MHz, 20,6 MHz, 20,4 MHz
Schnittstelle(n)	Ethernet

Tabelle A.1.: Die Tabelle zeigt eine Übersicht über die technischen bzw. optischen Eigenschaften der O3D201AB (PMD-Kamera) von ifm. Quelle: ifm

Eigenschaft	Wert
Messwerte	640×480 (Tiefen- und Vollfarbbild)
Messberreich	ca. 0,8 m bis 3,5 m
Öffnungswinkel	$58° \times 40°$
aktive Lichtquelle	830 nm (Infrarot Laser)
Schnittstelle(n)	USB

Tabelle A.2.: Die Tabelle zeigt eine Übersicht über die technischen bzw. optischen Eigenschaften der Microsoft Kienct. Quelle: Microsoft, PrimeSense

Eigenschaft	Wert
Öffnungswinkel	$180° \times 180°$ (L11 Objektiv)
Ausflösung(en)	max. 2048×1536 @ 10 Hz (QXGA), 640×480 @ 30 Hz (VGA)
Schnittstelle(n)	Ethernet (PoE)

Tabelle A.3.: Die Tabelle zeigt eine Übersicht über die technischen bzw. optischen Eigenschaften der Q24M von MOBOTIX. Quelle: MOBOTIX

Sensor	Daten pro Einzelbild	max. Bildrate	resultierende Datenrate
PMD-Kamera	$64 \times 50 \times 4 \times 4 = 50$ kB	~ 15 Hz	$0,73\,^{\mathrm{MB}}/_{\mathrm{s}}$
Microsoft Kinect	$320 \times 240 \times 7 = 525$ kB	30 Hz	$15,38\,^{\mathrm{MB}}/_{\mathrm{s}}$
	$640 \times 480 \times 7 = 2100$ kB	30 Hz	$61,52\,^{\mathrm{MB}}/_{\mathrm{s}}$
	$1280 \times 1024 \times 7 = 8960$ kB	15 Hz	$131,25\,^{\mathrm{MB}}/_{\mathrm{s}}$

Tabelle A.4.: Die Tabelle zeigt eine Übersicht über die anfallenden Datenmengen bei der Bildakquisition unter Berücksichtigung verschiedener Sensoren und Akquisitions-Modi.

Eigenschaft	Wert
Nutzlast	$\leq 40\,\mathrm{kg}$
Eigengewicht	$134\,\mathrm{kg}$
Länge	$1135\,\mathrm{mm}$
Breite	$706\,\mathrm{mm}$
Höhe	$350\,\mathrm{mm}$
Lastaufnahmemittel	zu beiden Seiten ausfahrbarer Teleskoparm
Motor	$24\,\mathrm{V}$ DC
Antrieb	Angetriebsräder im Heck, frei laufende Räder an der Front (Schiene) Differentialantrieb im Heck, Stützrad an der Front (Boden)
Geschwindigkeit	$2\,\mathrm{m/s}$ (Schiene) $1\,\mathrm{m/s}$ (Boden)
Beschleunigung	$1\,\mathrm{m/s^2}$
Sensorik	Odometrie-Sensoren Inertial Measurement Unit (Beschleunigungssensoren & Gyroskope) Laserscanner 5 GHz WLAN-Modul nanoLOC (wird noch nachgerüstet)

Tabelle A.5.: Die Tabelle zeigt eine Übersicht über die technischen Eigenschaften des Multishuttle-Move von Dematic.

Eigenschaft	Wert
Prozessor	Intel Atom 330
Prozessorkerne	2
Taktfrequenz	1600 MHz
Arbeitsspeicher	max. 4 GB
Leistungsaufnahme	8 W
Grafikprozessor	GeForce 9400M
Streaming Multiprozessoren	2
Streaming Prozessoren	2×8
Taktfrequenz	450 MHz
Shader Takt	1100 MHz
Shader Model	4.0
CUDA	Ja
Grafik Arbeitsspeicher	512 MB

Tabelle A.6.: Die Tabelle zeigt eine Übersicht über einige technische Eigenschaften des ION Chipsatzes. Quelle: NVIDIA, Alternate

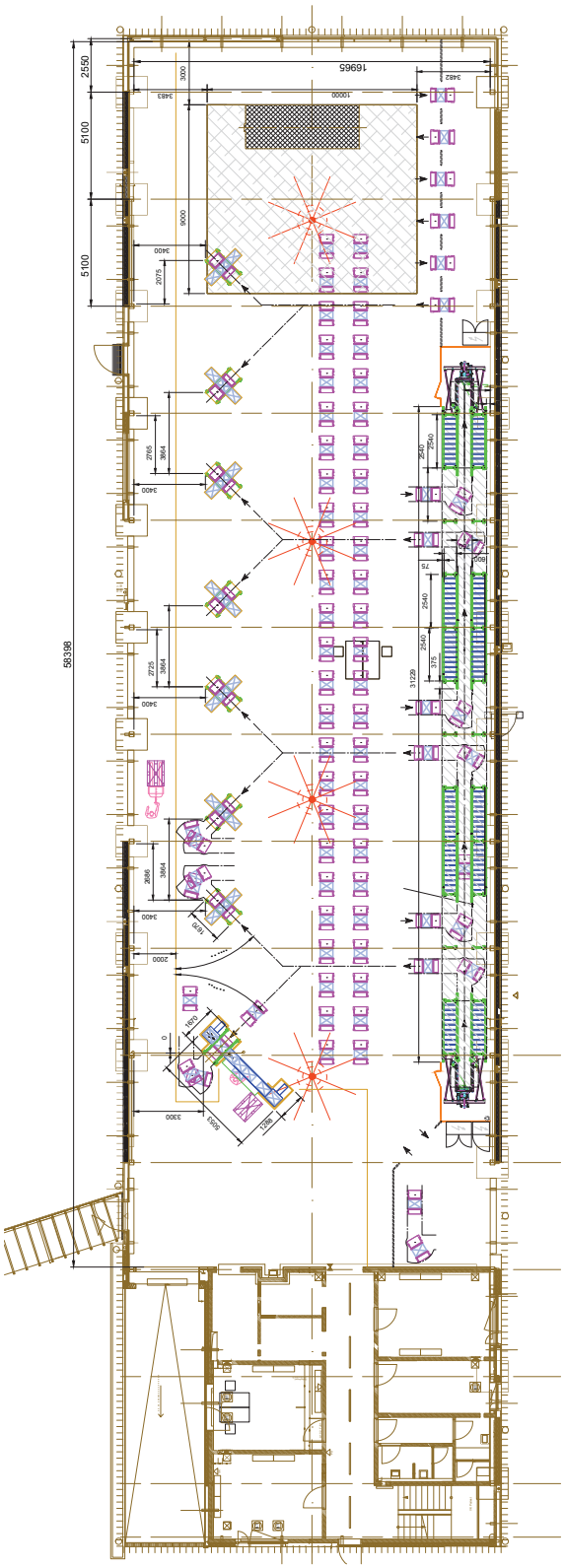


Abbildung A.3.: Die Abbildung zeigt die ZFT-Halle in Draufsicht mit Bemaßungen, u. a. unter Berücksichtigung der Positionen der Kommissionierstationen. Die vier an der Decke angebrachten Fischaugen-Kameras (in der Abbildung ● hervorgehoben) weisen sich partiell überlappende Sichtbereiche auf. Angedeutet sind ferner mehrere Fahrerlose Transportsysteme (● hervorgehoben), die u. a. beim Einfädelungsvorgang in die Kommisionierstationen gezeigt werden.

B. Spezifikationen

B.1. ION

Eigenschaft	Wert
Prozessor	Intel Atom 330
Prozessorkerne	2
Taktfrequenz	1600 MHz
Arbeitsspeicher	max. 4 GB
Leistungsaufnahme	8 W
Grafikprozessor	GeForce 9400M
Streaming Multiprozessoren	2
Streaming Prozessoren	2×8
Gigaflops	54
Taktfrequenz	450 MHz
Shader Takt	1100 MHz
Shader Model	4.0
CUDA	Ja
Grafik Arbeitsspeicher	512 MB

Tabelle B.1.: Die Tabelle zeigt eine Übersicht über einige technische Eigenschaften des ION Chipsatzes. [NVI11]

Glossar

agglomerativ	Bei einem agglomerativen Verfahren werden sukzessive jeweils zwei Teilmengen fusioniert.
CCD-Sensor	Der CCD-Sensor ist ein lichtempfindlicher elektronischer Baustein, der zur Aufnahme von zweidimensionalen Bildern eingesetzt wird.
Cluster	Ein Cluster ist eine Menge von Punkten, die räumlich dicht zusammenliegen und ein gemeinsames Objekt beschreiben.
Clustering	Durch Clustering wird eine große Menge Daten in kleinere, jeweils zusammenhängende Cluster partitioniert.
CSV	*CSV* steht für Comma Separated Values und bezeichnet ein Dateiformat zur textuellen Speicherung einfacher, strukturierter Daten.
CUDA	CUDA wird akkürzend für Compute Unified Device Architecture benutzt und beschreibt eine von NVIDIA entwickelte Technik, mit deren Hilfe Programme auf unterschiedlichen Hardwarekomponenten, wie der GPU, ausgeführt werden können.
Downhill-Simplex	Ein Optimierungs-Verfahren für nichtlineare Funktionen mit multiplen Parametern.
Downsampling	Downsampling ist eine Methode, mit der eine große Menge Daten auf eine kleinere, repräsentative Menge reduziert wird.
Drift	Der Drift oder Systemfehler bezeichnet die Abweichung eines Parameters von seinem Nominalwert.
Echtzeitumgebungen	Innerhalb einer Echtzeitumgebung müssen alle Berechnungen zu einem bestimmten, vorher festgelegten Zeitpunkt fertiggestellt sein.

FPGA ..... *FPGA* steht für Field Programmable Gate Array und bezeichnet programmierbaren intergrierten Schaltkreis

frontolateral ..... Schräg von der Seite betrachtet.

FTF ..... *FTF* ist die Abkürzung für Fahrerloses Transportfahrzeug und bezeichnet ein Fahrzeug, dass automatisch gesteuert wird und dem Materialtransport dient.

FTS ..... Die Abkürzung *FTS* steht für Fahrerloses Transportsystem. Ein Fahrerloses Transportsystem besteht aus mehreren automatisch fahrenden Fahrerlosen Transportfahrzeugen und dient primär dem Zweck des Materialtransportes.

FTSP ..... *FTSP* ist die Abkürzung für Flooding Time Synchronization Protocol und beschreibt ein Protokoll zur Zeitsynchronisation.

GPU ..... Die Abkürzung GPU steht für Graphics Processing Unit und dient zu Berechnung von Grafikausgaben, die an den Bildschirm eines Computers gesendet werden.

IP ..... *IP* steht für Internet Protocol und bezeichnet ein grundlegendes Netzwerkprotokoll der Vermittlungsschicht.

Jitter ..... Mit Jitter wird in der Netzwerktechnik die Varianz der Laufzeit von Datenpaketen bezeichnet.

KLT ..... Ein Kleinladungsträger (KLT) ist ein Ladungsträger kleiner Bauform. Bisweilen wird er auch Kleinlastträger genannt.

Kommissionierplatz ..... An einem Kommissionierplatz werden die dort angelieferten Waren gemäß vorhandener Aufträge zusammengestellt.

Normalenvektor ..... Als Normalenvektor wird ein Vektor bezeichnet, der orthogonal auf einer Fläche, Kurve oder ähnlichem geometrischen Objekt steht.

NTP ..... *NTP* ist die Abkürzung für Network Time Protocol. Es handelt sich bei NTP um ein Protokoll zur Zeitsynchronisation verschiedener Geräte innerhalb eines Netzwerkes.

OpenCL ..... OpenCL steht für Open Computing Language und ist ein Framework zur hochparallelen Berechnung auf Grafikkarten.

PMD ..... Ein Photomischdetektor, auch PMD-Sensor genannt (englisch: Photonic Mixing Device), ist ein optischer Sensor, dessen Funktion auf dem Lichtlaufzeitverfahren beruht.

PTP ..... *PTP* steht für Precision Time Protocol. Das Precision Time Protocol dient der Zeitsynchronisation verschiedener Geräte innerhalb eines Netzwerkes.

Punktwolke ..... Eine Punktwolke ist eine Menge von Punkten, die innerhalb eines bestimmten Raumbereichs liegen.

RANSAC ..... RANSAC steht für RANdom SAmple Consensus und bezeichnet einen Algorithmus, der aus einer Reihe von Messwerten die Parameter eines Modells schätzt.

RBS ..... *RBS* ist die Abkürzung für Reference-Broadcast Synchronization und beschreibt ein Protokoll zur Zeitsynchronisation.

RFC ..... *RFC* steht für Request for Comments und bezeichnet eine Reihe von technischen und organisatorischen Dokumenten zum Internet.

salienter Punkt ..... Als salienter Punkt wird ein in seinem Kontext besonders hervorgehobener Punkt bezeichnet.

Sensorzentrum ..... Bezeichnet den Koordinaten-Ursprung (meist der Mittelpunkt oder die linke obere Ecke) eines Sensors.

SNTP ..... *SNTP* steht für Simple Network Time Protocol und stellt eine einfache Version von NTP dar.

SPS ..... *SPS* ist kurz für speicherprogrammierbare Steuerung. Eine SPS wird zur Steuerung von Maschienen und Anlagen eingesetzt und kann den Anforderungen entsprechend programmiert werden

Stetigförderer ..... Stetigförderer sind Systeme in der Fördertechnik, die einen kontinuierlichen Transportstrom erzeugen.

TCP ..... *TCP* steht für Transmission Control Protocol und bezeichnet ein zuverlässiges, paketvermittelndes und verbindungsorientiertes Netzwerkprotokoll.

TDMA ..... Time Division Multiple Access (kurz TDMA) bezeichnet ein Multiplex-Verfahren bei der Signalübertragung, bei dem den Sendern zur Kollisionsvermeidung bei der Kommunikation disjunkte Zeitschlitze zugewiesen werden.

Tiefenbild ..... In einem Tiefenbild werden pro Bildpunkt anstelle von Farbwerten so genannte Tiefenwerte gespeichert, die die Distanz des Punktes vom Sensorzentrum angeben.

UDP ..... *UDP* steht für User Datagram Protocol und bezeichnet ein verbindungsloses Netzwerkprotokoll.

VFH ..... Das Viewpoint-Feature-Histogram (kurz VFH) ist ein Verfahren zu Generierung Blickwinkel- bzw. Distanzunabhängiger Cluster-Signaturen die u. a. für die Klassifikation von Clustern genutzt werden können.

VFO ..... *VFO* steht für Variable Frequency Oscillator und bildet den Oberbeggriff für einstellbare Oszillatoschaltungen.

Viewpoint ..... Mit dem engl. Begriff viewpoint wird i. A. die Ausrichtung einer (virtuellen) Kamera bzgl. einer Szene bezeichnet.

Vorzone ..... Als Vorzone wir der Bereich vor einem Hochregallager bezeichnet. In diesem Bereich werden die Waren zum Hochregallager transportiert und aus dem Hochregallager abgeholt.

ZFT ..... *ZFT* ist die Abkürzung für Zellulare Fördertechnik

Literaturverzeichnis

[Bec10] Beckhoff Automation GmbH: *Dokumentation für die PTP-Klemme EL6688*. http://infosys.beckhoff.com/content/1031/el6688/html/bt_el6688_title.htm?id=6279. Version: November 2010

[Bec12] Beckhoff Automation GmbH: *Beckhoff New Automation Technology*. http://www.beckhoff.de. Version: Mai 2012

[BJ95] Bunke, Horst ; Jiang, Xiaoyi: *Dreidimensionales Computersehen: Gewinnung und Analyse von Tiefenbildern*. Springer, 1995

[BK91] Borenstein, Johann ; Koren, Yoram: The Vector Field Histogram – Fast Obstacle Avoidance For Mobile Robots. In: *IEEE Journal of Robotics and Automation* 7 (1991), S. 278–288

[Cor10] Corsaro, Angelo: *A gentle introduction – OpenSplice DDS*. August 2010

[EGE02] Elson, Jeremy ; Girod, Lewis ; Estrin, Deborah: Fine-grained network time synchronization using reference broadcasts. In: *SIGOPS Oper. Syst. Rev.* 36 (2002), Dezember, 147–163. http://dx.doi.org/10.1145/844128.844143. – DOI 10.1145/844128.844143. – ISSN 0163–5980

[FB81] Fischler, Martin A. ; Bolles, Robert C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In: *Commun. ACM* 24 (1981), Juni, 381–395. http://dx.doi.org/10.1145/358669.358692. – DOI 10.1145/358669.358692. – ISSN 0001–0782

[Fin11] Fink, Gernot A.: *Vorlesung Mustererkennung*. http://ls12-www.cs.tu-dortmund.de/patrec/teaching/WS10/mustererkennung/index.html. Version: Januar 2011

[FPF96] Fitzgibbon, Andrew ; Pilu, Maurizio ; Fisher, Robert B.: Direct least squares fitting of ellipses. In: *Pattern Recognition, 1996., Proceedings of the 13th International Conference on* Bd. 1 IEEE, 1996, S. 253–257

[Fre08] Frey, Jochen: *Bildsensorik mit Tiefgang – Stand der PMD-Technik, Roadmap & Perspektiven*. 2008

[GC02] Ge, Shuzhi S. ; Cui, Yun J.: Dynamic Motion Planning for Mobile Robots Using Potential Field Method. In: *Auton. Robots* 13 (2002), November, 207–222. http://dx.doi.org/10.1023/A:1020564024509. – DOI 10.1023/A:1020564024509. – ISSN 0929–5593

[HR05] Hauck, Franz J. ; Reiser, Hans P.: *Algorithmen für Verteilte Systeme*. http://www-vs.informatik.uni-ulm.de/teach/ws05/avs/folien/avs-2-3.pdf. Version: 2005

[HR06] Hagebeuker, Bianca ; Ringbeck, Thorsten: *Mehrdimensionale Objekterfassung mittels PMD-Sensorik*. 2006

[HR07] Hagebeuker, Bianca ; Ringbeck, Thorsten: Dreidimensionale Objekterfassung in Echtzeit. In: *Allgemeine Vermessungsnachrichten* 7 (2007), S. 263–270

[IEE08] IEEE Instrumentation and Measurement Society: *IEEE Std 1588-2008, IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*. 2008

[iFi10] iFixit: *Microsoft Kinect Teardown*. http://www.ifixit.com/Teardown/Microsoft-Kinect-Teardown/4066. Version: 2010

[ifm07] ifm electronic GmbH: *O3D1xx Programmers Guide*. 2007

[ifm10a] ifm electronic GmbH: *O3D2xx Programmers Guide*. http://www.ifm.com/ifmweb/downcont.nsf/files/O3D2xx_Programmers_GuideV1-3/$file/O3D2xx_Programmers_GuideV1-3.pdf. Version: 2010

[ifm10b] ifm electronic GmbH: *Operating instructions PMD 3D sensor*. http://www.ifm.com/mounting/704668UK.pdf. Version: 2010

[Int11] Intel: *Intel Atom*. http://ark.intel.com/Product.aspx?id=35641. Version: April 2011

[KR12] Kariluoma, Matti ; Reetz, Daniel: *Kinect Hacking 103: Looking at Kinect IR Patterns*. http://www.futurepicture.org/?p=116. Version: Mai 2012

[Mü11] Müller, Heinrich: *Vorlesung – Graphische Datenverarbeitung*. http://ls7-www.cs.tu-dortmund.de/cms/de/node/2792. Version: April 2011

[Mei12] Meinberg Funkuhren GmbH & Co. KG: *Meinberg – NTP-Implementierung*. http://www.meinberg.de/german/sw/ntp.htm. Version: Juni 2012

[Mic04] Microsoft: *Windows Time Service*. http://technet.microsoft.com/en-us/library/bb490605.aspx. Version: August 2004

[Mic10a] Microsoft: *How the Windows Time Service Works*. http://technet.microsoft.com/en-us/library/cc773013(WS.10).aspx. Version: März 2010

[Mic10b] Microsoft: *PrimeSense Supplies 3-D-Sensing Technology to "Project Natal" for Xbox 360.* http://www.microsoft.com/Presspass/press/2010/mar10/03-31PrimeSensePR.mspx. Version: 2010

[Mic11] Microsoft: *Support boundary to configure the Windows Time service for high accuracy environments.* http://support.microsoft.com/kb/939322/en-us. Version: Oktober 2011

[Mil10] Mills, David L.: *Network Time Protocol 4: Protocol and Algorithms Specification.* Internet Request for Comments RFC 5905, Juni 2010

[MKSL04] Maróti, Miklós ; Kusy, Branislav ; Simon, Gyula ; Lédeczi Ákos: The flooding time synchronization protocol. In: *Proceedings of the 2nd international conference on Embedded networked sensor systems.* New York, NY, USA : ACM, 2004 (SenSys '04). – ISBN 1–58113–879–2, 39–49

[ML09] Muja, Marius ; Lowe, David G.: Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. In: *VISAPP (1)*, 2009, S. 331–340

[Muj11] Muja, Marius: *FLANN, Fast Library for Approximate Nearest Neighbors.* http://mloss.org/software/view/143. Version: 2011

[Mun08] Munshi, Aaftab: *The OpenCL Specification.* http://www.khronos.org/registry/cl/specs/opencl-1.0.29.pdf. Version: August 2008

[NVI10] NVIDIA: *OpenCL Programming Guide for the CUDA Architecture Version 3.1.* http://developer.download.nvidia.com/compute/cuda/3_1/toolkit/docs/NVIDIA_OpenCL_ProgrammingGuide.pdf. Version: Mai 2010

[NVI11] NVIDIA: *NVIDIA.* http://www.nvidia.de/object/geforce_9400m_g_mgpu_de.html. Version: April 2011

[Obj09] Object Management Group: *Documents associated with the Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol specification (DDSI), v2.1.* http://www.omg.org/spec/DDSI/2.1. Version: Januar 2009

[Obj12] Object Management Group: *Object Management Group.* http://www.omg.org. Version: Juni 2012

[PMD11] PMDTec: *CamCube 3.0 (Datasheet V. No. 20100601).* http://www.pmdtec.com/fileadmin/pmdtec/downloads/documentation/datenblatt_camcube3.pdf. Version: 2011

[Pri10] PrimeSense: *PrimeSensor Reference Design 1.08.* http://www.primesense.com/files/FMF_2.PDF. Version: 2010

[Pri12] PrismTech Ltd.: *OpenSplice DDS Data Distribution Service for Real-Time Systems.* http://www.prismtech.com/opensplice. Version: Juni 2012

[RBB09] Rusu, Radu B. ; Blodow, Nico ; Beetz, Michael: Fast Point Feature Histograms (FPFH) for 3D Registration. In: *The IEEE International Conference on Robotics and Automation (ICRA)*. Kobe, Japan, 05/2009 2009

[Reg08] Regele, Ralf: *Kooperative Multi-Roboter-Wegplanung durch heuristische Prioritätsanpassung*. Logos Berlin, 2008

[Rusa] Rusu, Radu B.: Aligning Point Cloud Views using Persistent Feature Histograms. In: *Proceedings of the 21st IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*

[Rusb] Rusu, Radu B.: Learning Informative Point Classes for the Acquisition of Object Model Maps. In: *Proceedings of the 10th International Conference on Control, Automation, Robotics and Vision (ICARCV)*

[Rus09] Rusu, Radu B.: *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*, Computer Science department, Technische Universität Müchen, Germany, Diss., October 2009

[Sch01] Schaufler, Ralf: *Bahnplanung redundanter Roboter auf der Basis einer wiederholbaren inversen Kinematik*. VDI Verlag, 2001

[Sch03] Schneider, Bernd: *Der Photomischdetektor zur schnellen 3D-Vermessung für Sicherheitssysteme und zur Informationsübertragung im Automobil*, Universität Siegen, Diss., 2003

[Win99] Winer, Dave: *XML-RPC Specification*. `http://www.xmlrpc.com/spec`. Version: 1999

[ZSMG06] Zalevsky, Zeev ; Shpunt, Alexander ; Maizels, Aviad ; Garcia, Javier: *Method and System for Object Reconstruction*. 2006. – US Patent App. 20,100/177,164