Universität der Bundeswehr München
Fakultät für Luft- und Raumfahrttechnik
Institut für Technik Autonomer Systeme

Fusion von Kamera und LiDAR zur modellbasierten autonomen Navigation

Dipl.-Inform. Sebastian Schneider

Promotionsausschuss

Vorsitzender:	Prof. Dr.-Ing. Peter Stütz
1. Berichterstatter:	Prof. Dr.-Ing. Hans-Joachim Wünsche
2. Berichterstatter:	Prof. Dr.-Ing. Michael Heizmann

Tag der Prüfung

13. April 2023

Mit der Promotion erlangter akademischer Grad

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

Neubiberg, den 06. Mai 2023

Vorwort

Die hier vorliegende Arbeit beschreibt die Ergebnisse meiner Tätigkeit als Wissenschaftlicher Mitarbeiter am Institut für Technik Autonomer Systeme (TAS) der Universität der Bundeswehr München. An dieser Stelle möchte ich die Gelegenheit nutzen, um den Menschen zu danken, ohne die diese Dissertation nie entstanden wäre.

Zunächst danke ich meinem Doktorvater Prof. Dr.-Ing. Hans-Joachim Wünsche, der mir nicht nur die Möglichkeit gegeben hat, am autonomen Fahren zu forschen, sondern auch die notwendigen Rahmenbedingungen dazu geschaffen hat und mir mit guten Ratschlägen zur Seite stand. Besonderer Dank gilt außerdem meinen TAS-Kollegen Michael Himmelsbach für interessante, zielführende Diskussionen, Pair Programming und unser erfolgreiches sportliches Engagement in inneruniversitären Wettkämpen, Thorsten Lüttel für die Arbeiten am Fahrzeug und Unmengen guter Ratschläge, Michael Manz für neue Sichtweisen und interessante Diskussionen u.a. beim Mittagessen, André Roskopf für Gedankenexperimente, Diskussionen und sein algorithmisches Wissen, Gerhard Rohe für die fachlichen und motivierenden Gespräche und die gemeinsame musikalische Leidenschaft und Martin Ruß für die Ratschläge, Reflexionen und die gemeinsamen freizeitlichen Interessen.

Außerdem bedanke ich mich bei meinen übrigen Kollegen von TAS: Sebastian Bayerl, Martina Falter, Dennis Faßbender, Carsten Fries, Falk Hecker, Benjamin Heinrich, Günther Hofbauer, Hanno Jaspers, Jan Kallwies, Michael Kusenbach, Georg Müller, Anton Pröls, Matthias Schmid, Michael Schweitzer, Alois Unterholzner, Felix von Hundelshausen und Rudi Waldherr. Als Team haben sie mit Hilfsbereitschaft, Motivation und kreativen Ideen die richtige Mischung zwischen Ernsthaftigkeit und Spaß an der Arbeit gefunden und dabei ein extrem angenehmes Arbeitsklima geschaffen.

Danken möchte ich auch den Mitgliedern des Promotionsausschusses, dem Vorsitzenden Prof. Dr.-Ing. Peter Stütz und dem 2. Gutachter Prof. Dr.-Ing. Michael Heizmann für ihre Beteiligung an meinem Promotionsverfahren.

Ganz besonderer Dank gilt meiner Frau Angelika für die jahrelange liebevolle Unterstützung, den notwendigen Ausgleich und dafür, dass sie mir den Rücken freigehalten hat, damit ich mich auf das Schreiben dieser Arbeit konzentrieren kann. Ebenso danke ich meinen Eltern Günter und Hannelore sowie meinem Bruder Daniel dafür, dass sie bereits mein ganzes Leben hinter mir stehen und mich in allen Entscheidungen beraten, unterstützen und motivieren. Außerdem danke ich meinen Schwiegereltern Josef und Barbara für ihre Unterstützung und Motivation sowie die konstruktiven Anmerkungen zu dieser Arbeit. Abschließend bedanke ich mich bei meinen Freunden für die animierenden Gespräche und die vielen Möglichkeiten zum Ausgleich.

Feldkirchen, den 26.10.2022

Hiermit versichere ich, dass ich diese Dissertation ohne fremde Hilfe erstellt, bei der Abfassung keine anderen als die im Schriftenverzeichnis angeführten Hilfsmittel benutzt und die wissenschaftlichen Leistungen eigenständig erbracht habe.

Neubiberg, den 26.10.2022

Zusammenfassung

Autonome Navigation steht seit Jahrzehnten im Fokus vieler Forschergruppen. Die Erfolge der ersten Arbeiten auf diesem Gebiet erleben Kunden heute täglich in Form moderner Fahrerassistenzsysteme, die dem Menschen die Fahraufgabe erleichtern und ihn dabei unterstützen. Anfang dieses Jahrtausends sorgten zwei Entwicklungen zu einem noch stärkeren Wachstum der Forschergemeinde: Einerseits wurde eine Vielzahl unterschiedlichster Sensoren günstiger und damit verfügbarer. Andererseits erreichte die Rechenleistung moderner Computer ein Niveau, das die Verarbeitung dieser Sensordaten erst ermöglichte.

Motiviert durch die Vielzahl unterschiedlicher Sensoren behandelt die vorliegende Arbeit das Thema Sensorfusion mit dem Ziel eine kollisionsfreie autonome Navigation zu ermöglichen. Betrachtet werden Kameras, Laserscanner und inertiale Navigationssysteme auf Seite der Sensorik und u.a. Optimierungsverfahren, zeitliche Filter und Belegungskarten auf der algorithmischen Seite.

Zunächst stellt die Arbeit ein Framework vor, das die zeitliche Synchronisation der verwendeten Sensoren ermöglicht. Dieses Vorgehen erlaubt eine einfache Referenzierung der verschiedenen Sensormessungen zueinander auf zeitlicher Ebene. Auf der geometrischen Ebene zeigt diese Dissertation zwei Verfahren zur extrinsischen Sensorkalibrierung. Das erste dieser Verfahren legt besonderes Augenmerk auf eine korrekte Behandlung von Sensorcharakteristika, wie Auflösung oder Sichtbereich, um eine schnelle und genaue Kalibrierung zu erreichen. Das zweite Verfahren, ein online Verfahren, bestimmt kontinuierlich die relative Lage zweier Sensoren nur anhand der von beiden Sensoren aufgrund ihrer unterschiedlichen Einbaulagen unterschiedlich wahrgenommenen Fahrzeugbewegung.

Mit Hilfe der synchronisierten und kalibrierten Sensoren wird dann ein rekursives Verfahren zur dreidimensionalen Beschreibung der Fahrzeugumgebung in Form von 3D-Punkten entwickelt. Das System vereint die Vorteile von Kamera und Laserscanner und stellt zudem für jeden 3D-Punkt die Unsicherheit der Schätzung bereit. Zusätzlich erlaubt die Parametrierung des Verfahrens eine einfache Auslegung zu Gunsten einer höheren Genauigkeit in der Umgebungsdarstellung oder zu Gunsten eines geringeren Rechenbedarfs.

Schließlich findet eine Kombination dieses Verfahrens mit Algorithmen zur Hindernisvermeidung auf Belegungskarten statt. Dadurch entsteht ein System zur autonomen Navigation, dass durch die Fusion von Kamera, Laserscanner und inertialem Navigationssystem je nach Einsatzzweck und verfügbarer Rechenleistung parametrierbar ist. Integriert auf dem Versuchsträger Munich Cognitive Autonomous Robot Car 4^{th} Generation (MuCAR-4) konnte die Funktionalität in realen, straßenähnlichen Szenarien demonstriert werden.

Summary

Autonomous driving is a key research topic of many research groups worldwide. The initial success stories of that research have meanwhile found their way into customer products in form of driver assistance systems that people experience on a daily basis. With the start of this century research in the field of autonomous driving gained much more momentum due to mainly two reasons: Different types of sensors became affordable and computational power made processing of all that sensor data possible.

Motivated by the amount of different types of sensors this work addresses the research topic sensor fusion, with the goal to provide collision free autonomous navigation. From a sensor point of view this work focuses on cameras, laser scanners and inertial navigation systems. From an algorithmic standpoint this dissertation discusses optimization algorithms, temporal filters and occupancy grids.

The work starts with providing a framework for temporal synchronization of different sensors. This allows for referencing sensor measurements of different sensors on a temporal level. On a geometrical level two methods are proposed that calculate the extrinsic sensor calibration. Focusing on special sensor characteristics like resolution and field of view, the first method provides a fast and accurate sensor calibration, while the second method, an online algorithm, continously provides the relative sensor poses employing only how differently both sensors perceive the vehicle movement due to their different viewports.

Given the availability of synchronized, calibrated sensors the dissertation proposes a recursive algorithm that represents the vehicle's environment in form of a 3D point cloud. Combining the advantages of both cameras and laser scanners this algorithm also provides the estimated uncertainty of each individual 3D point. The parameterization of the system allows for either a more fine grained representation of the environment or a coarser representation at the benefit of lower computational requirements.

Finally, the dissertation combines the 3D pointcloud based environment representation with occupancy grid based obstacle avoidance algorithms. Being based on the fusion of camera, laser scanner and inertial navigation system this combination provides a system for autonomous navigation that can be fine-tuned according to available computational power and intended use. The system was integrated into Munich Cognitive Autonomous Robot Car 4th Generation (MuCAR-4) and its functionality has been demonstrated in real world urban scenarios.

Inhaltsverzeichnis

Symbolverzeichnis

Abkürzungen

Abb.	Abbildung
ACC	Adaptive Cruise Control
AVT	Allied Vision Technologies GmbH
Bd.	Band
BM	Blockmatching
BRIEF	Binary Robust Independent Elementary Features
bzgl.	bezüglich
bzw.	beziehungsweise
CAN	Controller Area Network
CCD	Charge Coupled Device
CEP	Circular Error Probability
CMOS	Complementary Metal Oxide Semiconductor
CPU	Central Processing Unit
d.h.	das heißt
DARPA	Defense Advanced Research Projects Agency
DBW	Drive by Wire
DGPS	Differential Global Positioning System
DOF	Degrees of Freedom
EKF	Extended Kalman Filter
ELAS	Efficient Large-scale Stereo Matching
EM	Expectation Maximization
ESP	Electronic Stability Control
FAST	Features from Accelerated Segment Test
ggf.	gegebenenfalls
GigE	Gigabit Ethernet
GMM	Gauß'sches Mischmodell
GNSS	Globales Navigationssatellitensystem
GPS	Global Positioning System
HDRC	High Dynamic Range CMOS
HTM	Homogene Transformationsmatrix
i.d.R.	in der Regel
ICP	Iterative Closest Point

IMU	Inertial Measurement Unit
INS	Inertial Navigation System
IP	Internet Protocol
ISF	Institut für Systemdynamik und Flugmechanik
KESM	Kinematisches Einspurmodell
KF	Kalmanfilter
KL	Kullback-Leibler Divergenz
KLT	Kanade Lucas Tomasi Feature Tracker
KogMo-RTDB	KogniMobil Real Time Data Base
KogniMobil	Sonderforschungsbereich Transregio 28 – „Kognitive Automobile“
LAN	Local Area Network
LiDAR	Light Detection and Ranging
LM	Levenberg-Marquardt
MarVEye-8	Multifocal Active / Reactive Vehicle Eye 8th Generation
MEMS	Micro Electro Mechanical System
MMSE	Minimum Mean Squared Error
MSE	Mean Squared Error
MuCAR-3	Munich Cognitive Autonomous Robot Car 3rd Generation
MuCAR-4	Munich Cognitive Autonomous Robot Car 4th Generation
OpenCV	Opensource Computer Vision
ORB	Oriented FAST and Rotated BRIEF
OxTS	Oxford Technical Solutions
PF	Partikelfilter
Pixel	Picture Element
Pkw	Personenkraftwagen
RaDAR	Radio Detection and Ranging
RANSAC	Random Sample Consensus
RAS	Rekursives Automotives Stereo
RGB	Rot-Grün-Blau
RMS	Root Mean Square
ROI	Region of Interest
ROS	Robot Operating System
SBAS	Satellite Based Augmentation System
SDK	Software Development Kit
SfM	Structure from Motion
SGM	Semiglobal Matching
SIFT	Scale-Invariant Feature Transform
SIR	Sequential Importance Resampling

SLAM	Simultaneous Localization and Mapping
SLIC	Simple Linear Iterative Clustering
SM	Shared Memory
sog.	sogenannt
SSD	Solid State Drive
SURF	Speeded Up Robust Features
TAS	Institut für Technik Autonomer Systeme
TOF	Time of Flight
u.a.	unter anderem
u.U.	unter Umständen
UDP	User Datagram Protocol
UKF	Unscented Kalmanfilter
USA	United States of America
UT	Unscented Transform
VaMoRs	Versuchsfahrzeug für autonome Mobilität und Rechnersehen
VaMP	VaMoRs-Pkw
WDF	Wahrscheinlichkeitsdichtefunktion
WLAN	Local Area Network
z.B.	zum Beispiel
z.Dt.	zu Deutsch

Formelzeichen

$\mathbf{M}$	Matrix
$\boldsymbol{p}$	3D-Punkt
$\mathbf{P}$	6-DOF-Pose
q	Quaternion
s	Skalar
$\mathcal{P}$	Menge aller Posen
$\bar{\mathrm{p}}$	Pixel ohne Korrektur der Linsenverzeichnung
$\bar{\mathrm{u}}$	Horizontale Pixelkoordinate ohne Korrektur der Linsenverzeichnung
$\bar{\mathrm{v}}$	Vertikale Pixelkoordinate ohne Korrektur der Linsenverzeichnung
$\bar{\mathrm{q}}$	Einheitsquaternion
$\bar{\boldsymbol{v}}$	Einheitsvektor
$\boldsymbol{v}$	Vektor
l^{Basis}	Stereobasis
b	Bildweite

$c_{[u]}$	Horizontale Hauptpunktverschiebung [pix]
$c_{[v]}$	Vertikale Hauptpunktverschiebung [pix]
f	Brennweite
$f_{[u]}$	Horizontale Brennweite [pix], $f_{[u]} = b \cdot k_{[y]}$
$f_{[v]}$	Vertikale Brennweite [pix], $f_{[v]} = b \cdot k_{[z]}$
$\kappa_1, \kappa_2, \kappa_3$	Radial-symmetrische Verzeichnungsparameter
$k_{[y]}$	Skalierungsparameter in y-Richtung [pix/m]
$k_{[z]}$	Skalierungsparameter in z-Richtung [pix/m]
τ_1, τ_2	Radial-asymmetrische und tangentiale Verzeichnungsparameter

Ψ	Gierwinkel
Φ	Rollwinkel
Θ	Nickwinkel

$\mathbf{F}$	Zustandsübergangsmatrix
$f(\cdot)$	Systemgleichungen
$g(\cdot)$	Messgleichungen
$p(\cdot)$	Wahrscheinlichkeitsdichtefunktion
$\mathbf{K}$	Kalman Gain
$\boldsymbol{n}$	Messrauschvektor
$\mathbf{P}$	Schätzfehlerkovarianzmatrix
$\mathbf{Q}$	Systemfehlerkovarianzmatrix
$\mathbf{R}$	Messfehlerkovarianzmatrix
$\mathbf{S}$	Innovationskovarianzmatrix
$\boldsymbol{u}$	Steuergrößen
$\boldsymbol{v}$	Systemrauschvektor
$\boldsymbol{x}$	Zustandsvektor
$\boldsymbol{y}$	Messvektor

f	Frequenz
T	Zeitspanne
t	Zeitpunkt

$\hat{(\cdot)}$	geschätzt
$(\cdot)^{-1}$	invertiert
$(\cdot)^{*}$	prädiziert
$(\cdot)^{T}$	transponiert

$\langle \cdot, \cdot \rangle$	Skalarprodukt
$\|\|\cdot\|\|$	Vektornorm, L2-Norm

Koordinatensysteme

$\mathbf{S}$	Kartesisches Koordinatensystem

$\mathbf{T}$	Kugel-/Polarkoordinatensystem
$(\cdot)$	Beliebiges Koordinatensystem. Verwendet in generischen Gleichungen, in denen das Koordinatensystem anwendungsspezifisch definiert wird.
$\ulcorner$	Koordinatensystem in der linken oberen Ecke eines Bildes
$\boxdot$	Koordinatensystem in der Mitte eines Bildes
ego	Koordinatensystem im Fahrzeugreferenzpunkt
ego,p	Koordinatensystem in der Fahrbahnebene unterhalb des Fahrzeugreferenzpunktes
INS	Koordinatensystem eines Intertialen Navigationssystems
Kamera	Koordinatensystem einer Videokamera
LiDAR	Koordinatensystem im starren Fuß eines LiDAR
MarVEye-8	Koordinatensystem der Kameraplattform MarVEye-8

Glossar

Adaptive Cruise Control

Ein Fahrerassistenzsystem zur automatisierten Längsregelung eines Fahrzeugs, auch bekannt als Abstandsregeltempomat.

Anwendung

Eine Anwendung im Sinne des `tas::app`-Frameworks.

Auslösen

Als Kennzeichnung eines Zeitstempels: Auslösezeitpunkt einer Kamera.

Bayer-Matrix

Geometrische Anordnung von Farbfiltern vor CCD-RGB-Chips zur Kodierung von Farbinformationen in Graubildern.

Belichtung

Als Kennzeichnung eines Zeitstempels: Zeitdauer der Belichtung eines Kamerabildes.

Bildweite

Die Bildweite beschreibt den Abstand zwischen Bildsensor und optischen Zentrum im Lochkameramodell.

Binary Robust Independent Elementary Features

Ein bildbasiertes Verfahren zur Extraktion und Beschreibung von Schlüsselpunkten [Calonder et al., 2010].

Blockmatching

Ein Verfahren zur Berechnung der Disparität von Stereobildern [Chen et al., 2001].

Brennweite

Die Brennweite beschreibt den Abstand zwischen dem Brennpunkt einer Linse und der zugehörigen Hauptebene der Linse.

Central Processing Unit

Zentrale Recheneinheit.

Controller Area Network

Ein serielles Fahrzeugbussystem.

Daten

Als Kennzeichnung eines Zeitstempels: Zeitpunkt der Aufzeichnung von Daten.

Defense Advanced Research Projects Agency

Eine Behörde des Verteidigungsministeriums der United States of America (USA), die Forschungs-Projekte für die Streitkräfte beauftragt.

Differential Global Positioning System

Ein auf Global Positioning System (GPS) basierendes System, das mit Hilfe zusätzlicher Korrekturdaten die Positionsgenauigkeit der Lokalisierung erhöht.

Drive by Wire

Eine Vorrichtung zur Computer-basierten Fahrzeugsteuerung.

dSPACE MicroAutoBox

Echtzeitsystem für schnelles Funktions-Prototyping der Firma dSPACE.

Efficient Large-scale Stereo Matching

Ein Verfahren zur Berechnung der Disparität von Stereobildern [Geiger et al., 2010].

Electronic Stability Control

Ein Fahrerassistenzsystem, das durch gezieltes Abbremsen einzelner Räder dem Ausbrechen eines Fahrzeugs entgegenwirkt.

Erzeuger-Verbraucher-Problem

Klassisches Problem der Prozess- bzw. Thread-Synchronisation zur Regelung der Zugriffsreihenfolge auf geschützte Daten.

Expectation Maximization

Iterative Methode zur Bestimmung der unbekannten Parameter eines statistischen Modells, die die Wahrscheinlichkeit der gegebenen Daten maximieren.

Features from Accelerated Segment Test

Ein bildbasiertes Verfahren zur Extraktion und Beschreibung von Schlüsselpunkten [Rosten und Drummond, 2006].

FireWire

FireWire, i.LINK oder IEEE1394 ist ein Bus für serielle Datenübertragung. Es gibt zwei Versionen: IEEE1394a mit bis zu 400 Mbit/s und IEEE1394b mit bis zu 800 Mbit/s.

Firmware

Hardwarenahe Software, die zumeist die Schnittstelle zwischen Hardware und Anwendungssoftware darstellt.

Framework

Ein Programmiergerüst in der Softwareentwicklung.

Gauß'sches Mischmodell

Ein statistisches Modell bestehend aus der Superposition mehrerer, linear kombinierter Normalverteilungen.

Grand Challenge

Wettbewerb für autonome Fahrzeuge, der in den Jahren 2004 und 2005 durch die Defense Advanced Research Projects Agency (DARPA) veranstaltet wurde.

Ground-Truth

Z.Dt. Grundwahrheit oder Bodenwahrheit. Referenzdaten, deren Korrektheit als gewährleistet gilt.

Inertial Measurement Unit

Ein System zur Messung von Beschleunigungen und Drehraten.

Inertial Navigation System

Ein System, das durch Integrieren der Messungen einer Inertial Measurement Unit (IMU) die räumliche Bewegung bspw. eines Roboters berechnet und i.d.R. mit den Daten eines Positionssensors (z.B. Globales Navigationssatellitensystem (GNSS)) fusioniert.

Iterative Closest Point

Ein Verfahren zur Registrierung von Punktwolken.

Kanade Lucas Tomasi Feature Tracker

Ein bildbasiertes Verfahren zur Extraktion und Beschreibung von Schlüsselpunkten [Lucas und Kanade, 1981].

Klasse

Ein abstraktes Modell der objektorientierten Programmierung zur Beschreibung einer Reihe ähnlicher Objekte.

KogniMobil Real Time Data Base

Ein Echtzeitdatenbanksystem für kognitive Automobile, das im Rahmen des Sonderforschungsbereichs Transregio 28 - "Kognitive Automobile" (KogniMobil) entwickelt wurde.

Kullback-Leibler Divergenz

Ein Maß für den Unterschied zweier Wahrscheinlichkeitsverteilungen.

Levenberg-Marquardt

Numerisches Optimierungsverfahren, benannt nach Kenneth Levenberg und Donald Marquardt, zur Lösung nichtlinearer Probleme nach der Least Squares-Methode.

Likelihood

Ein Maß für die Wahrscheinlichkeit verschiedener unbekannter Werte eines Parameters.

Mean Squared Error

Mittlerer quadratischer Fehler.

Middleware

Anwendungsneutrale Software, die der Vermittlung und dem Datenaustausch anderer Anwendungen dient.

Minimum Mean Squared Error

Minimaler mittlerer quadratischer Fehler.

Mutex

Engl. ***mut****ual* ***ex****clusion*, z.Dt. wechselseitiger Ausschluss. Datenstruktur zum Schutz eines kritischen Abschnitts vor dem Zugriff durch nebenläufige Prozesse oder Threads.

Opensource Computer Vision

Eine C/C++ Softwarebibliothek zur Bildverarbeitung.

Oriented FAST and Rotated BRIEF

Ein bildbasiertes Verfahren zur Extraktion und Beschreibung von Schlüsselpunkten [Rublee et al., 2011].

Prozess

Eine Anwendung im Sinne einer eigenständigen Softwareapplikation.

Ringpuffer

Ringförmige Datenstruktur mit jeweils einem Zeiger auf das nächste zu lesende und das nächste freie Element.

Scale-Invariant Feature Transform

Ein bildbasiertes Verfahren zur Extraktion und Beschreibung von Schlüsselpunkten [Lowe et al., 2004].

Semaphor

Eine mutex-ähnliche Datenstruktur mit integrierter Zählvariablen zur Synchronisation von Threads und Prozessen.

Semiglobal Matching

Ein Verfahren zur Berechnung der Disparität von Stereobildern [Hirschmüller, 2005, Hirschmüller, 2008].

Shared Memory

Teil des Arbeitsspeichers, der von mehreren Prozessen gleichzeitig genutzt werden kann.

Simple Linear Iterative Clustering

Ein Verfahren zur Bildsegmentierung [Achanta et al., 2012].

Speeded Up Robust Features

Ein bildbasiertes Verfahren zur Extraktion und Beschreibung von Schlüsselpunkten [Bay et al., 2006].

Stereobasis

Die Stereobasis beschreibt den Abstand zwischen den zwei Projektionszentren eines Stereokamerasystems.

Structure from Motion

Ein Verfahren zur Berechnung der 3D-Umbebung oder -Struktur von Objekten basierend auf Aufnahmen eines (bildgebenden) bewegten Sensors.

`tas::app`

Am Institut für Technik Autonomer Systeme (TAS) entwickeltes Framework.

Template

Z.Dt. Vorlage, Schablone. Parametrierbarer Datentyp in der Programmierung.

Thread

Z.Dt. Faden, Strang. Eigenständiger Anwendungsstrang innerhalb eines Prozesses.

Übergabe

Als Kennzeichnung eines Zeitstempels: Zeitpunkt der Übergabe von Daten.

Umdrehung

Als Kennzeichnung eines Zeitstempels: Zeitdauer der Umdrehung eines Sensors.

Urban Challenge

Wettbewerb für autonome Fahrzeuge, der im Jahr 2007 durch die DARPA veranstaltet wurde.

User Datagram Protocol

Ein verbindungsloses, IP-basiertes Netzwerkprotokoll.

variadisch

Eindeutschung des englischen Adjektivs *variadic*, in der Anzahl variabel.

Velodyne HDL-64E S1/S2

High Definition LiDAR der Firma Velodyne. In dieser Arbeit werden die Generationen S1 und S2 verwendet.

Videokamera

Ein Gerät zur Aufnahme bewegter digitaler Bilder.

1 Einleitung

Die Erfindung des Begriffs Robotik geht auf das Jahr 1941 zurück und wurde erstmals in einer Kurzgeschichte des russisch-amerikanischen Schriftstellers und Professors der Biochemie Isaac Asimov verwendet. Interessanterweise war sich Asimov der Erfindung dieses Begriffs nicht bewusst, sondern ging in Analogie zur Elektronik – der Lehre der Steuerung von Elektronen – davon aus, dass der Begriff Robotik die Lehre der Entwicklung von Robotern beschreibt. Die Bezeichnung Roboter entstammt ebenfalls der Feder eines Autors und zwar der des tschechischen Schriftstellers Karel Čapek. Es basiert auf dem slavischen Wort *robota*, z. Dt. Arbeit.

Seither befasst sich die Robotik mit der Erfindung und dem Bau von Maschinen zur Verrichtung von Arbeit, die einerseits den Menschen entlastet und andererseits für Menschen u.U. gar nicht verrichtbar ist, wie beispielsweise unter Wasser oder im Inneren von Vulkanen. Die technologischen Fortschritte der Robotik im letzten Jahrhundert sind enorm. Dadurch übertreffen Roboter in vielen Disziplinen zum Teil schon seit längerem die Fähigkeiten von Menschen: Industrieroboter heben schwerere Lasten und fertigen Teile schneller und genauer als der Mensch. Forschungsroboter führen Missionen unter Wasser, im luftleeren Raum ohne Schwerkraft oder in Bereichen hoher Strahlung aus. Medizinische Roboter unterstützen Ärzte bei Operationen oder ermöglichen gehbehinderten Menschen in Form intelligenter Prothesen das Laufen.

In den genannten Fällen unterscheiden sich Roboter gegenüber dem Menschen vor allem durch Schnelligkeit und Robustheit in der Erledigung einzelner Aufgaben. Auf der kognitiven Seite sind derartige Systeme dem Menschen noch immer unterlegen. Dies wird insbesondere bei autonomen Robotern deutlich, die in der Lage sein müssen, auf ihre Umgebung selbständig zu reagieren. Eine Schlüsselrolle kommt dabei der Wahrnehmung zu. Wie weit die kognitiven Fähigkeiten der Robotik im Bereich der Wahrnehmung dem Menschen unterlegen sind, lässt sich leicht bei Kindern beobachten, die innerhalb von nur wenigen Jahren Dinge erlernen, an denen im Vergleich dazu im Bereich der Robotik von einer Vielzahl an Forschern weltweit bereits seit Jahrzehnten geforscht wird, wie bspw. Greifen, Fortbewegung oder Lesen.

Auch wenn die erzielten Leistungen beeindrucken, sind sie das Ergebnis jahrelanger Forschung und Implementierung im Vergleich zu einer eher kurzen Periode weitgehend selbständigen Lernens bei Kindern.

Wie wichtig die Wahrnehmung insbesondere für autonome Fahrzeuge ist, hat Professor Dickmanns bereits in den 1980-er Jahren erforscht und neben unzähligen entwicklungsbegleitenden Veröffentlichungen später in seinem Buch [Dickmanns, 2007] veröffentlicht. Mit Hilfe des 4D-Ansatzes [Wünsche, 1988] entstanden autonom fahrende Fahrzeuge, wie das Versuchsfahrzeug für autonome Mobilität und Rechnersehen (VaMoRs) und etwas später die Pkw-Variante (Personenkraftwagen) VaMoRs-Pkw (VaMP). Trotz der im Vergleich zu heute geringen Rechenleistung konnten beide Fahrzeuge rein kamerabasiert u.a. autonom Fahrspuren folgen [Dickmanns und Zapp, 1987a,b]. Diese Technologie bildet die Grundlage der heutigen Spurhalteassistenten und Spurverlassenswarner in modernen Pkw.

Das autonome Fahren erfuhr erneute Aufmerksamkeit durch die von der Defense Advanced Research Projects Agency (DARPA) organisierten Wettbewerbe *Grand Challenge* in den Jahren 2004 und 2005 und der darauf folgenden *Urban Challenge* im Jahr 2007. Im Gegensatz zu VaMoRs und VaMP waren die 2005 erfolgreichsten Fahrzeuge *Stanley* [Thrun et al., 2006] sowie *Sandstorm* und *H1ghlander* [Whittaker et al., 2006] nicht mehr nur mit Kameras ausgerüstet. Stanley und *Sandstorm* verfügten zusätzlich über RaDAR-Sensoren (Radio Detection and Ranging) und LiDAR-Sensoren (Light Detection and Ranging), *H1ghlander* sogar nur über LiDAR-Sensoren.

Zur *Urban Challenge* zwei Jahre später traten weitere Teams, wie der Gewinner *BOSS* [Urmson et al., 2008] der Carnegie Mellon Universität sowie *Junior* [Montemerlo et al., 2008] und auch einige überwiegend deutsche Teams wie *AnnyWAY* [Kammel et al., 2008], *CarOLO* [Basarke et al., 2008] und *Spirit of Berlin* [Rojo et al., 2007] erfolgreich an. In den letzten Jahren beschäftigen sich zunehmend auch privatwirtschaftliche Unternehmen mit dem autonomen Fahren. [Daily et al., 2017] zeigt, dass neben klassischen Automobilherstellern, wie BMW [Frickenstein et al., 2020], auch zunehmend Softwareunternehmen dem Trend des autonomen Fahrens folgen. So hat die Google Muttergesellschaft Alphabet schon vor einigen Jahren Waymo [Pennecot et al., 2015] gegründet, die Fahrdienstvermittler UBER und Lyft arbeiten an eigenen selbstfahrenden Autos und auch in China investiert Baidu [Li et al., 2020] sowie eine Reihe von Start-Ups in autonomes Fahren. Auch hier ging mit der überwiegenden Verwendung von Kameras, RaDAR und LiDAR der Trend in Richtung Multi-Sensorsystem weiter.

Derartige Multi-Sensorsysteme haben den Vorteil, dass sich die Eigenschaften der Sensoren ergänzen. Beispielsweise liefert ein LiDAR vergleichsweise genaue Entfernungsmessungen, ist hinsichtlich der lateralen Auflösung im Allgemeinen jedoch Kameras unterlegen. RaDAR-Sensoren sind unempfindlicher gegenüber Nebel, Staub und anderen Witterungseinflüssen und messen durch den Doppler-Effekt sogar die relative Geschwindigkeit anderer Objekte. Allerdings sprechen RaDAR-Sensoren insbesondere auf metallische Objekte an. Farbinformationen sind wiederum nur durch eine Kamera zugänglich.

1.1 Forschungsbeitrag

Neben den genannten Vorteilen birgt die Multi-Sensorfusion auch die Notwendigkeit der Sensorkalibrierung und Synchronisation: Um die Sensordaten eines Sensors mit denen eines anderen zu fusionieren, müssen die unterschiedliche Verbauposition und ein etwaiger zeitlicher Versatz der Messungen berücksichtigt werden.

Zu diesem Zweck beschreibt diese Arbeit zunächst ein System zur zeitlichen Synchronisation von mehreren Kameras sowie zwischen einem LiDAR-Sensor und einer Kamera. Basierend auf der Drehfrequenz eines Velodyne HDL-64E und der Belichtungszeit einer Allied Vision Technologies GmbH (AVT) Guppy F-036C Kamera wird der Auslösezeitpunkt dieser extern triggerbaren Kamera derart gewählt, dass beide Sensoren den selben Umgebungsbereich möglichst zeitgleich vermessen.

Besondere Aufmerksamkeit widmet diese Arbeit dem Thema Sensorkalibrierung. Während die intrinsische Kalibrierung auf bekannten, robusten Verfahren beruht und als gegeben angesehen wird, untersucht diese Arbeit eingehend die Kalibrierung von LiDAR und Kamera unter Berücksichtigung der speziellen Sensorcharakteristika, wie u.a. lateraler Auflösung und Entfernungsauflösung. Dazu beschreibt das Verfahren zunächst die Detektion und zeitliche Verfolgung eines schachbrettartigen Kalibrierkörpers in Punktwolken und Kamerabildern. Ausgehend von geschätzten 3D-Koordinaten der vier Eckpunkte bzgl. beider Sensoren wird die relative Pose beider Sensoren iterativ mit Hilfe numerischer Optimierung berechnet. Schrittweise findet anschließend eine Anpassung der Zielfunktion der Optimierung an die Sensorcharakteristika statt. Letztlich führt das zu einer hybriden Metrik, die der rein Eckpunkt-basierten Lösung sowohl in der Genauigkeit als auch in der Zahl der benötigten Iterationen überlegen ist.

Motiviert durch die Entwicklung im Bereich der *Visuellen Odometrie* zeigt diese Arbeit ebenfalls ein neuartiges Verfahren zur extrinsischen *online*-Kalibrierung verschiedener Sensoren. Das Verfahren zeigt, dass es möglich ist, rein auf Basis synchronisierter Eigenbewegungen von Sensoren die Relativlage und -Orientierung der Sensoren zu ermitteln. Das Verfahren wird dazu verwendet, um die Pose eines Stereo-Kamerasystems bzgl. eines Inertialen Navigationssystems kontinuierlich während der Fahrt zu schätzen. Die Arbeit vergleicht drei aufeinander aufbauende Ansätze, die sich hauptsächlich in der Berücksichtigung und Behandlung von Messfehlern unterscheiden. Da das rekursive Verfahren auf einem Unscented Kalmanfilter (UKF) basiert, wird insbesondere auf die Verwendung der Unscented Transform (UT) eingegangen, die eine Transformation Gauß'scher Verteilungen durch nicht-lineare Funktionen ermöglicht. Diese, vom UKF losgelöste Verwendung der UT wird in der vorliegenden Arbeit mehrfach verwendet.

Das zweite Hauptgebiet dieser Arbeit widmet sich dem Thema Sensorfusion. Im Rahmen der Fusion von Rohdaten eines LiDAR und einer Kamera, d.h. von Punktwolken und Bildern, wird eine Lösung für die Verdeckungsrechnung entwickelt, deren Notwendigkeit in den unterschiedlichen Einbaulagen der Sensoren begründet ist. Basierend auf den Stereobildern eines Kamerapaares zeigt diese Arbeit einen neuartigen, rekursiven Stereoalgorithmus, in dem die Umgebung durch eine Menge zeitlich gefilterter Punkte repräsentiert wird. Anfangs nur für statische Umgebungen geeignet wird der Algorithmus schrittweise auf dynamische Umgebungen erweitert. Damit sind für jeden Punkt Position und Geschwindigkeit samt Schätzunsicherheiten bekannt. Im ersten Schritt findet eine Evaluierung und ein Vergleich mit LiDAR-Messungen statt, bevor das Verfahren nochmals erweitert wird und damit auch Entfernungsmessungen eines LiDAR mit eingehen.

Der Nutzen kalibrierter Sensoren und deren Fusion, wie in dieser Arbeit gezeigt, wird abschließend anhand der autonomen Hindernisvermeidung unseres Versuchsträgers Munich Cognitive Autonomous Robot Car 4^{th} Generation (MuCAR-4) beschrieben und mit realen Fahrten evaluiert. Dabei wird ebenfalls auf Themen wie lokale Pfadplanung und Belegungskarten eingegangen.

Einzelne Teile dieser Arbeit wurden bereits auf internationalen Konferenzen veröffentlicht und zeigen damit die wissenschaftliche Relevanz der behandelten Themen auf. Die entsprechenden Arbeiten sind in den jeweiligen Fachkapiteln referenziert.

1.2 Struktur der Arbeit

Zum thematischen Überblick beschreibt Kapitel 2 den aktuellen Stand der Forschung auf Basis verwandter Veröffentlichungen aus dem Bereich des autonomen Fahrens. Diese Auflistung ist dabei in vier Hauptgebiete untergliedert. Abschnitt 2.1 behandelt das Thema Sensorkalibrierung, bevor Abschnitt 2.2 eine Zusammenfassung der Veröffentlichungen im Bereich Sensorfusion gibt. Das Thema 3D-Rekonstruktion und Mapping deckt Abschnitt 2.3 ab. Zum Abschluss gibt Abschnitt 2.4 den Stand der Forschung anhand ausgewählter Projekte zu autonomen Fahrzeugen wider.

Das folgende Kapitel 3 geht detailliert auf die in dieser Arbeit verwendeten Versuchsträger Munich Cognitive Autonomous Robot Car 3rd Generation (MuCAR-3) und MuCAR-4 ein. Dabei stellen die Abschnitte 3.1 bis 3.3 zunächst die im Fahrzeug verbaute Sensorik vor, bevor Abschnitt 3.4 einen Überblick über das verwendete Software-Framework zum Datenaustausch und zur Datensynchronisation gibt.

Da viele der in dieser Arbeit vorgestellten Verfahren auf modellbasierter Schätzung beruhen, gibt Kapitel 4 einen Überblick über die Bayes'sche Zustandsschätzung am Beispiel verschiedener Varianten des Kalmanfilter (KF) sowie des Partikelfilter (PF).

Anschließend folgt das Kapitel 5, welches sich dem Thema Sensorkalibrierung von Kamera, LiDAR und INS widmet. Neben entsprechenden Sensormodellen in den Abschnitten 5.1 und 5.2 befassen sich die weiteren Kapitel mit der Sensorsynchronisation und der extrinsischen Sensorkalibrierung. Herauszuheben sind dabei die Abschnitte 5.7 und 5.9, in denen zunächst ein Kalibrierungsverfahren vorgestellt wird, welches die speziellen Sensorcharakteristika von Kamera und LiDAR berücksichtigt, und im Weiteren ein auf Odometrie basierendes *online*-Kalibrierungsverfahren für Kamera und INS vorgestellt wird.

Aufbauend auf den Kalibrierungsergebnissen dieses Kapitels beschreibt Kapitel 6 einzelne Beispiele zur Fusion von Kamera, LiDAR und INS. Der Fokus liegt dabei in Abschnitt 6.1 auf der Rohdatenfusion, während Abschnitt 6.2 auf Merkmal-basierte Fusion eingeht.

Die Anwendung einer solchen Datenfusion wird durch Kapitel 7 am Beispiel einer autonomen Hindernisvermeidung dargestellt. Das Kapitel gibt außerdem einen Überblick über Belegungskarten, bevor die Arbeit mit einer Zusammenfassung und einem Ausblick in Kapitel 8 abgeschlossen wird.

An dieser Stelle sei außerdem auf den Anhang verwiesen, der eingehend die in dieser Arbeit verwendete Notation beschreibt, die relevanten Datenblätter der Sensoren enthält sowie Umrechnungsformeln zum Umgang mit homogenen Transformationsmatrizen und Posen bereitstellt.

2 Stand der Forschung

Wie eingangs erwähnt beschäftigt sich die Wissenschaft schon seit Jahrzehnten mit der Forschung im Bereich der Robotik. Eine vollständige Darstellung der Forschungserfolge aus dieser Zeit würde den Rahmen dieser Arbeit sprengen. Daher konzentriert sich der hier dargestellte Stand der Forschung auf die zu dieser Arbeit thematisch relevanten Forschungsthemen wie Sensorkalibrierung, Sensorfusion, 3D-Rekonstruktion und Mapping sowie autonome Fahrzeuge vor dem Hintergrund autonomer Navigation außerhalb von Gebäuden.

Da die Forschung an autonomen Fahrzeugen seit der bereits erwähnten Defense Advanced Research Projects Agency (DARPA) *Grand Challenge* im Jahr 2005 besondere Aufmerksamkeit erfuhr, konzentriert sich dieser Überblick überwiegend auf die seit 2005 entstandenen Forschungsarbeiten. Hauptsächlich wurde die Entwicklung an autonom fahrenden Fahrzeugen auf internationalen Konferenzen vorgestellt. Zu diesen Konferenzen zählen u.a. *Intelligent Vehicles Symposium (IV)*, *International Conference on Robotics and Automation (ICRA)* und *International Conference on Intelligent Robots and Systems (IROS)*. Dementsprechend entstammt auch ein Großteil der hier dargestellten Veröffentlichungen einer dieser Konferenzen. Weitere relevante Publikationen gehen auf die bekannten Schriftreihen, wie *Journal of Field Robotics* oder *Journal of Computer Vision*, zurück.

2.1 Sensorkalibrierung

Im Hinblick auf die extrinsische und intrinsische Kalibrierung von Stereokameras gehört die sog. *Matlab Calibration Toolbox* [Bouguet, 2003] zu den wohl bekanntesten Veröffentlichungen. Die Bereitstellung einer quelloffenen Software zur Kamerakalibrierung sorgte für eine weite Verbreitung und einigen Erweiterungen, wie der von Hartley und Kang [2007]. Das Verfahren basiert auf der Erkennung von Schachbrettmustern, deren Eckpunkte aufgrund der Linsenverzeichnung in den Kamerabildern nicht auf Geraden liegen. Durch manuelle oder automatisierte Erkennung dieser Eckpunkte

lässt sich in einem ersten Schritt ein Modell für die Linsenverzeichnung parametrieren, um anschließend mit Hilfe der bekannten Schachbrettgröße auf die Relativpose zweier Kameras zu schließen, die dasselbe Schachbrett zur gleichen Zeit beobachten. Strand und Hayman [2005] verfolgen das gleiche Ziel, allerdings mit einer anderen Optimierungsstrategie. Anstatt, wie bei Bouguet [2003], die Modellparameter derart zu verändern, dass gekrümmte Linien im Zielbild gerade werden, besteht das Ziel bei Strand und Hayman [2005] darin, die Parameter für Kreisbögen im verzerrten Eingangsbild zu bestimmen, was die iterative Suche der Modellparameter beschleunigt.

Dennoch ist diese iterative Suche zum einen rechenaufwändig und erfordert zum anderen eine bestehende Sequenz an Kalibrierungsbildern mit Kalibriermuster. Ändern sich Einbaulage oder Linsenparameter des Kamerasystems, ist eine Neukalibrierung notwendig. Dang [2009] beschreibt daher in seiner Dissertation und entsprechenden Vorarbeiten Dang et al. [2009] ein *online*-Verfahren zur Kalibrierung der extrinsischen Kalibrierungsparameter eines Stereokamerasystems. Das beschriebene Verfahren setzt voraus, dass beide Kameras zuvor intrinsisch, beispielsweise nach der Methode von Bouguet [2003], kalibriert wurden, und dass sich die Sichtbereiche beider Kameras überlappen, um Korrespondenzen zwischen beiden Bildern zu erhalten. Diesem Problem begegnen Pagel [2010] und Lébraly et al. [2011] durch die Verwendung eindeutig identifizierbarer Marker. Im Rahmen der Kalibrierungsequenz ist es nötig, mit jeder Kamera möglichst mehrere Marker zu sehen. Die Verfahren schätzen dann, neben der Lage der Marker im Raum zueinander, ebenfalls die Kalibrierungsparameter der Kameras. Für Fälle, in denen die Verwendung von Markern nicht gewünscht oder nicht möglich ist, beschreiben Knorr et al. [2013] ein Verfahren, welches es ermöglicht, mehrere Kameras extrinsisch zueinander zu kalibrieren, auch ohne dass sich die Sichtfelder der Kameras überlappen. Allerdings setzen Knorr et al. [2013] voraus, dass sich das Fahrzeug in einer Ebene bewegt und damit weder nickt noch rollt. Von Hong et al. [2015] stammt ein Verfahren, welches einerseits zwar überlappende Sichtbereiche der zu kalibrierenden Kameras erfordert, dabei jedoch auf einem analytischen statt iterativen Lösungsalgorithmus basiert.

Werden die Bilddaten mit den Daten anderer Sensoren fusioniert, ist es ebenfalls nötig, die Relativpose zwischen diesen Sensoren zu kennen. Eine Möglichkeit ist es dabei, die Einbaulage aller beteiligten Sensoren bzgl. eines festen Fahrzeugkoordinatensystems zu bestimmen. Miksch et al. [2010] beschreiben in ihrer Veröffentlichung eine solche Methode für ein monokulares Kamerasystem, bei dem Anforderungen an die Fahrzeugbewegung in Form einer exakten Geradeausfahrt auf einer ebenen Fahrbahn gestellt werden. Moderne autonome Fahrzeuge verfügen i.d.R. über einen Sensor

zur Ermittlung der Eigenbewegung, wie eine Inertial Measurement Unit (IMU) oder auch ein Inertial Navigation System (INS), mit dem die Bewegung des Fahrzeug bzgl. eines fahrzeugfesten Punktes angegeben wird. Daher existieren Verfahren, die mit Hilfe der Sensordaten und eben diesen Bewegungsdaten die Einbaulage des Sensors bzgl. des fahrzeugfesten Referenzpunktes bestimmen. Ein solches Verfahren ist das von Fleps et al. [2011], welches die Relativpose zwischen einer Kamera und einer IMU berechnet. Allerdings handelt es sich dabei um ein *offline*-Verfahren, d.h. es erfolgt keine automatische Nachkalibrierung. Mit *CamOdoCal* stellen Heng et al. [2013] ein weiteres *offline*-Verfahren vor, welches im Gegensatz zu Fleps et al. [2011] mehrere Kameras bzgl. des Eigenbewegungssensors kalibriert. Dabei liefert *CamOdoCal* neben der Einbaulage auch die intrinsische Kalibrierung der Kameras. Eine *online*-Lösung zur Kalibrierung einer Kamera zu einem Eigenbewegungssensor wird von Guo et al. [2012] vorgestellt. Bei dieser Lösung muss die Höhe der Kamera bzgl. des Referenzpunktes angegeben werden und kann nicht automatisch ermittelt werden. Tang und Liu [2018] beschreiben eine iterative Lösung zur extrinsichen Kamerakalibrierung, die zusätzlich auch eine Lösung liefert, um eine initiale Schätzung des Zustandsvektors zu finden. Dazu muss sich der Roboter jedoch ausschließlich auf einer ebenen Fläche bewegen.

Wie bereits erwähnt kommen in autonomen Fahrzeugen zusätzlich zu Kameras häufig auch LiDAR-Sensoren (Light Detection and Ranging) zum Einsatz. Für die *VIAC Challenge* wurde von Mazzei et al. [2012] ein Kalibrierkörper-basiertes *offline*-Kalibrierungsverfahren für Kameras und LiDAR entworfen, in dem beide Sensoren bzgl. des Fahrzeugreferenzpunktes kalibriert werden. Ebenfalls auf einem Schachbrettkalibrierkörper basierend stammt von Geiger et al. [2012] ein Kalibrierungsverfahren für LiDAR und Kamera, welches nur eine einzelne Aufnahme eines überlappenden Sichtbereichs beider Sensoren benötigt. Der Vorteil liegt in dem einfachen Messaufbau. Als nachteilig ist anzusehen, dass das Verfahren aufgrund der hohen Rechenlast nicht echtzeitfähig ist. Kämpchen [2007] stellt in seiner Dissertation ebenfalls ein Kalibrierkörper-basiertes Verfahren zur LiDAR-Kamerakalibrierung vor. Auch die Algorithmen von Huang und Barth [2009], Kwak et al. [2011] beschäftigen sich mit dem gleichen Thema, liefern ihre Ergebnisse allerdings ebenfalls erst im Nachhinein. In den letzten Jahren wurden drei weitere Verfahren [Taylor et al., 2013, Bok et al., 2014, Guindel et al., 2017] zur extrinsischen LiDAR-Kamera Kalibrierung vorgestellt. Bei diesen handelt es sich ebenfalls um *offline*-Verfahren, wobei zumindest das Verfahren von Bok et al. [2014] keine überlappenden Sichtbereiche der Sensoren erfordert. In dieser Arbeit findet jeweils nur ein LiDAR-Sensor Verwendung. Dennoch existieren auch Verfahren zur LiDAR-LiDAR Kalibrierung. Stellvertretend sei hier die Veröffentlichung von Maddern et al. [2012] genannt.

Insgesamt zeigt die Literatur im Bereich der Sensorkalibrierung, dass die intrinsische Kalibrierung i.d.R. als statisch angesehen wird, was eine einmalige *offline*-Kalibrierung rechtfertigt. In dieser Arbeit wird daher die gleiche Strategie verfolgt. Darüber hinaus sind Kalibrierungsverfahren meist auf eine bestimmte Sensorkonfiguration (z.B. Kamera-Eigenbewegungssensor, LiDAR-Kamera) zugeschnitten. Generische Kalibrierungsverfahren sind deutlich seltener und stellen im Allgemeinen eher Anforderungen an die Umgebung oder die Bewegung des Fahrzeugs während der Kalibrierung.

2.2 Sensorfusion

Ein interessantes System zur Sensorfusion stammt von Oskiper et al. [2007], bei dem zwei Stereokamerasysteme, mit Blick nach vorne und hinten, mit einer IMU gekoppelt werden. Das System wird von einem Menschen als eine Art Rucksack getragen und dient als Sensoriksystem für u.a. Simultaneous Localization and Mapping (SLAM) innerhalb von Gebäuden. Außerhalb von Gebäuden, speziell im automotiven Umfeld, werden Kamerasysteme häufig mit RaDAR-Sensoren (Radio Detection and Ranging) fusioniert. Dabei sind u.a. die Publikationen von Liu et al. [2008] und Richter et al. [2008] zu nennen. In beiden werden Objekte sowohl in RaDAR-Messungen als auch in Kamerabildern zunächst erkannt und anschließend mit einem Kalmanfilter (KF) über der Zeit verfolgt. Von Wu et al. [2009] stammt eine Veröffentlichung, in der neben einem RaDAR-System auch ein Stereokamerasystem zum Einsatz kommt. Letztlich findet auch hier ein zeitliches Verfolgen der erkannten Objekte mit Hilfe eines KF statt. Manz [2013] fusioniert Kamera, INS und LiDAR sowohl zum zeitlichen Verfolgen der Fahrbahn als auch zum modellbasierten zeitlichen Verfolgen eines vorausfahrenden Fahrzeugs. Fries [2019] erweitert diesen Ansatz auf weitere Fahrzeuge sowie um Modellmerkmale einer Wärmebildkamera.

Während in den bereits genannten Veröffentlichungen das Erkennen und Verfolgen anderer Verkehrsteilnehmer im Vordergrund steht, geht es in der Publikation von Milella et al. [2011] um Freiraumanalyse bzw. Segmentierung der Fläche vor dem Fahrzeug. Ausgehend von einem bereits kalibrierten RaDAR-Kamerasystem werden die RaDAR-Daten dazu verwendet, die befahrbaren Bereiche im Bild zu markieren. Mit Hilfe dieser Information trainieren die Autoren unüberwacht einen Klassifikator für Bildbereiche, der in der Lage ist, zwischen Boden und Nicht-Boden in den Kamerabildern zu unterscheiden. Allerdings ist Sensorfusion keinesfalls auf nur zwei Arten von Sensorik beschränkt. Suzuki et al. [2010] fusionieren eine Kamera, einen RaDAR- und einen Eigenbewegungssensor zur Kollisionswarnung vor Fußgängern.

Als Seriensensorik in der Automobilindustrie bislang noch weniger verbreitet sind LiDAR-Sensoren. Die wissenschaftliche Relevanz zeigt sich u.a. auch in der Veröffentlichung von Münz et al. [2010]. Sie ist insofern besonders, als dass sie einen generischen Ansatz zur Sensorfusion verfolgt. Die Auswertung basiert auf realen Daten einer Kamera und eines LiDAR.

Die Literaturanalyse zum Thema Sensorfusion zeigt vielfältige Anwendungen für Sensorfusion. Interessanterweise vertieft kaum eine Publikation in diesem Zusammenhang das Thema Sensorkalibrierung. Die Art und Weise der Kalibrierung wird i.d.R. nicht erwähnt oder als statisch und bereits bekannt vorausgesetzt. Während sich der überwiegende Teil der Literatur speziellen Sensoren und Sensoraufbauten widmet, zeigen nur wenige Veröffentlichungen, wie die von Münz et al. [2010], Möglichkeiten zur generischen Kalibrierung von Multi-Sensorsystemen auf. Dementsprechend betonen auch nur Münz et al. [2010] die Notwendigkeit, bei der Sensorfusion die speziellen Sensorcharakteristika zu berücksichtigen.

Darüber hinaus haben sich in den letzten Jahren insbesondere Belegungskarten als eine Methode der Umgebungsrepräsentation stark verbreitet. Belegungskarten ermöglichen ebenfalls die Fusion von Daten aus unterschiedlichen Quellen bzw. Sensoren. Da sie thematisch näher an der 3D-Rekonstruktion als an der Sensorfusion liegen, wird auf entsprechende Veröffentlichung im nächsten Abschnitt verwiesen.

2.3 3D-Rekonstruktion und Mapping

Das Rekonstruieren der 3D-Umgebung ist eine seit langem verfolgte Aufgabe insbesondere im Bereich des Computersehens. Dementsprechend gibt es einige Veröffentlichungen zu diesem Thema auf einschlägigen Konferenzen, wie die von Sameer et al. [2009], in der ausgehend von einzelnen Bildern ein punktbasiertes 3D-Modell einer ganzen Stadt berechnet wird. Basierend auf derartigen Ansätzen ist es möglich, statische Umgebungskarten für autonome Fahrzeuge zu berechnen. Aufgrund der hohen Rechenanforderungen eignen sie sich nicht für die kontinuierliche Freiraumberechnung um ein autonom fahrendes Fahrzeug herum. Doch auch dies ist mit reinem Computersehen möglich: Das Verfahren von Martinec und Pajdla [2007] liefert neben einer Umgebungsrepräsentation auch die Bewegung des Roboters nur allein aus Kameradaten, sog. Visuelle Odometrie. Müssen feinere Strukturen aufgelöst werden, eignet sich die Publikation von Guo et al. [2009] zur Erkennung von Fahrbahnrändern aus Stereobildern.

Den Weg von der Straße in unwegsames Gelände unterstützen Chilian und Hirschmüller [2009] durch Umwandeln von Disparitätenbildern in 3D-Punktwolken. Ausgehend von einer solchen Punktwolke erstellen die Autoren eine 2D Belegungskarte. In jeder Zelle wird dabei die mittlere Höhe aller in diese Zelle fallenden 3D-Punkte gespeichert. Das Verfahren ist durch die Mittelwertberechnung anfällig für Messrauschen und modelliert keinerlei Sensorungenauigkeiten. Lategahn et al. [2010] beschreiben ebenfalls eine auf 3D-Punkten einer Stereokamera basierende Belegungskarte, die auf den automotiven Einsatz spezialisiert ist. Das Verfahren von Vatavu et al. [2012] beruht ebenfalls auf 3D-Punkten einer Stereokamera. Durch die Einbeziehung der Eigenbewegung und damit berechneter Differenz lokaler Belegungskarten ermöglichen sie zusätzlich eine Klassifikation von Hindernissen in bewegt und nicht bewegt.

Bezüglich Modellierung der Erfassungssensorik geht Andert [2009] einen Schritt weiter, indem er den virtuellen Sehstrahlen zu jedem 3D-Punkt ein inverses Sensormodell zu Grunde legt, um auf diese Weise die Ungenauigkeiten der Sensorik bzw. der Korrespondenzbestimmung zu berücksichtigen. Die Ungenauigkeiten treten dabei im Disparitätenraum auf und nicht in 3D. Außerdem wird jede von einem virtuellen Sehstrahl durchstreifte Zelle der Belegungskarte bis zu dem 3D-Punkt als hindernisfrei angesehen, was abhängig von der Einbauhöhe des Stereokamerasystems nur dann möglich ist, wenn das Kamerasystem nicht über etwaige Hindernisse hinwegblickt. Kim und Kim [2012] modellieren Sensor- bzw. Korrespondenzungenauigkeiten durch eine Mischung Gauß'scher Prozesse. Brandao et al. [2013] stellen ebenfalls eine Berechnung von Belegungskarten mit Hilfe inverser Sensormodelle in 3D vor. Im Gegensatz dazu modellieren Perrollaz et al. [2010] den Fehler korrekt im Disparitätenraum. Coue et al. [2006] erweitern den Belegungskartenansatz. Indem die Zellen der Belegungskarte einen Bewegungsvektor erhalten, ist es ihnen möglich, dynamische Objekte mit in die Belegungskarten zu integrieren.

Allerdings eignen sich Belegungskarten nicht nur für bildbasierte Ansätze. Wie bereits erwähnt, finden sie im Bereich der Sensorfusion ebenfalls vermehrt Anwendung. Homm et al. [2010] stellen ein Verfahren zur Berechnung von Belegungskarten vor, welches die Informationen von LiDAR und RaDAR vereint und dabei zusätzlich durch die Berechnung auf einer Grafikkarte beschleunigt wird. Schueler et al. [2012] kombinieren die Messungen von RaDAR und LiDAR mit einer Objektverfolgung zur zusätzlichen Berücksichtigung dynamischer Objekte. Für multi-modale Sensoreingänge entwickelt Tanzmeister [2016] Belegungskarten, die – basierend auf der Dempster-Shafer Theorie – Evidenzen zur Unterscheidung von statischer und dynamischer Umgebung liefern. In Kombination mit einem Partikelfilter (PF) zeigt Steyer

[2021], wie eine Objektverfolgung auf Basis dynamischer Evidenzen realisiert werden kann.

Natürlich basiert die Rekonstruktion der Umgebung nicht immer auf Belegungskarten. Anwendungen, wie die von Sagawa et al. [2005], basieren auf 3D-Punktwolken, die durch Aufnahmen aus mehreren Perspektiven mit Hilfe des ICP-Algorithmus (Iterative Closest Point) verfeinert werden. Andere Ansätze, wie die von Cappalunga et al. [2010] und Motooka et al. [2014], basieren auf Gitterstrukturen, deren Höhenverlauf durch Optimierungsverfahren bestimmt wird. Belegungskarten, die für jede Zelle die Höhe oder Hinderniswahrscheinlichkeit darstellen, liefern in der Regel pro Zelle nur einen einzelnen Wert, basierend auf allen Messungen, die innerhalb dieser Zelle liegen. Überhängende Strukturen, wie Äste oder Brücken, müssen daher gesondert behandelt werden. Eine Möglichkeit dabei ist es, neben der Länge und Breite, auch die Höhe in diskrete Zellen zu unterteilen, wodurch die Zellen zu Würfeln werden. Shade und Newman [2011] beschreiben ein solches Verfahren, in dem die Würfel mit Hilfe eines Octree effizient verwaltet werden. Neben den genannten Verfahren gibt es ebenfalls Rekonstruktionsalgorithmen, die sich auf Basis ihrer eigenen Berechnungen aus vorangegangenen Zeitschritten selbst plausibilisieren. Dazu gehören u.a. die beiden Stereoverfahren von Geiger et al. [2011] und Milella et al. [2011].

Mit der in den letzten Jahren gestiegenen Verfügbarkeit von Rechenressourcen und Programmierbibliotheken sowie den Entwicklungen im Bereich künstlicher neuronaler Netze finden zunehmend auch sog. tiefe neuronale Netze Anwendung im Bereich des autonomen Fahrens. So zeigen Garnett et al. [2017] einen Ansatz zur Erkennung und zeitlichen Verfolgung von unklassifizierten Objekten, der auf Kamera- und LiDAR-Daten basiert. Peng et al. [2020] geben einen Überblick über verschiedene neuronale Netztopologien und Trainingsstrategien, die in der quelloffenen Apollo software [Baidu, 2017] zum Einsatz kommen.

Zum Forschungsgebiet der 3D-Rekonstruktion gehören selbstverständlich auch sog. SLAM-Verfahren. Da sich diese Arbeit dem Thema Kartenerstellung nur in Bezug auf lokale Karten der Fahrzeugumgebung widmet, verzichtet dieser Literaturüberblick auf eine eingehende Beschreibung des Standes der Forschung im Bereich SLAM.

Unter Berücksichtigung der ansonsten dargestellten Forschungsergebnisse fällt auf, dass nur in wenigen Fällen die Sensorcharakteristika direkt berücksichtigt werden. Sofern den daraus resultierenden Ungenauigkeiten, wie im Fall von Belegungskarten, durch eine geeignete Wahl der Zellgrößen begegnet werden kann, ist die Vereinfachung plausibilisierbar. Insbesondere im Zusammenhang mit Stereoverfahren wirken

sich Messfehler und Korrespondenzfehler aufgrund der projektiven Abbildung mit zunehmender Entfernung vom Sensor auch entsprechend stärker aus, wodurch eine konstante Zellgröße entweder im Nahbereich die Genauigkeit unterschätzt oder im Fernbereich überschätzt. Darüber hinaus fällt – wie schon beim Themengebiet Sensorfusion – auf, dass die Literatur i.d.R. kaum Hinweise auf die Art und Weise der Sensorkalibrierung liefert.

2.4 Autonome Fahrzeuge

Wie eingangs bereits erwähnt führten sowohl die DARPA *Grand Challenge* im Jahr 2005 als auch die im Jahr 2007 folgende DARPA *Urban Challenge* zu einem starken Anstieg an autonome Forschungsfahrzeugen. Neben dem Siegerfahrzeug des Jahres 2005, *Stanley* [Thrun et al., 2006], das sowohl mit Kamera, LiDAR- und RaDAR-Sensoren ausgerüstet ist, waren auch die beiden RaDAR- und LiDAR-basierten Fahrzeuge *Sandstorm* und *H1ghlander* [Whittaker et al., 2006] äußerst erfolgreich. Das Fahrzeug *TerraMax* [Broggi et al., 2006] nahm ebenfalls an der *Grand Challenge* teil und zeichnet sich durch ein System zur Sensorfusion aus, welches den Nickwinkel zwischen Kamera und LiDAR selbständig nachkalibrieren kann. Mit einem ähnlichen Fahrzeug nahm das Team zwei Jahre später erneut teil und fusionierte dabei 11 Kameras sowie 4 LiDAR [Broggi et al., 2010]. Dennoch konnte es sich nicht gegen das Siegerfahrzeug *BOSS* [Urmson et al., 2008] der Carnegie Mellon Universität oder auch *Junior* [Montemerlo et al., 2008] behaupten. Zur *Urban Challenge* traten außerdem die überwiegend deutschen Teams *AnnyWAY* [Kammel et al., 2008], *CarOLO* [Basarke et al., 2008] und *Spirit of Berlin* [Rojo et al., 2007] an.

Neben den durch die DARPA organisierten Wettkämpfen haben außerdem Leibe et al. [2007] ein Fahrzeug vorgestellt, welches mit Hilfe von Stereokameras bestimmte Aufgaben, wie Structure from Motion (SfM), Erkennung und Tracking von Fahrzeugen und Personen in 3D sowie 3D-Lokalisierung, erledigt. Das Konzeptfahrzeug *CARAI* [Wanielik et al., 2008] der Technischen Universität Chemnitz fusioniert Kamera und RaDAR-Sensoren zur Umfelderkennung. Ein weiteres Versuchsfahrzeug stammt von Munz et al. [2009] und untersucht u.a. die Fusion von Kamera und LiDAR zur Verfolgung anderer Fahrzeuge. Schließlich haben Wille et al. [2010] mit dem Projekt Stadtpilot erfolgreich einen Versuchsträger aufgebaut, der in der Lage ist, mit Hilfe von Kamera und LiDAR den Braunschweiger Stadtring autonom zu befahren.

Während die Entwicklung in akademischen Einrichtungen wie bspw. mit Broggi et al. [2015] und Maddern et al. [2017] weiterging, entwickelten zunehmend auch Firmen mit kommerziellen Absichten autonome Fahrzeuge [Daily et al., 2017]. Beiträge dieser Firmen finden sich jedoch überwiegend in Patenten der jeweiligen Firmen. So patentierte bspw. Waymo ein autonomes Fahrzeug mit mehreren LiDAR-Sensoren [Pennecot et al., 2015]. BMW patentierte ein System zur Erkennung des befahrbaren Raums [Frickenstein et al., 2020], während Baidu ein System zum Verfolgen von Hindernissen vorstellte [Li et al., 2020].

3 Versuchsträger, Sensorik und Middleware

Die in dieser Arbeit entwickelten Verfahren finden ihre Verwendung vor allem auf den beiden institutseigenen Fahrzeugen Munich Cognitive Autonomous Robot Car 3rd Generation (MuCAR-3) und Munich Cognitive Autonomous Robot Car 4th Generation (MuCAR-4). Daher werden im Folgenden beide Fahrzeuge in Hinblick auf die verbaute Sensorik, Hardware und Middleware beschrieben.

(a) MuCAR-3 **(b)** MuCAR-4

Abbildung 3.1: Versuchsträger MuCAR-3 und MuCAR-4

Bei MuCAR-3 (Abbildung 3.1a) handelt es sich um einen Volkswagen Touareg, der am Institut für Technik Autonomer Systeme (TAS) seit dem Jahr 2006 aufgebaut und kontinuierlich weiterentwickelt wird. In der dritten Generation folgt es dem Versuchsfahrzeug für autonome Mobilität und Rechnersehen (VaMoRs) und dem VaMoRs-Pkw (VaMP), die Prof. Ernst D. Dickmanns mit seinem Team zuvor am Institut für Systemdynamik und Flugmechanik (ISF) entwickelt hat.

MuCAR-3 hat von seinen Vorgängern einige Konzepte übernommen. Darunter fällt z.B. die rotierbare Kameraplattform hinter der Windschutzscheibe. Als geländegängiges Fahrzeug mit vollständiger DBW-Fähigkeit (Drive by Wire), höhenverstellbarer Luftfederung und Unterbodenschutz eignet es sich im Gegensatz zu seinen Vorgängern auch zur autonomen Navigation abseits befestigter Straßen.

MuCAR-4 (Abbildung 3.1b) basiert auf einem Volkswagen Tiguan und wurde bei TAS ab dem Jahr 2010 aufgebaut und ebenfalls weiterentwickelt. Die Unterschiede

zwischen MuCAR-4 und MuCAR-3 finden sich hauptsächlich in der Aktorik. Da die Ansteuerung beider Fahrzeuge softwareseitig weitestgehend abstrahiert wurde, ist die Aktorik für diese Arbeit nur von untergeordneter Bedeutung.

Die folgenden Abschnitte beschreiben Sensorik, Koordinatensysteme und Middleware anhand von MuCAR-3 und gelten i.d.R. auch für MuCAR-4. Sofern Unterschiede auftreten, wird gesondert darauf hingewiesen.

3.1 Sensorik

Um die Aufgabe der autonomen Navigation zu bewältigen, verfügt MuCAR-3 über eine Reihe an Sensoren, die sowohl zum Erfassen des eigenen Zustands als auch der Umgebung geeignet sind.

3.1.1 Seriensensorik

Besondere Bedeutung kommt dabei der CAN-Schnittstelle (Controller Area Network) zu, die Zugriff auf die Daten der Seriensensorik des Fahrzeugs ermöglicht. Dazu gehört vor allem die Geschwindigkeit des Fahrzeugs, die durch Auswertung der Drehzahlen an allen vier Rädern ermittelt wird. MuCAR-3 verfügt außerdem über ein Electronic Stability Control (ESP). Daher liegen auf dem CAN-Bus auch der Lenkwinkel sowie die Beschleunigungen und Drehraten des Fahrzeugs entlang aller drei Raumachsen bereit. Darüber hinaus kann die Einfederung aller vier Federbeine ausgelesen werden.

3.1.2 Kameraplattform MarVEye-8

Zur optischen Wahrnehmung der Umgebung kommt in MuCAR-3 die multifokale Kameraplattform MarVEye-8 (Abbildung 3.2) zum Einsatz. Wie einleitend beschrieben geht die Entwicklung dieser Plattform auf ihre Vorgänger aus den Fahrzeugen VaMoRs und VaMP zurück. Die aktuelle Version ist eine Erweiterung des Vorgängers [Unterholzner, 2006, 2007] und ist in Unterholzner et al. [2010], Unterholzner und Wuensche [2010] genauer beschrieben. MarVEye-8 besteht aus den drei folgenden Hauptkomponenten:

Befestigungsbasis Damit wird die Kameraplattform an der Windschutzscheibe im Bereich des Rückspiegels befestigt.

Hohlwellenmotor Der Hohlwellenmotor ermöglicht das Drehen der montierten Kameras um die Gierachse.

Aufnahmeplatte Die Aufnahmeplatte verfügt beidseitig über Befestigungsmöglichkeiten für Adapterplatten.

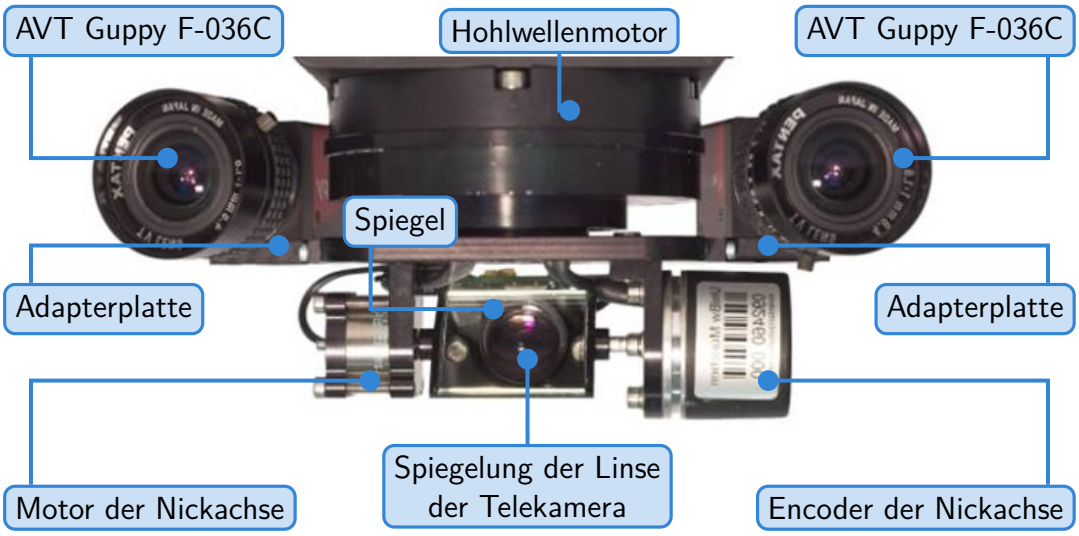


Abbildung 3.2:
Kameraplattform MarVEye-8. Hier dargestellt mit zwei nach vorne blickenden Allied Vision Technologies GmbH (AVT) Guppy F-036C Kameras und einer AVT Guppy F-036C Telekamera, die vertikal durch die Hohlwelle über einen Spiegel blickwinkelstabilisiert nach vorne blickt.

Passend zur Aufnahmeplatte, die seitlich über Passstifte verfügt, existieren eine Reihe von Adapterplatten, mit denen Kameras an der Aufnahmeplatte angebracht werden. Die Passstifte erlauben, dass Adapterplatten mit ihren Kameras demontiert und anschließend mit hoher Genauigkeit wieder montiert werden können. Dies macht eine erneute extrinsische Kalibrierung der Kameras nach ihrem Wiedereinbau unnötig. Die beschriebenen Adapterplatten existieren für die folgenden Kameratypen: Automotive Kamera des Herstellers Aglaia, AVT Guppy F-036C, Basler Ace acA1300-30gc und Bumblebee XB3. Während auf MuCAR-3 früher Aglaia Kameras eingesetzt wurden, sind derzeit ausschließlich Kameras des Typs AVT Guppy F-036C verbaut. Auf MuCAR-4 sind hingegen Basler Ace acA1300-30gc Kameras sowie eine Bumblebee XB3 verbaut. Detaillierte Angaben zu den Kameras sind im Anhang B.1 zu finden.

Generell lässt sich sagen, dass die Entwicklung der Kameras in den letzten Jahren besonders leistungsfähige Kameras, Sensorchips sowie Objektive hervorgebracht hat. Entsprechend vielfältig ist nicht nur die Auswahl an Komponenten, sondern auch deren Spezialisierung auf bestimmte Anwendungsbereiche. Für den automotiven Bereich sind Kameras notwendig, die mit kurzen Belichtungszeiten – zur Verringerung von Bewegungsunschärfe – sowie einem hohen Dynamikumfang – zur Robustheit gegenüber Blendungseffekten – ausgestattet sind. Bei den beschriebenen Kameras handelt es sich daher um sog. HDRC-Kameras (High Dynamic Range CMOS). Im Gegensatz zu traditionellen CCD-Kameras (Charge Coupled Device) verfügen die Pixel einer solchen Kamera über ein nahezu logarithmisches Verhältnis zwischen einfallender Lichtintensität und Ausgangsspannung, wodurch insbesondere im Dunkeln noch hohe Kontraste erzielt werden. Außerdem liest eine HDRC-Kamera jedes Pixel individuell aus, was die von CCD-Kameras bekannten *Blooming*- oder *Smear*–Effekte reduziert. Der Nachteil der HDRC-Kameras gegenüber CCD-Kameras liegt in der typischerweise schlechteren Farbwiedergabe. Bei Farb-HDRC-Kameras kommt i.d.R. ein sog. *Bayer pattern* zum Einsatz. Dabei werden rote, grüne und blaue Farbfilter nach einem vorgegebenen Muster über die einzelnen Bildpixel gelegt. Da gleichzeitig die Auflösung des Chips nicht erhöht wird, muss der RGB-Farbton (Rot-Grün-Blau) eines Pixels aus den umliegenden farbgefilterten Pixeln interpoliert werden. Dies führt zu Unschärfe und damit zu einer weiteren Verschlechterung der Farbwiedergabe.

Alle genannten Kameras verfügen über einen externen Eingang zum Auslösen. Dieser ermöglicht, wie in Abschnitt 5.3.1 gezeigt, nicht nur die Synchronisation mehrerer Kameras, sondern auch die Synchronisation mehrerer Kameras mit einem Laserscanner, die in Abschnitt 5.3.2 beschrieben ist. Der Hohlwellenmotor lässt sich um $\pm 50° \approx \pm 0{,}87\,\text{rad}$ um die Vertikalachse gieren. Dadurch vergrößert sich der von den Kameras beobachtbare Bereich vor dem Fahrzeug. Eine Kamera, deren Objektiv einen horizontalen Öffnungswinkel von 60° aufweist, kann durch Drehung der Plattform auf diese Weise einen horizontalen Bereich von 160° beobachten. Derartige Sichtfelder lassen sich ohne Kameraplattform nur mit mehreren Kameras oder durch die Verwendung eines Fischaugenobjektivs erfassen.

Mehrere Kameras verursachen allerdings auch höhere Kosten, einen höheren Kalibrieraufwand und größere Datenmengen, die verarbeitet oder gespeichert werden müssen. Die Verwendung eines Fischaugenobjektivs bringt einerseits einen vergleichbaren Sichtbereich, andererseits sind die Aufnahmen stark verzerrt und statt der genannten 60° wird der gesamte Sichtbereich von 160° auf den Chip der Kamera abgebildet. Bei gleichem Chip reduziert sich damit die laterale Auflösung des Linsen-Kamera-Systems entsprechend. Die Veröffentlichung von Manz et al. [2011b] zeigt außerdem, dass im

Fall des optischen Verfolgens vorausfahrender Fahrzeuge eine hohe Auflösung die Genauigkeit erwartungsgemäß erhöht. Das genannte Verfahren verwendet daher sogar ein Linsen-Kamera-System im Telebereich mit einem horizontalen Öffnungswinkel von lediglich 7° bei einer Auflösung des Kamerachips von 752 pix × 480 pix.

Die Verwendung von Teleobjektiven kann zu Verwacklungen im Kamerabild führen, wenn das Fahrzeug durch Schlaglöcher oder unwegsames Gelände fährt. Darum wurde an MarVEye-8 eine zusätzliche AVT Guppy F-036C Kamera angebracht, die vertikal durch die Hohlachse des Motors über einen Spiegel nach vorne blickt. Dieser Spiegel ist drehend gelagert und wird derart geregelt, dass das Bild der Telekamera weitestgehend von den Vibrationen und Vertikalbewegungen des Fahrzeugs entkoppelt wird [Unterholzner, 2007]. Insgesamt bietet die Plattform damit Platz für 3 bis 5 Monokameras plus Bumblebee XB3 und liefert somit in Summe bis zu 8 Kamerabilder.

3.1.3 Velodyne HDL-64E LiDAR

Der Velodyne HDL-64E ist ein Laserscanner, der seit dem Jahr 2007 vertrieben wird. Die erste Variante S1 wurde zum ersten Mal auf der durch die Defense Advanced Research Projects Agency (DARPA) organisierten Urban Challenge 2007 eingesetzt. Die Varianten S2 und S3 sind Weiterentwicklungen. Diese Arbeit macht jedoch nur Gebrauch von den Varianten S1 und S2. Vom Grundprinzip her arbeiten alle Velodyne HDL-64E identisch. Im Vergleich zu seinem Vorgänger zeichnet sich der Velodyne HDL-64E S2 vor allem durch eine höhere laterale Auflösung der unteren 32 Laser aus. Der Unterschied ist später in Abb. 6.2 zu sehen.

Wie in Abb. 3.3a gezeigt besteht der Sensor aus einer festen Basis und einem rotierenden Oberteil. Die Basis dient der Befestigung des Sensors auf dem Versuchsträger. An ihr befindet sich außerdem der Anschluss zur Stromversorgung und eine LAN-Schnittstelle (Local Area Network). Im rotierenden oberen Teil des Sensors tasten 64 Infrarotlaser in gleichmäßigen Pulsen die Umgebung nach dem TOF-Prinzip (Time of Flight) ab und generieren nach diesem Muster pro Sekunde je nach Modell etwa zwischen 1000000 und 1300000 Messungen. Diese Messungen werden blockweise per User Datagram Protocol (UDP) über die LAN-Schnittstelle an den Fahrzeugrechner übertragen. Dieser errechnet daraus eine, wie in Abb. 3.3b dargestellte, 3D-Punktwolke. Detaillierte Daten über beide Laserscanner sind im Anhang B.2 zu finden.

(a) Velodyne HDL-64E LiDAR

(b)
Punktwolke eines Velodyne HDL-64E S2

Abbildung 3.3: Velodyne HDL-64E LiDAR und erzeugte 3D-Punktwolke

3.2 Inertial Navigation System (INS)

Um die eigene Bewegung des Fahrzeugs genauer als mit der Seriensensorik allein erfassen zu können, verfügt MuCAR-3 über ein Inertial Navigation System (INS) vom Typ OxTS RT3003 [Oxford Technical Solutions Ltd., 2011]. Die darin verbaute Inertial Measurement Unit (IMU) besteht aus je drei MEMS-Drehraten- und Beschleunigungssensoren (Micro Electro Mechanical System) und ermöglicht damit die Bestimmung der Fahrzeugbewegung in allen sechs Freiheitsgraden (DOF). Die IMU ist zusätzlich mit einem DGPS (Differential Global Positioning System) und einem Raddrehzahlsensor gekoppelt.

Dies ermöglicht nicht nur die Bestimmung der globalen Fahrzeugposition, sondern auch die Kompensation auftretender Fehler. Beispielsweise wirken eine kontinuierliche DGPS-Positionsbestimmung und Messung der Raddrehzahl dem Driften der MEMS-Kreisel entgegen. Außerdem kompensiert die IMU zusätzlich Sprünge in der DGPS-Position, die beispielsweise aufgrund von plötzlicher Abschattung oder Mehrfachreflektion des GPS-Signals (Global Positioning System) durch Vegetation oder in Häuserschluchten entstehen. Überlagert stützt zusätzlich ein kinematisches Bewegungsmodell des Fahrzeugs [Luettel et al., 2009] die INS-Messungen. Die wesentlichen Daten des Oxford Technical Solutions (OxTS) RT3003 INS sind im Anhang B.3 aufgeführt.

3.3 dSPACE MicroAutoBox

(a)
dSPACE MicroAutoBox im Tischaufbau.

(b)
AVT Guppy F-036C mit FireWire- und Auslöse-Steckverbindung.

Abbildung 3.4: dSPACE MicroAutoBox und Kamera mit Steckverbindungen.

Wie in Abschnitt 3.1.2 beschrieben, kann das Auslösen der verwendeten Kameras extern gesteuert werden. Dazu wurden die entsprechenden Anschlüsse der Kameras (Abbildung 3.4b) mit einer im Fahrzeug verbauen dSPACE MicroAutoBox verbunden. Neben dem zeitgenauen Auslösen der angeschlossenen Kameras erlaubt diese Verbindung auch das Messen der Belichtungszeit einer jeden angeschlossenen Kamera, indem die Kamera während der Belichtung einen digitalen Eingang der dSPACE MicroAutoBox auf logisch *high* setzt. Bei der dSPACE MicroAutoBox handelt es sich um ein Echtzeitsystem, so dass die Belichtungszeiten mit hoher Genauigkeit gemessen werden können.

Die Belichtungszeit wird durch die verwendeten Kameras, d.h. deren Firmware selbst bestimmt. Diese versuchen nach Möglichkeit, die Belichtungszeit derart zu wählen, dass der Dynamikbereich des gesamten Bildes oder einer Region of Interest (ROI) voll ausgeschöpft wird. Dadurch sollen die Bilder weder über- noch unterbelichtet werden und einen möglichst hohen Kontrast aufweisen. Die Belichtungszeit hängt daher stark von der Umgebungshelligkeit ab, ändert sich von Bild zu Bild i.d.R. allerdings nur geringfügig.

Darüber hinaus übernimmt die dSPACE MicroAutoBox die Regelung der Kameraplattform MarVEye-8 [Unterholzner, 2006, 2007, Unterholzner et al., 2010, Unterholzner und Wuensche, 2010]. Kombiniert mit Wahrnehmungsalgorithmen lässt sich damit eine Aufmerksamkeitssteuerung umsetzen [Unterholzner und Wuensche, 2013].

3.4 Middleware und Framework

Die Verarbeitung und Interpretation der Sensordaten findet auf einem im Versuchsträger mitgeführten Rechner statt. Dabei handelt es sich bei beiden Versuchsträgern um einen Rechner mit einer Intel Xeon L5640 Dual Hexacore Central Processing Unit (CPU) und 12 GB Arbeitsspeicher. Die Rechner sind jeweils mit einer Reihe von externen Schnittstellen ausgestattet, die einerseits dem Einzug der Sensordaten, aber auch der Kommunikation mit dem Fahrzeug oder anderen Fahrzeugen dienen. Abbildung 3.5 gibt einen Überblick über die verwendeten Schnittstellen.

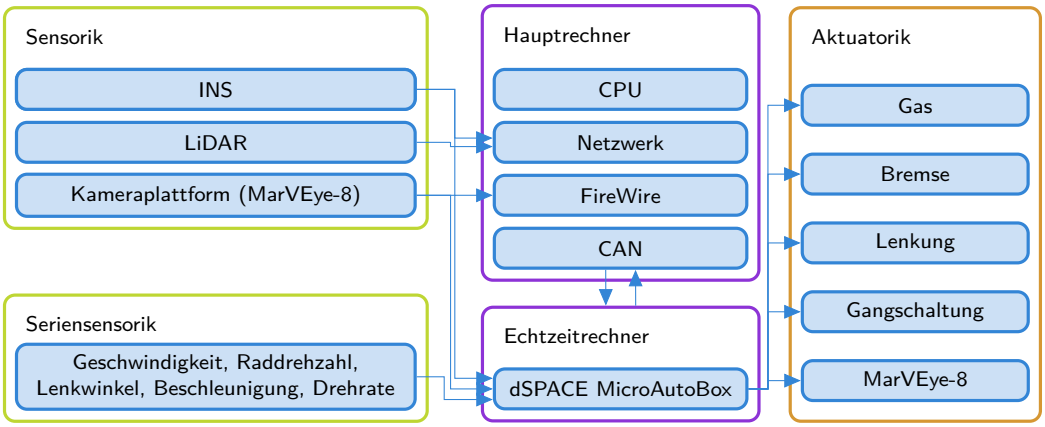


Abbildung 3.5: Verwendete Hardware und Schnittstellen auf MuCAR-3 bzw. MuCAR-4

Auf beiden Versuchsträgern befindet sich das Betriebssystem Arch Linux auf einer Solid State Drive (SSD). Da SSD-Festplatten über keine beweglichen Teile verfügen, sind sie robust gegenüber Vibrationen und Stößen durch das Fahrzeug. Außerdem liegen die Lese- und Schreibgeschwindigkeiten mit etwa 500 MB/s etwa um den Faktor 5 höher als bei herkömmlichen magnetischen Festplatten. Wie im nächsten Abschnitt gezeigt wird, sind zum Aufzeichnen der Sensordaten derartige Geschwindigkeiten notwendig.

3.4.1 Echtzeitdatenbank für kognitive Automobile (KogMo-RTDB)

Der Einzug und die Verarbeitung der Sensordaten sind bei TAS modular aufgebaut. D.h. der Einzug der Daten und die einzelnen Verarbeitungsschritte bis zur Ausgabe eines Fahrkommandos sind in mehrere Anwendungen unterteilt. Zur Kommunikation zwischen den beteiligten Prozessen wird als Middleware die KogniMobil Real Time Data Base (KogMo-RTDB) eingesetzt. Diese wurde im Rahmen des Sonderfor-

schungsbereich Transregio 28 – „Kognitive Automobile“ (KogniMobil) entwickelt [Goebl et al., 2008].

Die KogMo-RTDB stellt eine SM-Schnittstelle (Shared Memory) dar. Darüber können Prozesse effizient große Datenmengen austauschen. Durch die Verwendung des SM fallen lediglich `memcpy`-Kommandos an, d.h. die Daten verbleiben im Arbeitsspeicher. Im Vergleich zur Middleware Robot Operating System (ROS) müssen die Daten dabei nicht aufwendig serialisiert und deserialisiert werden. Dafür ist die KogMo-RTDB im Gegenzug nicht für verteilte Rechnerarchitekturen geeignet. Diese Einschränkung kann z.T. durch zusätzliche Software und Hardware, wie einer WLAN-Brücke (Local Area Network), aufgehoben werden.

Der Zugriff auf die Daten der KogMo-RTDB geschieht über sog. KogMo-RTDB-Objekte. Dabei handelt es sich um Instanzen von Klassen, die sowohl das Lesen als auch das Speichern bestimmter zuvor definierter Datenstrukturen, sog. KogMo-RTDB-Structs, ermöglichen. Auch wenn die KogMo-RTDB es zulässt, verschiedene Datenstrukturen über dasselbe KogMo-RTDB-Objekt zu schreiben, ist i.a. jedem KogMo-RTDB-Struct oder einer festen Gruppe von KogMo-RTDB-Structs ein spezielles KogMo-RTDB-Objekt zugeordnet.

Für ein bestimmtes KogMo-RTDB-Objekt legt der schreibende Prozess neben einem Bezeichner für die Daten auch die Größe des Puffers fest, in dem die Daten eine Zeit lang vorgehalten werden. Dies ermöglicht später die Synchronisation verschiedener Datenstrukturen, die von verschiedenen Prozessen zu unterschiedlichen Zeiten geschrieben und gelesen werden. Dazu werden Daten in der KogMo-RTDB grundsätzlich mit zwei Zeitstempeln versehen: einem Datenzeitstempel t^{Daten}, über den sich das Alter der Daten errechnen lässt, und einen Übergabezeitstempel $t^{\text{Übergabe}}$, der den Zeitpunkt angibt, zu dem ein Datum in die KogMo-RTDB geschrieben wurde. Die Unterscheidung zwischen beiden Zeitstempeln ist notwendig, da die Daten zum Zeitpunkt ihrer Aufnahme noch nicht sofort in der KogMo-RTDB zur Verfügung stehen. Typischerweise sind Latenzen in den Sensoren, aber auch die Verarbeitungszeiten für einen Zeitversatz zwischen t^{Daten} und $t^{\text{Übergabe}}$ verantwortlich. Dies wird am Besten an einem Beispiel deutlich, welches in Abb. 3.6 visualisiert ist.

Angenommen, eine Kamera nimmt ein Bild der Umgebung auf, welches in der KogMo-RTDB abgelegt werden soll. Der Zeitpunkt des Bildes $t^{\text{Daten, Bild}}$ wird auf den mittleren Belichtungszeitpunkt festgelegt. Dieser Zeitpunkt liegt noch während der Aufnahme. Das Bild kann daher unmöglich schon in der KogMo-RTDB

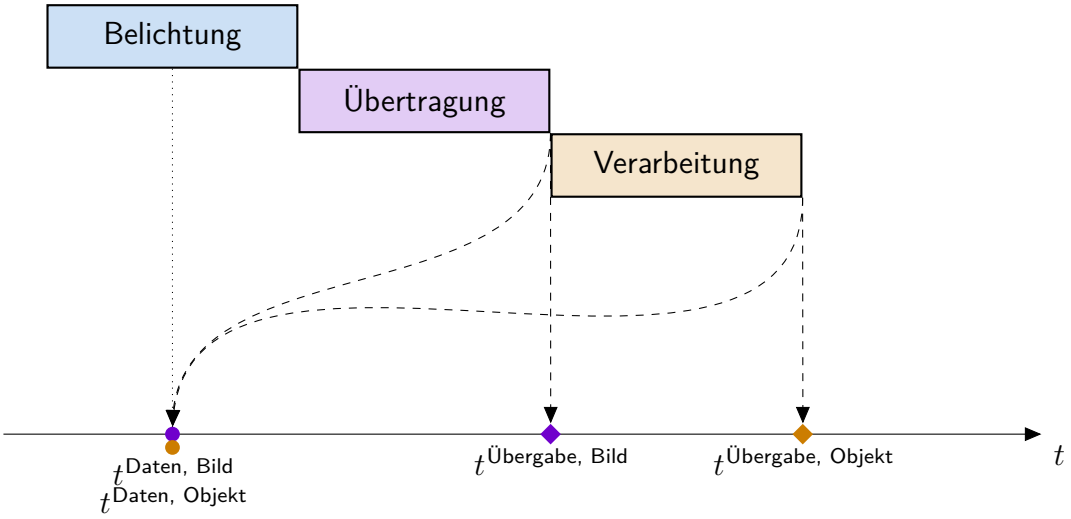


Abbildung 3.6:
Unterschied zwischen Datenzeitstempel t^{Daten} und Übergabezeitstempel $t^{\text{Übergabe}}$

stehen. Nach der Belichtung des Kamerachips muss dieser ausgelesen werden und die Daten müssen über die Schnittstelle der Kamera an den Rechner übertragen werden $t^{\text{Übergabe, Bild}}$. Erst danach kann das Bild an die KogMo-RTDB übergeben werden. Die Ursprungszeit des Bildes ist unabhängig davon, wie lange die Übertragung gedauert hat. Daher wird das Bild mit dem Datenzeitstempel $t^{\text{Daten, Bild}}$ in die KogMo-RTDB geschrieben. Erst nach dem Übergabezeitstempel $t^{\text{Übergabe, Bild}}$ ist es möglich, dass ein Programm die Verarbeitung des Kamerabildes übernimmt. Sobald dieses Programm die Verarbeitung abgeschlossen hat und beispielsweise ein Objekt darin erkannt hat, wird auch dieses Objekt in die KogMo-RTDB geschrieben. Dies geschieht zum Zeitpunkt $t^{\text{Übergabe, Objekt}}$. Die Daten bzgl. des Objekts entstammen allerdings einem Bild mit dem Datenzeitstempel $t^{\text{Daten, Bild}}$. Daher wird auch der Datenzeitstempel des Objekts $t^{\text{Daten, Objekt}}$ auf diesen Zeitpunkt gesetzt. Ein Prozess, der die Objektdaten aus der KogMo-RTDB liest, muss immer den Übergabezeitstempel des Objekts beachten. Falls es die Anfrage nach den aktuellen Objektdaten zeitlich vor $t^{\text{Übergabe, Objekt}}$ stellt, liefert ihm die KogMo-RTDB möglicherweise die Daten des Objekts aus dem vorherigen Bild. Daher ist es wichtig, dass der Entwickler die Zykluszeiten der einzelnen Prozesse kennt, sofern er die Daten dieser Prozesse mit möglichst geringen Zeitunterschieden lesen möchte.

Beim Lesen von Daten aus der KogMo-RTDB kann jedes Programm die Daten zu einem bestimmten Datenzeitstempel t^{Daten} anfordern. Ist der exakte Zeitstempel nicht bekannt, stehen verschiedene Aufrufe zur Verfügung, um beispielsweise den nächst älteren oder jüngeren Datensatz zu lesen. Da die nächst jüngeren Daten

u.U. noch nicht an die KogMo-RTDB übertragen wurden, besteht hierbei auch die Möglichkeit, durch einen blockierenden Aufruf auf diese Daten zu warten.

Die KogMo-RTDB erlaubt das Aufzeichnen und Abspielen des Datenstroms. Dies ermöglicht, dass sämtliche Sensordaten und ggf. auch Zwischenergebnisse während der Fahrt mit dem Versuchsträger auf einer Festplatte abgelegt werden. Die aufgezeichneten Daten können später an einem beliebigen Arbeitsplatz abgespielt werden, um die Fahrt zu rekonstruieren bzw. Modifikationen oder neue Algorithmen zu evaluieren. Das Aufzeichnen sowie das Abspielen kann auf bestimmte KogMo-RTDB-Objekte beschränkt werden. Dennoch muss gerade beim Aufzeichnen dafür Sorge getragen werden, dass die Festplatte zum Speichern der Daten zum einen groß genug und zum anderen schnell genug ist. Typischerweise entstehen beim Aufzeichnen der Sensordaten aller Kameras und dem LiDAR Datenmengen von 30 MB/s bis 50 MB/s. Damit sind herkömmliche Magnet-Festplatten schon an der Grenze ihrer Schreibgeschwindigkeiten, weshalb in beiden Versuchsträgern SSD-Festplatten zum Aufzeichnen verwendet werden.

3.4.2 Das `tas::app`-Framework

Bei der Synchronisation mehrerer KogMo-RTDB-Objekte muss beachtet werden, dass die Daten der Objekte asynchron und mit unterschiedlichen Zykluszeiten in die KogMo-RTDB geschrieben werden. Die Middleware ROS bietet für einen solchen Fall einen automatisieren Nachrichtenfilter, namens *Approximate Time Message Filter*, an. ROS arbeitet hingegen nur mit Datenzeitstempeln und vernachlässigt dabei den Übertragungszeitpunkt der Daten.

Da die vollständig manuelle Synchronisation der KogMo-RTDB-Objekte kompliziert und fehleranfällig ist, wurde im Rahmen dieser Arbeit an einem einheitlichen Framework namens `tas::app` gearbeitet, welches einerseits den Programmierer bei der Synchronisation der KogMo-RTDB-Objekte unterstützt und andererseits durch eine feste Programmstruktur und die Einbindung gemeinsam genutzter Bibliotheken die Verständlichkeit und Wiederverwendbarkeit des Quelltextes fördert.

Eine grundlegende Eigenschaft des `tas::app`-Frameworks ist die Trennung von Datenquelle und Anwendung. Dabei handelt es sich um eine 1:n-Beziehung, d.h. eine Quelle kann mit mehreren Anwendungen innerhalb desselben Prozesses verbunden sein. Damit lässt sich verhindern, dass die gleichen Daten aus der KogMo-RTDB für mehrere Anwendungen mehrfach gelesen werden müssen.

Die Datenquelle ist verantwortlich für das automatisierte Auslesen zeitlich beieinanderliegender Datenstrukturen mit Hilfe von KogMo-RTDB-Objekten. Der Anwender spezifiziert mit Hilfe variadischer Templates eine Liste von KogMo-RTDB-Objekten, die er synchronisiert auslesen möchte. Gleichzeitig definiert er eine Liste an Bezeichnern, unter denen die entsprechenden Objekte in der KogMo-RTDB zu finden sind. Anschließend erhält jedes gelistete KogMo-RTDB-Objekt eine der Eigenschaften *auslösend*, *notwendig* oder *optional*.

Die Datenquelle, die innerhalb des Prozesses als eigener Thread realisiert ist, wartet zyklisch auf neue Daten des als *auslösend* bezeichneten KogMo-RTDB-Objekts (z.B. eine LiDAR-Punktwolke). Erst wenn für dieses Objekt neue Daten übergeben wurden, versucht die Datenquelle die als *notwendig* und *optional* bezeichneten KogMo-RTDB-Objekte zu lesen (z.B. Bilder mehrerer Kameras). Dabei werden die Daten ausgewählt, die zeitlich am nächsten zum Datenzeitstempel des *auslösenden* KogMo-RTDB-Objekts liegen und deren Übergabezeitstempel älter oder gleich dem Übergabezeitstempel des *auslösenden* Objekts sind. Dadurch wird klar, dass nur jeweils ein KogMo-RTDB-Objekt als *auslösend* bezeichnet werden darf. Abbildung 3.7 zeigt ein Beispiel für die daraus folgende Synchronisation von drei KogMo-RTDB-Objekten.

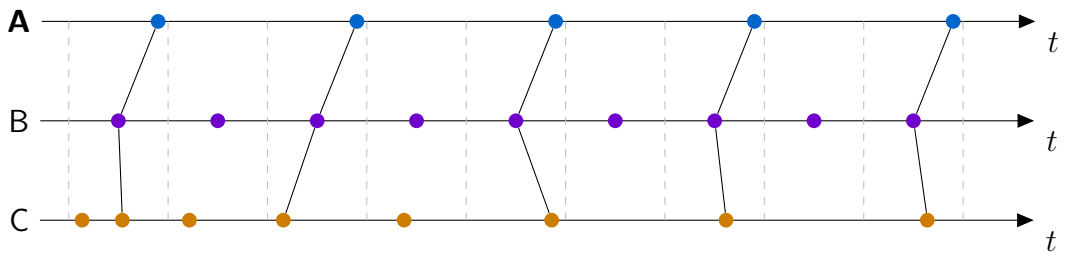


Abbildung 3.7:
Beispiel für die Synchronisation von drei KogMo-RTDB-Objekten. Zur Vereinfachung wird angenommen, dass für jedes der Objekte t^{Daten} gleich $t^{\text{Übergabe}}$ ist.

Objekt **A** (die LiDAR-Punktwolke) ist hierbei als *auslösend* definiert, so dass zu jedem Eintrag aus **A** ein zeitlich passender Datensatz aus B und C (Bilder verschiedener Kameras) gesucht wird. Ein zusammenhängendes Tripel ist hierbei durch verbundene Linien gekennzeichnet. In dem dargestellten Beispiel werden jeweils Tripel mit zeitlich nahe beieinanderliegenden Daten gefunden. Auch wenn die derart erfolgte Synchronisation sinnvoll erscheint, zeigt das Beispiel aus Abb. 3.8, dass die Synchronisation nicht immer optimal ist.

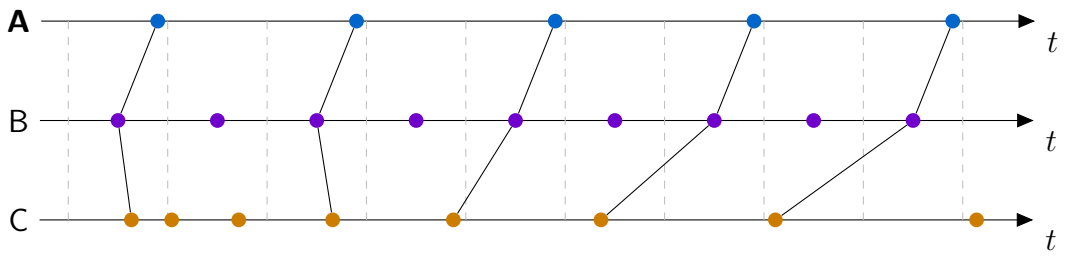


Abbildung 3.8:
Zweites Beispiel für die Synchronisation von drei KogMo-RTDB-Objekten. Im Vergleich zu Abb. 3.7 wurden die Daten des Objekts C zeitlich verzögert.

Im Vergleich zu Abb. 3.7 wurden hier die Daten des Objekts C zeitlich verzögert. Dadurch entstehen Tripel mit zeitlich stärker abweichenden Datenzeitstempeln. Die Datenzeitstempel der Tripel liegen deutlich weiter auseinander. Diesem Effekt kann, wie Abb. 3.9 dargestellt, durch einen Wechsel des *auslösenden* Objekts begegnet werden, d.h. statt der LiDAR-Punktwolke wird eine der Kameras das auslösende Objekt. Es zeigen sich – bis auf den farblich hervorgehobenen Fall – deutlich besser synchronisierte Tripel.

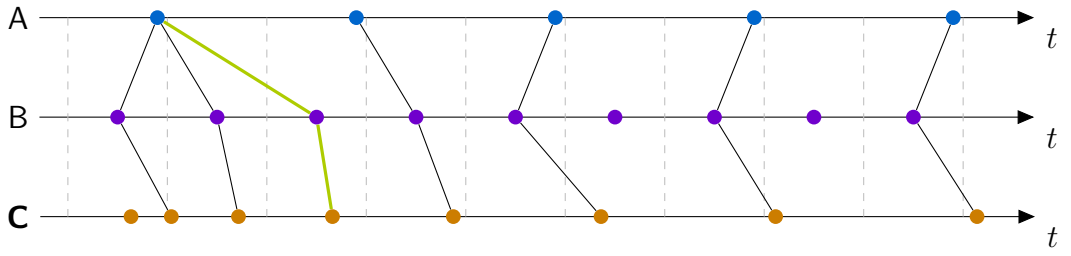


Abbildung 3.9:
Drittes Beispiel für die Synchronisation von drei KogMo-RTDB-Objekten. Im Gegensatz zu Abb. 3.8 ist nun **C** das *auslösende* Objekt.

Wie aus den Beispielen ersichtlich ist, hat die Wahl des *auslösenden* Objekts einen hohen Einfluss auf das Synchronisationsergebnis. Bei der Entscheidung muss berücksichtigt werden, ob ein Objekt periodisch in die KogMo-RTDB geschrieben wird, mit welcher Frequenz es geschrieben wird sowie – und das wurde in den Beispielen nicht berücksichtigt – mit welcher Verzögerung ($t^{\text{Daten}} - t^{\text{Übergabe}}$) es geschrieben wird. Wie eingangs erwähnt treten derartige Verzögerungen durch den Datentransport sowie durch die Weiterverarbeitung der Daten auf. In den meisten Fällen ist es daher sinnvoll, als *auslösendes* KogMo-RTDB-Objekt das Objekt zu wählen, welches die größte Verzögerung aufweist. Dadurch ist sichergestellt, dass die weiteren Objekte mit ähnlichen Zeitstempeln bereits in die KogMo-RTDB geschrieben wurden.

Nachdem die Datenquelle versucht hat, alle KogMo-RTDB-Objekte zu lesen, überprüft sie, ob jedes *notwendige* Objekt gelesen werden konnte. Das Lesen aus der KogMo-RTDB schlägt fehl, wenn beispielsweise keine andere Anwendung existiert, die das gewünschte KogMo-RTDB-Objekt schreibt. Nur wenn das *auslösende* und alle *notwendigen* Objekte gelesen werden konnten, übermittelt die Datenquelle das gelesene Datentupel nacheinander an alle mit ihr verknüpften Anwendungen.

Die Anwendungen selber laufen ebenfalls als eigene Threads im selben Prozessraum wie die Datenquelle. Es kann daher vorkommen, dass während die Datenquelle die neu gelesenen Daten zu einer der Anwendungen kopieren möchte, diese noch mit der Verarbeitung der zuvor übertragenen Daten beschäftigt ist. Daher muss das anwendungsinterne Datentupel vor konkurrierendem Zugriff durch Datenquelle und Anwendungen geschützt werden. Dieses Problem ist als das Erzeuger-Verbraucher-Problem bekannt.

Die typische Lösung dieses Problems besteht aus einen Ringpuffer in Kombination mit zwei Semaphoren zur Anzeige der noch freien und noch zu lesenden Einträge. Diese Lösung setzt allerdings voraus, dass der Verbraucher, d.h. die Anwendung, die Daten im Mittel mindestens so schnell liest, wie sie vom Erzeuger in den Ringpuffer geschrieben werden. Andernfalls läuft der Ringpuffer voll, was dazu führt, dass der Erzeuger, d.h. die Datenquelle, blockiert, bis wieder freie Einträge im Ringpuffer verfügbar sind. Ist die Datenquelle blockiert, kann sie auch keine Daten an die anderen mit ihr verbundenen Anwendungen mehr weiterleiten. Es ist somit möglich, dass eine einzelne langsame Anwendung alle anderen Anwendungen, die mit der gleichen Datenquelle verbunden sind, ausbremst. Darüber hinaus besteht bei der typischen Lösung des Erzeuger-Verbraucher-Problems die Annahme, dass der Verbraucher alle Daten des Erzeugers lesen möchte. Dies ist im Fall der autonomen Navigation nicht unbedingt gegeben. Hier soll die Entscheidung über das aktuelle Fahrmanöver möglichst auf Basis aktueller Daten getroffen werden. Deshalb wird es toleriert, wenn durch zu lange Berechnungszeiten einzelne Datensätze übersprungen werden.

Aus den genannten Gründen wird das Erzeuger-Verbraucher-Problem im `tas::app`-Framework durch zwei Mutexe zur Synchronisation gelöst. Der Prozessmutex sorgt dafür, dass die Anwendung nur dann Berechnungen anstellt, wenn neue Daten zur Verfügung stehen. Der Datenmutex schützt die Daten der Anwendung davor, überschrieben zu werden, falls die Anwendung ihre Berechnungen noch nicht abgeschlossen hat. Das Zusammenspiel der beiden Mutexe wird durch die Quelltexte 3.1 und 3.2 beschrieben.

```
1 void run()
2 {
3   process_mutex_.lock();
4   ⋮
5   data_mutex_.unlock();
6 }
```

Quelltext 3.1:
Zyklisch aufgerufene Funktion der Anwendung, in der alle Berechnungen stattfinden.

```
1 void update( const DataTuple data )
2 {
3   if ( data_mutex_.try_lock() )
4   {
5     data_ = data;
6     process_mutex_.unlock();
7   }
8 }
```

Quelltext 3.2:
Aktualisierung der Daten einer Anwendung. Die dargestelle Funktion wird durch die Datenquelle aufgerufen. Das Argument data enthält die neu eingelesenen Daten.

Die `run`-Funktion läuft dabei im Anwendungsthread und wird zyklisch aufgerufen. Die `update`-Funktion wird von der Datenquelle nach erfolgreichem Einlesen eines neuen Datentupels aufgerufen. Damit die Anwendung nach dem Starten zunächst auf neue Daten wartet, wird der Prozessmutex process_mutex_ bereits gesperrt initialisiert. Deshalb führt der Aufruf in Quelltext 3.1, Zeile 3 zum Blockieren des Anwendungsthreads. Sobald die Datenquelle ein neues Datentupel eingelesen hat, ruft sie die `update`-Funktion in Quelltext 3.2 auf. In Zeile 3 versucht sie dabei den Datenmutex data_mutex_ zu sperren, was beim ersten Aufruf erfolgreich ist. Darauf kopiert die Datenquelle die Daten (Zeile 5) und entsperrt schließlich den Prozessmutex process_mutex_ (Zeile 6). Das löst die Blockade der Anwendung, so dass diese ihrerseits nun den Prozessmutex process_mutex_ sperrt und ihre Berechnungen durchführt.

Für den weiteren Verlauf gibt es zwei Möglichkeiten. Im ersten Fall ist die Anwendung mit ihren Berechnungen fertig, bevor die Datenquelle neue Daten bereitstellen möchte. Da die `run`-Funktion zyklisch aufgerufen wird, aber der Prozessmutex process_mutex_ noch aus dem letzten Aufruf gesperrt ist, blockiert der Anwendungsthread nach erneutem Eintritt in die `run`-Funktion. Der Programmablauf steht wieder in der Ausgangssituation und erst nach dem Aufruf der `update`-Funktion durch die Datenquelle führt die Anwendung ihre Berechnungen auf Basis der neuen

Daten aus. Im zweiten Fall versucht die Datenquelle bereits neue Daten zu übergeben, obwohl die Anwendung ihre Berechnungen noch nicht abgeschlossen hat. Die Anwendung entsperrt den Datenmutex data_mutex_ erst nach Ende der Berechnungen. Deshalb ist der erneute Sperrversuch durch die Datenquelle in Quelltext 3.2 Zeile 3 erfolglos. Da es sich hierbei allerdings um einen `try_lock`-Aufruf handelt, blockiert die Datenquelle nicht, kehrt aus dem Funktionsaufruf zurück und versucht ggf. anderen Anwendungen die Daten bereitzustellen. Für die genannte Anwendung verfallen die Daten allerdings.

Das Zusammenspiel zwischen Datenquelle und Anwendungen spiegelt nur einen Aspekt des `tas::app`-Frameworks wider. Weitere Teile des Frameworks sind die Generierung einer graphischen Oberfläche für jede Anwendung sowie die Möglichkeit zur Interprozesskommunikation zwischen mehreren Anwendungen. Da diese Aspekte für die vorliegende Arbeit allerdings weniger relevant sind, wird hier nicht näher darauf eingegangen.

4 Modellbasierte Schätzung

Die Abschnitte 3.1 und 3.2 beschreiben Sensoren, die die Umgebung um das Fahrzeug sowie dessen Bewegung zeit-diskret abtasten. Dabei treten eine Reihe von Ungenauigkeiten auf. Beispielsweise wird in einer Kamera das Abbild der Umgebung durch die Projektion auf den Chip der Kamera diskretisiert. Im Fall des Light Detection and Ranging (LiDAR) werden Entfernungsmessungen nur in bestimmten Winkeln vorgenommen und auch das Inertial Navigation System (INS) arbeitet unter Zuhilfenahme von GPS-Signalen (Global Positioning System), die atmosphärischen Störungen ausgesetzt sind und durch Mehrfachreflexion die Laufzeitmessungen beeinträchtigen können, um nur einige Beispiele zu nennen. Glücklicherweise sind die Größenordnungen der Störungen der in dieser Arbeit verwendeten Sensoren bekannt oder lassen sich empirisch bestimmen.

Auf der anderen Seite führt das Zusammenspiel eines autonomen Fahrzeugs und seiner Umgebung zu einem komplexen dynamischen System. Nicht nur die Bewegung des Fahrzeugs selbst, auch die Bewegungen anderer Verkehrsteilnehmer oder – im Fall der drehbaren Kameraplattform – der Sensoren am Fahrzeug müssen betrachtet werden. Im Rahmen dieser Arbeit werden derartige Bewegungen durch Modelle beschrieben. Da diese Modelle, beispielsweise durch Vernachlässigung von Reibung, nur Vereinfachungen der realen Welt sind, treten auch hier Unsicherheiten auf, die an den jeweiligen Stellen quantifiziert werden.

Zusammenfassend bewegt sich diese Arbeit in einem System, in dem sowohl Messungen als auch Modelle mit Unsicherheiten behaftet sind, die in ihrem Umfang jedoch abschätzbar sind.

4.1 Zustandsraumbeschreibung

Zur Beschreibung derartiger Systeme kommt ein nicht-lineares Markov'sches Zustandsraummodell zum Einsatz. Die Forderung nach Nicht-Linearität ergibt sich

bereits aus den ebenfalls nicht-linearen Abbildungsgleichungen für Kameras, die in Abschnitt 5.1 vorgestellt werden. Die Markov-Eingenschaft ist aus Gründen der Speicherverwaltung im laufenden System erforderlich. Ohne diese Eigenschaft müsste die Historie einer jeden Zustandsgröße gespeichert werden, was zu einem linearen Anwachsen des Speicherbedarfs und ggf. auch Rechenbedarfs führt.

Das Zustandsraummodell gliedert sich in ein sog. System- oder auch Prozessmodell und ein Messmodell. Das Systemmodell besteht aus den stochastischen Systemgleichungen $f(\cdot)$, die Dynamik und Geometrie der Umgebung bzw. der bewegten Sensorik beschreiben. Das Messmodell beschreibt ebenfalls auf Basis stochastischer Gleichungen, den sog. Messgleichungen $g(\cdot)$, den Messprozess, d.h. die Abbildung der realen Welt durch Sensorik zu einzelnen Messungen $\boldsymbol{y}$. Auch wenn es möglich ist, die Systemgleichungen in ihrer kontinuierlichen Form aufzustellen, verwendet diese Arbeit ausschließlich die zeit-diskrete Form, da aufgrund der taktgetriebenen Verarbeitung auch die Schätzung zeit-diskret erfolgt. Die Diskretisierung bezieht sich allein auf die Formulierung der System- und Messgleichungen. Die Zustandsgrößen, die später zum Beispiel Entfernungen oder Winkel repräsentieren, werden als kontinuierlich angesehen.

Die System- und Messgleichungen lassen sich ganz allgemein in folgender Form beschreiben

$$\boldsymbol{x}_k = f_{k-1}\left(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{v}_{k-1}\right) \text{ und} \tag{4.1}$$

$$\boldsymbol{y}_k = g_k\left(\boldsymbol{x}_k, \boldsymbol{n}_k\right). \tag{4.2}$$

Die eingangs besprochenen Unsicherheiten werden hier einerseits durch den Systemrauschvektor $\boldsymbol{v}_{k-1}$ und andererseits durch den Messrauschvektor $\boldsymbol{n}_k$ repräsentiert. Beide müssen stochastisch voneinander unabhängiges weißes Rauschen beschreiben. Unterliegt ein Teil des System- oder Messrauschens etwa farbigem Rauschen, ist dieser Anteil entweder dem Zustand $\boldsymbol{x}$ oder den Steuergrößen $\boldsymbol{u}$ hinzuzufügen.

Die Systemgleichungen definieren die Änderung des Zustands $\boldsymbol{x}$ vom Zeitpunkt t_{k-1} zum Zeitpunkt t_k unter dem Einfluß der Steuergrößen $\boldsymbol{u}_{k-1}$. Die Messgleichungen beschreiben daraufhin die Abhängigkeit der Messung $\boldsymbol{y}_k$ vom aktuellen Zustand $\boldsymbol{x}_k$. Aufgrund der stochastischen Natur der System- und Messgleichungen werden in dieser Arbeit probabilistische Zustandsschätzer verwendet, die statt dem Zustand alleine die Wahrscheinlichkeitsdichtefunktion (WDF) für $\boldsymbol{x}_k$ schätzen

$$p\left(\boldsymbol{x}_k \middle| \boldsymbol{y}_{1:k}, \boldsymbol{u}_{1:k-1}\right). \tag{4.3}$$

Diese WDF wird auch a-posteriori WDF des Zustands $\boldsymbol{x}_k$ genannt. Sie ist bedingt durch alle Steuergrößen der Zeitpunkte $1, \ldots, k-1$, kurz $\boldsymbol{u}_{1:k-1}$ sowie alle Messungen der Zeitpunkte $1, \ldots, k$, kurz $\boldsymbol{y}_{1:k}$. Im Rahmen der Zustandsschätzung steht häufig allerdings nur die WDF zur Verfügung, in die die aktuelle Messung noch nicht eingeflossen ist

$$p\left(\boldsymbol{x}_k | \, \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-1}\right). \tag{4.4}$$

Diese WDF nennt sich a-priori WDF bzw. Prädiktion. Aufgabe eines Zustandsschätzers ist es nun, aus der a-priori WDF und der tatsächlichen, aktuellen Messung die a-posteriori WDF zu berechnen. Ein Beispiel für derartige Zustandsschätzer sind die sog. Bayes-Filter.

4.2 Bayes'sche Zustandsschätzung

Bayes-Filter beschreiben eine Gruppe von stochastischen Zustandsschätzern, die zur Schätzung der in Gleichung (4.3) vorgestellten WDF herangezogen werden können. Allen Bayes-Filtern liegt das sog. Bayes-Theorem zugrunde

$$p\left(a|b\right) = \frac{p\left(b|a\right) p\left(a\right)}{p\left(b\right)}. \tag{4.5}$$

Die Anwendung von Gleichung (4.5) auf Gleichung (4.3) ergibt

$$\underbrace{p\left(\boldsymbol{x}_k | \, \boldsymbol{y}_{1:k}, \boldsymbol{u}_{1:k-1}\right)}_{bel\left(\boldsymbol{x}_k\right)} \overset{Bayes}{=} \frac{p\left(\boldsymbol{y}_k | \, \boldsymbol{x}_k, \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-1}\right) p\left(\boldsymbol{x}_k | \, \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-1}\right)}{p\left(\boldsymbol{y}_k | \, \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-1}\right)}. \tag{4.6}$$

Für den linken Teil der Gleichung $bel\left(\boldsymbol{x}_k\right)$ ist häufig auch der Name *belief* (aus dem Englischen *belief*, z. dt. Glaube, Vorstellung) geläufig. Bei Betrachtung des ersten Terms des Zählers unter Berücksichtigung der Zustandsraumdefinition, insbesondere von Gleichung (4.2), fällt auf, dass $\boldsymbol{y}_k$ hauptsächlich von $\boldsymbol{x}_k$ abhängt. Damit ergibt sich die folgende Vereinfachung

$$p\left(\boldsymbol{y}_k | \, \boldsymbol{x}_k, \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-1}\right) = p\left(\boldsymbol{y}_k | \, \boldsymbol{x}_k\right). \tag{4.7}$$

Durch Erweiterung des zweiten Term des Zählers, der Filter-Prädiktion, mit dem Gesetz der totalen Wahrscheinlichkeit ergibt sich

$$p\left(\boldsymbol{x}_k|\, \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-1}\right) = \int p\left(\boldsymbol{x}_k|\, \boldsymbol{x}_{k-1}, \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-1}\right) p\left(\boldsymbol{x}_{k-1}|\, \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-1}\right) \mathrm{d}\boldsymbol{x}_{k-1}. \tag{4.8}$$

Die Markov-Eigenschaft fordert, dass der aktuelle Zustand $\boldsymbol{x}_k$ *vollständig* ist, und daher nur von seinem vorherigen Zustand und den daraus resultierten Steuergrößen abhängt. Damit lässt sich Gleichung (4.8) vereinfachen zu

$$p\left(\boldsymbol{x}_k|\, \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-1}\right) \overset{Markov}{=} \int p\left(\boldsymbol{x}_k|\, \boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}\right) p\left(\boldsymbol{x}_{k-1}|\, \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-1}\right) \mathrm{d}\boldsymbol{x}_{k-1}. \tag{4.9}$$

Ferner wissen wir, dass $\boldsymbol{x}_{k-1}$ in Wahrheit nicht von $\boldsymbol{u}_{k-1}$ abhängt, da die Steuergrößen $\boldsymbol{u}_{k-1}$ erst aus $\boldsymbol{x}_{k-1}$ berechnet werden. Damit lässt sich der zweite Teil des Integrals vereinfachen

$$p\left(\boldsymbol{x}_k|\, \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-1}\right) \overset{Markov}{=} \int p\left(\boldsymbol{x}_k|\, \boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}\right) \underbrace{p\left(\boldsymbol{x}_{k-1}|\, \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-2}\right)}_{bel\left(\boldsymbol{x}_{k-1}\right)} \mathrm{d}\boldsymbol{x}_{k-1}. \tag{4.10}$$

Im nächsten Schritt führt die Anwendung des Gesetzes der totalen Wahrscheinlichkeit auf den Nenner von Gleichung (4.6) zur Gleichung

$$p\left(\boldsymbol{y}_k|\, \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-1}\right) = \int p\left(\boldsymbol{y}_k|\, \boldsymbol{x}_k, \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-1}\right) p\left(\boldsymbol{x}_k|\, \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-1}\right) \mathrm{d}\boldsymbol{x}_k. \tag{4.11}$$

Die gleiche Vereinfachung wie für Gleichung (4.7) liefert

$$p\left(\boldsymbol{y}_k|\, \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-1}\right) = \int p\left(\boldsymbol{y}_k|\, \boldsymbol{x}_k\right) p\left(\boldsymbol{x}_k|\, \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-1}\right) \mathrm{d}\boldsymbol{x}_k. \tag{4.12}$$

Der erste Teil des Integrals ist bereits aus Gleichung (4.7) bekannt. Der zweite Teil entspricht Gleichung (4.10). Damit sind alle Terme beschrieben. Durch Einsetzen der Gleichungen (4.7), (4.10) und (4.12) in Gleichung (4.6) und Substituierung von

$p\left(\boldsymbol{x}_k|\, \boldsymbol{y}_{1:k}, \boldsymbol{u}_{1:k-1}\right)$ durch $bel\left(\boldsymbol{x}_k\right)$ wird die rekursive Eigenschaft der Bayes-Filter deutlich

$$p\left(\boldsymbol{x}_k|\, \boldsymbol{y}_{1:k}, \boldsymbol{u}_{1:k-1}\right) = \frac{p\left(\boldsymbol{y}_k|\, \boldsymbol{x}_k\right) \int p\left(\boldsymbol{x}_k|\, \boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}\right) p\left(\boldsymbol{x}_{k-1}|\, \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-2}\right) \mathrm{d}\boldsymbol{x}_{k-1}}{\int p\left(\boldsymbol{y}_k|\, \boldsymbol{x}_k\right) p\left(\boldsymbol{x}_k|\, \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-1}\right) \mathrm{d}\boldsymbol{x}_k} \tag{4.13}$$

$$bel\left(\boldsymbol{x}_k\right) = \frac{p\left(\boldsymbol{y}_k|\, \boldsymbol{x}_k\right) \int p\left(\boldsymbol{x}_k|\, \boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}\right) bel\left(\boldsymbol{x}_{k-1}\right) \mathrm{d}\boldsymbol{x}_{k-1}}{\int p\left(\boldsymbol{y}_k|\, \boldsymbol{x}_k\right) p\left(\boldsymbol{x}_k|\, \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-1}\right) \mathrm{d}\boldsymbol{x}_k}. \tag{4.14}$$

Der Übersichtlichkeit halber wird der Nenner durch η^{-1} ersetzt und es ergibt sich

$$bel\left(\boldsymbol{x}_k\right) = \underbrace{\eta p\left(\boldsymbol{y}_k|\, \boldsymbol{x}_k\right) \underbrace{\int p\left(\boldsymbol{x}_k|\, \boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}\right) bel\left(\boldsymbol{x}_{k-1}\right) \mathrm{d}\boldsymbol{x}_{k-1}}_{\text{Prädiktionsschritt}}}_{\text{Innovationsschritt}}. \tag{4.15}$$

Abbildung 4.1 zeigt eine graphische Interpretations des rekursiven Zusammenhangs in Form einer Markov-Kette.

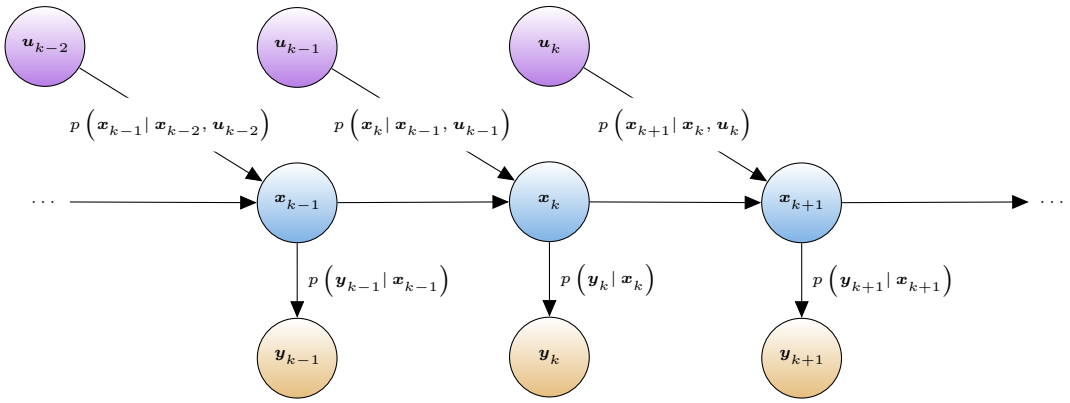


Abbildung 4.1: Darstellung der Bayes'schen Zustandsschätzung als Markov-Kette.

Für eine konkrete Implementierung muss neben dem Systemmodell $p\left(\boldsymbol{x}_k|\, \boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}\right)$ und dem Messmodell $p\left(\boldsymbol{y}_k|\, \boldsymbol{x}_k\right)$ auch die sog. a-priori Wahrscheinlichkeit $p\left(\boldsymbol{x}_0\right)$ bekannt sein. Je nachdem, welche Annahmen für diese drei WDF getroffen werden, ergibt sich eine Variante des Bayes-Filters. Im Folgenden werden vier dieser Varianten vorgestellt. Dabei dienen die ersten beiden lediglich als Grundlage für die letzten beiden, die wiederum in dieser Arbeit verwendet werden.

4.3 Kalmanfilter (KF)

Das Kalmanfilter (KF) [Kalman, 1960] gehört zur Familie der Gauß'schen Filter und ist zudem eines der bekanntesten Filter in der Robotik. Als Gauß'sches Filter wird die WDF aus Gleichung (4.3) als multivariate Gaußverteilung mit Mittelwert $\boldsymbol{\mu}$ und Kovarianzmatrix $\mathbf{P}$ angesehen und kann damit durch die folgende Formel beschrieben werden

$$p\left(\boldsymbol{x}\right) = \frac{1}{\left(2\pi\right)^{n} \det\left(\mathbf{P}\right)} e^{\left(-\frac{1}{2}\left(\boldsymbol{x}-\boldsymbol{\mu}\right)^{T} \mathbf{P}^{-1} \left(\boldsymbol{x}-\boldsymbol{\mu}\right)\right)}, \text{ wobei} \tag{4.16}$$

$$n = \dim(\boldsymbol{x}). \tag{4.17}$$

Die WDF von $\boldsymbol{x}$ ist damit vollständig durch den Mittelwert $\boldsymbol{\mu}$ und die Kovarianzmatrix $\mathbf{P}$ beschreibbar,

$$p\left(\boldsymbol{x}_k | \, \boldsymbol{y}_{1:k}, \boldsymbol{u}_{1:k-1}\right) = \mathcal{N}\left(\boldsymbol{x}_k | \, \boldsymbol{\mu}_k, \mathbf{P}_k\right). \tag{4.18}$$

Sie ist damit ebenfalls *unimodal*, d.h. sie besitzt nur ein einziges globales Maximum. Damit Gleichung (4.18) gilt, müssen – zusätzlich zur Markov-Eigenschaft der Bayes-Filter – die folgenden drei Bedingungen erfüllt sein:

1. Linearität der Systemgleichungen $f\left(\cdot\right)$ mit additivem Gauß'schen Rauschen

$$\boldsymbol{x}_k = \mathbf{F}_{k-1} \, \boldsymbol{x}_{k-1} + \mathbf{B}_{k-1} \, \boldsymbol{u}_{k-1} + \mathbf{D}_{k-1} \, \boldsymbol{v}_{k-1}. \tag{4.19}$$

   Dabei wird $\mathbf{F}_{k-1}$ die sog. Zustandsübergangsmatrix mit Dimension $n \times n$ und $n = \dim\left(\boldsymbol{x}\right)$ genannt. $\mathbf{B}_{k-1}$ beschreibt eine $n \times o$-Matrix mit $o = \dim\left(\boldsymbol{u}\right)$ und $\mathbf{D}_{k-1}$ eine $n \times l$-Matrix. $\boldsymbol{v}_{k-1}$ ist eine Zufallsvariable mit Dimension l, Mittelwert $\mathbf{0}$ und Kovarianzmatrix $\mathbf{Q}_{k-1}$, die die Ungenauigkeit $\boldsymbol{v}_{k-1} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{Q}_{k-1}\right)$ im Zustandsübergang repräsentiert. Es gilt

$$p\left(\boldsymbol{x}_k | \, \boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}\right) = \mathcal{N}\left(\boldsymbol{x}_k | \, \mathbf{F}_{k-1} \, \boldsymbol{x}_{k-1} + \mathbf{B}_{k-1} \, \boldsymbol{u}_{k-1}, \mathbf{D}_{k-1} \, \mathbf{Q}_{k-1} \, \mathbf{D}_{k-1}^{T}\right). \tag{4.20}$$

2. Linearität der Messgleichungen $g\left(\cdot\right)$ mit additivem Gauß'schen Rauschen

$$\boldsymbol{y}_k = \mathbf{C}_k \, \boldsymbol{x}_k + \boldsymbol{n}_k. \tag{4.21}$$

$\mathbf{C}_k$ ist eine $m \times n$-Matrix und $\boldsymbol{n}_k$ eine mittelwertfreie Gauß'sche Zufallsvariable mit Kovarianzmatrix $\mathbf{R}_k$: $\boldsymbol{n}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$ und $m = \dim(\boldsymbol{y})$. Es gilt

$$p(\boldsymbol{y}_k | \boldsymbol{x}_k) = \mathcal{N}(\boldsymbol{y}_k | \mathbf{C}_k \boldsymbol{x}_k, \mathbf{R}_k). \tag{4.22}$$

3. Gauß'sche Verteilung der a-priori Wahrscheinlichkeit $p(\boldsymbol{x}_0)$

$$p(\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_0 | \boldsymbol{\mu}_0, \mathbf{P}_0). \tag{4.23}$$

Daraus ergeben sich die folgenden KF-Gleichungen für den Prädiktionsschritt [Kalman, 1960]

$$\boldsymbol{x}_k^* = \mathbf{F}_{k-1} \hat{\boldsymbol{x}}_{k-1} + \mathbf{B}_{k-1} \boldsymbol{u}_{k-1} \tag{4.24}$$

$$\mathbf{P}_k^* = \mathbf{F}_{k-1} \mathbf{P}_{k-1} \mathbf{F}_{k-1}^T + \mathbf{D}_{k-1} \mathbf{Q}_{k-1} \mathbf{D}_{k-1}^T \tag{4.25}$$

$$\mathbf{K}_k = \mathbf{P}_k^* \mathbf{C}_k^T \left(\mathbf{C}_k \mathbf{P}_k^* \mathbf{C}_k^T + \mathbf{R}_k \right)^{-1}. \tag{4.26}$$

Die Gleichungen für den Innovationsschritt lauten damit

$$\hat{\boldsymbol{x}}_k = \boldsymbol{x}_k^* + \mathbf{K}_k (\boldsymbol{y}_k - \mathbf{C}_k \boldsymbol{x}_k^*) \tag{4.27}$$

$$\mathbf{P}_k = \mathbf{P}_k^* - \mathbf{K}_k \mathbf{C}_k \mathbf{P}_k^* \tag{4.28}$$

$$\mathbf{S}_k = \left(\mathbf{C}_k \mathbf{P}_k^* \mathbf{C}_k^T + \mathbf{R}_k \right)^{-1}. \tag{4.29}$$

Hierbei bezeichnet $\mathbf{K}_k$ das sog. Kalmangain. Es drückt aus, mit welchem Gewicht das Residuum, d.h. die Abweichung zwischen realer Messung $\boldsymbol{y}_k$ und prädizierter Messung $\mathbf{C}_k \boldsymbol{x}_k^*$, in den Zustand aufgenommen wird. Die Matrix $\mathbf{S}_k$ ist die sog. Innovationskovarianzmatrix. Sie wird häufig zum Aussortieren einzelner Messungen verwendet. Die Ausgabe des KF besteht aus dem korrigierten Zustandsvektor $\hat{\boldsymbol{x}}_k$ und der Kovarianzmatrik $\mathbf{P}_k$.

Unter den genannten Bedingungen linearer Gauß'scher Modelle ist das KF ein optimaler Zustandsschätzer im Sinne der kleinsten Fehlerquadrate. Die Komplexität wird hauptsächlich durch die Berechnung des Kalmangains (Gleichung (4.26)) bestimmt. Ist die Dimension der Messung ($\dim(\boldsymbol{y})$) deutlich größer als die Dimension des Zustands ($\dim(\boldsymbol{x})$), überwiegt die Inversion der Innovationskovarianzmatrix und die Komplexität ergibt sich zu $\mathcal{O}\left(\dim(\boldsymbol{y})^{<=3}\right)$. Ist hingegen die Dimension des Zustands größer als die der Messung, überwiegen die Matrixmultiplikationen mit $\mathcal{O}\left(\dim(\boldsymbol{x})^2\right)$.

4.4 Erweitertes Kalmanfilter (EKF)

Die Annahme eines linearen System- oder Messmodells kann in vielen Anwendungen der Robotik nicht getroffen werden. Ein Beispiel dafür ist die Projektion der dreidimensionalen Welt auf den Chip einer Kamera. Das Erweiterte Kalmanfilter (EKF) lockert die Anforderung an lineare Systemgleichungen $f(\cdot)$ und Messgleichungen $g(\cdot)$ des KF. Anstelle von Gleichung (4.19) schreiben wir

$$\boldsymbol{x}_k = f_{k-1}\left(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}\right) + \mathbf{D}_{k-1}\,\boldsymbol{v}_{k-1}. \tag{4.30}$$

Analog wird aus Gleichung (4.20)

$$\boldsymbol{y}_k = g_k(\boldsymbol{x}_k) + \boldsymbol{n}_k. \tag{4.31}$$

Dabei ist festzustellen, dass durch die Verwendung nicht-linearer System- oder Messmodelle $p\left(\boldsymbol{x}_k|\,\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}\right)$ bzw. $p(\boldsymbol{y}_k|\,\boldsymbol{x}_k)$ nicht mehr Gauß-verteilt sind. Um dennoch mit Gauß'schen Größen rechnen zu können, trifft das EKF die Annahme, dass die nicht-linearen System- und Messmodelle zumindest lokal linearisierbar sind. Dazu berechnet das EKF das erste Glied der Taylorreihendarstellung beider Modelle und setzt den korrigierten letzten Zustand bzw. die Prädiktion des aktuellen Zustands ein. Es ergeben sich die Jakobimatrizen

$$\mathbf{F}_{k-1}^{*} = \left.\frac{\partial f_{k-1}\left(\boldsymbol{x}, \boldsymbol{u}_{k-1}\right)}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_{k-1}} \tag{4.32}$$

$$\mathbf{C}_{k}^{*} = \left.\frac{\partial g_k(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}_k^{*}}. \tag{4.33}$$

Die Genauigkeit dieser Approximation hängt zum einen davon ab, wie linear die beiden Modelle um die gewählten Arbeitspunkte wirklich sind. Zum anderen spielt

natürlich die Größe der Unsicherheit, d.h. der Kovarianzen um $\hat{\boldsymbol{x}}_{k-1}$ bzw. $\boldsymbol{x}_k^*$, eine Rolle. Die Gleichungen des EKF lauten damit:

$$\boldsymbol{x}_k^* = f_{k-1}\left(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}\right) \tag{4.34}$$

$$\mathbf{P}_k^* = \mathbf{F}_{k-1}^* \, \mathbf{P}_{k-1} \, \mathbf{F}_{k-1}^{*T} + \mathbf{D}_{k-1} \, \mathbf{Q}_{k-1} \, \mathbf{D}_{k-1}^T \tag{4.35}$$

$$\mathbf{K}_k = \mathbf{P}_k^* \, \mathbf{C}_k^{*T} \underbrace{\left(\mathbf{C}_k^* \, \mathbf{P}_k^* \, \mathbf{C}_k^{*T} + \mathbf{R}_k\right)}_{\mathbf{S}_k^*}^{-1} \tag{4.36}$$

$$\hat{\boldsymbol{x}}_k = \boldsymbol{x}_k^* + \mathbf{K}_k \left(\boldsymbol{y}_k - g_k\left(\boldsymbol{x}_k\right)\right) \tag{4.37}$$

$$\mathbf{P}_k = \mathbf{P}_k^* - \mathbf{K}_k \, \mathbf{C}_k^* \, \mathbf{P}_k^* \tag{4.38}$$

Trotz der Linearisierung der beiden Modelle wird auch beim EKF die Komplexität hauptsächlich durch die Berechnung des Kalmangains bestimmt. Dementsprechend besitzt das EKF die gleiche Komplexität wie das KF.

4.5 Unscented Kalmanfilter (UKF)

Das EKF setzt voraus, dass die Systemgleichungen und die Messgleichungen linearisierbar sind. Dies erfordert zudem die Berechnung der Jakobimatrizen in den Gleichungen (4.32) und (4.33), was für nicht-differenzierbare Funktionen nicht möglich ist. Anstatt Annahmen für $f\left(\cdot\right)$ oder $g\left(\cdot\right)$ zu treffen, fokussiert sich das Unscented Kalmanfilter (UKF) auf die Gauß'sche Verteilung der Zufallsvariablen. Die Idee, die zugleich die Grundlage der sog. Unscented Transform (UT) darstellt und dem UKF seinen Namen gibt, lässt sich vereinfacht darstellen.

- Erzeugung eines Satzes diskreter Punkte, der sog. Sigma-Punkte, zur Approximation der Gauß'schen WDF der Zufallsvariable.
- Transformation eines jeden Sigma-Punktes einzeln durch $f\left(\cdot\right)$ bzw. $g\left(\cdot\right)$.
- Rekonstruktion der Gauß'schen WDF der transformierten Zufallsvariablen aus den transformierten Sigma-Punkten.

Auch wenn die UT in der Lage ist, auch die höheren Momente einer WDF, wie Schiefe oder Wölbung, zu erfassen, genügen im Falle Gauß'scher Zufallsvariablen die ersten beiden Momente: Mittelwert und Kovarianz. Dazu werden $2n + 1$ Sigma-Punkte benötigt, wobei n die Dimension der Zufallsvariablen darstellt. Die Darstellung der

UT analog zu Wan und Merwe [2000], Van der Merwe und Wan [2002] als Funktion ergibt

$$\left\{\boldsymbol{\mu}^{\mathsf{y}}, \boldsymbol{\Sigma}^{\mathsf{yy}}, \boldsymbol{\mu}^{(i)\,\mathsf{x}}, \boldsymbol{\mu}^{(i)\,\mathsf{y}}\right\} = ut\left(\boldsymbol{\mu}^{\mathsf{x}}, \boldsymbol{\Sigma}^{\mathsf{xx}}, g\left(\cdot\right), \boldsymbol{\Sigma}^{\mathsf{gg}}\right). \tag{4.39}$$

$\boldsymbol{\mu}^{\mathsf{x}}$ und $\boldsymbol{\Sigma}^{\mathsf{xx}}$ beschreiben den Mittelwert und die zugehörige Kovarianzmatrix. Die Funktion $g\left(\cdot\right)$ ist eine beliebige nicht-lineare Transformationsfunktion. Die Kovarianzmatrix $\boldsymbol{\Sigma}^{\mathsf{gg}}$ dient der Berücksichtigung von Ungenauigkeiten, die durch $g\left(\cdot\right)$ verursacht werden. Die Ausgabe der Funktion besteht aus dem transformierten Mittelwert $\boldsymbol{\mu}^{\mathsf{y}}$ und der transformierten Kovarianzmatrix $\boldsymbol{\Sigma}^{\mathsf{yy}}$ sowie den berechneten Sigma-Punkten $\boldsymbol{\mu}^{(i)\,\mathsf{x}}$ und deren Transformationen $\boldsymbol{\mu}^{(i)\,\mathsf{y}}$. Dabei führt $ut\left(\boldsymbol{\mu}^{\mathsf{x}}, \boldsymbol{\Sigma}^{\mathsf{xx}}, g\left(\cdot\right), \boldsymbol{\Sigma}^{\mathsf{gg}}\right)$ die folgenden Berechnungen durch

$$n = \dim\left(\boldsymbol{\mu}^{\mathsf{x}}\right) \tag{4.40}$$

$$\lambda = \alpha^2\left(n + \kappa\right) \tag{4.41}$$

$$\boldsymbol{\mu}^{(i)\,\mathsf{x}} = \boldsymbol{\mu}^{\mathsf{x}} + (-1)^i \left(\sqrt{\left(n+\lambda\right)\boldsymbol{\Sigma}^{\mathsf{xx}}}\right)_{\left[\lfloor\frac{i+1}{2}\rfloor\right]},\ i = 0,\ldots,2n \tag{4.42}$$

$$\boldsymbol{\mu}^{(i)\,\mathsf{y}} = g\left(\boldsymbol{\mu}^{(i)\,\mathsf{x}}\right),\ i = 0,\ldots,2n \tag{4.43}$$

$$w^{(0)\,\mathsf{m}} = \frac{\lambda}{n+\lambda} \tag{4.44}$$

$$w^{(i)\,\mathsf{m}} = \frac{1}{2\left(n+\lambda\right)},\ i = 1,\ldots,2n \tag{4.45}$$

$$w^{(0)\,\mathsf{c}} = \frac{\lambda}{n+\lambda} + \left(1 - \alpha^2 + \beta\right) \tag{4.46}$$

$$w^{(i)\,\mathsf{c}} = w^{(i)\,\mathsf{m}},\ i = 1,\ldots,2n \tag{4.47}$$

$$\boldsymbol{\mu}^{\mathsf{y}} = \sum_{i=0}^{2n} w^{(i)\,\mathsf{m}}\,\boldsymbol{\mu}^{(i)\,\mathsf{y}} \tag{4.48}$$

$$\boldsymbol{\Sigma}^{\mathsf{yy}} = \sum_{i=0}^{2n} w^{(i)\,\mathsf{c}}\left(\boldsymbol{\mu}^{(i)\,\mathsf{y}} - \boldsymbol{\mu}^{\mathsf{y}}\right)\left(\boldsymbol{\mu}^{(i)\,\mathsf{y}} - \boldsymbol{\mu}^{\mathsf{y}}\right)^T + \boldsymbol{\Sigma}^{\mathsf{gg}}. \tag{4.49}$$

Der Skalierungsparameter λ bestimmt die Streuung der Sigma-Punkte um den Mittelwert. Dabei ist α typischerweise ein kleiner positiver Wert (beispielsweise 0,001) und κ in der Regel 0 [Wan und Merwe, 2000, Van der Merwe und Wan, 2002]. Der Matrixoperator $\left(\cdot\right)_{[i]}$ liefert die i-te Spalte einer Matrix und $\mathbf{0}$ für $i = 0$. Anhand von β ist es möglich, Vorwissen über die Verteilung von $\boldsymbol{\mu}^{\mathsf{x}}$ bzw. $\boldsymbol{\mu}^{\mathsf{y}}$ einzubringen. Für den Fall Gauß-verteilter Zufallsvariablen ist $\beta = 2$ optimal [Julier, 2002].

Nach der Berechnung von n und λ folgt in Gleichung (4.42) die Berechnung der Sigmapunkte $\boldsymbol{\mu}^{(i)\,\mathsf{x}}$. Diese werden anschließend einzeln durch $g\left(\cdot\right)$ transformiert,

wobei $g(\cdot)$ später dem System- oder Messmodell entspricht. Gleichung (4.48) berechnet mit Hilfe der Mittelwertgewichte $w^{(i)\,\mathsf{m}}$ den neuen Mittelwert $\boldsymbol{\mu}^{\mathsf{y}}$, bevor in Gleichung (4.49) die dazugehörige Kovarianzmatrix $\boldsymbol{\Sigma}^{\mathsf{yy}}$ bestimmt wird.

Unter Zuhilfenahme von $ut\left(\boldsymbol{\mu}^{\mathsf{x}}, \boldsymbol{\Sigma}^{\mathsf{xx}}, g(\cdot), \boldsymbol{\Sigma}^{\mathsf{gg}}\right)$ lauten die Gleichungen des UKF

$$\left\{\boldsymbol{x}_k^*, \mathbf{P}_k^*, \cdot, \cdot\right\} = ut\left(\hat{\boldsymbol{x}}_{k-1}, \mathbf{P}_{k-1}, f_{k-1}\left(\cdot, \boldsymbol{u}_{k-1}\right), \mathbf{D}_{k-1}\,\mathbf{Q}_{k-1}\,\mathbf{D}_{k-1}^T\right) \tag{4.50}$$

$$\left\{\boldsymbol{y}_k^*, \mathbf{S}_k, \boldsymbol{x}_k^{(i)\,*}, \boldsymbol{y}_k^{(i)\,*}\right\} = ut\left(\boldsymbol{x}_k^*, \mathbf{P}_k^*, g_k(\cdot), \mathbf{R}_k\right) \tag{4.51}$$

$$\mathbf{P}_k^{\mathsf{xy}} = \sum_{i=0}^{2n} w^{(i)\,\mathsf{c}}\left(\boldsymbol{x}_k^{(i)\,*} - \boldsymbol{x}_k^*\right)\left(\boldsymbol{y}_k^{(i)\,*} - \boldsymbol{y}_k^*\right)^T \tag{4.52}$$

$$\mathbf{K}_k = \mathbf{P}_k^{\mathsf{xy}}\,\mathbf{S}_k^{-1} \tag{4.53}$$

$$\hat{\boldsymbol{x}}_k = \boldsymbol{x}_k^* + \mathbf{K}_k\left(\boldsymbol{y}_k - \boldsymbol{y}_k^*\right) \tag{4.54}$$

$$\mathbf{P}_k = \mathbf{P}_k^* - \mathbf{K}_k\,\mathbf{S}_k\,\mathbf{K}_k^T. \tag{4.55}$$

Dabei bezeichnet $\mathbf{S}_k$ die sog. Innovationskovarianzmatrix. Im Vergleich zum EKF benötigt das UKF keine Jakobimatrizen, was die Implementierung vereinfacht bzw. für einige nicht-lineare Gleichungssysteme erst möglich macht. Da die System- und Messgleichungen für jeden Sigma-Punkt durchlaufen werden, liegt der Rechenaufwand des UKF leicht über dem des EKF, bei dem die System- bzw. Messgleichungen nur jeweils einmal aufgerufen werden. Dennoch liegt der Rechenaufwand beider Filter in der gleichen Größenordnung [Wan und Merwe, 2000]. Die approximierten Mittelwerte und Kovarianzen sind beim UKF genauer als beim EKF, was insbesondere bei stark nicht-linearen System- bzw. Messgleichungen im Vergleicht zum EKF zu einer genaueren Schätzung führt.

4.6 Partikelfilter (PF)

Die vorgestellten Varianten des Kalmanfilters gehen alle von Gauß'schen Zufallsvariablen aus. Sind die Zufallsvariablen, d.h. der Zustand und die Messung nicht Gauß-verteilt, muss die WDF anders repräsentiert werden. Eine Möglichkeit sind gitterbasierte Filter, die auf einer Diskretisierung des Raums der Zufallsvariablen arbeiten. Da derartige Filter insbesondere bei hochdimensionalen Problemen aufgrund des hohen Rechenbedarfs nicht mehr praktikabel sind, werden sie in dieser Arbeit nicht weiter betrachtet.

In eine ähnliche, d.h. diskretisierende, Richtung geht das Partikelfilter (PF). Hier wird das a-posteriori $bel\left(\boldsymbol{x}_k\right)$ durch eine diskrete Menge an Stichproben $\boldsymbol{x}_k^{(1)}, \ldots, \boldsymbol{x}_k^{(M)}$ und zugehörigen Gewichten $w_k^{(1)}, \ldots, w_k^{(M)}$ repräsentiert. Das Paar aus Stichprobe und Gewicht $\left[\boldsymbol{x}_k^{(i)}, w_k^{(i)}\right]$ wird dabei als Partikel bezeichnet. Im Gegensatz zu gitterbasierten Filtern sind die Partikel nicht notwendigerweise gleichmäßig über den Zustandsraum verteilt. Stattdessen wird versucht, über die Verteilung der Partikel die WDF möglichst gut zu beschreiben. Die Grundlage hierfür liefert die sog. Monte Carlo Integration.

4.6.1 Monte Carlo Integration

Ist die analytische Berechnung des Integrals einer Funktion nicht möglich oder nicht gewünscht, steht eine Reihe numerischer Verfahren zur Verfügung. Auch hier gibt es gitterbasierte Verfahren, die den Wertebereich der Funktion gleichverteilt abtasten. Zusätzlich existieren auch stochastische Verfahren, die das Integral der Funktion über Stichproben berechnen. Ein solches Verfahren ist die Monte Carlo Integration. Zur Bestimmung des Integrals

$$I = \int g\left(\boldsymbol{x}\right) \, \mathrm{d}\boldsymbol{x} \tag{4.56}$$

wird dabei die Funktion $g\left(\boldsymbol{x}\right)$ wie folgt faktorisiert

$$g\left(\boldsymbol{x}\right) = k\left(\boldsymbol{x}\right) \pi\left(\boldsymbol{x}\right) \tag{4.57}$$

und $\pi\left(\boldsymbol{x}\right)$ als WDF interpretiert, von der im Folgenden einzelne Stichproben $\boldsymbol{x}^{(i)}$ gezogen werden. Ristic et al. [2004] zeigen, dass der Mittelwert aller $k\left(\boldsymbol{x}^{(i)}\right)$ unter gewissen Nebenbedingungen mit einer steigenden Anzahl von Stichproben zum Wert des gesuchten Integrals I konvergiert

$$\begin{aligned} I &= \int k\left(\boldsymbol{x}\right) \pi\left(\boldsymbol{x}\right) \, \mathrm{d}\boldsymbol{x} \\ &\approx \frac{1}{M} \sum_{i=1}^{M} k\left(\boldsymbol{x}^{(i)}\right), \; \boldsymbol{x}^{(i)} \sim \pi\left(\boldsymbol{x}\right). \end{aligned} \tag{4.58}$$

Die Konvergenzrate ist dabei – im Gegensatz zu deterministischen Verfahren – bei der Monte Carlo Integration unabhängig von der Dimension des Integranden [Ristic et al., 2004]. Allerdings setzt die Monte Carlo Integration voraus, dass die Stichproben von $\pi\left(\boldsymbol{x}\right)$ gezogen werden können. Häufig ist dies nur schwer möglich, beispielsweise wenn

$\pi(\boldsymbol{x})$ sowohl multimodal als auch mehrdimensional ist. In solchen Fällen bietet die Methode der wesentlichen Stichproben eine weit verbreitete Lösungsmöglichkeit.

4.6.2 Methode der wesentlichen Stichproben

Die Anwendung der Methode der wesentlichen Stichproben eignet sich, wenn das Ziehen von Stichproben einer WDF $\pi(\boldsymbol{x})$ nur aufwändig möglich ist, die WDF selbst aber zumindest bis auf Proportionalität auswertbar ist. Die Stichproben werden dabei von einer Vorschlagswahrscheinlichkeit $q(\boldsymbol{x})$, anstelle von $\pi(\boldsymbol{x})$, gezogen. Die Erweiterung des bereits bekannten Integrals um die Vorschlagswahrscheinlichkeit $q(\boldsymbol{x})$ führt zu

$$\begin{aligned} I &= \int k(\boldsymbol{x})\,\pi(\boldsymbol{x})\;\mathrm{d}\boldsymbol{x} \\ &= \int k(\boldsymbol{x})\,\frac{\pi(\boldsymbol{x})}{q(\boldsymbol{x})} q(\boldsymbol{x})\;\mathrm{d}\boldsymbol{x}. \end{aligned} \tag{4.59}$$

Durch Anwendung der Monte Carlo Integration erhalten wir

$$I \approx \frac{1}{M}\sum_{i=1}^{M} k\left(\boldsymbol{x}^{(i)}\right)\tilde{w}^{(i)}, \text{ mit} \tag{4.60}$$

$$\tilde{w}^{(i)} = \frac{\pi\left(\boldsymbol{x}^{(i)}\right)}{q\left(\boldsymbol{x}^{(i)}\right)} \text{ und } \boldsymbol{x}^{(i)} \sim q(\boldsymbol{x}). \tag{4.61}$$

Durch Normalisieren der Gewichte $\tilde{w}^{(i)}$ ergeben sich die sog. wesentlichen Gewichte (*engl. importance weights*)

$$I \approx \sum_{i=1}^{M} k\left(\boldsymbol{x}^{(i)}\right) w^{(i)}, \text{ mit} \tag{4.62}$$

$$w^{(i)} = \frac{\tilde{w}^{(i)}}{\sum_{j=1}^{M}\tilde{w}^{(j)}} \text{ und } \boldsymbol{x}^{(i)} \sim q(\boldsymbol{x}). \tag{4.63}$$

Abbildung 4.2 veranschaulicht die Methode der wesentlichen Stichproben an einem Beispiel. Zunächst werden die Stichproben $\boldsymbol{x}^{(i)}$ von der Vorschlagswahrscheinlichkeit $q(\boldsymbol{x})$ gezogen (blaue Kreise). Anschließend werden die wesentlichen Gewichte $w^{(i)}$ aus dem Quotienten von $\pi\left(\boldsymbol{x}^{(i)}\right)$ und $q\left(\boldsymbol{x}^{(i)}\right)$ berechnet. Die Gewichte sind entsprechend ihres Wertes in Größe und Farbe kodiert. Im gezeigten Beispiel sind die Vorschlagswahrscheinlichkeit und $\pi\left(\boldsymbol{x}^{(i)}\right)$ recht unterschiedlich, was dazu führt, dass die Maxima von $\pi\left(\boldsymbol{x}^{(i)}\right)$ nur ungenau abgetastet werden. Diesem Effekt kann

durch eine ähnlichere Vorschlagswahrscheinlichkeit oder durch eine Erhöhung der Stichprobenanzahl begegnet werden.

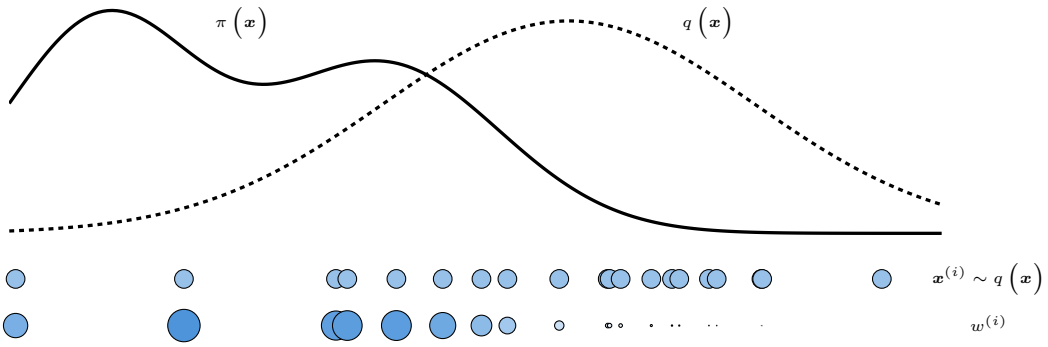


Abbildung 4.2:
Beispiel der Methode der wesentlichen Stichproben. Die Stichproben $\boldsymbol{x}^{(i)}$ werden von der Vorschlagswahrscheinlichkeit $q\left(\boldsymbol{x}\right)$ gezogen. Die wesentlichen Gewichte $w^{(i)}$ werden über den Quotienten von $\pi\left(\boldsymbol{x}^{(i)}\right)$ und $q\left(\boldsymbol{x}^{(i)}\right)$ berechnet.

4.6.3 SIR-Partikelfilter

Gemäß Mählisch [2009], Mahler [2007] führt die Anwendung der Monte Carlo Integration und der Methode der wesentlichen Stichproben auf die Gleichungen der Bayes'schen Zustandsschätzung aus Abschnitt 4.2 zum sog. SIR-Partikelfilter (Sequential Importance Resampling). Auch hierbei wird zwischen Prädiktions- und Innovationsschritt unterschieden.

Die Approximation von Gleichung (4.10) mit Hilfe der Monte Carlo Integration ergibt die Prädiktionsgleichung

$$p\left(\boldsymbol{x}_k|\, \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-1}\right) \approx \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \delta\left(\boldsymbol{x}_k - f_{k-1}\left(\boldsymbol{x}_{k-1}^{(i)}, \boldsymbol{u}_{k-1}, \boldsymbol{v}_{k-1}^{(j)}\right)\right), \text{ mit} \tag{4.64}$$

$$\boldsymbol{x}_{k-1}^{(i)} \sim bel\left(\boldsymbol{x}_{k-1}\right) \text{ und } \boldsymbol{v}_{k-1}^{(j)} \sim p\left(\boldsymbol{v}\right). \tag{4.65}$$

Eine detaillierte Herleitung dieser Gleichung ist in Mählisch [2009], Mahler [2007] zu finden. Zu erwähnen ist hierbei, dass sich die Gesamtzahl der Stichproben zur Approximation der a-priori WDF $p\left(\boldsymbol{x}_k|\, \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-1}\right)$ aus dem Produkt der Zahl der Stichproben von der a-posteriori WDF $bel\left(\boldsymbol{x}_{k-1}\right)$ und der Zahl der Stichproben des Systemrauschens $\boldsymbol{v}$ ergibt. Für $N > 1$ steigt damit die Gesamtzahl der Stichproben

mit jedem Rekursionsschritt an. Da dies in der Praxis zu einem immer größer werdenden Rechenaufwand führt, verwendet diese Arbeit stets $N = 1$. Damit gilt

$$p\left(\boldsymbol{x}_k|\, \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-1}\right) \approx \frac{1}{M} \sum_{i=1}^{M} \delta\left(\boldsymbol{x}_k - \boldsymbol{x}_k^{(i)*}\right), \text{ mit} \tag{4.66}$$

$$\boldsymbol{x}_k^{(i)*} = f_{k-1}\left(\boldsymbol{x}_{k-1}^{(i)}, \boldsymbol{u}_{k-1}, \boldsymbol{v}_{k-1}^{(i)}\right), \tag{4.67}$$

$$\boldsymbol{x}_{k-1}^{(i)} \sim bel\left(\boldsymbol{x}_{k-1}\right) \text{ und } \boldsymbol{v}_{k-1}^{(i)} \sim p\left(\boldsymbol{v}\right). \tag{4.68}$$

Ausgehend von der a-posteriori WDF werden nun M Stichproben $\boldsymbol{x}_{k-1}^{(i)}$ generiert, durch das Systemmodell $f_{k-1}\left(\boldsymbol{x}_{k-1}^{(i)}, \boldsymbol{u}_{k-1}, \boldsymbol{v}_{k-1}^{(i)}\right)$ prädiziert und letztlich mit Rauschen gemäß $p\left(\boldsymbol{v}\right)$ beaufschlagt. Damit ergeben sich M prädizierte Stichproben $\boldsymbol{x}_k^{(i)*}$. Die Dirac-Funktion $\delta\left(\cdot\right)$ drückt schließlich aus, dass die a-priori WDF $p\left(\boldsymbol{x}_k|\, \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-1}\right)$ nur an diesen M Stellen den Wert $\frac{1}{M}$ annimmt und ansonsten gleich 0 ist.

Das Ziel der Filter-Innovation ist es, ausgehend von dieser a-priori WDF mit Hilfe der aktuellen Messungen die neue a-posteriori WDF $p\left(\boldsymbol{x}_k|\, \boldsymbol{y}_{1:k}, \boldsymbol{u}_{1:k-1}\right)$ zu berechnen. Durch Anwendung der Methode der wesentlichen Stichproben auf Gleichung (4.15) erhalten wir

$$p\left(\boldsymbol{x}_k|\, \boldsymbol{y}_{1:k}, \boldsymbol{u}_{1:k-1}\right) \approx \sum_{i=1}^{M} w_k^{(i)} \delta\left(\boldsymbol{x}_k - \boldsymbol{x}_k^{(i)}\right). \tag{4.69}$$

Hierbei müssten die Stichproben $\boldsymbol{x}_k^{(i)}$ im Idealfall von der a-posteriori Verteilung gezogen werden. Auf Grund der Rekursion steht diese allerdings erst nach dem Innovationsschritt zur Verfügung. Aus Gleichungen (4.61) und (4.63) ist bekannt, dass zur Berechnung der wesentlichen Gewichte $w_k^{(i)}$ lediglich die Proportionalität zwischen $\pi\left(\boldsymbol{x}\right)$ und $q\left(\boldsymbol{x}\right)$ entscheidend ist. Wie in Gleichung (4.6) zu sehen, ist die a-posteriori WDF proportional zum Zähler derselben Gleichung. Unter Berücksichtigung von Gleichung (4.7) folgt

$$p\left(\boldsymbol{x}_k|\, \boldsymbol{y}_{1:k}, \boldsymbol{u}_{1:k-1}\right) \propto p\left(\boldsymbol{y}_k|\, \boldsymbol{x}_k\right) p\left(\boldsymbol{x}_k|\, \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-1}\right). \tag{4.70}$$

Weiterhin stellt sich die Frage, von welcher Vorschlagsverteilung die Stichproben $\boldsymbol{x}_k^{(i)}$ gezogen werden. Im SIR-Partikelfilter tritt die a-priori WDF in die Rolle der Vorschlagsverteilung. Die nicht-normierten Partikelgewichte ergeben sich damit zu

$$\tilde{w}_k = \frac{p\left(\boldsymbol{y}_k \middle| \boldsymbol{x}_k\right) p\left(\boldsymbol{x}_k \middle| \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-1}\right)}{p\left(\boldsymbol{x}_k \middle| \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-1}\right)} = p\left(\boldsymbol{y}_k \middle| \boldsymbol{x}_k\right) \text{ bzw.} \tag{4.71}$$

$$\tilde{w}_k^{(i)} = p\left(\boldsymbol{y}_k \middle| \boldsymbol{x}_k^{(i)*}\right), \ \boldsymbol{x}_k^{(i)*} \sim p\left(\boldsymbol{x}_k \middle| \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-1}\right). \tag{4.72}$$

Die Normalisierung folgt gemäß Gleichung (4.63). Ein Vorteil dieser Vorschlagswahrscheinlichkeit ist, dass die Stichproben $\boldsymbol{x}_k^{(i)*}$ bereits aus dem Prädiktionsschritt bekannt sind. Eine Neuberechnung ist daher nicht notwendig. Außerdem können die Gewichte direkt aus dem Messmodell $p\left(\boldsymbol{y}_k \middle| \boldsymbol{x}_k\right)$ berechnet werden.

Auf der anderen Seite hängt die Position der Stichproben stark vom Systemmodell ab. Fehler oder Ungenauigkeiten des Systemmodells können dazu führen, dass Stichproben abdriften und in Bereiche geringer Likelihood fallen. Da derartigen Stichproben nur sehr kleine Gewichte zugeteilt werden, steigen im Verhältnis die Gewichte der anderen Stichproben, so dass die a-posteriori WDF effektiv von weniger Partikeln getragen wird. Dieses Phänomen wird Degeneration genannt. Wenn auf diese Weise alle Partikel in Bereich geringer Likelihood fallen, führt dies ggf. zur Divergenz des Filters.

Es existieren viele Varianten des Partikelfilters, die auf unterschiedliche Weise versuchen, der Degeneration entgegenzuwirken. Eine dieser Varianten ist das *Systematic Resampling* [Kitagawa, 1996, Arulampalam et al., 2002], welches im SIR-Partikelfilter zum Einsatz kommt.

Das Systematic Resampling findet in Rahmen der Filter-Innovation statt. Ziel ist es, Partikel mit geringen Gewichten aus der Partikelmenge zu entfernen und stattdessen Partikel mit hohen Gewichten zu vervielfältigen. Die Gesamtzahl der Partikel bleibt dabei erhalten. Die Auswahl der zu verwerfenden und zu vervielfältigenden Partikel geschieht über die kumulativen Gewichte

$$c_k^{(i)} = \sum_{j=1}^{i} w_k^{(j)} \tag{4.73}$$

und die stochastischen Selektoren

$$s_k^{(i)} = s_k^{(i-1)} + \frac{1}{M}, \text{ mit} \tag{4.74}$$

$$s_k^{(1)} \sim \mathcal{U}\left[0, \frac{1}{M}\right]. \tag{4.75}$$

An die Stelle der Stichprobe $\boldsymbol{x}_k^{(i)*}$ tritt nun $\bar{\boldsymbol{x}}_k^{(i)}$ mit

$$\bar{\boldsymbol{x}}_k^{(i)} = \boldsymbol{x}_k^{(m)*}, \text{ wobei gilt} \tag{4.76}$$

$$m = \underset{m}{\operatorname{argmin}}\{c_k^{(m)} - s_k^{(i)} : c_k^{(m)} > s_k^{(i)}\}. \tag{4.77}$$

Die entsprechenden Partikelgewichte $\bar{w}_k^{(i)}$ sind uniform

$$\bar{w}_k^{(i)} = \frac{1}{M}. \tag{4.78}$$

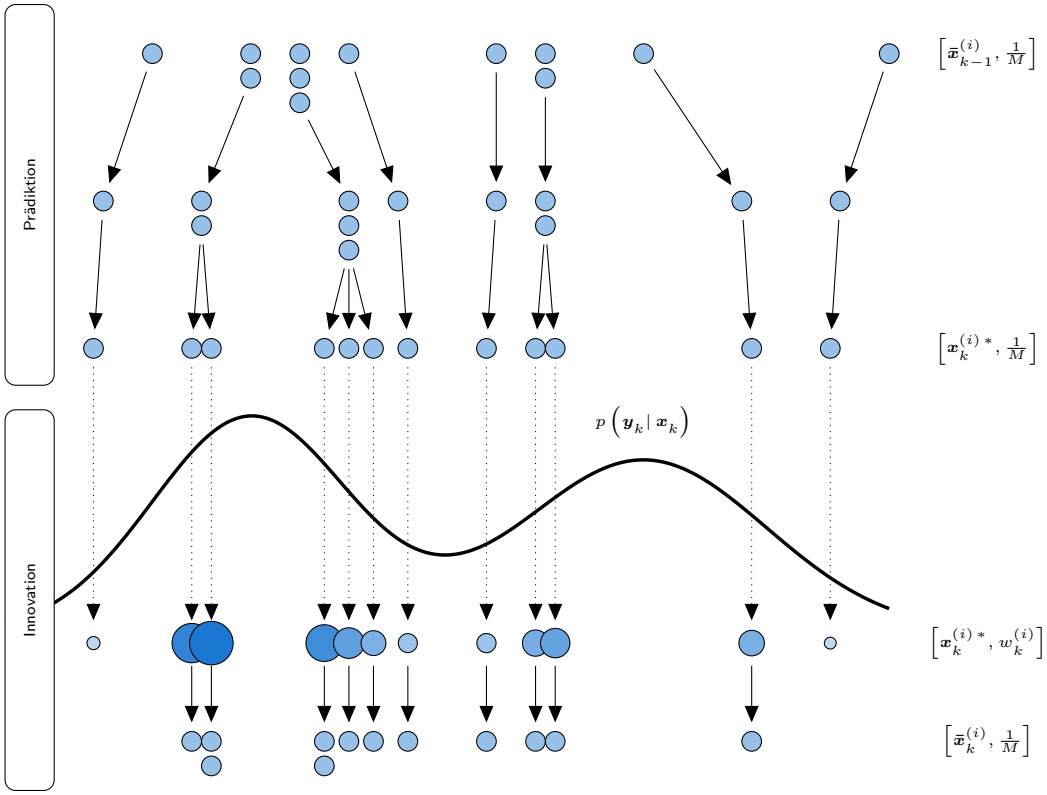


Abbildung 4.3: Rekursionsschritt eines SIR-Partikelfilters.

Abbildung 4.3 zeigt ein graphisches Beispiel eines Rekursionsschritts des SIR-Partikelfilters. Die Partikel durchlaufen zunächst die Filter-Prädiktion entsprechend der Gleichung (4.67) und werden dabei verrauscht. Die Filter-Innovation bestimmt

zunächst die wesentlichen Gewichte $w_k^{(i)}$ und führt anschließend das Systematic Resampling durch.

Auch wenn die Repräsentation der a-priori und a-posteriori WDF im PF nur eine Näherung darstellt, lassen sich mit dieser nicht-parametrischen Darstellung weitaus komplexere Verteilungen beschreiben, als dies bei den parametrischen Darstellungen des KF, EKF oder UKF der Fall ist. Die Anzahl der verwendeten Partikel entscheidet über die Genauigkeit dieser Näherung. In der Realität muss daher zwischen gewünschter Genauigkeit und Rechenbedarf abgewogen werden. Typischerweise sind Partikelmengen in der Größenordnung von $M = 500$ bis 1000 ausreichend. Dabei wertet das PF allerdings sowohl das System- als auch das Messmodell M-mal aus. Aus diesem Grund finden PF erst in den letzten Jahren vermehrt Verwendung, da zuvor die benötigten Rechenressourcen häufig nicht vorhanden waren. Auf Grund der Partikelgewichtung muss beim PF – im Gegensatz zu KF, EKF und UKF – kein Residuum berechnet werden. Anstelle der z.T. aufwändigen Transformation des prädizierten Zustands in den Messraum tritt eine Gewichtung, die besagt, wie gut die aktuelle Messung den Zustand der einzelnen Partikel erklärt.

4.6.4 Extraktion der Zustandsschätzung

Das Ergebnis der Zustandsschätzung mit dem SIR-Partikelfilter ist die Approximation der a-posteriori WDF $p\left(\boldsymbol{x}_k|\, \boldsymbol{y}_{1:k}, \boldsymbol{u}_{1:k-1}\right)$. Allerdings interessiert in der Praxis häufig ein spezieller Wert, beispielsweise die Position und Geschwindigkeit eines vorausfahrenden Fahrzeugs. Ein möglicher solcher Wert ist das Maximum der a-posteriori WDF. Das Maximum a-posteriori ergibt sich aus

$$\boldsymbol{x}_k^{\mathrm{MAP}} = \underset{\boldsymbol{x}_k}{\operatorname{argmax}}\{p\left(\boldsymbol{x}_k|\, \boldsymbol{y}_{1:k}, \boldsymbol{u}_{1:k-1}\right)\} \tag{4.79}$$

$$\approx \underset{\boldsymbol{x}_k}{\operatorname{argmax}}\{\sum_{i=1}^{M} w_k^{(i)} \delta\left(\boldsymbol{x}_k - \boldsymbol{x}_k^{(i)}\right)\}. \tag{4.80}$$

Wie in Gleichung (4.80) zu sehen ist, nimmt $\boldsymbol{x}_k^{\mathrm{MAP}}$ immer den Wert eines der Partikel an. Damit entspricht $\boldsymbol{x}_k^{\mathrm{MAP}}$ nur dann dem wahren Maximum a-posteriori, wenn zufälligerweise auch eines der Partikel genau auf dem Maximum der WDF liegt. Je geringer die Anzahl der Partikel ist, desto unwahrscheinlicher ist dieser Fall. Damit hängt der Fehler von $\boldsymbol{x}_k^{\mathrm{MAP}}$ stark von der Anzahl der Partikel M ab.

Als Alternative dazu bietet es sich bei unimodalen Verteilungen an, den Erwartungswert von $\boldsymbol{x}_k$ als extrahierten Zustand zu verwenden. Der Erwartungswert entspricht dem Minimum Mean Squared Error (MMSE)

$$\boldsymbol{x}_k^{\text{MMSE}} = \mathbb{E}\left(\boldsymbol{x}_k | \, \boldsymbol{y}_{1:k}, \boldsymbol{u}_{1:k-1}\right) = \int \boldsymbol{x}_k p\left(\boldsymbol{x}_k | \, \boldsymbol{y}_{1:k}, \boldsymbol{u}_{1:k-1}\right) \, \mathrm{d}\boldsymbol{x}_k. \tag{4.81}$$

$\boldsymbol{x}_k^{\text{MMSE}}$ lässt sich ebenfalls mit Hilfe der Methode der wesentlichen Stichproben approximieren

$$\boldsymbol{x}_k^{\text{MMSE}} \approx \sum_{i=1}^{M} w_k^{(i)} \, \boldsymbol{x}_k^{(i)*} \tag{4.82}$$

und ist damit aus den Partikeln nach jeder Iteration des Filters leicht berechenbar. Da im PF keinerlei Annahmen an die Modalität der a-posteriori WDF getroffen werden, ist es möglich, dass $\boldsymbol{x}_k^{\text{MMSE}}$ bei multimodalen Verteilungen in einem Bereich niedriger Dichte liegt, beispielsweise zwischen zwei Maxima. Damit wäre $\boldsymbol{x}_k^{\text{MMSE}}$ eine schlechte Schätzung.

Unter der Annahme, dass die Partikel mit den höchsten Gewichten nahe am gleichen Maximum liegen, ergibt sich eine weitere Möglichkeit. Dazu werden die Partikel absteigend entsprechend ihrer Gewichte sortiert und anschließend der Erwartungswert nur anhand der ersten N stärksten Gewichte berechnet

$$\boldsymbol{x}_k^{\text{MMSE-}p} = \sum_{i=1}^{N} \frac{w_k^{(i)}}{\sum_{j=1}^{N} w_k^{(j)}} \, \boldsymbol{x}_k^{(i)*}, \text{ wobei} \tag{4.83}$$

$$0 < p \leq 1, \tag{4.84}$$

$$N = \lfloor p \cdot M \rfloor \text{ und} \tag{4.85}$$

$$\forall i : w_k^{(i)} > w_k^{(i+1)}. \tag{4.86}$$

p beschreibt dabei den Prozentsatz der am höchsten gewichteten Partikel, aus denen der Erwartungswert bestimmt wird. Ein typischer Wert ist $p = 0{,}1$. Daher verwendet diese Arbeit $\boldsymbol{x}_k^{\text{MAP}} = \boldsymbol{x}_k^{\text{MMSE-0,1}}$. Wie bereits beschrieben bietet sich MMSE nur für Anwendungen an, in denen eine unimodale Verteilung angenommen wird. Darüber hinaus sind weitere Formen der Berechnung möglich, beispielsweise durch Klusterbildung, werden in dieser Arbeit allerdings nicht weiter untersucht.

5 Sensorkalibrierung

5.1 Kameramodellierung

Eine Kamera erzeugt zweidimensionale Bilder der dreidimensionalen Welt. Dabei werden Raumpunkte durch das Objektiv einer Kamera auf einen lichtempfindlichen Bildsensor projiziert. Diese Abbildung wird mathematisch als Perspektivische Projektion modelliert.

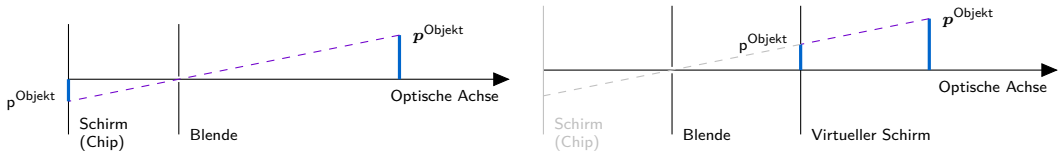
Allerdings kommt es gerade bei kostengünstigen Objektiven zu Bildverzerrungen. Gerade Linien werden dadurch im Bild nicht originalgetreu wiedergegeben. Dieser Effekt wird durch die zusätzliche Modellierung einer Linsenverzeichnung korrigiert. Abschnitte 5.1.1 und 5.1.2 beschreiben das Kameramodell mit Linsenverzeichnung, welches in dieser Arbeit für jeden Kameratyp verwendet wird.

5.1.1 Perspektivische Projektion

Das folgende Kapitel beschreibt die Perspektivische Projektion von Raumpunkten in ein Kamerabild. Eine detailliertere Darstellung ist in Hartley und Zisserman [2003] und Faugeras et al. [2001] zu finden. Neben den Grundlagen führt dieses Kapitel auch in die Notation und die wesentlichen Koordinatensysteme ein.

Die Perspektivische Projektion basiert auf dem sog. Lochkameramodell. Abbildung 5.1a zeigt, wie dabei dreidimensionale Raumpunkte durch das Loch einer Blende auf einen dahinter liegenden Schirm projiziert werden. Dadurch steht das Bild auf dem Kopf, genauer gesagt, findet eine Rotation des Bildes um die optische Achse statt.

Dieses Prinzip lässt sich mathematisch durch homogene Koordinaten beschreiben. Um den Rotationseffekt rückgängig zu machen, wird ein virtueller Schirm, wie in Abb. 5.1b, vor der Blende positioniert. Durch die Interpretation dieses virtuellen



(a) Projektion auf realen Schirm **(b)** Projektion auf virtuellen Schirm

Abbildung 5.1: Projektion nach dem Lochkameramodell

Schirms als Bildsensor einer Kamera lässt sich das in Abb. 5.2 gezeigte Modell entwickeln. Das Modell zeigt die xz-Ebene der perspektivischen Projektion. Der Bildsensor liegt virtuell, um die Bildweite[1] b entfernt, vor dem optischen Zentrum, d.h. dem Koordinatenursprung. Der dreidimensionale Raumpunkt $\boldsymbol{p}$ erzeugt auf dem Bildsensor den Bildpunkt mit der Vertikalkoordinate v.

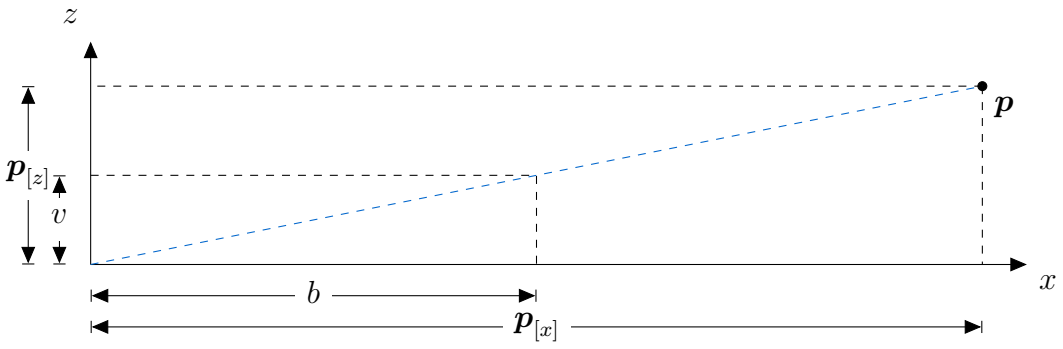


Abbildung 5.2: Symbolische Beschreibung des Lochkameramodells in der xz-Ebene

Über den Strahlensatz lässt sich die folgende Beziehung herstellen

$$\frac{\boldsymbol{p}_{[x]}}{\boldsymbol{p}_{[z]}} = \frac{b}{v} \tag{5.1}$$

$$\Rightarrow v = b\frac{\boldsymbol{p}_{[z]}}{\boldsymbol{p}_{[x]}}. \tag{5.2}$$

Analog dazu gilt für die xy-Ebene

$$u = b\frac{\boldsymbol{p}_{[y]}}{\boldsymbol{p}_{[x]}}. \tag{5.3}$$

[1] In der Literatur werden Bildweite b und Brennweite f häufig synonym gebraucht, auch wenn dies nur für unendlich weit entfernte Objekte zutrifft.

Dabei ist zu bemerken, dass die Koordinaten u und v in dieser Form noch in der Einheit [m] statt in [pix] angegeben sind. Zur Umrechnung in Pixelkoordinaten $\bar{\mathsf{u}}$ und $\bar{\mathsf{v}}$ bedarf es einer Skalierung mit $k_{[y]}$ bzw. $k_{[z]}$. Es folgt daher

$$\bar{\mathsf{u}} = u \cdot k_{[y]} = b \cdot k_{[y]} \frac{\boldsymbol{p}_{[y]}}{\boldsymbol{p}_{[x]}} \text{ und} \tag{5.4}$$

$$\bar{\mathsf{v}} = v \cdot k_{[z]} = b \cdot k_{[z]} \frac{\boldsymbol{p}_{[z]}}{\boldsymbol{p}_{[x]}}. \tag{5.5}$$

Die beiden Skalierungsfaktoren $k_{[y]}$ und $k_{[z]}$ sind konstant und abhängig vom verwendeten Bildsensor. Für das letztlich zu verarbeitende Bild ist es jedoch unerheblich, ob ein kleiner Bildsensor mit geringer Bildweite oder ein größerer Bildsensor mit entsprechend größerer Bildweite verwendet wird, solange das Produkt aus Bildweite und Skalierungsfaktor identisch ist. Eine sensorunabhängige Beschreibung ergibt sich damit durch das Zusammenfassen von b mit den Skalierungsfaktoren $k_{[y]}$ und $k_{[z]}$:

$$\bar{\mathsf{u}} = f_{[\mathsf{u}]} \frac{\boldsymbol{p}_{[y]}}{\boldsymbol{p}_{[x]}} = \bar{\mathsf{y}} \tag{5.6}$$

$$\bar{\mathsf{v}} = f_{[\mathsf{v}]} \frac{\boldsymbol{p}_{[z]}}{\boldsymbol{p}_{[x]}} = \bar{\mathsf{z}} \text{ mit} \tag{5.7}$$

$$f_{[\mathsf{u}]} = b \cdot k_{[y]} \tag{5.8}$$

$$f_{[\mathsf{v}]} = b \cdot k_{[z]} \tag{5.9}$$

Durch Verwendung von homogenen Koordinaten lassen sich die beschriebenen Operationen in Matrixform schreiben:

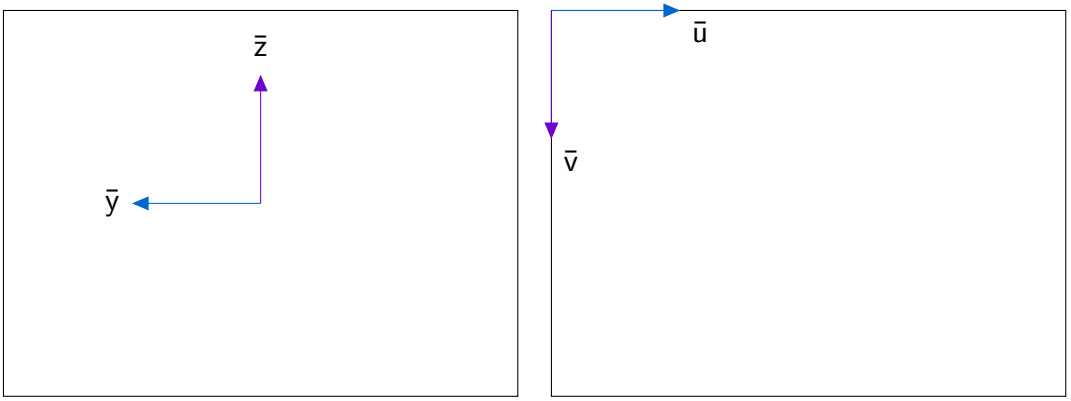
$$\bar{\bar{\mathsf{p}}}_{\boxdot} = \begin{pmatrix} \bar{\bar{\mathsf{p}}}_{[\mathsf{u}]\,\boxdot} \\ \bar{\bar{\mathsf{p}}}_{[\mathsf{v}]\,\boxdot} \\ \bar{\bar{\mathsf{p}}}_{[\mathsf{w}]\,\boxdot} \end{pmatrix} = \begin{bmatrix} 0 & f_{[\mathsf{u}]} & 0 & 0 \\ 0 & 0 & f_{[\mathsf{v}]} & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{pmatrix} \boldsymbol{p}_{[x]} \\ \boldsymbol{p}_{[y]} \\ \boldsymbol{p}_{[z]} \\ 1 \end{pmatrix} \tag{5.10}$$

Damit liegt der Ursprung des Bildkoordinatensystems, wie in Abb. 5.3a dargestellt, in der Mitte des Bildes $\boxdot$. In vielen Fällen intuitiver ist hingegen das in Abb. 5.3b gezeigte Koordinatensystem. Dazu wird das Koordinatensystem zunächst rotiert und anschließend verschoben. Da das optische Zentrum der Projektion ohnehin nicht immer in der Bildmitte liegt, wird eine sog. Hauptpunktverschiebung $(c_{[\mathsf{u}]}, c_{[\mathsf{v}]})^T$ in

[pix] eingeführt, die den Koordinatenursprung vom optischen Zentrum aus in die linke obere Ecke des Bildes $\ulcorner$ verschiebt:

$$\tilde{\bar{\mathrm{p}}}_{\ulcorner} = \begin{pmatrix} \tilde{\bar{\mathrm{p}}}_{[\mathrm{u}]\,\ulcorner} \\ \tilde{\bar{\mathrm{p}}}_{[\mathrm{v}]\,\ulcorner} \\ \tilde{\bar{\mathrm{p}}}_{[\mathrm{w}]\,\ulcorner} \end{pmatrix} = \begin{bmatrix} 1 & 0 & c_{[\mathrm{u}]} \\ 0 & 1 & c_{[\mathrm{v}]} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & f_{[\mathrm{u}]} & 0 & 0 \\ 0 & 0 & f_{[\mathrm{v}]} & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{pmatrix} \boldsymbol{p}_{[x]} \\ \boldsymbol{p}_{[y]} \\ \boldsymbol{p}_{[z]} \\ 1 \end{pmatrix} \tag{5.11}$$

$$\Rightarrow \tilde{\bar{\mathrm{p}}}_{\ulcorner} = \begin{pmatrix} \tilde{\bar{\mathrm{p}}}_{[\mathrm{u}]\,\ulcorner} \\ \tilde{\bar{\mathrm{p}}}_{[\mathrm{v}]\,\ulcorner} \\ \tilde{\bar{\mathrm{p}}}_{[\mathrm{w}]\,\ulcorner} \end{pmatrix} = \begin{bmatrix} c_{[\mathrm{u}]} & -f_{[\mathrm{u}]} & 0 & 0 \\ c_{[\mathrm{v}]} & 0 & -f_{[\mathrm{v}]} & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{pmatrix} \boldsymbol{p}_{[x]} \\ \boldsymbol{p}_{[y]} \\ \boldsymbol{p}_{[z]} \\ 1 \end{pmatrix} \tag{5.12}$$



(a)
Kartesisches Koordinatensystem in der Mitte des Bildes $\mathbf{S}_{\square}$ nach Gleichung (5.10)

(b)
Kartesisches Koordinatensystem in der linken oberen Ecke des Bildes $\mathbf{S}_{\ulcorner}$ nach Gleichung (5.12)

Abbildung 5.3: Vergleich verschiedener Bildkoordinatensysteme

Die Pixelkoordinaten $\bar{\mathrm{p}}_{\ulcorner}$ der Perspektivischen Projektion von $\boldsymbol{p}$ ergeben sich nach Division durch $\tilde{\bar{\mathrm{p}}}_{[\mathrm{w}]\,\ulcorner}$ zu

$$\bar{\mathrm{p}}_{\ulcorner} = \begin{pmatrix} \tilde{\bar{\mathrm{p}}}_{[\mathrm{u}]\,\ulcorner} / \tilde{\bar{\mathrm{p}}}_{[\mathrm{w}]\,\ulcorner} \\ \tilde{\bar{\mathrm{p}}}_{[\mathrm{v}]\,\ulcorner} / \tilde{\bar{\mathrm{p}}}_{[\mathrm{w}]\,\ulcorner} \\ 1 \end{pmatrix} = \begin{pmatrix} \bar{\mathrm{p}}_{[\mathrm{u}]\,\ulcorner} \\ \bar{\mathrm{p}}_{[\mathrm{v}]\,\ulcorner} \\ 1 \end{pmatrix} = \begin{pmatrix} \bar{\mathrm{u}} \\ \bar{\mathrm{v}} \\ 1 \end{pmatrix}. \tag{5.13}$$

Die vier Parameter $f_{[\mathrm{u}]}$, $f_{[\mathrm{v}]}$, $c_{[\mathrm{u}]}$ und $c_{[\mathrm{v}]}$ sind die sog. intrinsischen Kameraparameter. Die Literatur beschreibt häufig einen weiteren Parameter, der die Schiefe, *engl. skew*, der Pixel beschreibt. Bei den in dieser Dissertation beschriebenen Kameras, sowie bei fast allen modernen Kameras heutzutage, kommt diese Schiefe allerdings nicht vor, weshalb sie in dieser Arbeit nicht weiter behandelt wird.

5.1.2 Linsenverzeichnung

Wie eingangs erwähnt verursacht die Verwendung von kostengünstigen bzw. allgemein Weitwinkelobjektiven sog. Linsenverzeichnungen. Ein Beispiel für die Verzeichnung und ein entsprechendes verzeichnungsfreies Bild sind in Abb. 5.4 dargestellt.

(a) Kamerabild mit Verzeichnung **(b)** Kamerabild ohne Verzeichnung

Abbildung 5.4: Vergleich zwischen Bild mit und ohne Linsenverzeichnung

Die Linsenverzeichnung $\Delta_{\ulcorner}^{\text{Linse}}$ wurde bereits von Brown [1971] und Fryer und Brown [1986] untersucht. Demnach lassen sich die auftretenden Verzeichnungen in radial-symmetrische $\Delta_{[\text{radial}]\,\ulcorner}^{\text{Linse}}$ sowie radial-asymetrische und tangentiale Verzeichnung $\Delta_{[\text{tangential}]\,\ulcorner}^{\text{Linse}}$ unterscheiden. Beide Verzeichnungsformen werden dabei additiv mit der Perspektivischen Projektion verknüpft. Daher gilt für die korrigierten Pixelkoordinaten $\mathsf{p}_{\ulcorner}$ des Raumpunktes $\boldsymbol{p}$

$$\mathsf{p}_{\ulcorner} = \bar{\mathsf{p}}_{\ulcorner} + \Delta_{[\text{radial}]\,\ulcorner}^{\text{Linse}} + \Delta_{[\text{tangential}]\,\ulcorner}^{\text{Linse}} \tag{5.14}$$

$$\Delta_{[\text{radial}]\,\ulcorner}^{\text{Linse}} = \begin{pmatrix} \bar{\mathsf{p}}_{[\mathsf{u}]\,\ulcorner} - c_{[\mathsf{u}]} \\ \bar{\mathsf{p}}_{[\mathsf{v}]\,\ulcorner} - c_{[\mathsf{v}]} \\ 0 \end{pmatrix} \cdot \left(\kappa_1\, r_{\ulcorner}^2 + \kappa_2\, r_{\ulcorner}^4 + \kappa_3\, r_{\ulcorner}^6 \right) \tag{5.15}$$

$$\Delta_{[\text{tangential}]\,\ulcorner}^{\text{Linse}} = \begin{pmatrix} f_{[\mathsf{u}]} \left(2\tau_1 \left(\frac{\bar{\mathsf{p}}_{[\mathsf{u}]\,\ulcorner} - c_{[\mathsf{u}]}}{f_{[\mathsf{u}]}} \right) \left(\frac{\bar{\mathsf{p}}_{[\mathsf{v}]\,\ulcorner} - c_{[\mathsf{v}]}}{f_{[\mathsf{v}]}} \right) + \tau_2 \left(r_{\ulcorner}^2 + 2 \left(\frac{\bar{\mathsf{p}}_{[\mathsf{u}]\,\ulcorner} - c_{[\mathsf{u}]}}{f_{[\mathsf{u}]}} \right)^2 \right) \right) \\ f_{[\mathsf{v}]} \left(\tau_1 \left(r_{\ulcorner}^2 + 2 \left(\frac{\bar{\mathsf{p}}_{[\mathsf{v}]\,\ulcorner} - c_{[\mathsf{v}]}}{f_{[\mathsf{v}]}} \right)^2 \right) + 2\tau_2 \left(\frac{\bar{\mathsf{p}}_{[\mathsf{u}]\,\ulcorner} - c_{[\mathsf{u}]}}{f_{[\mathsf{u}]}} \right) \left(\frac{\bar{\mathsf{p}}_{[\mathsf{v}]\,\ulcorner} - c_{[\mathsf{v}]}}{f_{[\mathsf{v}]}} \right) \right) \\ 0 \end{pmatrix} \tag{5.16}$$

$$r_{\ulcorner}^2 = \left(\frac{\bar{\mathsf{p}}_{[\mathsf{u}]\,\ulcorner} - c_{[\mathsf{u}]}}{f_{[\mathsf{u}]}} \right)^2 + \left(\frac{\bar{\mathsf{p}}_{[\mathsf{v}]\,\ulcorner} - c_{[\mathsf{v}]}}{f_{[\mathsf{v}]}} \right)^2 . \tag{5.17}$$

Die Linsenverzeichnung $\Delta_{\ulcorner}^{\text{Linse}}$ verschiebt die Pixelposition radial bzw. tangential bzgl. des Kamerahauptpunktes. Dies wird auch in den Gleichungen (5.15) bis (5.17) deutlich, in denen von $\bar{\mathsf{p}}_{[\mathsf{u}]\,\ulcorner}$ bzw. $\bar{\mathsf{p}}_{[\mathsf{v}]\,\ulcorner}$ jeweils $c_{[\mathsf{u}]}$ bzw. $c_{[\mathsf{v}]}$ subtrahiert wird. Es stellt sich daher die Frage, ob die Linsenverzeichnung nicht besser vor der Transformation von $\mathbf{S}_{\square}$ in $\mathbf{S}_{\ulcorner}$ durchgeführt werden sollte. Außerdem fällt in den Gleichungen (5.16) und (5.17) auf, dass die genannten Differenzen durch die jeweilige Brennweite dividiert werden. Die Brennweiten lassen sich den Gleichungen (5.6) und (5.7) entnehmen, vereinfacht gesagt der Umwandlung von metrischen Bildkoordinaten in Pixelkoordinaten. Analog zu Zhang [1999] wird Gleichung (5.14) derart umformuliert, dass die Verzeichnung bereits vor der Umwandlung in Pixelkoordinaten durchgeführt wird.

Durch Berechnung von $\bar{u}$ und $\bar{v}$ aus

$$\bar{u} = \frac{\boldsymbol{p}_{[y]}}{\boldsymbol{p}_{[x]}} \tag{5.18}$$

$$\bar{v} = \frac{\boldsymbol{p}_{[z]}}{\boldsymbol{p}_{[x]}} \tag{5.19}$$

ergeben sich die metrischen Bildkoordinaten des Punktes $\boldsymbol{p}$ einer virtuellen Kamera mit Bildweite $b = 1$. Anstelle von Gleichung (5.14) ergibt sich damit

$$\mathsf{p}_{\ulcorner} = {}^{\ulcorner}\mathbf{H}_{\square} \cdot \left(\begin{pmatrix} 1 \\ \bar{u} \\ \bar{v} \end{pmatrix} + \Delta^{\text{Linse}}_{[\text{radial}]\,\square} + \Delta^{\text{Linse}}_{[\text{tangential}]\,\square} \right) \text{ mit} \tag{5.20}$$

$$\begin{aligned} {}^{\ulcorner}\mathbf{H}_{\square} &= \begin{bmatrix} 1 & 0 & c_{[\mathsf{u}]} \\ 0 & 1 & c_{[\mathsf{v}]} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & f_{[\mathsf{u}]} & 0 \\ 0 & 0 & f_{[\mathsf{v}]} \\ 1 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} c_{[\mathsf{u}]} & -f_{[\mathsf{u}]} & 0 \\ c_{[\mathsf{v}]} & 0 & -f_{[\mathsf{v}]} \\ 1 & 0 & 0 \end{bmatrix}, \end{aligned} \tag{5.21}$$

$$\Delta^{\text{Linse}}_{[\text{radial}]\,\square} = \begin{pmatrix} 0 \\ \bar{u} \\ \bar{v} \end{pmatrix} \cdot \left(\kappa_1\, r_{\square}^2 + \kappa_2\, r_{\square}^4 + \kappa_3\, r_{\square}^6 \right) \text{ und} \tag{5.22}$$

$$\Delta^{\text{Linse}}_{[\text{tangential}]\,\square} = \begin{pmatrix} 0 \\ -2\tau_1\, \bar{u}\, \bar{v} - \tau_2 \left(r_{\square}^2 + 2\, \bar{u}^2 \right) \\ -\tau_1 \left(r_{\square}^2 + 2\, \bar{v}^2 \right) - 2\tau_2\, \bar{u}\, \bar{v} \end{pmatrix} \tag{5.23}$$

$$r_{\square}^2 = \bar{u}^2 + \bar{v}^2, \tag{5.24}$$

wobei ${}^{\lceil}\mathbf{H}_{\boxdot}$ die homogene Transformationsmatrix beschreibt, die metrische Bildkoordinaten bzgl. der Bildmitte einer virtuellen Kamera mit Bildweite $b = 1$ in Pixelkoordinaten bzgl. der linken oberen Bildecke transformiert. Weiterhin ist zu bemerken, dass die Gleichungen (5.14) und (5.20) äquivalent sind.

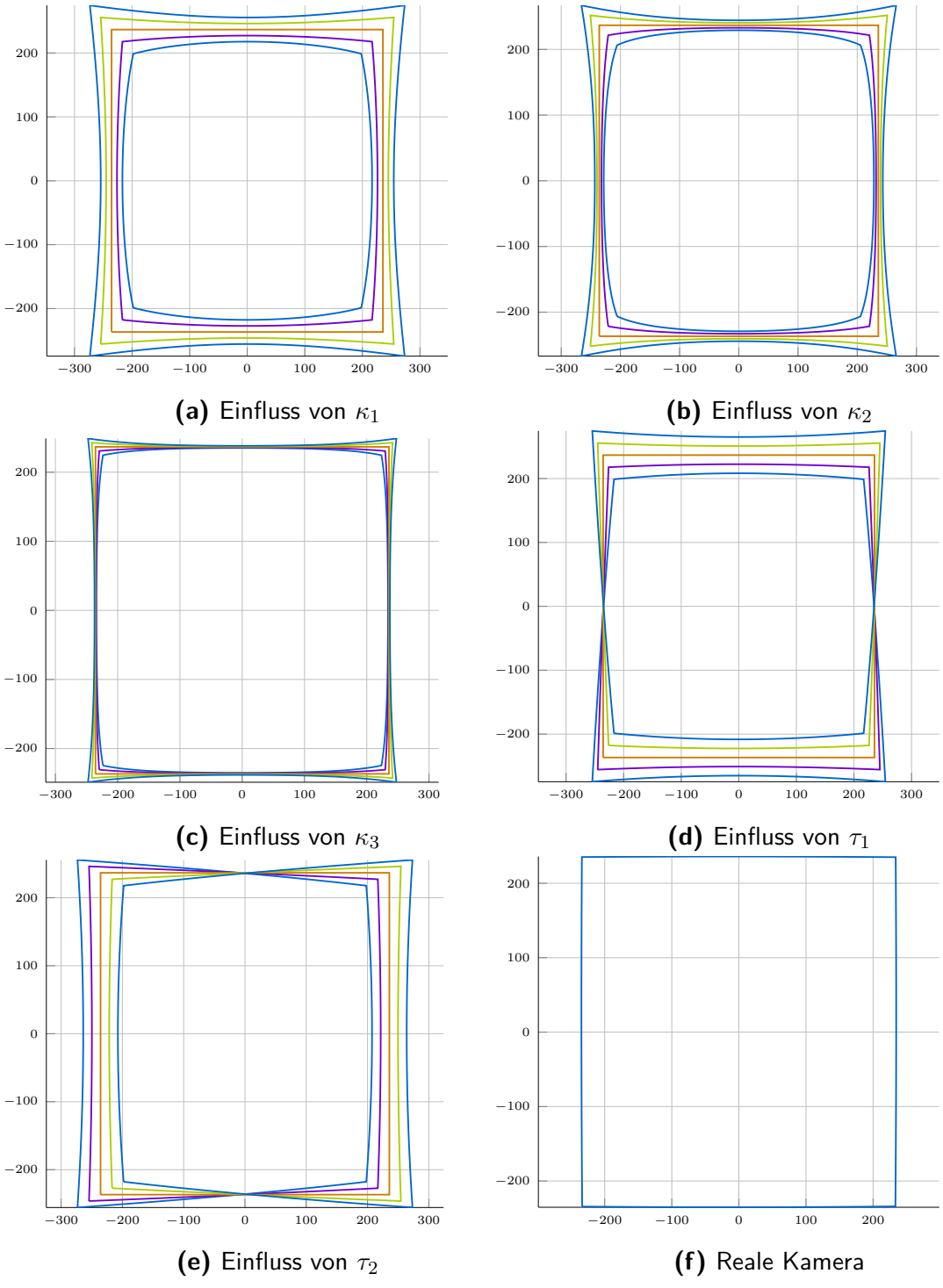
Die Berücksichtigung der Linsenverzeichnung erweitert die aus Abschnitt 5.1.1 bekannten intrinsischen Kameraparameter um die fünf Verzeichnungsparameter κ_1, κ_2, κ_3, τ_1 und τ_2. Das intrinsische Kameramodell besteht daher aus insgesamt neun Parametern. Die intrinsischen Kalibrierungsparameter der in dieser Arbeit verwendeten Kameras stammen alle aus Berechnungen mit Hilfe der *Camera Calibration Toolbox for Matlab* von Bouguet [2003].

Der Einfluss der fünf Linsenverzeichnungsparameter κ_1, κ_2, κ_3, τ_1 und τ_2 ist in Abb. 5.5 dargestellt. Dabei wird ein mittig vor einer virtuellen Kamera platziertes Quadrat mit Hilfe von Gleichung (5.20) abgebildet. Das Quadrat hat eine Kantenlänge von 2 m und befindet sich im Abstand von 5 m vor der Kamera. Die Brennweiten sind hier analog zu einer realen, kalibrierten Kamera mit $f_{[u]} = 1178{,}16$ pix und $f_{[v]} = 1183{,}38$ pix gewählt. Für jede der Abb. 5.5a bis 5.5e wird nur jeweils ein Verzeichnungsparameter verändert und die anderen Parameter auf 0 gesetzt. Die Werte der einzelnen Parameter sind in Tabelle 5.1 für Abb. 5.5a bis 5.5c von innen nach außen, für Abb. 5.5d von unten nach oben und Abb. 5.5e von links nach rechts angegeben.

		■	■	■	■	■
Abb. 5.5a	κ_1	−2,0	−1,0	0,0	1,0	2,0
Abb. 5.5b	κ_2	−20,0	−10,0	0,0	10,0	20,0
Abb. 5.5c	κ_3	−100,0	−50	0,0	50,0	100,0
Abb. 5.5d	τ_1	−0,2	−0,1	0,0	0,1	0,2
Abb. 5.5e	τ_2	−0,2	−0,1	0,0	0,1	0,2
Abb. 5.5f		κ_1 −0,120	κ_2 0,184	κ_3 −0,036	τ_1 0,002	τ_2 −0,001

Tabelle 5.1: Verzeichnungsparameter der Abb. 5.5

Anhand von Abb. 5.5a bis 5.5c ist erkennbar, dass über die radialen Verzeichnungsparameter eine kissen- bzw. tonnenförmige Verzerrung modelliert wird. Jedes Pixel wird dabei abhängig von seinem Abstand zum Bildmittelpunkt weiter auf diesen zubewegt ($\kappa_{1,2,3} < 0$) oder von ihm wegbewegt ($\kappa_{1,2,3} > 0$). Der Abstand zum Mittelpunkt geht dabei für κ_1 quadratisch, für κ_2 mit der vierten Potenz und für



(a) Einfluss von κ_1

(b) Einfluss von κ_2

(c) Einfluss von κ_3

(d) Einfluss von τ_1

(e) Einfluss von τ_2

(f) Reale Kamera

Abbildung 5.5: Einfluss der verschiedenen Parameter der Linsenverzeichnung.

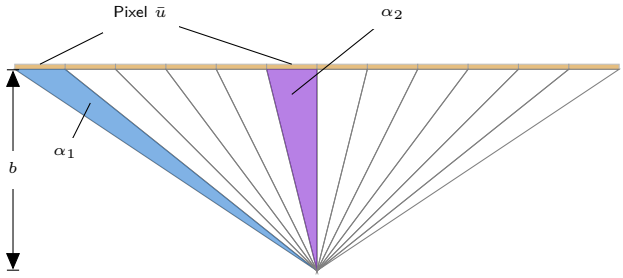
κ_3 mit der sechsten Potenz ein (vgl. Gleichung (5.22)). Der Grund dafür, dass im vorliegenden Beispiel die beiden Verzeichnungsparameter κ_2 und κ_3 einen relativ geringen Einfluss ausüben, liegt darin, dass für alle projizierten Punkte des Quadrats der quadrierte Radius $r^2_{\square}$ und damit auch $r^4_{\square}$ und $r^6_{\square}$ kleiner 0 sind. Erst für Punkte mit einem Winkel $\alpha > 45°$ zur optischen Achse werden $r^2_{\square}$, $r^4_{\square}$ und $r^6_{\square}$ größer 0 und damit auch der Einfluss von κ_2 und κ_3.

Zum Vergleich zeigt Abb. 5.5f die Abbildung des gleichen Quadrats gemäß den Verzeichnunsparametern einer Basler Ace acA1300-30gc Kamera aus Munich Cognitive Autonomous Robot Car 4$^{\text{th}}$ Generation (MuCAR-4), deren Verzeichnungsparameter mit der *Camera Calibration Toolbox for Matlab* [Bouguet, 2003] bestimmt wurden. Die Verzeichnungsparameter κ_1, κ_2, κ_3, τ_1 und τ_2 sind, wie in Tabelle 5.1 zu sehen, vergleichsweise klein. Dies erklärt die kaum erkennbare Tonnenverzeichnung. Trotzdem lohnt sich die Verwendung der fünf zusätzlichen Kalibrierungsparameter, da die Verzeichnung zum Bildrand hin zunimmt und der Chip der hier verwendeten Kamera etwa die dreifache Breite und die doppelte Höhe des hier dargestellten Ausschnitts besitzt. Gleichzeitig ist die Winkelauflösung einer Kamera im Randbereich des Bildes höher. Dies liegt daran, dass der von einem Pixel abgedeckte Öffnungswinkel zum Bildrand hin, aufgrund des tangentialen Zusammenhangs zwischen Blickwinkel und Bildspalte bzw. Bildzeile, kleiner wird (Abb. 5.6). Deshalb sind genaue Messungen gerade im Randbereich des Bildes notwendig. Der horizontale Öffnungswinkel $\alpha\left(\bar{\mathsf{p}}_{[\mathsf{u}]\,\square}\right)$ eines Pixels berechnet sich nach folgender Formel

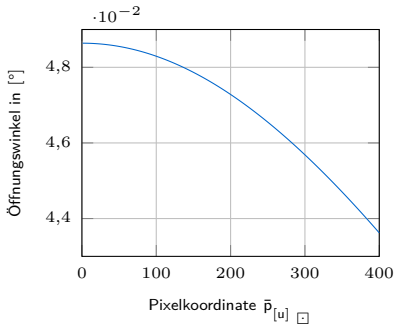
$$\alpha\left(\bar{\mathsf{p}}_{[\mathsf{u}]\,\square}\right) = \tan^{-1}\left(\frac{\bar{\mathsf{p}}_{[\mathsf{u}]\,\square}}{f_{[\mathsf{u}]}}\right) - \tan^{-1}\left(\frac{\bar{\mathsf{p}}_{[\mathsf{u}]\,\square} - 1}{f_{[\mathsf{u}]}}\right). \tag{5.25}$$

Der vertikale Öffnungswinkel eines Pixels in Abhängigkeit von der Bildzeile $\bar{\mathsf{p}}_{[\mathsf{v}]\,\square}$ berechnet sich analog.

In vielen Fällen, beispielsweise bei der Umwandlung von Disparitätenbildern in 3D-Punktwolken bzw. zur Initialisierung eines 3D-Zustands aus Bildinformationen im 4D-Ansatz [Wünsche, 1988], ist die Rückprojektion von Pixeln in 3D-Punkte notwendig. Dazu ist entweder der Abstand des zu projizierenden Bildpunktes notwendig oder anhand von Modellwissen (z.B. Annahme einer planen Bodenebene) notwendig. Ohne Linsenverzeichnung ließe sich dann die Rückprojektion trigonometrisch herleiten und analytisch berechnen. Da sich die inverse Abbildung zur Linsenverzeichnung in Gleichung (5.20) nicht analytisch berechnen lässt, schlagen Heikkilä und Silvén [1997] die folgende iterative Lösung vor.



(a) Schematische Darstellung



(b) Öffnungswinkel in Abhängigkeit von $\bar{p}_{[u]\,\square}$

Abbildung 5.6:
Zusammenhang zwischen Blickwinkel und Bildspalte. Die Winkelauflösung nimmt zum Randbereich des Bildes zu.

Im ersten Schritt findet eine Rücktransformation von $p_{\ulcorner}$ in das bildzentrierte metrische Koordinatensystem $\mathbf{S}_{\square}$ der virtuellen Kamera mit Bildweite $b = 1$ statt:

$$p_{\square,0} = {}^{\square}\mathbf{H}_{\ulcorner} \cdot p_{\ulcorner} = \begin{pmatrix} 1 \\ \bar{u}_0 \\ \bar{v}_0 \end{pmatrix} \text{ mit} \tag{5.26}$$

$${}^{\square}\mathbf{H}_{\ulcorner} = {}^{\ulcorner}\mathbf{H}_{\square}^{-1} = \begin{bmatrix} 0 & 0 & 1 \\ -1/f_{[u]} & 0 & 0 \\ 0 & -1\,f_{[v]} & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & -c_{[u]} \\ 0 & 1 & -c_{[v]} \\ 0 & 0 & 1 \end{bmatrix} \tag{5.27}$$

$$= \begin{bmatrix} 0 & 0 & 1 \\ -1/f_{[u]} & 0 & c_{[u]}/f_{[u]} \\ 0 & -1/f_{[v]} & c_{[v]}/f_{[v]} \end{bmatrix} \tag{5.28}$$

Anschließend werden die beiden Verzeichnungsarten $\Delta^{\mathsf{Linse}}_{[\mathsf{radial}]\ \square,i}$ und $\Delta^{\mathsf{Linse}}_{[\mathsf{tangential}]\ \square,i}$ sowie $\mathsf{p}_{\square,i+1}$ gemäß folgender Formeln iterativ berechnet:

$$r^2_{\square,i} = \bar{u}_i^2 + \bar{v}_i^2 \tag{5.29}$$

$$\Delta^{\mathsf{Linse}}_{[\mathsf{radial}]\ \square,i} = \frac{1}{1 + \kappa_1\, r^2_{\square,i} + \kappa_2\, r^4_{\square,i} + \kappa_3\, r^6_{\square,i}} \tag{5.30}$$

$$\Delta^{\mathsf{Linse}}_{[\mathsf{tangential}]\ \square,i} = \begin{pmatrix} 0 \\ 2\tau_1\, \bar{u}_i\, \bar{v}_i + \tau_2 \left(r^2_{\square,i} + 2\, \bar{u}_i^2 \right) \\ \tau_1 \left(r^2_{\square,i} + 2\, \bar{v}_i^2 \right) + 2\tau_2\, \bar{u}_i\, \bar{v}_i \end{pmatrix} \tag{5.31}$$

$$\mathsf{p}_{\square,i+1} = \left(\mathsf{p}_{\square,i} - \Delta^{\mathsf{Linse}}_{[\mathsf{tangential}]\ \square,i} \right) \cdot \Delta^{\mathsf{Linse}}_{[\mathsf{radial}]\ \square,i} = \begin{pmatrix} 1 \\ \bar{u}_{i+1} \\ \bar{v}_{i+1} \end{pmatrix} \tag{5.32}$$

In der Praxis wird die iterative Näherung typischerweise für 20 Iterationen berechnet. Gleichung (5.32) repräsentiert dabei bereits den Richtungsvektor des zu $\mathsf{p}_{\ulcorner}$ korrespondierenden 3D-Sehstrahls. Ist der Abstand des zu $\mathsf{p}_{\ulcorner}$ korrespondierenden 3D-Punktes bekannt, errechnen sich durch Normalisieren von $\mathsf{p}_{\ulcorner}$ und Multiplizieren mit dem Abstand die entsprechenden 3D-Koordinaten.

Für den weiteren Verlauf der Arbeit werden die Kalibrierungsparameter $c_{[\mathsf{u}]}$, $c_{[\mathsf{v}]}$, $f_{[\mathsf{u}]}$, $f_{[\mathsf{v}]}$, κ_1, κ_2, κ_3, τ_1 und τ_2 als bekannt vorausgesetzt. Zur Vereinfachung der Schreibweisen wird die folgende Funktion zur Projektion von 3D-Punkten bzw. 4D-homogenen Koordinaten in korrigierte Pixelkoordinaten bzgl. der linken oberen Bildecke definiert

$$\mathsf{p}\left(\boldsymbol{p}\right) = \begin{pmatrix} \mathsf{p}_{[\mathsf{u}]\ulcorner} \\ \mathsf{p}_{[\mathsf{v}]\ulcorner} \end{pmatrix} = \begin{pmatrix} \mathsf{u} \\ \mathsf{v} \end{pmatrix}. \tag{5.33}$$

Die inverse Funktion zur Umwandlung korrigierter Pixelkoordinaten in 3D-Punkte bzw. 4D-homogene Koordinaten wird definiert als

$$\mathsf{p}^{-1}\left(\mathsf{p}_{\ulcorner}, d\right) = \boldsymbol{p}, \tag{5.34}$$

wobei d die metrische Tiefe des entsprechenden Bildpunktes repräsentiert.

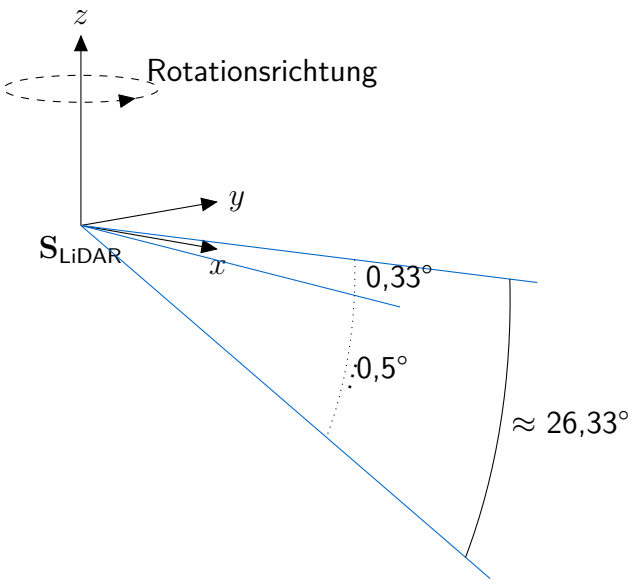


Abbildung 5.7:
Schematische Darstellung des Velodyne HDL-64E Light Detection and Ranging (LiDAR)

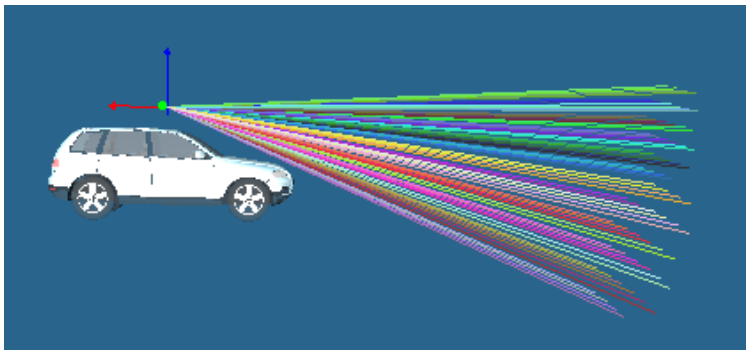
5.2 LiDAR-Modellierung

LiDAR-Sensoren messen Entfernungen entlang eines Laserstrahls nach dem TOF-Prinzip (Time of Flight). Wie in Abschnitt 3.1.3 beschrieben verwendet diese Arbeit die Versionen S1 und S2 des Velodyne HDL-64E. Zunächst findet eine Beschreibung der Version S2 statt, bevor anschließend auf die Unterschiede zu Version S1 eingegangen wird.

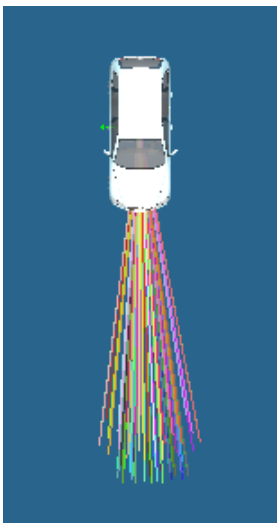
Im speziellen Fall des Velodyne HDL-64E (Abb. 3.3a) besteht der Sensor aus 64 Laserdioden. Abbildung 5.7 zeigt eine schematische Darstellung der Version S2, die einen vertikalen Bereich von etwa 26,33° mit einer vertikalen Winkelauflösung von 0,33° im Bereich der oberen 10,33° und 0,5° im Bereich der unteren 16° abdeckt. Während der Entfernungsmessungen der einzelnen Laser rotieren die Laserdioden um die Vertikalachse mit einstellbarer Geschwindigkeit. Damit bietet der Sensor eine Rundumsicht von 360°. Da jede der 64 Laserdioden der Version S2 mit einer Frequenz von ca. 20 835 Hz Entfernungen misst, beeinflusst die Rotationsgeschwindigkeit gleichzeitig auch die laterale Winkelauflösung des gesamten Sensors. Bei einer Rotationsfrequenz von 10 Hz, wie sie für diese Arbeit eingestellt ist, liefert jeder der 64 Laser etwa 2083 Entfernungsmessungen pro Umdrehung. Damit ergibt sich eine laterale Winkelauflösung von 0,1728° pro Umdrehung.

Der wesentliche Unterschied zur Version S1 liegt in der Ansteuerung der 64 Laserdioden. Im Vergleich zum S2 sind diese beim S1 in einen oberen und einen unteren Block partitioniert. Dabei misst ein einzelner Laser im oberen Block etwa 3125 Entfernungsmessungen pro Umdrehung und erreicht damit eine höhere laterale Winkelauflösung von 0,1152° pro Umdrehung. Die Laser im unteren Block hingegen messen nur etwa 781,25 Entfernungsmessungen pro Umdrehung. Die laterale Winkelauflösung sinkt damit im Vergleich zur Version S2 auf 0,4608° pro Umdrehung. Auch die vertikale Anordnung der 64 Laserdioden unterscheidet sich. In der Version S1 sind die Dioden vertikal in einem Winkel von ca. 0,4° zueinander angeordnet und umfassen einen vertikalen Öffnungswinkel von 26,8°.

Bei der Fertigung und Bestückung der Trägerplatine für die 64 Laserdioden kommt es produktionsbedingt zu leichten Ungenauigkeiten im Gier- und Nickwinkel der einzelnen Dioden. Dadurch unterscheiden sich der vertikale Öffnungswinkel und entsprechend auch die vertikale Auflösung von Sensor zu Sensor. Jeder Laserscanner wird daher vom Hersteller intrinsisch kalibriert und mit einer entsprechenden Kalibierdatei ausgeliefert. Abb. 5.8 visualisiert die intrinsische Kalibrierung eines Velodyne HDL-64E Laserscanners.



(a) Intrinsische Kalibrierung aus der Seitenansicht.



(b) Intrinsische Kalibrierung aus der Vogelperspektive.

Abbildung 5.8:
Intrinsische Kalibrierung eines Velodyne HDL-64E Laserscanners. Jeder der 64 Laserstrahlen ist andersfarbig dargestellt.

Laut Datenblatt sind die gemessenen Distanzen bei Version S2 auf 0,02 m (1-σ Abstand) genau. Damit liegt eine Messung in 99,73 % aller Fälle nicht weiter als

$\pm$0,06 m von der gemessenen Distanz entfernt. Zur korrekten Betrachtung dieses Fehlers bietet sich die Darstellung in Polarkoordinaten, d.h. horizontalen und vertikalen Winkeln sowie Distanz an. Die Messungenauigkeiten liegen damit näherungsweise auf einem Kugelsegment. Allerdings sind die Winkelungenauigkeiten und insbesondere die Entfernungsungenauigkeiten derart klein, dass eine Approximation des Fehlers in kartesischen Koordinaten möglich ist. Daher werden in dieser Arbeit für die gemessenen Punkte kartesische Koordinaten verwendet, da dies die Transformation der Punkte mit Hilfe von homogenen Transformationen ermöglicht. Außerdem wird die Strahlaufweitung in dieser Arbeit außer Acht gelassen. Genauere Untersuchungen dazu sind in Schmid [2012] zu finden.

5.3 Sensorsynchronisation

Werden die Sensordaten mehrerer Sensoren miteinander verrechnet, ist zur korrekten Verarbeitung eine zeitliche Registrierung, d.h. der gemeinsame Bezug auf eine Zeitbasis, notwendig. Dabei wird zwischen synchronen und asynchronen Sensoren unterschieden. In einem asynchronen Sensoraufbau misst jeder Sensor in einem für ihn individuellen Takt, oder auch Ereignis-gesteuert. Beispielsweise zeigen Lynen et al. [2013] ein Fusionsverfahren basierend auf asynchronen Messungen verschiedener Sensoren. Musicki und Evans [2004] beschreiben ebenfalls ein Verfahren zur asynchronen Sensorfusion. Die Grundlage derartiger Verfahren ist immer Modellwissen, welches über die Zeit durch die fortlaufenden Messungen aktualisiert wird.

Dennoch gibt es Verfahren, die synchrone Sensormessungen, d.h. alle Sensoren messen zu einem vorgegebenen Zeitpunkt, voraussetzen. Dazu gehören u.a. die meisten Stereoalgorithmen, wie Blockmatching (BM) [Chen et al., 2001], Semiglobal Matching (SGM) [Hirschmüller, 2005, Hirschmüller, 2008] oder Efficient Large-scale Stereo Matching (ELAS) [Geiger et al., 2010]. Da derartige Verfahren auch in dieser Arbeit entwickelt und verwendet werden, beschreibt Abschnitt 5.3.1 die Synchronisation mehrerer Kameras und Abschnitt 5.3.2 die Synchronisation von Kameras mit einem LiDAR.

5.3.1 Synchronisation mehrerer Kameras

Die Synchronisation der auf beiden Versuchsträgern verbauten Kameras des Typs Allied Vision Technologies GmbH (AVT) Guppy F-036C, Basler Ace acA1300-30gc

und Bumblebee XB3 geschieht über die in Abschnitt 3.3 beschriebene dSPACE MicroAutoBox. Diese sendet die Auslösesignale per Kabelverbindung an die angeschlossenen Kameras und misst gleichzeitig die Belichtungszeit $T^{\text{Belichtung}}$ jeder einzelnen Kamera. Das Echtzeitbetriebssystem der dSPACE MicroAutoBox sorgt dafür, dass die voreingestellte Auslösefrequenz $f^{\text{Auslösen}}$ möglichst exakt eingehalten wird. Die typische Auslösefrequenz im reinen Kamerabetrieb liegt bei 20 Hz.

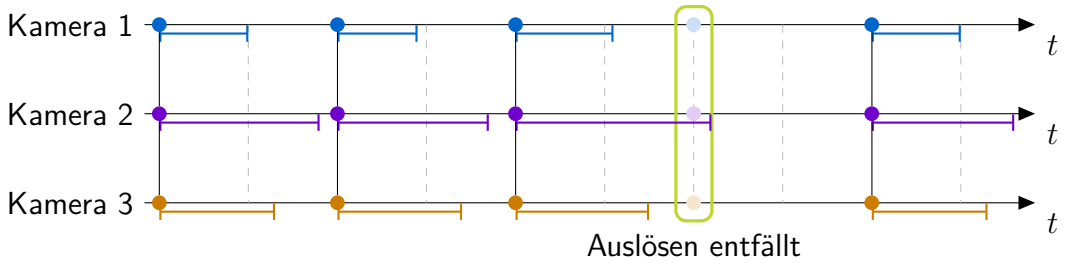


Abbildung 5.9:
Synchronisation mehrerer Kameras unter Berücksichtigung der individuellen Belichtungszeit.

Abbildung 5.9 zeigt eine beispielhafte Sequenz der Synchronisation von drei Kameras mit unterschiedlichen Belichtungszeiten. Die Auslösezeitpunkte sind hierbei durch • dargestellt. Die Belichtungszeit einer jeden Kamera wird durch ⊢⊣ gekennzeichnet. Belichtet, wie in Abb. 5.9 gezeigt, eine der Kameras länger als bis zum nächsten geplanten Auslösezeitpunkt, entfällt dieses Auslösesignal. Auch wenn dadurch einzelne Messzeitpunkte und damit Messungen fehlen, wird dennoch sichergestellt, dass die Aufnahme neuer Bilder immer von allen Kameras zum gleichen Zeitpunkt beginnt.

Als Nächstes stellt sich die Frage, wie die Datenzeitstempel t^{Daten} der jeweiligen Bilder berechnet werden. Dabei gibt es die folgenden Punkte zu beachten. Zum einen findet die Ansteuerung der Kameras auf der dSPACE MicroAutoBox statt, während die Bilder über die jeweiligen Schnittstellen (FireWire bzw. Gigabit Ethernet (GigE)) an den zentralen Fahrzeugrechner übertragen werden. Zum anderen wird ein Kamerabild nicht etwa zu einem Zeitpunkt aufgenommen, sondern das Bild ist das Ergebnis einer Helligkeitsintegration innerhalb der Belichtungszeit $T^{\text{Belichtung}}$. Abbildung 5.10 zeigt schematisch die Bestimmung des Datenzeitstempels.

Zur Klärung der Frage, welches Alter ein Bild hat bzw. zu welchem Zeitpunkt es aufgenommen wurde, gibt es mehrere Möglichkeiten, wie Start oder Ende der Belichtungszeit. Beide Zeitpunkte sind argumentierbar, allerdings sind die im Bild aufgenommenen Effekte eine Akkumulation aller Ereignisse innerhalb des Zeitraums zwischen beiden Zeitpunkten. Ein sich parallel zur Bildebene konstant bewegendes

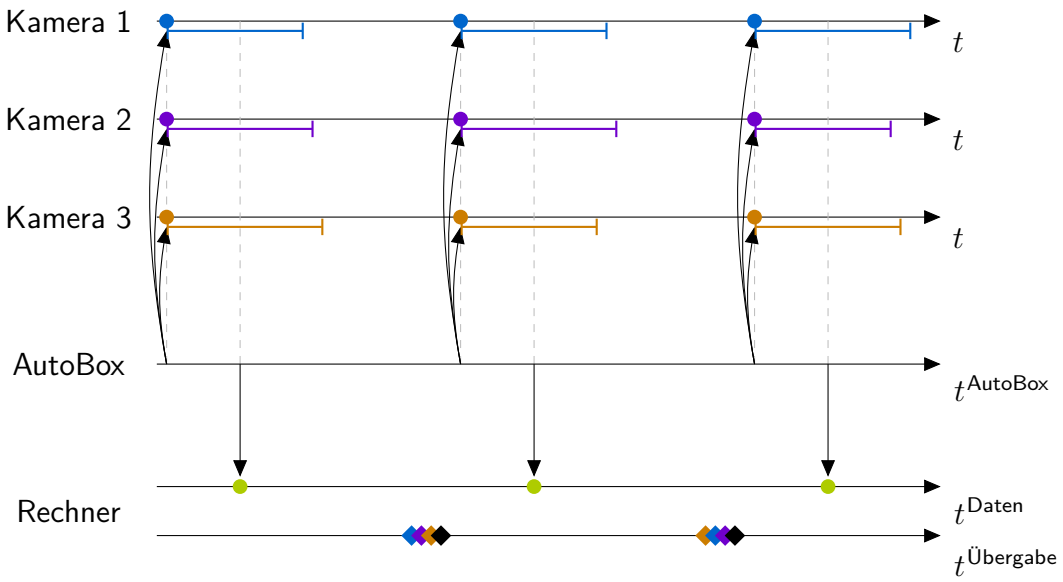


Abbildung 5.10:
Bestimmung des Datenzeitstempels unter Berücksichtigung der individuellen Belichtungszeit mehrerer Kameras.

Objekt würde aufgrund der Bewegungsunschärfe den deutlichsten Bildeindruck für seine Position zum mittleren Belichtungszeitpunkt hinterlassen. Die Bewegungsunschärfe verblasst hin zu der entsprechenden Position zum Belichtungsstart und Belichtungsende. Daher wird der mittlere Belichtungszeitpunkt, d.h. der Zeitpunkt in der Mitte zwischen Belichtungsstart und Belichtungsende dem Bildzeitstempel am gerechtesten. Dieser ist zu Beginn der Belichtung allerdings noch nicht bekannt. Unter der Annahme, dass sich die Belichtungszeit zwischen zwei Aufnahmen der Kamera kaum ändert, wird der Zeitpunkt aus der Belichtungszeit des vorherigen Bildes berechnet. Der Datenzeitstempel t_k^{Daten} eines Bildes ergibt sich damit zu

$$t_k^{\text{Daten}} = t_k^{\text{Auslösen}} + \frac{1}{2}\, T_{k-1}^{\text{Belichtung}}. \tag{5.35}$$

Werden die Bilder aller an der Kameraplattform MarVEye-8 befestigten Kameras aufgenommen, errechnet sich die mittlere Belichtungszeit aus der mittleren Belichtungszeit aller Kameras zur vorherigen Aufnahme. Dies ist möglich, da sich die Belichtungszeiten der Kameras erfahrungsgemäß ohnehin kaum unterscheiden. Andernfalls würde das in Abschnitt 3.4.2 beschriebene Verfahren zum Synchronisieren verschiedener KogMo-RTDB-Objekte (KogniMobil Real Time Data Base) alte und

neue Bilder von unterschiedlichen Kameras womöglich mischen. Für n Kameras ergibt sich damit

$$t_k^{\text{Daten}} = t_k^{\text{Auslösen}} + \frac{1}{2} \cdot \frac{1}{n} \sum_{i=0}^{n-1} T_{k-1}^{(i)\,\text{Belichtung}}. \tag{5.36}$$

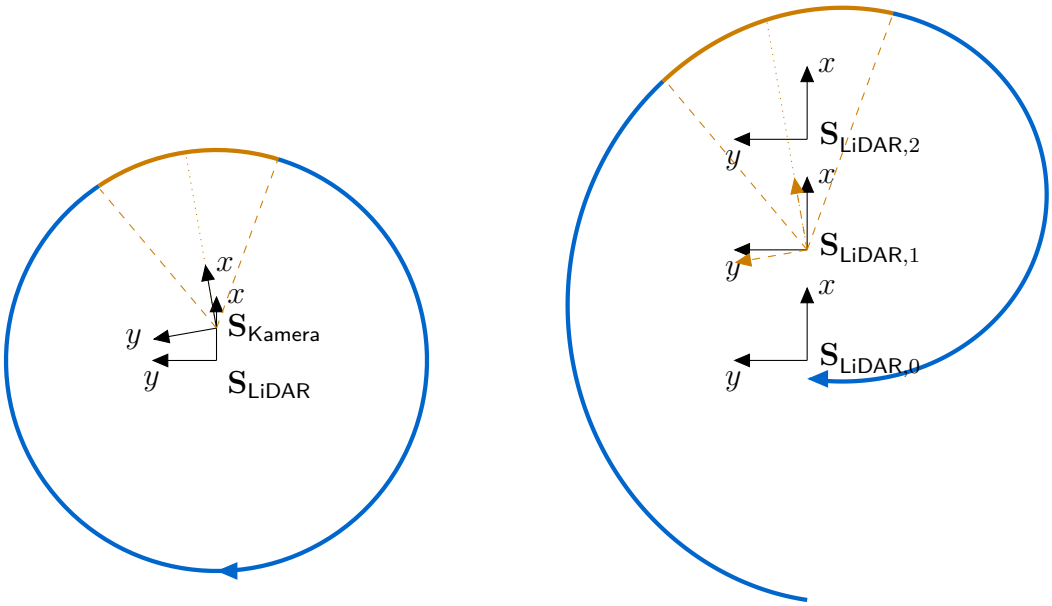
Die dSPACE MicroAutoBox und der zentrale Fahrzeugrechner sind über einen CAN-Bus (Controller Area Network) miteinander verbunden. Zum Zeitpunkt t_k^{Daten} sendet die dSPACE MicroAutoBox über CAN eine Nachricht an den Fahrzeugrechner. Trifft diese Nachricht auf dem Fahrzeugrechner ein, bestimmt dieser die aktuelle Systemzeit und damit den Datenzeitstempel auf dem Fahrzeugrechner, unter dem die Bilder nach ihrem Eintreffen in der KogMo-RTDB abgelegt werden. Das Signal enthält außerdem die beiden Winkel der Kameraplattform, die zur Berechnung der Pose der Kameras notwendig sind. Die Übertragungszeit der Nachricht wird dabei aufgrund ihrer geringen Größe als vernachlässigbar klein angenommen.

Wie die Zeitachse für $t^{\text{Übergabe}}$ in Abb. 5.10 zeigt, werden die Bilder nacheinander in die KogMo-RTDB geschrieben. Dabei kann sich deren Reihenfolge aufgrund leicht unterschiedlicher Belichtungszeiten theoretisch ändern. Damit stellt sich für eine nachgeschaltete Anwendung, die mehrere Bilder verarbeiten möchte, die Frage, welches Kamerabild sie als *auslösend* definiert. Die Reihenfolge des Eintreffens aller Bilder ist nicht vorhersagbar. Allerdings ist der Anwendung, die die Kamerabilder über die externen Schnittstellen abruft, aufgrund der Fahrzeugkonfiguration bekannt, welche Kameras gemeinsam angesteuert werden und damit einen synchronisierten Satz an Bildern generieren. Daher schreibt diese Einzugsanwendung nach Erhalt aller Bilder ein spezielles KogMo-RTDB-Objekt, welches die abgeschlossene Übergabe aller Bilder eines Satzes an die KogMo-RTDB kennzeichnet. Der Zeitpunkt, zu dem dieses KogMo-RTDB-Objekt geschrieben wird, ist in Abb. 5.10 durch das ⧫-Symbol gekennzeichnet. Die nachfolgende Anwendung muss nun lediglich dieses Kennzeichnungsobjekt als auslösend definieren und findet dadurch mit dem entsprechenden Datenzeitstempel alle gemeinsam aufgezeichneten Bilder mit dem gleichen Datenzeitstempel.

5.3.2 Synchronisation von Kamera und LiDAR

Wie in Abschnitt 5.2 beschrieben, rotieren die beiden Velodyne HDL-64E mit einer Frequenz von ca. 10 Hz. Damit liefern die genannten Kameras ihre Bilder etwa

doppelt so häufig wie einer der LiDAR-Sensoren. Andererseits erstellen die beiden LiDAR ein 360° Abbild der Umgebung, während der horizontale Öffnungswinkel der Kameras nur bei ca. 60° liegt. Unter der Annahme einer statischen Umgebung und einem nicht bewegten Versuchsträger könnte die Asynchronität der beiden Sensoren vernachlässigt werden. Im hier behandelten Fall muss mit einem bewegten Fahrzeug und einer dynamischen Umgebung gerechnet werden. Daher müssen die Sensoren derart synchronisiert werden, dass beide zum gleichen Zeitpunkt den gleichen Ausschnitt der Umgebung abtasten.



(a)
Synchronisation von Kamera und LiDAR bei stehendem Fahrzeug.

(b)
Synchronisation von Kamera und LiDAR bei bewegtem Fahrzeug.

Abbildung 5.11: Synchronisation von Kamera und LiDAR.

Abbildung 5.11a zeigt die Situation zunächst für einen stehenden Versuchsträger aus der Vogelperspektive. Der Laserscanner liefert kontinuierlich Tiefenmessungen. Im Gegensatz zu den Kameras bietet dieser LiDAR keine Möglichkeit, über einen externen Auslöser angesteuert zu werden. Allerdings lässt sich zu jedem Datenpaket der aktuelle Rotationswinkel bestimmen. Der Beginn einer Umdrehung wird daher als der Zeitpunkt definiert, zu dem der Rotationswinkel des LiDAR bei $\pm 180°$ liegt, d.h. exakt entgegen der x-Achse des Fahrzeugs, d.h. entgegen der Fahrtrichtung, orientiert ist. Zu diesem Zeitpunkt wird per CAN-Bus ein Signal an die dSPACE MicroAutoBox gesendet.

Die dSPACE MicroAutoBox hat, wie auch in Abschnitt 5.3.1, die Aufgabe, den Auslösezeitpunkt der Kamera zu bestimmen und die Belichtungszeit zu messen. Wie

eingangs erwähnt sollen sowohl Kamera als auch LiDAR den gleichen Ausschnitt der Umgebung zum gleichen Zeitpunkt abtasten. Allerdings ist dies je nach Belichtungszeit nur teilweise möglich. Belichtet die Kamera beispielsweise sehr kurz, wird ggf. nur ein Teil ihres Sichtbereichs vom LiDAR abgetastet. Da die Belichtungszeit von der Kamera selbst bestimmt wird, liegt die einzige Möglichkeit der Einflussnahme in der Berechnung des Auslösezeitpunktes der Kamera. Dieser soll derart gewählt werden, dass nach Ablauf der halben Belichtungszeit die Orientierung des LiDAR mit der Blickrichtung, d.h. x-Achse, der Kamera übereinstimmt.

Wie Abb. 5.11a zeigt, befinden sich Kamera und LiDAR nicht im selben Koordinatenursprung. Unter Berücksichtigung des realen xy-Abstands von Kamera und LiDAR im Bereich weniger Dezimeter und dem xy-Abstand zu Hindernissen von einigen Metern wird zur Berechnung des Auslösezeitpunktes vereinfacht angenommen, dass die Koordinatensysteme zumindest in x- und y-Richtung übereinstimmen. Darüber hinaus sind die Kameras auf einer drehbaren Plattform montiert. Daher geht der Gierwinkel $\Psi_{\text{MarVEye-8}}$ der Plattform ebenfalls in die Berechnung des Auslösezeitpunktes ein. Vereinfacht wird angenommen, dass der Gierwinkel gemäß der Rechten-Hand-Regel positiv im Gegenuhrzeigersinn bezogen auf Abb. 5.11 dreht. Sind $T^{\text{Umdrehung}}_{\text{LiDAR},k-1}$ die Dauer einer Umdrehung des LiDAR und $T^{\text{Belichtung}}_{\text{Kamera},k-1}$ die Belichtungszeit der Kamera für die Aufnahme $k-1$ bekannt, dann ergibt sich die Auslöseverzögerung $T^{\text{Auslösen}}_{\text{Kamera},k}$ für die Kamera ab Beginn einer neuen Umdrehung des LiDAR zu

$$T^{\text{Auslösen}}_{\text{Kamera},k} = T^{\text{Umdrehung}}_{\text{LiDAR},k-1} \frac{180^\circ - \Psi^*_{\text{MarVEye-8},k}}{360^\circ} - \frac{1}{2}\, T^{\text{Belichtung}}_{\text{Kamera},k-1}. \tag{5.37}$$

Für den Gierwinkel der Kameraplattform $\Psi^*_{\text{MarVEye-8},k}$ muss hier auf eine Schätzung zurückgegriffen werden, da sich die Plattformwinkel aufgrund der Blickrichtungssteuerung auch nach der Berechnung der Auslöseverzögerung noch ändern können.

Abbildung 5.11b zeigt die gleiche Situation für einen bewegten Versuchsträger. Hier müssen zwei Effekte beachtet werden. Da sich das Fahrzeug nun bewegt, verändert sich global auch der Koordinatenursprung des LiDAR während einer Umdrehung. Die Messungen können daher nicht einfach auf die Pose des LiDAR zu Beginn oder Ende einer Umdrehung bezogen werden. Theoretisch müsste für jede Entfernungsmessung die exakte Position des LiDAR bzw. dessen Bewegung seit Beginn der Umdrehung bekannt sein. Da die beiden LiDAR-Varianten ihre Messungen nicht einzeln, sondern in Form von gebündelten Paketen versenden, kann die Referenzierung der Messungen nur bei Eintreffen eines solchen Pakets vorgenommen werden. Dieser Vorgang nennt sich inertiale Korrektur. In Abb. 5.11b entsprechen die Koordinatensysteme $\mathbf{S}_{\text{LiDAR},0}$

und $\mathbf{S}_{\text{LiDAR},2}$ der Pose des LiDAR zu Beginn bzw. zum Ende einer Umdrehung. Dazwischen gibt es je nach Anzahl der referenzierten Pakete entsprechend weitere Koordinatensysteme.

Der zweite Effekt entsteht durch die Auslöseverzögerung der Kamera. Gewünscht sind am Ende ein Kamerabild und eine dazu zeitsynchrone Punktwolke. Nach Ablauf einer LiDAR-Umdrehung muss daher die gesamte inertial korrigierte Punktwolke in den Zeitpunkt der Aufnahme des Kamerabildes transformiert werden. Das entsprechende Koordinatensystem $\mathbf{S}_{\text{LiDAR},1}$ entspricht der Pose des LiDAR zum errechneten mittleren Belichtungszeitpunkt der Kamera. Der Datenzeitstempel der Punktwolke wird dann auf den Datenzeitstempel des Bildes gesetzt.

Bei der Anordnung der Sensoren in den beiden Versuchsträgern Munich Cognitive Autonomous Robot Car 3rd Generation (MuCAR-3) und MuCAR-4 wird davon ausgegangen, dass die Aufzeichnung und Übertragung der Kamerabilder zum Ende einer jeden LiDAR-Umdrehung bereits abgeschlossen ist. Für nachfolgende Anwendungen, die zeitlich synchronisierte Kamerabilder und Punktwolken verarbeiten möchten, muss daher das zur Punktwolke korrespondierende KogMo-RTDB-Objekt das *auslösende* Objekt sein. Wie auch im rein kamerabasierten Fall enthält auch hier das Zeitstempelsignal der dSPACE MicroAutoBox an den Fahrzeugrechner die Winkel der Kameraplattform. Damit können, unter der Annahme bekannter Kalibrierungen, Punkte eingefärbt oder einzelnen Pixeln Tiefeninformationen mitgegeben werden. Nachdem sowohl Kameras als auch die LiDAR gemäß den Abschnitten 5.1 und 5.2 als intrinsisch kalibriert angenommen werden können, widmen sich die folgenden Abschnitte der extrinsischen Kalibrierung.

5.4 Extrinsische und intrinsische Kalibrierung von Kameras

Zur Fusion verschiedener Sensoren muss neben den intrinsischen Sensorparametern auch die relative Lage, d.h. die extrinsische Kalibrierung der Sensoren zueinander bekannt sein. Dadurch wird es möglich, die Messungen des einen Sensors in das Koordinatensystem des anderen Sensors zu transformieren und damit zu vergleichen. Die extrinsische Kalibrierung zweier Sensoren wird typischerweise dadurch gelöst, dass beide Sensoren gleichzeitig dieselbe Szene beobachten und dabei Korrespondenzen zwischen den Bildern beider Sensoren gebildet werden.

Eine der bekanntesten Veröffentlichungen zur Kalibrierung zweier Kameras ist die *Camera Calibration Toolbox for Matlab* von Bouguet [2003]. Dabei wird eine Sequenz

von zeitlich synchronisierten Bildpaaren aufgenommen, in der ein Schachbrettmuster vor beiden Kameras bewegt wird. Anschließend müssen die Bilder beider Kameras von Hand annotiert werden. D.h. die Eckpunkte der aufgenommenen Schachbrettmuster müssen manuell im Bild markiert werden. Daraus berechnet die Software neben der extrinsischen Kalibrierung auch die intrinsischen Kameraparameter, wie in Abschnitt 5.1 beschrieben. Für diese Kalibrierung eignen sich Schachbrettmuster vor allem deshalb, weil die starken Kontraste zwischen den Schachbrettfeldern von Mensch und Computer gut lokalisierbar sind. Dadurch konnte das Kalibrierungsverfahren zwischenzeitlich auch in die OpenCV-Bibliothek (Opensource Computer Vision) aufgenommen werden. Diese C/C++-Bibliothek übernimmt auch die Detektion des Schachbretts mit Hilfe von Ableitungsfiltern, so dass das manuelle Annotieren der Bildpaare entfällt.

Das Verfahren von Bouguet [2003] ist in sich abgeschlossen, d.h. die Kalibrierungswerte werden lediglich auf Basis der Eingangsbilder berechnet. Eine kontinuierliche Aktualisierung der Parameter findet nicht statt. Ändert sich die Relativlage beider Kameras beispielsweise durch einen Stoß oder aktive Blickrichtungssteuerung, muss der gesamte Kalibrierungsprozess auf Basis neuer Eingangsdaten wiederholt werden. Ist mit solchen Einflüssen zu rechnen, kann auf ein kontinuierliches Kalibrierungsverfahren wie das von Dang et al. [2009] zurückgegriffen werden. Dabei wird die Relativlage der Kameras kontinuierlich neu geschätzt. Sofern sich das Fahrzeug und damit die Sensoren nur in einer Ebene bewegen, kann auch auf das Verfahren von Tang und Liu [2018] zurückgegriffen werden.

Da diese Arbeit hauptsächlich die Fusion von Kamera und LiDAR betrachtet, beschreiben die folgenden Abschnitte die extrinsische Kalibrierung dieser beiden Sensorarten zueinander. Zum Teil lassen sich diese Verfahren auch zur Kamera-Kamera-Kalibrierung nutzen.

5.5 Extrinsische Kalibrierung von Kamera und LiDAR

Auch die extrinsische Kalibrierung einer Kamera mit einem LiDAR basiert typischerweise auf Korrespondenzen. Naikal et al. [2009] verwenden einen Kalibrierkörper und manuell annotierte Punktkorrespondenzen in den entstehenden LiDAR-Daten und Kamerabildern. In der Literatur finden sich mehrere Verfahren, in denen die Korrespondenzen zwischen Kanten statt zwischen Punkten gebildet werden. Für Luftaufnahmen zeigen die Publikationen von Deng et al. [2008] und Frueh et al.

[2004], dass sich Höhenkanten aus LiDAR-Daten und Bildkanten zur Problemlösung eignen. Da diese Verfahren die Existenz geradliniger Strukturen voraussetzen, sind sie gerade bei der Betrachtung von Städten und Gebäuden aus der Luft erfolgreich. Im innerstädtischen Bereich zeigen Stamos und Allen [2002] sowie Bileschi [2009], dass die Korrespondenzbildung zwischen Tiefenkanten und Bildkanten zur Bestimmung der extrinsischen Kalibrierung geeignet ist. Für Nahbereichsaufnahmen im Labor verfolgen Habib et al. [2004] ebenfalls einen kantenbasierten Ansatz. Guindel et al. [2017] beschreiben ein automatisches Verfahren zur extrinsischen Kalibrierung eines LiDAR mit einem Stereokamerasystem. Es beschreibt spezielle Kalibrierkörper, die von beiden Sensoren erkannt werden, um anschließend anhand der Korrespondenz die Relativlage zu bestimmen.

Trotz der unterschiedlichen Herangehensweisen bei der Korrespondenzfindung kommt bei den genannten Verfahren in einem zweiten Schritt immer eine Form der Fehlerminimierung, wie Levenberg-Marquardt (LM), zum Einsatz, um die gesuchte extrinsische Kalibrierung zu bestimmen. Die Fehler werden dabei auf unterschiedlichste Weise berechnet. Naikal et al. [2009] minimieren den Euklid'schen Abstand zwischen korrespondierenden 3D-Punkten. Bileschi [2009] hingegen minimiert den Rückprojektionsfehler, d.h. den Pixelabstand zwischen Bildpunkten und den Rückprojektionen der jeweiligen geschätzten 3D-Punkte in das Bild der Kamera. Jede der genannten Veröffentlichungen setzt zudem voraus, dass sich die Relativlage der beiden Sensoren nicht verändert, d.h. es findet keine kontinuierliche Kalibrierung statt.

Die zur Fehlerminimierung herangezogenen Korrespondenzen sind in keiner der beschriebenen Verfahren vollends an die Charakteristika der Sensoren angepasst. Zwar lassen sich 3D-Punkte in LiDAR-Daten genau bestimmen, die Schätzung von 3D-Punkten aus Bilddaten heraus ist hingegen häufig mit einer hohen Ungenauigkeit in der Distanz zur Kamera behaftet. Dieser Umstand wird in Abschnitt 6.2.3 weitergehend analysiert. Umgekehrt lassen sich Objektkonturen wie Kanten in Kamerabildern sehr genau extrahieren. In LiDAR-Punktwolken hingegen sind Kanten aufgrund der im Vergleich geringeren lateralen Auflösung schlecht lokalisierbar.

5.6 Berücksichtigung der Sensorcharakteristika in Fehlermetriken

Sind die Korrespondenzen, die zur Fehlerminimierung im Rahmen der extrinsischen Kalibrierung verwendet werden, nicht auf die Sensoreigenschaften der beteiligten

Sensoren abgestimmt, besteht die Möglichkeit, dass die resultierende Kalibrierung ungenau oder sogar fehlerhaft ist. Daher wird im Folgenden auf Möglichkeiten zur Einbeziehung der Sensorcharakteristika in die Minimierung durch verschiedene Fehlermetriken eingegangen. Die Grundlage hierfür liefert die Veröffentlichung von Himmelsbach et al. [2011b]. Zur Verbesserung der Genauigkeiten werden einzelne Metriken in diesem Abschnitt angepasst.

Entgegen der typischen merkmalbasierten Korrespondenzbestimmung zwischen LiDAR-Punktwolken und Kamerabildern wird in dieser Arbeit eine objektbasierte Korrespondenzbestimmung verwendet. Dazu ist es nötig, dass der Kalibierkörper, wie in Abb. 5.12, sowohl in LiDAR-Punktwolken, als auch in Kamerabildern wahrnehmbar ist.

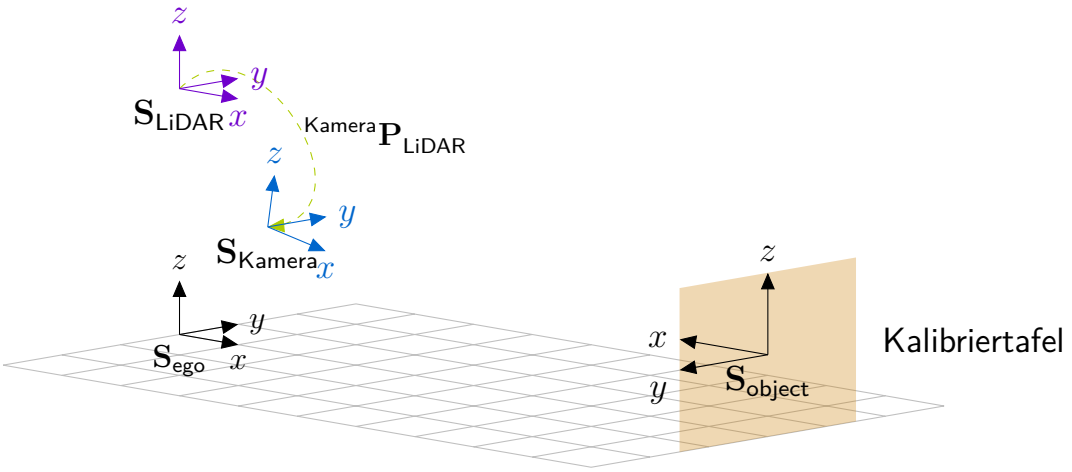


Abbildung 5.12: Wahrnehmung einer Kalibriertafel durch LiDAR und Kamera

Ein LiDAR-Sensor kann nach der Erkennung der Kalibriertafel direkt auf die Entfernung der Eckpunkte schließen. Da die Abmessungen der Kalibriertafel bekannt sind, ist dies auch aus Kamerabildern möglich. Liegen N zeitlich synchronisierte Aufnahmen der Kalibriertafel vor, können die vier Eckpunkte in 3D bzgl. der beiden Sensoren bestimmt werden. Umgekehrt kann dadurch auch die Pose der beiden Sensoren bzgl. der Kalibriertafel errechnet werden. Ziel ist es nun, die 6-DOF-Pose (Degrees of Freedom) ${}^{\text{Kamera}}\mathbf{P}_{\text{LiDAR}}$ zwischen diesen beiden Sensorkoordinatensystemen zu ermitteln. Mit der zu dieser Pose korrespondierenden Homogenen Transformationsmatrix ${}^{\text{Kamera}}\mathbf{H}_{\text{LiDAR}}$ lassen sich 3D-Punkte aus $\mathbf{S}_{\text{LiDAR}}$ in das Koordinatensystem der Kamera $\mathbf{S}_{\text{Kamera}}$ transformieren. Zur Vereinfachung werden in dieser Arbeit Homogene Transformationsmatrizen und 6-DOF-Posen je nach Zusammenhang synonym verwendet. Eine genauere Beschreibung findet sich im Anhang C.

5.6.1 Punkt zu Punkt-Metrik

Die Pose ${}^{\text{Kamera}}\mathbf{P}_{\text{LiDAR}}$ hängt von 6 Parametern ab: einer 3D-Translation entlang von x, y und z sowie den Rotationen um den Gierwinkel Ψ, den Nickwinkel Θ und den Rollwinkel Φ. Seien $\boldsymbol{p}_{\text{Kamera}}^{(i,j)}$ und $\boldsymbol{p}_{\text{LiDAR}}^{(i,j)}$ mit $i = 0, \cdots, N-1$ und $j = 0, \cdots, 3$ die jeweils vier extrahierten Eckpunkte der insgesamt N Aufnahmen. Dann lässt sich die extrinsische Kalibrierung ${}^{\text{Kamera}}\mathbf{P}_{\text{LiDAR}}$ durch Lösen des folgenden Minimierungsproblems bestimmen:

$$ {}^{\text{Kamera}}\mathbf{P}_{\text{LiDAR}} = \underset{\mathbf{P}\in\mathcal{P}}{\operatorname{argmin}} F_{pp}(\mathbf{P}) \text{ mit} \tag{5.38} $$

$$ F_{pp}(\mathbf{P}) = \sum_{i=0}^{N-1} \frac{1}{4} \sum_{j=0}^{3} \left\| \boldsymbol{p}_{\text{Kamera}}^{(i,j)} - \underbrace{\mathbf{P} \cdot \boldsymbol{p}_{\text{LiDAR}}^{(i,j)}}_{\boldsymbol{p}^{(i,j)}} \right\| . \tag{5.39} $$

Der für ein beliebiges i bestimmte Fehler dieser Punkt zu Punkt-Metrik $F_{pp}(\mathbf{P})$ ist in Abb. 5.13 dargestellt. Dabei werden für jedes i die orangefarbenen Abstände aufsummiert.

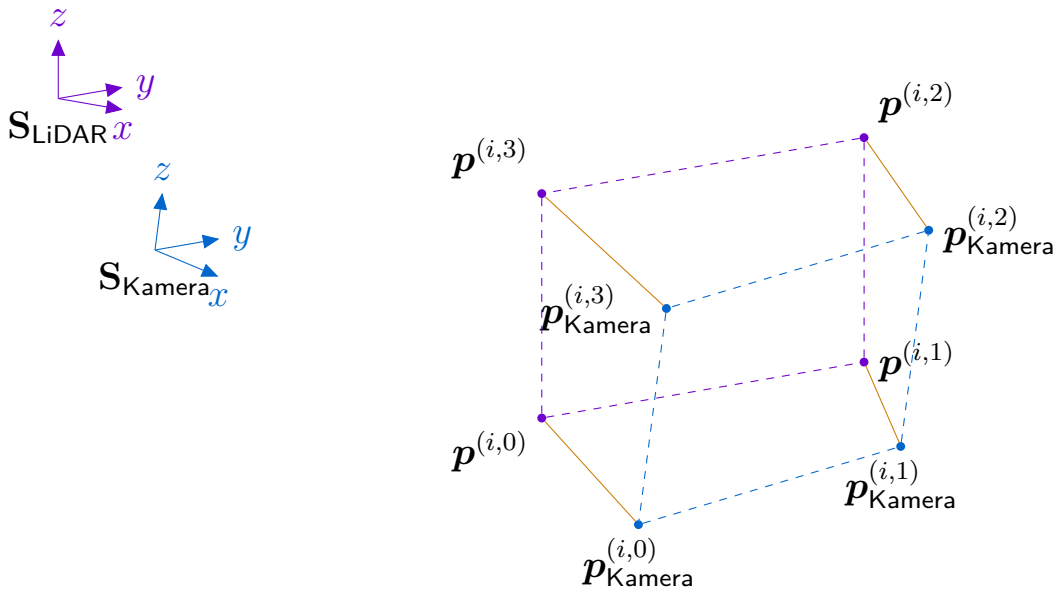


Abbildung 5.13:
Die Punkt zu Punkt-Metrik für ein beliebiges i. Der Fehler entspricht der Summe der orangefarbenen Abstände.

Die Punkt zu Punkt-Metrik vernachlässigt die Tatsache, dass aus Kamerabildern nicht direkt auf die Entfernung zu einem Punkt oder Objekt geschlossen werden kann. Die Ermittlung der 3D-Punkte $\boldsymbol{p}_{\text{Kamera}}^{(i,j)}$ wird nur dadurch möglich, dass ein Modell der Kalibriertafel bekannt ist. Doch selbst wenn die Pose der Kalibriertafel bzgl. der Kamera anhand dieses Modells über der Zeit geschätzt wird, verbleibt durch die Messungenauigkeit im Bild eine gewisse Unsicherheit in der Bestimmung des Abstands zur Kamera.

5.6.2 Punkt zu Strahl-Metrik

Auf der anderen Seite lässt sich jeder 2D-Bildpunkt als 3D-Sichtstrahl vom Kamerazentrum durch den Bildpunkt interpretieren. Die entsprechende Geradengleichung lautet

$$g\left(x\right) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + x \cdot \underbrace{\begin{pmatrix} 1 \\ \bar{u} \\ \bar{v} \end{pmatrix}}_{\boldsymbol{r}}, \tag{5.40}$$

wobei $\bar{u}$ und $\bar{v}$ die metrischen, verzerrungsfreien und bildzentrierten Koordinaten des 2D-Bildpunktes nach Gleichung (5.32) sind. Damit lassen sich für jede der N Aufnahmen vier Geraden $g^{(i,j)}\left(x\right) = x \cdot \boldsymbol{r}_{\text{Kamera}}^{(i,j)}$ bestimmen. Sofern die extrinsische Kalibrierung ${}^{\text{Kamera}}\mathbf{P}_{\text{LiDAR}}$ zwischen LiDAR und Kamera korrekt ist, müssen die Transformierten der Eckpunkte $\boldsymbol{p}^{(i,j)}$ auf den entsprechenden Geraden $g^{(i,j)}\left(x\right)$ liegen. Der Abstand der Punkte $\boldsymbol{p}^{(i,j)}$ zu diesen Geraden kann daher ebenfalls als Maß für die Korrektheit der extrinsischen Kalibrierung angesehen werden. Damit lässt sich das Optimierungsproblem unter Verwendung der sog. Punkt zu Strahl-Metrik, die in Abb. 5.14 dargestellt ist, umformulieren zu

$${}^{\text{Kamera}}\mathbf{P}_{\text{LiDAR}} = \underset{\mathbf{P} \in \mathcal{P}}{\operatorname{argmin}} F_{ps}\left(\mathbf{P}\right) \text{ mit} \tag{5.41}$$

$$F_{ps}\left(\mathbf{P}\right) = \sum_{i=0}^{N-1} \frac{1}{4} \sum_{j=0}^{3} \left|\left| \frac{\left(\mathbf{P} \cdot \boldsymbol{p}_{\text{LiDAR}}^{(i,j)}\right) \times \boldsymbol{r}_{\text{Kamera}}^{(i,j)}}{\left|\left|\boldsymbol{r}_{\text{Kamera}}^{(i,j)}\right|\right|} \right|\right|. \tag{5.42}$$

Dazu berücksichtigt die Punkt zu Strahl-Metrik die Sensoreigenschaften der Kamera, geht allerdings davon aus, dass der LiDAR genaue Eckpunkte $\boldsymbol{p}_{\text{LiDAR}}^{(i,j)}$ der Kalibriertafel

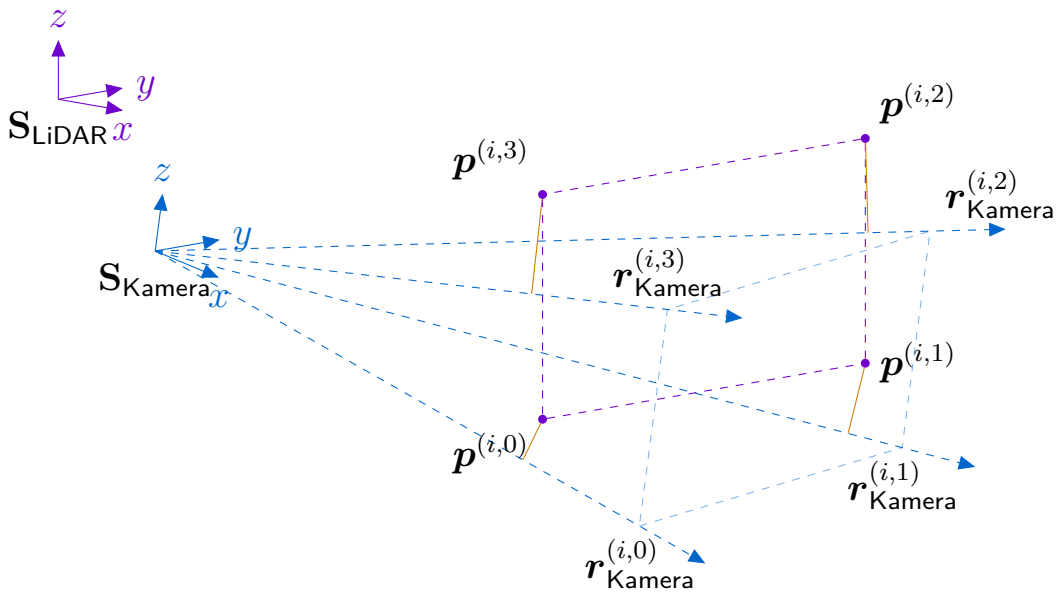


Abbildung 5.14:
Die Punkt zu Strahl-Metrik für ein beliebiges i. Der Fehler entspricht der Summe der orangefarbenen Abstände.

liefert. Im Vergleich zu einer Kamera besitzt der LiDAR eine wesentlich geringere laterale Auflösung. Dadurch kann der Rand der Kalibriertafel nicht exakt bestimmt werden.

5.6.3 Ebene zu Strahl-Metrik

Treffen genügend 3D-Messungen auf die Tafel, lässt sich der Abstand und die Orientierung der Ebene, in der die Kalibriertafel liegt, d.h. die Hesse'sche Normalform, bestimmen. Gänginge Methoden dafür sind die Lineare Ausgleichsrechnung oder das RANSAC-Verfahren (Random Sample Consensus). Die Ebene $e^{(i)}$ der Aufnahme i ist damit definiert durch die Menge aller Punkte $\boldsymbol{q}$, für die gilt

$$e^{(i)} : \boldsymbol{q} \cdot \bar{\boldsymbol{n}}^{(i)}_{\text{LiDAR}} - d^{(i)} = 0, \tag{5.43}$$

wobei $\bar{\boldsymbol{n}}^{(i)}_{\text{LiDAR}}$ der normierte Normalenvektor der i-ten Ebene bzgl. des LiDAR-Koordinatensystems ist und $d^{(i)}$ den Abstand der Ebene von $\mathbf{S}_{\text{LiDAR}}$ angibt.

Für eine korrekte extrinsische Kalibrierung zwischen $\mathbf{S}_{\text{LiDAR}}$ und $\mathbf{S}_{\text{Kamera}}$ spannen die vier Schnittpunkte $\boldsymbol{q}^{(i,j)}$ von $\boldsymbol{r}^{(i,j)}_{\text{Kamera}}$ und $e^{(i)}$ ein 3D-Rechteck auf, dessen Größe

mit der Kalibriertafel übereinstimmt. Stimmt die Kalibrierung nicht, degeneriert das Rechteck zu einem Viereck. Dieses Viereck ist in Abb. 5.15 orange dargestellt.

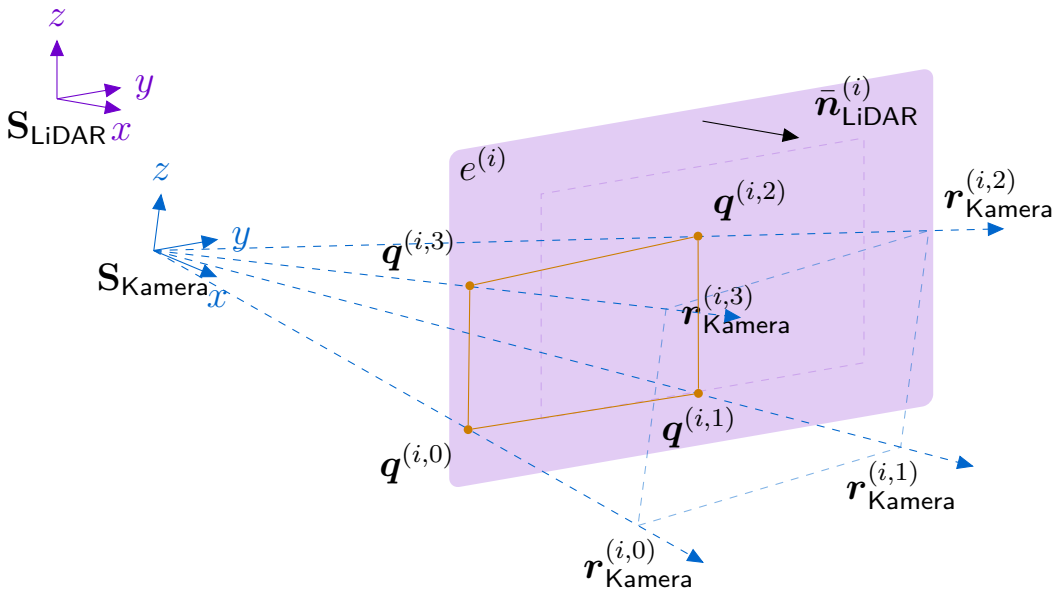


Abbildung 5.15: Die Ebene zu Strahl-Metrik für ein beliebiges i.

Gesucht ist damit die extrinsische Kalibrierung ${}^{\text{Kamera}}\mathbf{P}_{\text{LiDAR}}$, die dafür sorgt, dass das durch $\boldsymbol{q}^{(i,j)}$ aufgespannte Viereck zu einem Rechteck von der Größe der Kalibriertafel wird. Mit Hilfe der vier Kantenvektoren

$$\begin{aligned} \boldsymbol{v}^{(0,i)} &= \boldsymbol{q}^{(i,1)} - \boldsymbol{q}^{(i,0)}, & \boldsymbol{v}^{(1,i)} &= \boldsymbol{q}^{(i,2)} - \boldsymbol{q}^{(i,1)}, \\ \boldsymbol{v}^{(2,i)} &= \boldsymbol{q}^{(i,3)} - \boldsymbol{q}^{(i,2)} \text{ und} & \boldsymbol{v}^{(3,i)} &= \boldsymbol{q}^{(i,0)} - \boldsymbol{q}^{(i,3)} \end{aligned} \tag{5.44}$$

lassen sich die vier Kriterien Kantenlänge ($\leftrightarrow$), Rechtwinkligkeit ($\llcorner$), Parallelität ($\|$) und Seitenverhältnis ($\div$) ableiten. Für die gegebene Breite w und Höhe h der Kalibiertafel ergibt sich damit das neue Optimierungsproblem:

$$
{}^{\text{Kamera}}\mathbf{P}_{\text{LiDAR}} = \underset{\mathbf{P}\in\mathcal{P}}{\operatorname{argmin}} F_{es}\left(\mathbf{P}\right) \text{ mit} \tag{5.45}
$$

$$
F_{es}\left(\mathbf{P}\right) = \sum_{i=0}^{N-1} \left(F_{\leftrightarrow}^{(i)}\left(\mathbf{P}\right) \cdot F_{\llcorner}^{(i)}\left(\mathbf{P}\right) \cdot F_{\|}^{(i)}\left(\mathbf{P}\right) \cdot F_{\div}^{(i)}\left(\mathbf{P}\right) \right) \tag{5.46}
$$

$$
F_{\leftrightarrow}^{(i)}\left(\mathbf{P}\right) = 1 + \sum_{j=0,2} \left| \left|\left|\boldsymbol{v}^{(j,i)}\right|\right| - w \right| + \sum_{j=1,3} \left| \left|\left|\boldsymbol{v}^{(j,i)}\right|\right| - h \right| \tag{5.47}
$$

$$
F_{\llcorner}^{(i)}\left(\mathbf{P}\right) = 1 + \sum_{j=0,2} \sum_{k=1,3} \left| \bar{\boldsymbol{v}}^{(j,i)} \cdot \bar{\boldsymbol{v}}^{(k,i)} \right| \tag{5.48}
$$

$$
F_{\|}^{(i)}\left(\mathbf{P}\right) = 1 + \sum_{j=0,1} 1 - \left| \bar{\boldsymbol{v}}^{(j,i)} \cdot \bar{\boldsymbol{v}}^{(j+2,i)} \right| \tag{5.49}
$$

$$
F_{\div}^{(i)}\left(\mathbf{P}\right) = 1 + \left(\frac{w}{h} - \left| \frac{\boldsymbol{v}^{(0,i)}}{\boldsymbol{v}^{(1,i)}} \right| \right) + \left(\frac{w}{h} - \left| \frac{\boldsymbol{v}^{(2,i)}}{\boldsymbol{v}^{(3,i)}} \right| \right) \tag{5.50}
$$

Die vier Fehlerfunktionen $F_{\leftrightarrow}^{(i)}\left(\mathbf{P}\right)$, $F_{\llcorner}^{(i)}\left(\mathbf{P}\right)$, $F_{\|}^{(i)}\left(\mathbf{P}\right)$ und $F_{\div}^{(i)}\left(\mathbf{P}\right)$ sind derart gewählt, dass sie im günstigsten Fall den Wert 1 zurückliefern. In diesem Fall ergibt das Produkt in Gleichung (5.46) ebenfalls 1, so dass der Gesamtfehler der Funktion $F_{es}\left(\mathbf{P}\right)$ gleich N wird.

Die Fehlerfunktion $F_{\leftrightarrow}^{(i)}\left(\mathbf{P}\right)$ betrachtet dabei die Kantenlänge. Sie summiert die Differenzen zwischen realer Kantenlänge w bzw. h und den Längen der horizontalen bzw. vertikalen Kantenvektoren $\boldsymbol{v}^{(j,i)}$ auf. Die Orthogonalität der horizontalen Kantenvektoren wird durch die Fehlerfunktion $F_{\llcorner}^{(i)}\left(\mathbf{P}\right)$ überprüft. Dazu werden die Skalarprodukte jeder Paarung von normierten vertikalen und horizontalen Kantenvektoren aufsummiert. Sind alle Innenwinkel des Vierecks rechtwinklig, summieren sich die Skalarprodukte zu Null. Um die Rechteckform noch weiter zu forcieren, erhöht die Fehlerfunktion $F_{\|}^{(i)}\left(\mathbf{P}\right)$ den Fehler beim Abweichen von der Parallelität gegenüberliegender Kantenvektoren. Auch wenn die Kantenlänge bereits betrachtet wird, gewichtet die Fehlerfunktion $F_{\div}^{(i)}\left(\mathbf{P}\right)$ nochmals gesondert ein Abweichen des Seitenverhältnisses zwischen horizontalen und vertikalen Kantenvektoren.

5.6.4 Hybride Metrik

Um zu überprüfen, ob die Genauigkeit einer Metrik durch Kombination mit einer anderen erhöht werden kann, wird ein weiteres Optimierungsproblem formuliert. Von

besonderem Interesse ist dabei die Kombination der Metrik, die die Sensoreigenschaften am stärksten vernachlässigt, mit der, die die Sensoreigenschaften am stärksten berücksichtigt. Durch multiplikative Verknüpfung der Punkt zu Punkt-Metrik mit der Ebene zu Strahl-Metrik ergibt sich das folgende Optimierungsproblem:

$$^{\text{Kamera}}\mathbf{P}_{\text{LiDAR}} = \underset{\mathbf{P}\in\mathcal{P}}{\operatorname{argmin}} F_{\text{hybrid}}(\mathbf{P}) \text{ mit} \tag{5.51}$$

$$F_{\text{hybrid}}(\mathbf{P}) = F_{pp}(\mathbf{P}) \cdot F_{es}(\mathbf{P}) \tag{5.52}$$

5.7 Vergleich verschiedener Metriken zur extrinsischen Kalibrierung

Nach Vorstellung der Metriken gilt es zu untersuchen, welche der in Abschnitt 5.6 vorgestellten Metriken die extrinsische Kalibrierung zwischen einer Kamera und einem LiDAR am genauesten berechnet. Die genannten Metriken unterscheiden sich in der Anpassung an die Charakteristika der Sensoren. Um festzustellen, ob diese Anpassungen auch zu einer Steigerung der Genauigkeit beitragen, werden in diesem Kapitel umfangreiche Untersuchungen auf Basis synthetischer Daten vorgenommen. Die synthetischen Daten erlauben einen direkten Zugriff auf die sog. Ground-Truth, d.h. die in der Simulation bekannte extrinsische Kalibrierung. Durch Vergleich der errechneten Kalibrierungsergebnisse mit der Ground-Truth wird die Genauigkeit einer jeden Metrik messbar.

In der virtuellen Versuchsanordnung wird eine Kalibriertafel mit einer Breite von 3 m und einer Höhe von 2 m in einem Abstand von 5 m vor der Kamera platziert. Der LiDAR ist gegenüber der Kamera um 0,5 m nach hinten und 0,8 m nach oben versetzt. Zusätzlich ist die Kamera um 0,2 rad (ca. 11,5°) genickt. Diese Anordnung entspricht in etwa der echten Anordnung der Sensoren auf den Versuchsfahrzeugen und wird in den folgenden Experimenten als Ground-Truth $^{\text{LiDAR}}\mathbf{P}^{\text{Ground-Truth}}_{\text{Kamera}}$ verwendet.

Um die Ground-Truth herum werden nun synthetische Messungen erzeugt. Zur Berechnung der Fehlermetriken sind die folgenden Werte notwendig:

- Eckpunkte der Kalibriertafel
- Sehstrahlen von $\mathbf{S}_{\text{Kamera}}$ durch die Eckpunkte der Kalibriertafel
- Ebenengleichung der Kalibriertafel

Sowohl die Sehstrahlen als auch die Ebenengleichung lassen sich aus den Eckpunkten berechnen. Daher ist es zur Generierung der synthetischen Messungen ausreichend, lediglich Eckpunkte der virtuellen Kalibriertafel zu generieren.

Um die verschiedenen Sensorcharakteristika darzustellen, werden die Eckpunkte bezüglich der beiden Sensorkoordinatensysteme statistisch verrauscht. Aufgrund der Sensormodellierung aus den Abschnitten 5.1 und 5.2 lässt sich das Rauschverhalten der beiden Sensoren kartesisch nicht korrekt darstellen. Darum findet zunächst eine Transformation der Eckpunkte in Polarkoordinaten statt. Für einen beliebigen Eckpunkt $\boldsymbol{p}^{(i,j)}_{(\cdot),\,\mathbf{S}} = \left(\boldsymbol{p}^{(i,j)}_{[x]}, \boldsymbol{p}^{(i,j)}_{[y]}, \boldsymbol{p}^{(i,j)}_{[z]}\right)^T$ in kartesischen Koordinaten ergibt sich der entsprechende Punkt $\boldsymbol{p}^{(i,j)}_{(\cdot),\,\mathbf{T}} = \left(\boldsymbol{p}^{(i,j)}_{[r]}, \boldsymbol{p}^{(i,j)}_{[\Theta]}, \boldsymbol{p}^{(i,j)}_{[\Psi]}\right)^T$ in Polarkoordinaten durch

$$\boldsymbol{p}^{(i,j)}_{(\cdot),\,\mathbf{T}} = \begin{pmatrix} \boldsymbol{p}^{(i,j)}_{[r]} \\ \boldsymbol{p}^{(i,j)}_{[\Theta]} \\ \boldsymbol{p}^{(i,j)}_{[\Psi]} \end{pmatrix} = \begin{pmatrix} \left|\left|\boldsymbol{p}^{(i,j)}_{(\cdot),\,\mathbf{S}}\right|\right| \\ \tan^{-1}\left(\frac{\boldsymbol{p}^{(i,j)}_{[y]}}{\boldsymbol{p}^{(i,j)}_{[x]}}\right) \\ \cos^{-1}\left(\boldsymbol{p}^{(i,j)}_{[z]} \cdot \left|\left|\boldsymbol{p}^{(i,j)}_{(\cdot),\,\mathbf{S}}\right|\right|^{-1}\right) \end{pmatrix}. \tag{5.53}$$

Für die Umkehrung dieser Transformation gilt

$$\boldsymbol{p}^{(i,j)}_{(\cdot),\,\mathbf{S}} = \begin{pmatrix} \boldsymbol{p}^{(i,j)}_{[x]} \\ \boldsymbol{p}^{(i,j)}_{[y]} \\ \boldsymbol{p}^{(i,j)}_{[z]} \end{pmatrix} = \begin{pmatrix} \boldsymbol{p}^{(i,j)}_{[r]} \cdot \cos \boldsymbol{p}^{(i,j)}_{[\Theta]} \cdot \sin \boldsymbol{p}^{(i,j)}_{[\Psi]} \\ \boldsymbol{p}^{(i,j)}_{[r]} \cdot \sin \boldsymbol{p}^{(i,j)}_{[\Theta]} \cdot \sin \boldsymbol{p}^{(i,j)}_{[\Psi]} \\ \boldsymbol{p}^{(i,j)}_{[r]} \cdot \cos \boldsymbol{p}^{(i,j)}_{[\Psi]} \end{pmatrix}. \tag{5.54}$$

Verrauschte Koordinaten der Eckpunkte ergeben sich durch Addition eines zufälligen Rauschterms $\boldsymbol{n}_{\text{Kamera},\,\mathbf{T}} \sim \mathcal{N}\left(\mathbf{0}, \Sigma_{\text{Kamera},\,\mathbf{T}}\right)$ bzw. $\boldsymbol{n}_{\text{LiDAR},\,\mathbf{T}} \sim \mathcal{N}\left(\mathbf{0}, \Sigma_{\text{LiDAR},\,\mathbf{T}}\right)$ zu $\boldsymbol{p}^{(i,j)}_{\text{Kamera},\,\mathbf{T}}$ bzw. $\boldsymbol{p}^{(i,j)}_{\text{LiDAR},\,\mathbf{T}}$. Das Rauschen muss dabei an die Sensorcharakteristika angepasst werden. Eine Kamera zeichnet sich durch eine vergleichsweise hohe Winkelauflösung aus. Beispielsweise hat die bereits in Abschnitt 5.1 vorgestellte Kamera einen horizontalen Öffnungswinkel von ca. 57,5° und eine Bildbreite von 1294 pix. Das ergibt eine mittlere Winkelauflösung[2] von 0,044 °/pix bzw. $0{,}776 \times 10^{-3}$ rad/pix unter der Annahme, dass die Eckpunkte auf 1 pix genau gemessen werden. Da die Pixel quadratisch sind, gilt diese Näherung auch für die vertikale Winkelauflösung. Die Kamera führt keine Tiefenmessungen durch. Obwohl die Entfernung der Eckpunkte aufgrund der bekannten Größe der Kalibriertafel (Modellwissen) berechenbar ist, hängt die Entfernungsunsicherheit maßgeblich von der realen Entfernung der

[2] Zur Vereinfachung wird hier die in Abschnitt 5.1.2 beschriebene, zum Rand hin größer werdende Winkelauflösung vernachlässigt.

Kalibriertafel zur Kamera ab. Trotz eines u.U. konstanten Messfehlers im Bild wirkt sich dieser aufgrund der projektiven Abbildung mit steigendem Abstand der Kalibriertafel zur Kamera stärker aus. Im vorliegenden Fall befindet sich die Kalibriertafel in konstantem Abstand zur Kamera. Daher kann unter Verwendung des Strahlensatzes dennoch eine Abschätzung der Entfernungsvarianz für die beschriebene Versuchsanordnung ermittelt werden. Demnach entsprechen sich die Verhältnisse von Breite w zu Abstand d und Breite in Pixel w zu Brennweite f der Kalibriertafel,

$$\frac{w}{d} = \frac{\mathsf{w}}{f} \text{ und damit} \tag{5.55}$$

$$d = \frac{w}{\mathsf{w}} \cdot f. \tag{5.56}$$

Im vorliegenden Fall sind alle Größen bekannt bzw. lassen sich aus den Daten der Kamera und der Versuchsanordnung berechnen. So entspricht bspw. die Breite der Kalibriertafel im virtuellen Kamerabild ca. 695 pix. Auch wenn sich die Ecken der Kalibiertafel im Bild relativ genau vermessen lassen, so führen Kalibrierfehler und Ungenauigkeiten im Versuchsaufbau in der Realität erfahrungsgemäß zu einem Fehler in der Darstellung der Breite des Objekts im Bild und damit in der Schätzung der Distanz.

Indem w in Gleichung (5.56) als normalverteilt angenommen wird, kann mit Hilfe der Regeln zur Fehlerfortpflanzung auf die gesuchte Standardabweichung σ_d der Entfernung geschlossen werden:

$$d = y(\mathsf{w}) = \frac{w \cdot f}{\mathsf{w}} \tag{5.57}$$

$$y\left(\mathsf{w} + \Delta\,\mathsf{w}\right) - y\left(\mathsf{w}\right) = \frac{\partial y}{\partial\,\mathsf{w}} \cdot \Delta\,\mathsf{w} \tag{5.58}$$

$$\sigma_d = \left| \frac{\partial y}{\partial\,\mathsf{w}} \cdot \Delta\,\mathsf{w} \right| = \left| -\frac{w \cdot f}{\mathsf{w}^2} \cdot \Delta\,\mathsf{w} \right| \tag{5.59}$$

Unter der Annahme eines Fehlers in der Darstellung der Breite von 5 %, d.h. ca. 35 pix ergibt sich somit $\sigma_d \approx 0{,}25\,\mathrm{m}$. Insgesamt ergibt sich so

$$\Sigma_{\mathsf{Kamera},\,\mathbf{T}} = \begin{bmatrix} 0{,}25^2 & 0 & 0 \\ 0 & \left(0{,}776 \times 10^{-3}\right)^2 & 0 \\ 0 & 0 & \left(0{,}776 \times 10^{-3}\right)^2 \end{bmatrix}. \tag{5.60}$$

Für den LiDAR ergeben sich die erforderlichen Werte direkt aus dem Datenblatt (bzw. Anhang B.2). Die im Vergleich zum Velodyne HDL-64E S2 niedrigere late-

rale Winkelauflösung der unteren 32 Laserdioden des Velodyne HDL-64E S1 wird dabei ignoriert und für beide Sensortypen eine Rotationsgeschwindigkeit von 10 Hz angenommen. Vereinfacht wird für die vertikale Winkelauflösung ein konstanter Relativwinkel von 0,4° angenommen. Daraus folgt

$$\Sigma_{\text{LiDAR, }\mathbf{T}} = \begin{bmatrix} 0{,}02^2 & 0 & 0 \\ 0 & (3{,}02 \times 10^{-3})^2 & 0 \\ 0 & 0 & (6{,}98 \times 10^{-3})^2 \end{bmatrix}. \tag{5.61}$$

Auf Basis des beschriebenen Rauschens werden 1000 korrespondierende Paare von jeweils vier Eckpunkten bzgl. der beiden Koordinatensysteme $\mathbf{T}_{\text{LiDAR}}$ und $\mathbf{T}_{\text{Kamera}}$ generiert. Die Punkte sind in Abb. 5.16 dargestellt.

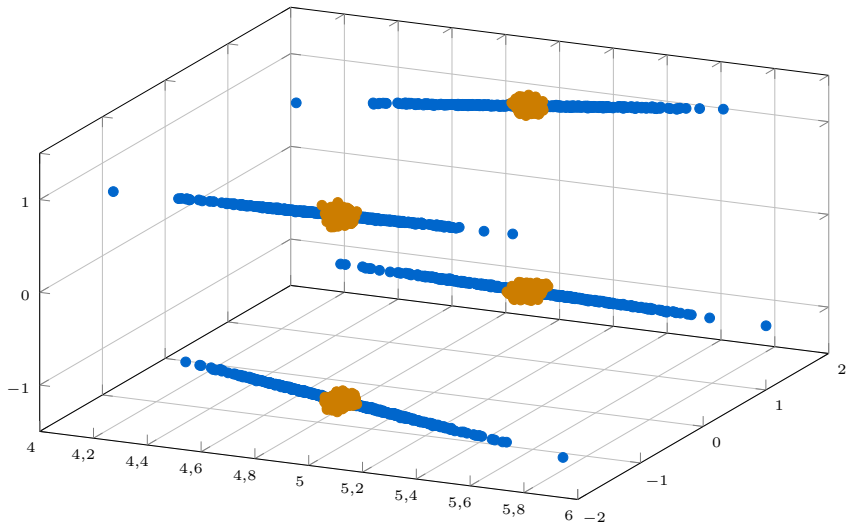


Abbildung 5.16:
Generierte Eckpunkte der Kalibriertafel. Punkte sind farbig entsprechend der Koordinatensysteme $\mathbf{T}_{\text{Kamera}}$ ● und $\mathbf{T}_{\text{LiDAR}}$ ● markiert.

Zur Umwandlung dieser generierten Punkte von Polarkoordinaten in kartesische Koordinaten dient Gleichung (5.54). Die Sehstrahlen der Punkt zu Strahl-Metrik sind mit Hilfe von Gleichung (5.40) berechenbar. Die Ebene durch die jeweils 4 generierten Punkte bzgl. $\mathbf{S}_{\text{LiDAR}}$ berechnet sich durch die Methode der kleinsten Fehlerquadrate.

Zur Minimierung der vorgestellten Fehlermetriken kommt das LM-Verfahren zum Einsatz. Ausgehend von einer initialen Schätzung berechnet dieses Verfahren iterativ Lösungen für nicht-lineare Optimierungsprobleme. Da derartige iterative Verfahren die Gefahr bergen, in lokale Extrema zu laufen, ist eine gute initiale Schätzung notwendig. Um diese zu erhalten, werden zunächst die beiden Mittelpunkte ausgehend von

den generierten Punkten bezüglich $\mathbf{S}_{\text{Kamera}}$ und $\mathbf{S}_{\text{LiDAR}}$ berechnet. Der kartesische Verschiebungsvektor zwischen diesen beiden Mittelpunkten dient schließlich als initiale Schätzung. Auf diese Weise werden unterschiedliche Sensororientierungen nicht berücksichtigt, was im hier vorliegenden Fall aufgrund der ohnehin ähnlichen Orientierungen akzeptabel ist.

Die 1000 generierten Eckpunktpaare repräsentieren nur einen einzigen virtuellen Satz an Messungen. Um statistisch belastbare Aussagen treffen zu können, wird der gesamte Vorgang, d.h. die Generierung der Punkte und die Optimierung der Fehlermetriken, einhundertmal wiederholt. Um zu beurteilen, welche der Metriken am Besten ist, findet ein Vergleich der von den Metriken geschätzten Posen mit der Ground-Truth ${}^{\text{LiDAR}}\mathbf{P}^{\text{Ground-Truth}}_{\text{Kamera}}$ statt. Dabei werden Positionierungsfehler und Winkelfehler getrennt betrachtet, zumal ohnehin kein einheitliches Vorgehen existiert, beide in Relation zu setzen. Für zwei Posen $\mathbf{P}^{(1)}$ und $\mathbf{P}^{(2)}$ beschreibt die Translationsdistanz $D^{\text{Translation}}$ den Euklid'schen Abstand zwischen den kartesischen Koordinaten der Posen

$$D^{\text{Translation}}\left(\mathbf{P}^{(1)}, \mathbf{P}^{(2)}\right) = \left\| \begin{pmatrix} \mathbf{P}^{(1)}_{[x]} \\ \mathbf{P}^{(1)}_{[y]} \\ \mathbf{P}^{(1)}_{[z]} \end{pmatrix} - \begin{pmatrix} \mathbf{P}^{(2)}_{[x]} \\ \mathbf{P}^{(2)}_{[y]} \\ \mathbf{P}^{(2)}_{[z]} \end{pmatrix} \right\|. \tag{5.62}$$

Die Rotationsdistanz D^{Rotation} zwischen den Posen $\mathbf{P}^{(1)}$ und $\mathbf{P}^{(2)}$ ergibt sich aus dem Winkel zwischen den beiden korrespondierenden Einheitsquaternionen $\bar{\mathrm{q}}^{(1)}$ and $\bar{\mathrm{q}}^{(2)}$. Quaternionen sind neben Eulerwinkeln eine weitere Möglichkeit der Beschreibung von 3D-Rotationen durch vier Parameter. Die vier Parameter repräsentieren dabei die 3D-Rotationsachse und den Winkel der Rotation. Die Rotationsdistanz D^{Rotation}, d.h. der Winkel zwischen zwei Quaternionen, berechnet sich wie folgt

$$D^{\text{Rotation}}\left(\mathbf{P}^{(1)}, \mathbf{P}^{(2)}\right) = \angle\left(\bar{\mathrm{q}}^{(1)}, \bar{\mathrm{q}}^{(2)}\right) \tag{5.63}$$

$$= \cos^{-1}\left(2 \cdot \left\langle \bar{\mathrm{q}}^{(1)}, \bar{\mathrm{q}}^{(2)} \right\rangle\right). \tag{5.64}$$

Abbildung 5.17 zeigt einen Vergleich der Positionsfehler in Form von Kastengrafiken. Die blauen Kästen umspannen dabei den Bereich vom 25-sten bis zum 75-sten Perzentil, während die mittleren horizontalen Linien die Position der Mediane anzeigen. Die gesamte Bereich einer jeden Verteilung, ohne Ausreißer, erstreckt sich über die jeweiligen gestrichelten Linien. Ausreißer sind in Form von $+$-Symbolen dargestellt.

Die Hybride Metrik liefert hierbei die genauesten Schätzungen. Der Positionsfehler liegt in einem Bereich weniger Millimeter. Die Punkt zu Punkt-Metrik liefert ebenfalls

gute Positionsschätzungen im Bereich einiger Millimeter bis weniger Zentimeter. Interessanterweise verschlechtert sich die Positionsgenauigkeit für die Punkt zu Strahl und die Ebene zu Strahl-Metrik, obwohl angenommen wurde, gerade durch die Beachtung der Sensorcharakteristika bessere Ergebnisse zu erzielen. Dieser Effekt wird auf den folgenden Seiten noch detaillierter betrachtet. Festzuhalten bleibt an dieser Stelle, dass die Kombination aus traditioneller Punkt zu Punkt-Metrik und sensorangepasster Ebene zu Strahl-Metrik die genauesten Positionsschätzungen liefert.

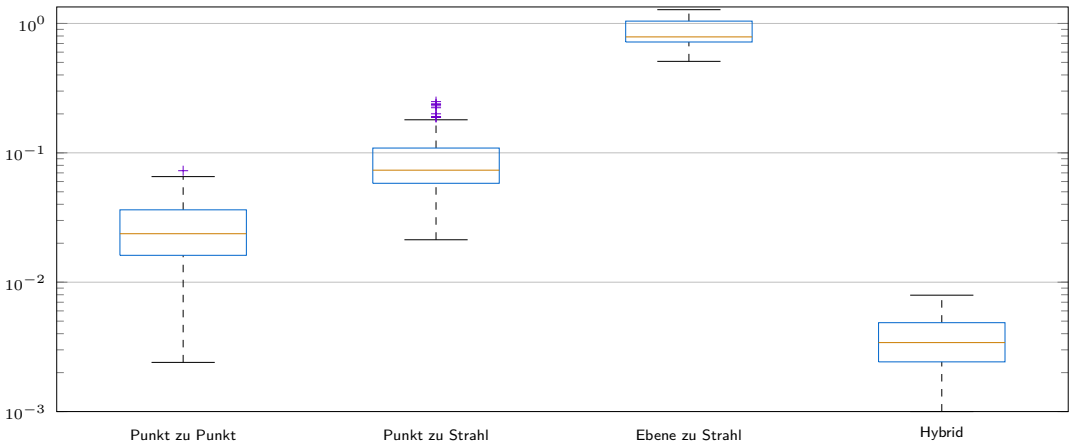


Abbildung 5.17: Translationsfehler der verschiedenen Metriken in [m].

Die entsprechende Grafik für den Rotationsfehler ist in Abb. 5.18 dargestellt. Hier zeigen sich vergleichbare Ergebnisse wie für den Positionsfehler. Die Kombination aus Punkt zu Punkt-Metrik und Ebene zu Strahl-Metrik liefert auch hier die genauesten Schätzungen in der Größenordnung von tausendstel Radian.

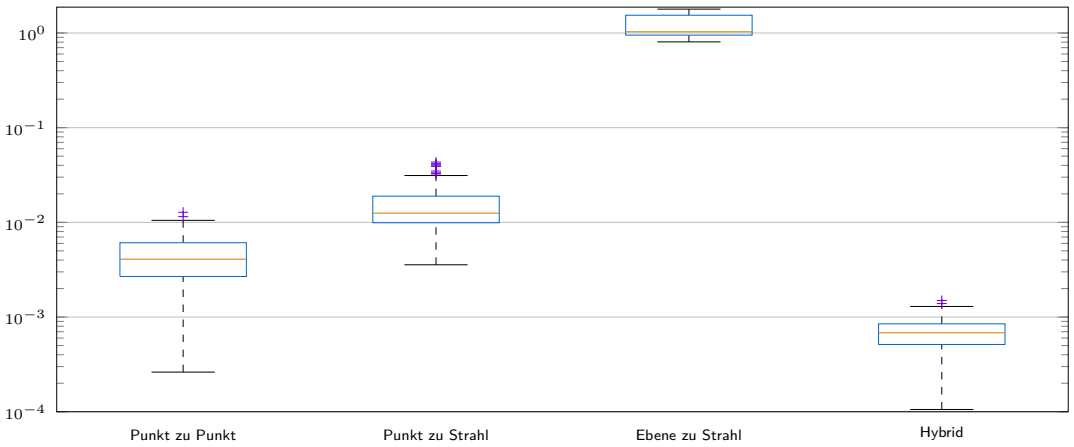


Abbildung 5.18: Rotationsfehler der verschiedenen Metriken in [rad].

Neben der Positions- und Winkelgenauigkeit spielt bei einem iterativen Optimierungsverfahren, wie dem LM-Verfahren, die Anzahl der Iterationen bis zur Konvergenz eine Rolle. Letztlich entscheidet sie maßgeblich über die Dauer, die zur Kalibrierung erforderlich ist. Einen Vergleich der benötigten Iterationen zeigt Abb. 5.19. Wie schon beim Positions- und Rotationsfehler erzielt auch hier die Hybride Metrik das beste Ergebnis. Nach im Schnitt etwa 200 Iterationen wird die Lösung gefunden. Die anderen Metriken benötigen dazu zum Teil über 1200 Iterationen.

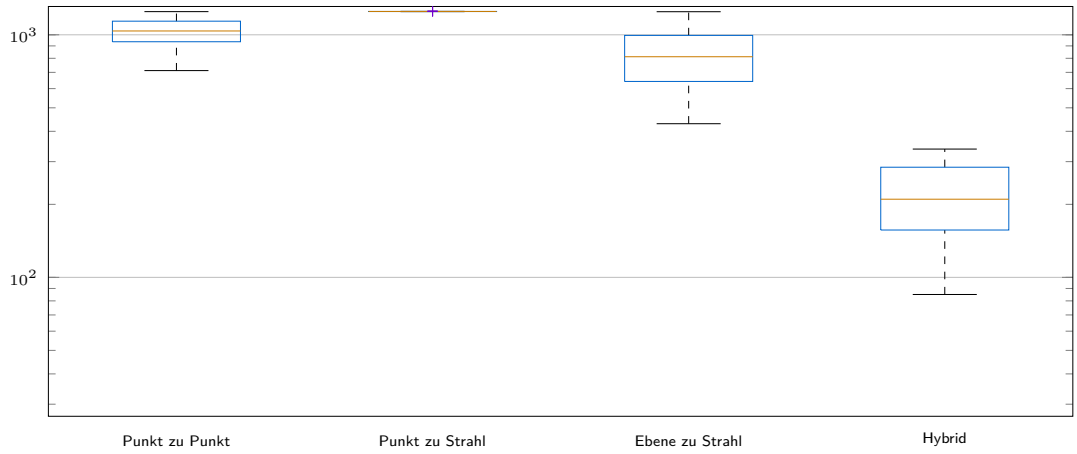


Abbildung 5.19: Anzahl der Iterationen bis zum Abbruch des LM-Verfahrens.

Gerade bei der Punkt zu Strahl-Metrik fällt auf, dass die Optimierung durch das LM-Verfahren häufig abgebrochen wurde, da sich die geschätzte Lösung gar nicht mehr, oder nur unwesentlich, geändert hat. Auch dieses Verhalten wird im Anschluss genauer erklärt. Zum detaillierten Vergleich der Metriken zeigt Tabelle 5.2 eine Übersicht über die statistischen Kenngrößen der beschriebenen Untersuchung. Diese Übersicht zeigt deutlich, dass die Hybride Metrik allen anderen Metriken in Bezug auf Positions- und Rotationsfehler sowie in der Anzahl der benötigten Iterationen überlegen ist.

Das Ergebnis dieser Untersuchung steht insofern im Widerspruch zu der Erwartung, dass eine genauere Berücksichtigung der Sensorcharakterisika auch zu genaueren Optimierungsergebnissen führt. Zur Erklärung findet im Folgenden eine Betrachtung des Raums statt, in dem das LM-Verfahren die Optimierung durchführt. Anstelle des Optimierungsverfahrens tastet dazu eine Gittersuche den sechsdimensionalen Raum der Posen um die Ground-Truth herum paarweise ab. D.h. für jede paarweise Kombination aus Pose-Koordinaten wird in einem 2D-Raster um die Ground-Truth herum die entsprechende Metrik über alle N Aufnahmen berechnet. Die jeweils anderen Koordinaten der Pose werden in diesem Fall gleich der Ground-Truth gesetzt. Der untersuchte Bereich erstreckt sich für die Pose-Koordinaten $\mathbf{P}_{[x]}$, $\mathbf{P}_{[y]}$ und $\mathbf{P}_{[z]}$

		F_{pp}	F_{ps}	F_{es}	F_{hybrid}
$D^{\text{Translation}}$	μ	0,0260	0,0910	0,8590	0,0040
	σ	0,0150	0,0510	0,1870	0,0020
	Median	0,0240	0,0730	0,7870	0,0030
	Minimum	0,0024	0,0213	0,5094	0,0001
	Maximum	0,0728	0,2489	1,2778	0,0079
D^{Rotation}	μ	0,0040	0,0160	1,1950	0,0001
	σ	0,0030	0,0090	0,3100	$<$0,0001
	Median	0,0040	0,0130	1,0290	0,0010
	Minimum	0,0003	0,0036	0,8055	0,0001
	Maximum	0,0128	0,0433	1,7861	0,0015
Iterationen	μ	1025,8510	1247,8220	831,2080	217,0990
	σ	133,4120	0,9630	231,0270	71,8280
	Median	1037	1248	812	210
	Minimum	712	1246	430	85
	Maximum	1246	1250	1245	338

Tabelle 5.2:
Statistische Kenngrößen der verschiedenen Metriken. Die besten und schlechtesten Werte einer Zeile sind jeweils farbig (■ bzw. ■) dargestellt.

über jeweils 4 m und wird in Schrittweiten von 0,1 m abgetastet. Für $\mathbf{P}_{[\Phi]}$, $\mathbf{P}_{[\Theta]}$ und $\mathbf{P}_{[\Psi]}$ gilt ein Winkelbereich von 20° und eine Schrittweite von 0,5°. Insgesamt ergeben sich damit pro Metrik 15 Fehlerflächen aus jeweils 40×40 Datenpunkten, und somit insgesamt 24 000 verschiedene Abtastpunkte. Die Intensitäten der gezeigten Fehlerflächen sind innerhalb der jeweils selben Metrik auf minimale und maximale Fehler normiert.

Ganz allgemein lässt sich feststellen, dass die Optimierung am schnellsten und genauesten ist, wenn die Fehlerflächen ein eindeutiges Minimum und hohe Gradienten aufweisen. Abbildung 5.20 zeigt die x,y-Fehlerflächen der vier Metriken $F_{pp}(\mathbf{P})$, $F_{ps}(\mathbf{P})$, $F_{es}(\mathbf{P})$ und $F_{hybrid}(\mathbf{P})$. Alle bis auf die Ebene zu Strahl-Metrik weisen hier ein eindeutiges Minimum auf. Die Fehlerfläche der Ebene zu Strahl-Metrik zeigt ein vertikal sehr stark ausgeprägtes Basin mit keinem eindeutigen Minimum für $\mathbf{P}_{[y]}$. Das führt dazu, dass $\mathbf{P}_{[x]}$ durch das Optimierungsverfahren zwar recht gut bestimmt werden kann, $\mathbf{P}_{[y]}$ im Vergleich dazu allerdings nur schlecht. Dies ist dadurch erklärbar, dass sich Änderungen in $\mathbf{P}_{[x]}$ direkt negativ auf $F_{\leftrightarrow}^{(i)}(\mathbf{P})$ auswirken. Änderungen in $\mathbf{P}_{[y]}$ hingegen würden unter idealen Messdaten den Fehler der Ebene zu Strahl-Metrik gar nicht beeinflussen, da die Kalibriertafel parallel zur yz-Ebene der Kamera aufgestellt ist. Nur aufgrund der künstlich verrauschten Aufnamen und der damit verbundenen Nicht-Parallelität von yz-Ebene und Kalibiertafel kommt es zu

einem kleinen Fehler für $\mathbf{P}_{[y]}$. Die Fehlerfläche der Punkt zu Punkt-Metrik erscheint symmetrisch, wodurch sich Abweichungen in $\mathbf{P}_{[x]}$ und $\mathbf{P}_{[y]}$ gleich stark auf den Fehler auswirken. Das Bild der hybriden Metrik zeigt die Kombination aus Punkt zu Punkt- und Ebene zu Strahl-Metrik, wodurch sich ein noch tieferes Basin und damit steilere Gradienten ergeben. Bei Betrachtung der Fehlerfläche der Punkt zu Strahl-Metrik fällt auf, wie die Metrik sensitiver auf eine Verschiebung in y-Richtung reagiert. Aufgrund des Versuchsaufbaus entspricht eine Verschiebung in y-Richtung einer zur Ebene parallelen Bewegung, welche sich direkt auf den Abstand zwischen den Eckpunkten der Ebene und den zugehörigen Sehstrahlen auswirkt. Eine Verschiebung in x-Richtung hingegen entspricht einer Bewegung orthogonal zur Ebene und wirkt sich damit – aufgrund der relativ kleinen Azimuth- und Elevationswinkel zwischen der Bewegungsrichtung und dem jeweiligen Sehstrahl – weniger stark aus.

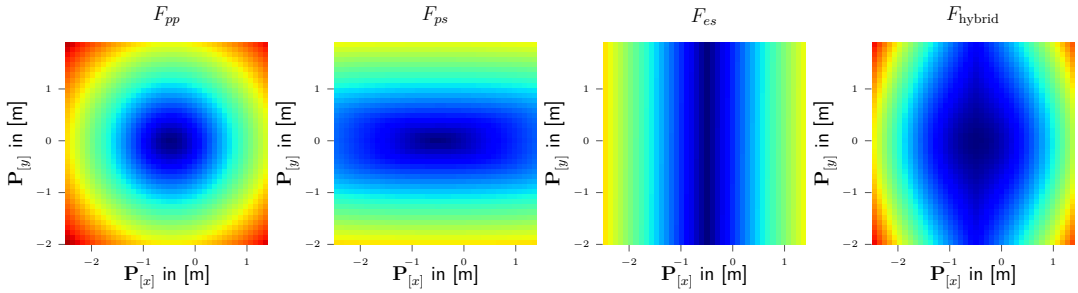


Abbildung 5.20:
x,y-Fehlerflächen der vier Metriken $F_{pp}(\mathbf{P})$, $F_{ps}(\mathbf{P})$, $F_{es}(\mathbf{P})$ und $F_{hybrid}(\mathbf{P})$.

Abbildung 5.21 zeigt ein zu Abb. 5.20 analoges Bild. Im Vergleich ändert sich in diesem Beispiel nur die Bewegungsrichtung in z-Richtung, welches aufgrund des Versuchsaufbaus zu vergleichbaren Resultaten führt.

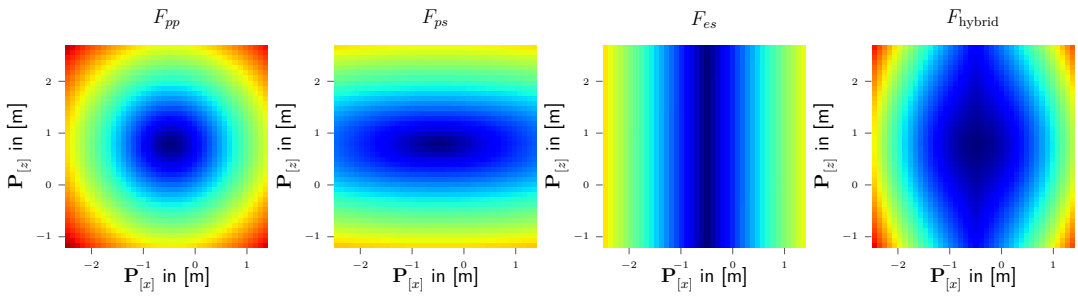


Abbildung 5.21:
x,z-Fehlerflächen der vier Metriken $F_{pp}(\mathbf{P})$, $F_{ps}(\mathbf{P})$, $F_{es}(\mathbf{P})$ und $F_{hybrid}(\mathbf{P})$.

In Abb. 5.22 ist erkennbar, wie sowohl die Punkt zu Punkt-Metrik, als auch die Punkt zu Strahl-Metrik gleichermaßen auf eine Verschiebung entlang y sowie z

reagieren. Aufgrund der schon angesprochenen Parallelität der Ebene zur xy-Ebene ergibt sich für die Ebene zu Strahl-Metrik kein eindeutiges Minimum und nur sehr geringe Gradienten. Dadurch führt auch die Kombination aus Punkt zu Punkt- und Ebene zu Strahl-Metrik zur Hybriden Metrik zu einer Abflachung der Gradienten.

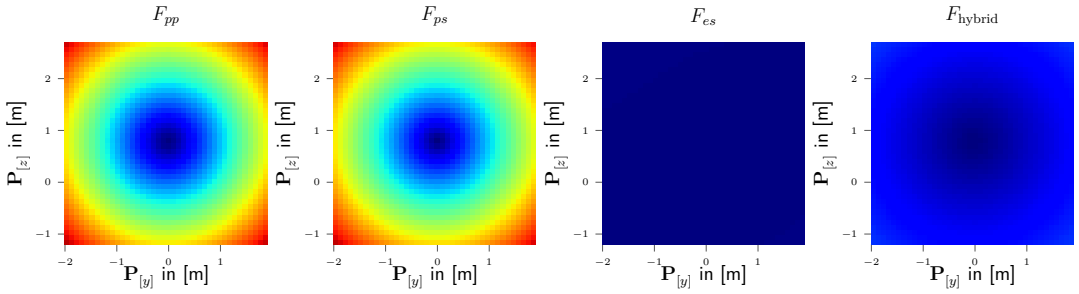


Abbildung 5.22:
y,z-Fehlerflächen der vier Metriken $F_{pp}(\mathbf{P})$, $F_{ps}(\mathbf{P})$, $F_{es}(\mathbf{P})$ und $F_{hybrid}(\mathbf{P})$.

Die bisherigen Betrachtungen der Fehlerflächen zeigen für die Ebene zu Strahl-Metrik in keinem der Fälle ein eindeutiges Minimum und lediglich in Hinblick auf $\mathbf{P}_{[x]}$ hohe Gradienten. Dies erklärt bereits den im Vergleich zu den anderen Metriken höheren Translationsfehler $D^{\text{Translation}}$ sowie die im Vergleich zur Hybriden Metrik höhere Zahl an Iterationen des Optimierungsverfahrens. Die Punkt zu Punkt-Metrik zeigt im Vergleich zur Punkt zur Strahl- und Ebene zu Strahl-Metrik in den bisherigen Fällen eindeutige Minima, woraus sich schließen lässt, dass sich die Punkt zu Punkt-Metrik generell zur Bestimmung des translatorischen Anteils der Kalibrierung eignet. Im Hinblick auf die Kalibrierung von $\mathbf{P}_{[x]}$ ist eine Kombination mit der Ebene zu Strahl-Metrik vorteilhaft, weil sie in diesen Fällen zu noch steileren Gradienten führt.

Im Weiteren findet eine Betrachtung der Raumwinkel statt. Abbildung 5.23 zeigt die Fehlerflächen für $\mathbf{P}_{[\Phi]}$ ggü. $\mathbf{P}_{[\Theta]}$ und $\mathbf{P}_{[\Phi]}$ ggü. $\mathbf{P}_{[\Psi]}$. Allgemein ist erkennbar, dass die Gradienten aller vier Metriken deutlich geringer ausfallen. Dies legt die Vermutung nahe, dass die Winkelgrößen im Vergleich zu den metrischen Größen durch das Optimierungsverfahren schwerer zu bestimmen sind. Dennoch sind für die Punkt zu Punkt- und die Punkt zu Strahl-Metrik Basins mit eindeutigen Minima zu erkennen. Die Basins sind entlang von $\mathbf{P}_{[\Phi]}$ ausgedehnter, d.h. $\mathbf{P}_{[\Phi]}$ ist durch das Optimierungsverfahren schwieriger zu bestimmen als $\mathbf{P}_{[\Theta]}$ und $\mathbf{P}_{[\Psi]}$. Auch dieses Verhalten ist durch den Versuchsaufbau erklärbar. Eine Veränderung von $\mathbf{P}_{[\Phi]}$ führt trotz des bestehenden Nickwinkels der Ground-Truth in der Hauptsache zu einer Rotation der korrespondierenden Eckpunkte in der Ebene der Kalibriertafel. Der Fehler der Metriken ist damit hauptsächlich durch den Sinus und Cosinus des

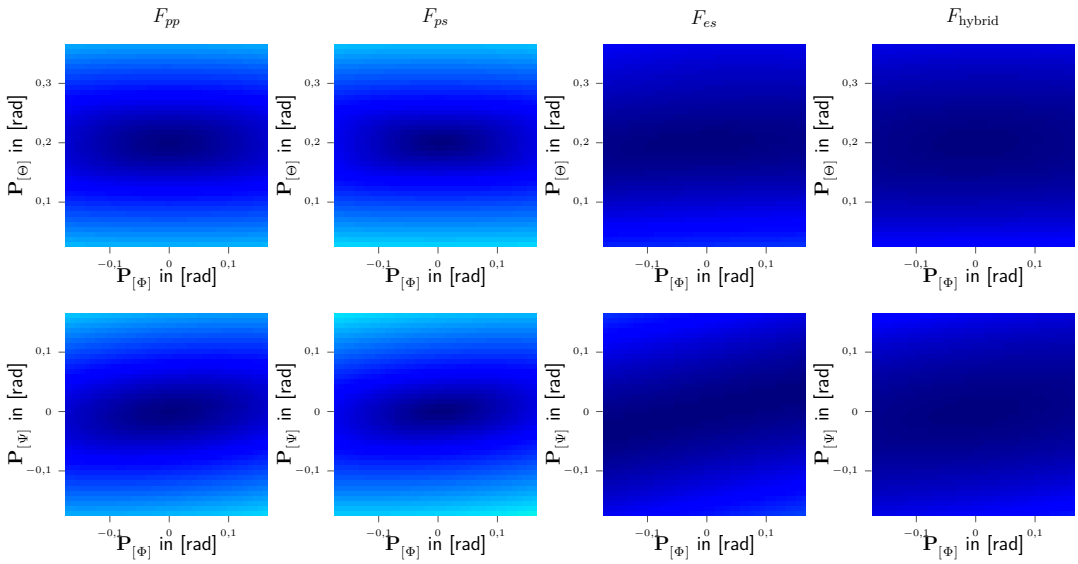


Abbildung 5.23:
Φ,Θ-Fehlerflächen und Φ,Ψ-Fehlerflächen der vier Metriken $F_{pp}(\mathbf{P})$, $F_{ps}(\mathbf{P})$, $F_{es}(\mathbf{P})$ und $F_{hybrid}(\mathbf{P})$.

Rotationswinkels $\mathbf{P}_{[\Phi]}$ beeinflusst. Für Nickwinkel $\mathbf{P}_{[\Theta]}$ und Gierwinkel $\mathbf{P}_{[\Psi]}$ hingegen wird der Fehler hauptsächlich durch den Tangens des Rotationswinkels beeinflusst und nimmt damit gerade bei größer werdenden Winkeln stärker zu. Für die Ebene zu Strahl- und die Hybride Metrik sind ein Minimum und kleine Gradienten gerade noch erkennbar.

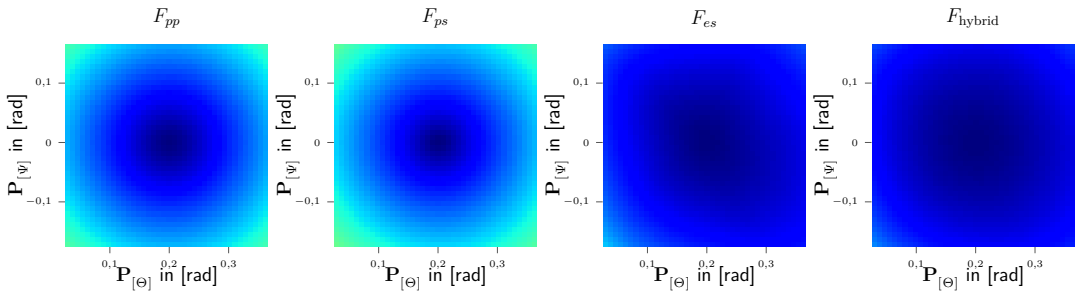


Abbildung 5.24:
Θ,Ψ-Fehlerflächen der vier Metriken $F_{pp}(\mathbf{P})$, $F_{ps}(\mathbf{P})$, $F_{es}(\mathbf{P})$ und $F_{hybrid}(\mathbf{P})$.

Abbildung 5.24 zeigt die Fehlerflächen für $\mathbf{P}_{[\Theta]}$ ggü. $\mathbf{P}_{[\Psi]}$. Hier sind für alle vier Metriken eindeutige Minima und kreisförmige Basins erkennbar. Zwar sind diese für die Punkt zu Punkt- und Punkt zu Strahl-Metrik deutlicher ausgeprägt und unterliegen höheren Gradienten, dennoch scheint das Optimierungsverfahren mit allen Metriken $\mathbf{P}_{[\Theta]}$ und $\mathbf{P}_{[\Psi]}$ bestimmen zu können.

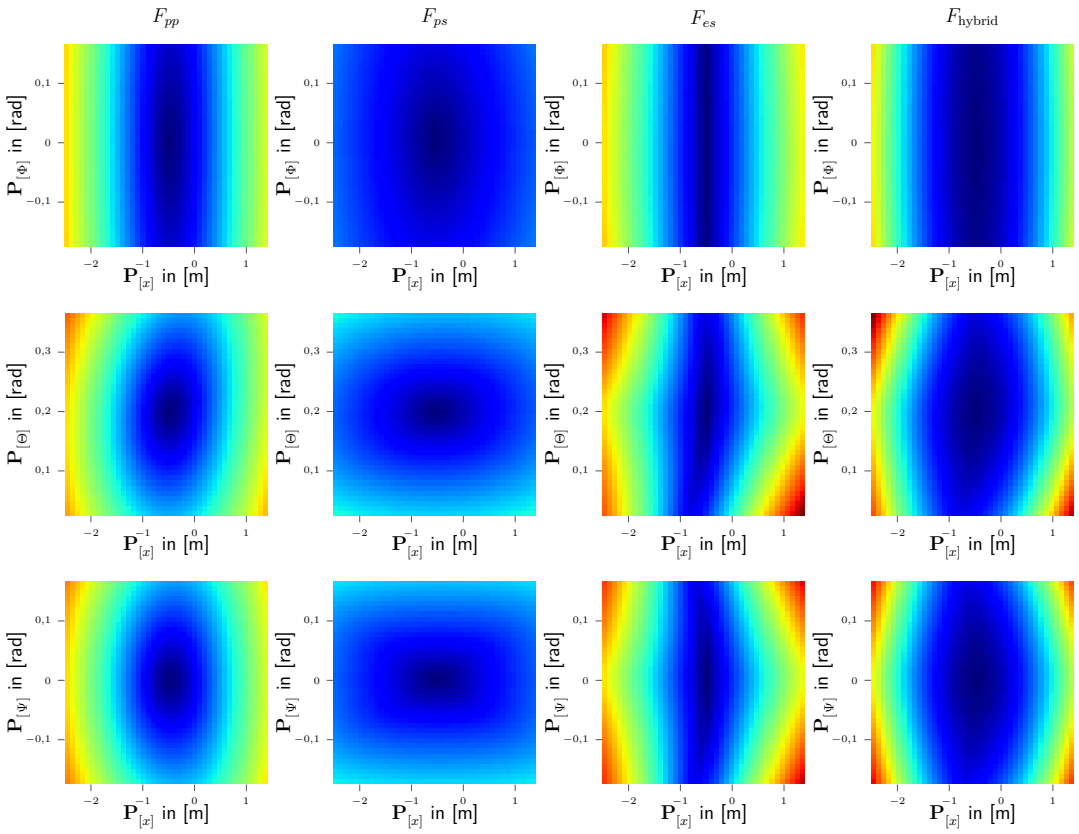


Abbildung 5.25:
x,Φ-Fehlerflächen, x,Θ-Fehlerflächen und x,Ψ-Fehlerflächen der vier Metriken $F_{pp}\,(\mathbf{P})$, $F_{ps}\,(\mathbf{P})$, $F_{es}\,(\mathbf{P})$ und $F_{hybrid}\,(\mathbf{P})$.

Interessant ist hierbei auch die Betrachtung von Fehlerflächen bzgl. einer Positions- und einer Winkelgröße. Abbildung 5.25, Abb. 5.26 und Abb. 5.27 zeigen die verbleibenden 9 Fehlerflächen für jede der 4 Metriken. Wie auch in den vorherigen Abbildungen zeigt sich, dass der Rollwinkel $\mathbf{P}_{[\Phi]}$ generell durch das Optimierungsverfahren schwieriger zu bestimmen ist als Nick- und Gierwinkel. Aber auch im Vergleich zu $\mathbf{P}_{[\Theta]}$ und $\mathbf{P}_{[\Psi]}$ sind die Gradienten generell steiler für die Positionsgrößen $\mathbf{P}_{[x]}$, $\mathbf{P}_{[y]}$ und $\mathbf{P}_{[z]}$. Dadurch wird erneut bestätigt, dass das Optimierungsverfahren die Positionsgrößen leichter finden kann als die Winkelgrößen. Für die Ebene zu Strahl-Metrik bestätigen die Abbildungen, dass $\mathbf{P}_{[y]}$, $\mathbf{P}_{[z]}$ und $\mathbf{P}_{[\Phi]}$ kaum bestimmbar sind.

Zusammenfassend zeigt die Punkt zu Punkt-Metrik zwar gute Ergebnisse in den metrischen Fehlerflächen, zeigt jedoch in der Kombination von Positions- und Winkelgrößen einige länglich ausgedehnte Basins. Gleiches gilt für die Punkt zu Strahl-Metrik. Die Anpassung an die Sensorcharakteristika in Form der Ebene zu Strahl-Metrik scheint in den meisten Fällen nur schwer optimierbar zu sein, was die hohe Zahl

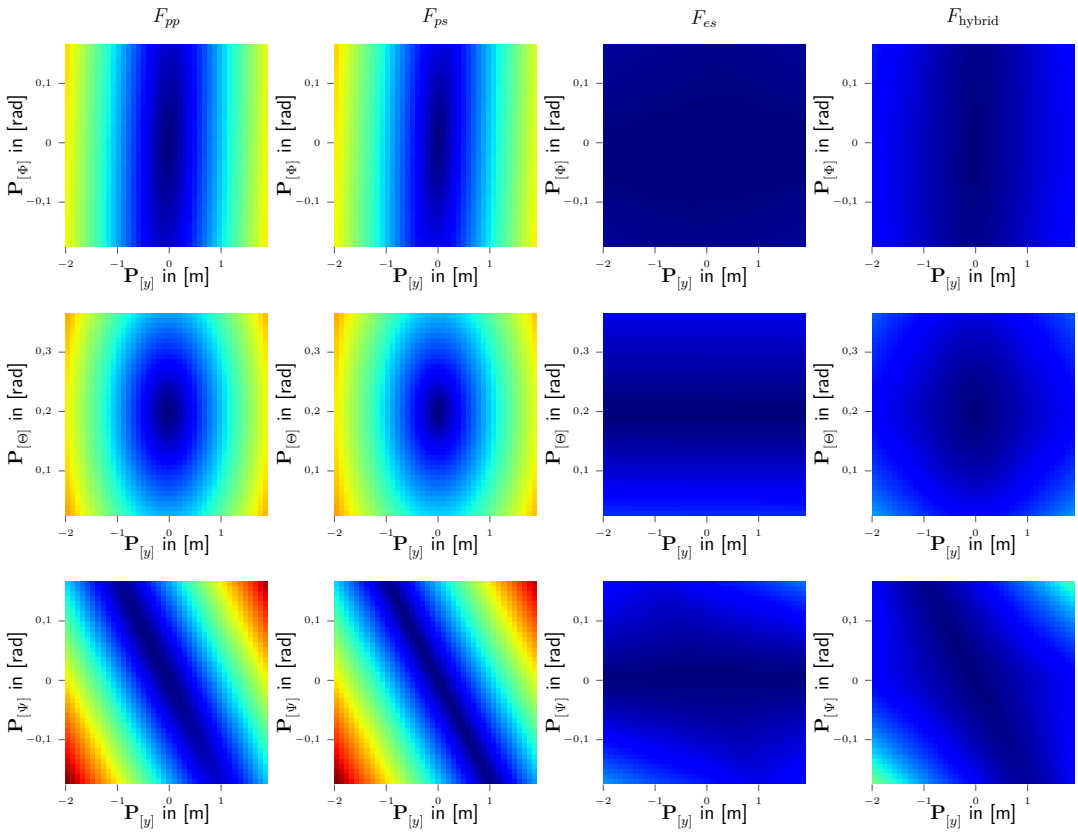


Abbildung 5.26:
y,Φ-Fehlerflächen, y,Θ-Fehlerflächen und y,Ψ-Fehlerflächen der vier Metriken $F_{pp}\,(\mathbf{P})$, $F_{ps}\,(\mathbf{P})$, $F_{es}\,(\mathbf{P})$ und $F_{hybrid}\,(\mathbf{P})$.

an Iterationen und die hohen Positions- und Winkelfehler erklärt. Bei genauerer Betrachtung von $\mathbf{P}_{[y]}$ ggü. $\mathbf{P}_{[\Theta]}$ und $\mathbf{P}_{[\Psi]}$ sowie $\mathbf{P}_{[z]}$ ggü. $\mathbf{P}_{[\Theta]}$ und $\mathbf{P}_{[\Psi]}$ fällt jedoch auf, dass sich in diesen vier Fällen die Punkt zu Punkt- sowie die Ebene zu Strahl-Metrik gut ergänzen. In diesen Fällen finden sich länglich ausgedehnte Basins in beiden Metriken, jedoch sind diese um ca. 60° bis 90° zueinander verdreht, so dass sich in der hybriden Metrik als Ergebnis gleichmäßiger ausgedehnte Basins ergeben. Dadurch findet ein Optimierungsverfahren, welches entlang der Gradienten absteigt, mit weniger Iterationen zum Minimum. Dies erklärt, warum die hybride Metrik im Vergleich auch die geringste Zahl an Iterationen benötigt. Auch wenn die Gradienten der hybriden Metrik im Vergleich zu den anderen Metriken in den meisten betrachteten Fällen geringer ausfallen, so scheint das Optimierungsverfahren dennoch einerseits mit weniger Iterationen und andererseits mit kleinerem Restfehler zum Minimum zu konvergieren, so das die Hybride Metrik letztendlich die besten Ergebnisse zeigt.

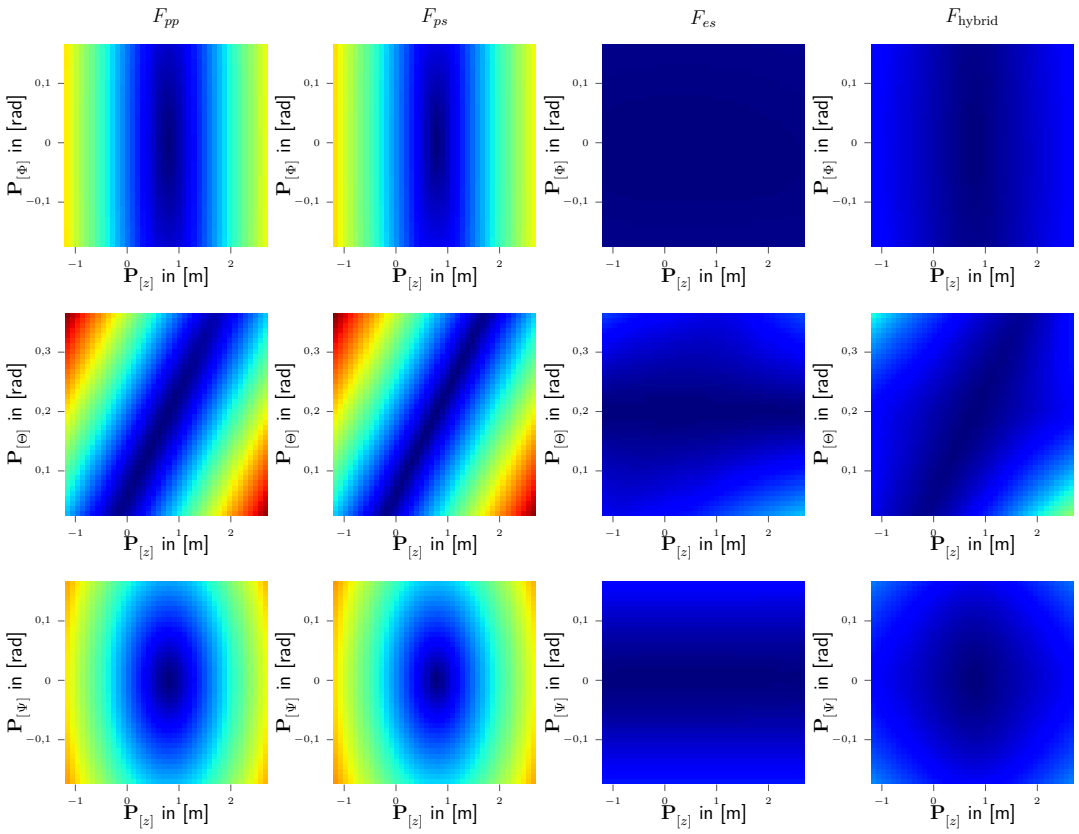


Abbildung 5.27:
z,Φ-Fehlerflächen, z,Θ-Fehlerflächen und z,Ψ-Fehlerflächen der vier Metriken $F_{pp}\,(\mathbf{P})$, $F_{ps}\,(\mathbf{P})$, $F_{es}\,(\mathbf{P})$ und $F_{hybrid}\,(\mathbf{P})$.

An dieser Stelle sei jedoch darauf hingewiesen, dass die Darstellungen starke Vereinfachungen des sechsdimensionalen Raums sind, in dem das Optimierungsverfahren arbeitet. Die dargestellten Fehlerflächen zeigen in den meisten Fällen Minima, obwohl alle anderen Koordinaten des Raums die Werte der Ground-Truth angenommen haben. Es ist zu erwarten, dass sich die Gradienten verschieben, sofern diese Vereinfachung gelockert wird. Außerdem ist anzunehmen, dass bestimmte Effekte einander kompensieren: So könnte bspw. ein leichter Gierwinkel bis zu einem gewissen Grad auch durch eine Translation entlang y kompensiert werden. Aufgrund solcher Effekte ist zu erwarten, dass sich in dem zu optimierenden Raum lokale Minima befinden. Dementsprechend sollte der Startwert der Optimierung nach Möglichkeit bereits möglichst nah am idealen Ergebnis gewählt werden.

5.8 Extrinsische Kalibrierung eines LiDAR und einer Kamera mit Hilfe der Hybriden Metrik

Ausgehend von den bisherigen Untersuchungen werden nun ein Velodyne HDL-64E und eine AVT Guppy F-036C mit Hilfe der Hybriden Metrik extrinsisch kalibriert. Die Hybride Metrik stellt außer Planität und Rechtwinkligkeit keine Anforderungen an den Kalibrierkörper. Zur Vereinfachung der Erkennung des Kalibrierkörpers mit beiden Sensoren kommt eine Schachbretttafel mit einer Breite von 2,64 m und einer Höhe von 1,92 m zum Einsatz. Die Kantenlänge der quadratischen Schachbrettfelder beträgt 0,24 m, wodurch sich ein Muster von 11×8 Feldern ergibt. Beide Sensoren werden durch das in Abschnitt 5.3.2 beschriebene Verfahren synchronisiert. Während und nach der Kalibrierung findet kein Gieren der Kameraplattform statt.

Anstelle synthetischer Daten wird in diesem Fall ein Datenstrom von etwa 1300 Bild-Punktwolken-Paaren aufgezeichnet. In den Daten steht die Kalibriertafel mit unterschiedlichen Gierwinkeln an verschiedenen Positionen, etwa 20 m bis 30 m in x-Richtung entfernt, vor dem Fahrzeug. Das führt zu einer – im Vergleich zum simulativen Versuch – größeren Variation innerhalb der Menge der später berechneten Posen-Paare. Dazu wird die Kalibriertafel von Hand nur sehr langsam bewegt, um verbleibende Ungenauigkeiten der Sensorsynchronisation, aber auch Bewegungsunschärfe zu reduzieren. Die Aufgabe besteht zunächst darin, die Kalibriertafel in den Sensordaten beider Sensoren zu detektieren. Dazu werden, wie im Folgenden beschrieben, die bekannten Größen der Kalibriertafel in ein 3D-Modell überführt und spezielle Verfahren entwickelt, um die 6D-Pose der Tafel sowohl in den Kamerabildern, als auch in den LiDAR-Punktwolken zu finden.

Das Ergebnis sind 1300 Posen-Paare der Kalibriertafel für beide Sensoren. Über die bekannten Abmessungen der Kalibriertafel können alle zur Optimierung mit Hilfe der Hybriden Metrik notwendigen Eckpunkte und Sehstrahlen berechnet werden. Die Optimierung wird dann mit Hilfe des LM-Verfahrens durchgeführt.

5.8.1 Detektion des Kalibrierkörpers in monokularen Kamerabildern

Im vorliegenden Fall wird die verwendete Kamera bereits als intrinsisch und extrinsisch kalibriert angenommen. Die Grundlagen dazu sind in Abschnitt 5.4 beschrieben. Durch die gute Detektierbarkeit von Schachbrettmustern in Kamerabildern existiert eine Vielzahl von Algorithmen, die diese Detektion durchführen. Ein Beispiel dafür ist `findChessboardCorners` aus der OpenCV-Bibliothek. Auch wenn dieses

Verfahren zuverlässig arbeitet, besteht ein großer Nachteil in der benötigten Rechenleistung. Jedes Bild wird individuell nach einem Schachbrett durchsucht, ohne dabei Vorwissen aus den vorhergehenden Bildern zu berücksichtigen.

In der hier beschriebenen Anwendung liegt eine kontinuierliche Sequenz an Schachbrettbildern vor. Das Problem der Schachbrettdetektion wird somit zu einem Tracking-Problem, d.h. dem zeitlichen Verfolgen des Schachbretts in einer Bildsequenz zeitlich aufeinanderfolgender Aufnahmen. Dabei wird das Schachbrett nur langsam und hauptsächlich in der xy-Ebene bewegt. Daher ist die Annahme, dass das Schachbrett im nächsten Bild die Position und Orientierung nur leicht verändert hat, legitim. Der Abgleich mit dem LiDAR verlangt nach einer Lokalisierung des Schachbretts in 3D. Gesucht ist daher die 6D-Pose des Kalibrierkörpers zu jedem Kamerabild und damit – aufgrund der Synchronität mit dem LiDAR – auch jeder Umdrehung des Laserscanners.

Zur Schätzung der relativen 6D-Pose aus Kamerabildern hat sich das Partikelfilter (PF) bewährt [Manz, 2013], welches in Abschnitt 4.6 vorgestellt wurde. Der Zustandsvektor $\boldsymbol{x}$ entspricht der 6D-Pose des Schachbretts

$$\boldsymbol{x} = \mathbf{P}^{\text{Schachbrett}} = \begin{pmatrix} \mathbf{P}_{[x]} \\ \mathbf{P}_{[y]} \\ \mathbf{P}_{[z]} \\ \mathbf{P}_{[\Phi]} \\ \mathbf{P}_{[\Theta]} \\ \mathbf{P}_{[\Psi]} \end{pmatrix}^{\text{Schachbrett}} . \tag{5.65}$$

Da die Kalibriertafel von Menschen bewegt wird, ist es schwierig, ein physikalisches Bewegungsmodell zu finden. Aufgrund der ohnehin langsamen Bewegung wird das Schachbrett als statisch angenommen. Die tatsächliche Bewegung wird folglich über größeres Systemrauschen $\boldsymbol{v}$ kompensiert. Das Systemmodell lautet damit

$$\boldsymbol{x}_k = f\left(\boldsymbol{x}_{k-1}, \boldsymbol{v}_{k-1}\right) = \boldsymbol{x}_{k-1} + \boldsymbol{v}_{k-1}. \tag{5.66}$$

Im Systemrauschen $\boldsymbol{v}_{k-1}$ wird die Ungenauigkeiten des Prozessmodells berücksichtigt. Dieses ergibt sich in jedem Schritt stochastisch aus dem Systemrauschen $\boldsymbol{v}_{k-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$. Die Wahl der Systemrauschmatrix $\mathbf{Q}$ basiert auf den folgenden Überlegungen.

Wie bereits beschrieben, findet die Bewegung der Kalibriertafel hauptsächlich in der xy-Ebene statt, wobei die Tafel in erster Linie auf die Kamera zu bewegt wird. Die Bewegung in y ist deutlich kleiner, noch geringer fällt die Bewegung in z-Richtung aus. Die Ständerkonstruktion der Kalibriertafel hält diese aufrecht, wodurch kaum Roll- oder Nickbewegungen erwartet werden. Hauptsächlich Gierbewegungen werden erwartet. Deshalb wird die Systemfehlerkovarianzmatrix $\mathbf{Q}$ als Diagonalmatrix mit den folgenden Einträgen beschrieben

$$\mathbf{Q} = \operatorname{diag}\left(0{,}1 \;\; 0{,}05 \;\; 0{,}01 \;\; 0{,}0001 \;\; 0{,}0001 \;\; 0{,}01\right) \tag{5.67}$$

$$= \begin{bmatrix} 0{,}1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0{,}05 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0{,}01 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0{,}0001 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0{,}0001 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0{,}01 \end{bmatrix}. \tag{5.68}$$

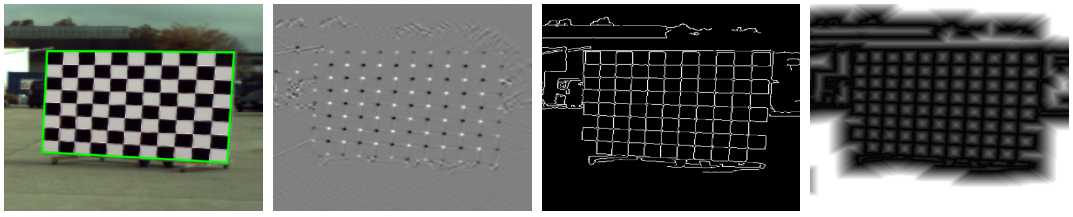
Die Diagonalwerte von $\mathbf{Q}$ sind derart groß gewählt, so dass eine grobe Schätzung der Position der Kalibriertafel als Initialisierung für die Zustandsvariable $\boldsymbol{x}$ ausreicht. Nach der anfänglichen Konvergenz des Filters könnte eine Kovarianzmatrix mit deutlich kleineren Diagonalwerten verwendet werden. Unter Berücksichtigung der Messungen ist es mit der genannten Systemfehlerkovarianzmatrix $\mathbf{Q}$ jedoch bereits möglich, die Kalibriertafel durch die gesamte Sequenz hindurch optisch zu verfolgen, so dass auf die Definition einer weiteren Kovarianzmatrix verzichtet wird.

Die Messung der Kalibriertafel stützt sich, wie eingangs erwähnt, auf die starken Kontraste zwischen den Schachbrettfeldern. Zum Auffinden dieser Bereiche hohen Kontrasts eignen sich in der Bildverarbeitung Ableitungsfilter. Dabei handelt es sich um spezielle diskrete Filterkernel, mit denen eine Faltung des Bildes durchgeführt wird und wobei das Ergebnis dieser Faltung der ersten Ableitung der Bildpunkt-Helligkeitswerte entspricht. Die Sobel-Operatoren sind solche Ableitungskernel. Diese

Arbeit verwendet einen, auf die Sobel-Operatoren zurückgehenden, Ableitungskernel

$$\mathbf{K}^{\text{Sobel}} = \begin{bmatrix} 1 & 4 & 5 & 0 & -5 & -4 & -1 \\ 4 & 16 & 20 & 0 & -20 & -16 & -4 \\ 5 & 20 & 25 & 0 & -25 & -20 & -5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -5 & -20 & -25 & 0 & 25 & 20 & 5 \\ -4 & -16 & -20 & 0 & 20 & 16 & 4 \\ -1 & -4 & -5 & 0 & 5 & 4 & 1 \end{bmatrix}, \tag{5.69}$$

der insbesondere an Ecken im Bild betraglich hohe Antworten liefert. Die Faltung des in Abb. 5.28a gezeigten Bildes mit $\mathbf{K}^{\text{Sobel}}$ führt zu dem in Abb. 5.28b gezeigten Bild. Positive Werte sind dabei in weiß dargestellt, negative Werte in schwarz. Die Kanten zwischen den Schachbrettfeldern sind in Abb. 5.28b zwar vorhanden, allerdings aufgrund der Skalierung des Wertebereichs nicht mehr zu erkennen. Mit Hilfe des Canny-Algorithmus [Canny, 1986] lassen sich die Kanten des Eingangsbildes in Form eines Binärbildes (Abb. 5.28c) extrahieren.



(a) Eingangsbild **(b)** Sobel-Bild **(c)** Canny-Bild **(d)** Kantendistanz

Abbildung 5.28:
Verarbeitungsschritte im Messmodell. (a) zeigt das Eingangsbild mit in grün markiertem Ergebnis. Dessen Sobel-basierte Ableitung ist in (b) dargestellt. Das Kantenbild zeigt (c) und das daraus berechnete Kantendistanzbild ist (d).

Im Rahmen der Filter-Innovation müssen die Gewichte eines jeden Partikels gemäß Gleichung (4.72) bestimmt werden. Dazu wird auf die bekannte Geometrie der Kalibriertafel zur Modellbildung und die Kalibrierung der Kamera zurückgegriffen. Die Menge aller Eckpunkte der Schachbrettfelder bzgl. der Mitte des Schachbretts wird beschrieben durch

$$\mathcal{P}^{\text{Schachbrett}} = \{\boldsymbol{p}^{(1,1)}, \ldots, \boldsymbol{p}^{(12,9)}\}. \tag{5.70}$$

Mit Hilfe der extrinsischen Kalibrierung der Kamera bzgl. des Fahrzeugreferenzpunktes ${}^{\text{Kamera}}\mathbf{H}_{\text{ego}}$ und der Projektionsfunktion $\mathsf{p}(\cdot)$ aus Gleichung (5.33) lässt sich somit jeder Eckpunkt des Schachbretts in das Kamerabild transformieren

$$\mathcal{P}_{\ulcorner,k}^{(i)\,\text{Schachbrett}} = \mathsf{p}\left({}^{\text{Kamera}}\mathbf{H}_{\text{ego}} \cdot \mathit{htm}\left(\boldsymbol{x}_k^{(i)\,*}\right) \cdot \mathcal{P}^{\text{Schachbrett}}\right). \tag{5.71}$$

$\mathit{htm}(\cdot)$ ist, wie in Anhang C.2 beschrieben, eine Funktion, die eine Pose, hier das Partikel $\boldsymbol{x}_k^{(i)\,*}$ in die zugehörige Homogene Transformationsmatrix (HTM) umwandelt, so dass die Matrix-Verkettung in Gleichung (5.71) durchführbar ist. Für jedes Partikel $\boldsymbol{x}_k^{(i)\,*}$ ergibt sich dadurch eine Menge an korrigierten 2D-Bildpunkten bzgl. der linken oberen Bildecke $\mathcal{P}_{\ulcorner,k}^{(i)\,\text{Schachbrett}}$ im aktuellen Kamerabild. Für M Partikel ergeben sich auf diese Weise M Mengen an projizierten Eckpunkten. Die Aufgabe ist nun, anhand dieser Eckpunkte die Gewichte eines jeden einzelnen Partikels zu bestimmen.

Wie in Abb. 5.28b zu sehen, weist das Sobel-Bild I^{Sobel} deutliche Maxima und Minima an den Eckpunkten der Schachbrettfelder auf. Unter der Annahme, dass der tatsächliche Wertebereich von I^{Sobel} auf den Bereich $[0, 1]$ linear skaliert ist, ist die Ecken-Zugehörigkeit eines Bildpunktes definiert als

$$c\left(\mathsf{p}, I_k^{\text{Sobel}}\right) = 2 \cdot \left| I_k^{\text{Sobel}}(\mathsf{p}) - \frac{1}{2} \right|. \tag{5.72}$$

Als Folge daraus ergibt sich die mittlere Eckigkeit eines Partikels durch

$$\bar{c}_k^{(i)} = \frac{1}{12 \cdot 9} \sum_{m=1}^{12} \sum_{n=1}^{9} c\left(\mathsf{p}^{(m,n)}\right),\ \forall\, \mathsf{p}^{(m,n)} \in \mathcal{P}_{\ulcorner,k}^{(i)\,\text{Schachbrett}}. \tag{5.73}$$

Die mittlere Eckigkeit eines Partikels $\bar{c}_k^{(i)}$ besitzt ihr Maximum bei 1, d.h. alle projizierten Eckpunkte fallen tatsächlich auf die Extrema des Sobel-Bildes I^{Sobel}. Daraus berechnet sich die Wahrscheinlichkeit, dass $\boldsymbol{x}_k^{(i)\,*}$ die Lage des gesuchten Schachbretts im Kamerabild beschreibt, durch Auswerten der Gaußverteilung $\mathcal{N}\left(1, \left(\frac{1}{20}\right)^2\right)$ an der Stelle $\bar{c}^{(i)}$

$$p_c\left(I_k^{\text{Sobel}} \middle|\, \boldsymbol{x}_k^{(i)\,*}\right) = \frac{1}{\frac{1}{20} \cdot \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\bar{c}_k^{(i)} - 1}{\frac{1}{20}}\right)^2}. \tag{5.74}$$

Die vergleichweise kleine Varianz von $\left(\frac{1}{20}\right)^2$ wurde empirisch ermittelt. Sie führt dazu, dass nur diejenigen Partikel hohe Gewichte erhalten, deren projizierte Eckpunkte besonders nahe an den Extrema des Sobel-Bildes I^{Sobel} liegen.

Mit diesem Messmodell allein wird das Partikelfilter instabil, da die Partikel häufig in Bereichen landen, in denen die mittlere Eckigkeit nahezu 0 wird. Das Partikelfilter degeneriert. Um diesem Effekt entgegenzuwirken, muss den Partikeln, vereinfach gesagt, der Weg zur optimalen Lösung gezeigt werden. Anders ausgedrückt sollten selbst leichte Unterschiede zwischen den Partikeln zu höheren Partikelgewichten derjenigen Partikel führen, die der optimalen Lösung näher liegen. Das in Abb. 5.28c gezeigte Kantenbild kann dabei behilflich sein. Es illustriert das Ergebnis des Canny-Algorithmus [Canny, 1986], d.h. ein Binärbild, das die kontrastreichen Kanten des Eingangsbildes zeigt. Die Berechnung des Euklid'schen Pixelabstands eines jeden Bildpunktes zu seiner nächsten Kante führt vom Kantenbild zu dem in Abb. 5.28d gezeigten Kantendistanzbild I^{Distanz}.

Je kleiner der Wert eines Bildpunktes im Kantendistanzbild ist, desto näher an bzw. auf einer Kante liegt dieser Bildpunkt. Analog zur Eckigkeit wird die sog. Nicht-Kantigkeit eines Bildpunktes definiert als

$$e_k\left(\mathsf{p}\right) = I_k^{\mathsf{Distanz}}\left(\mathsf{p}\right). \tag{5.75}$$

Die Berechnung der mittleren Nicht-Kantigkeit für die Bildpunktmenge eines Partikels liefert

$$\bar{e}_k^{(i)} = \frac{1}{12 \cdot 9} \sum_{m=1}^{12} \sum_{n=1}^{9} e_k\left(\mathsf{p}^{(m,n)}\right),\ \forall\, \mathsf{p}^{(m,n)} \in \mathcal{P}_{\ulcorner,k}^{(i)\,\mathsf{Schachbrett}}. \tag{5.76}$$

Im Idealfall, d.h. wenn $\boldsymbol{x}_k^{(i)*}$ exakt der Pose der Kalibriertafel entspricht, nimmt die mittlere Nicht-Kantigkeit eines Partikels den Wert 0 an. In diesem Fall liegen alle Eckpunkte genau auf einer Linie des Kantenbildes. Die Betrachtung als Wahrscheinlichkeit, dass $\boldsymbol{x}_k^{(i)*}$ die Lage des gesuchten Schachbretts im Kamerabild beschreibt, erfolgt ähnlich zur Eckigkeit durch Auswerten der Normalverteilung $\mathcal{N}\left(0, \left(\frac{1}{100}\right)^2\right)$ an der Stelle $\bar{e}^{(i)}$:

$$p_e\left(I_k^{\mathsf{Distanz}} |\, \boldsymbol{x}_k^{(i)*}\right) = \frac{1}{\frac{1}{100} \cdot \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\bar{e}_k^{(i)}}{\frac{1}{100}}\right)^2} \tag{5.77}$$

Auch hier wurde die Varianz von $\left(\frac{1}{100}\right)^2$ im Rahmen des Versuchs empirisch ermittelt. Sie sorgt dafür, dass Partikel, deren projizierte Eckpunkte abseits der Bildkanten liegen, geringere Gewichte erhalten.

Das Gesamtgewicht eines jeden Partikels ergibt sich abschließend aus der Verbundwahrscheinlichkeit

$$\tilde{w}_k^{(i)} = p\left(\boldsymbol{y}_k = I_k | \, \boldsymbol{x}_k^{(i)*}\right) = p_c\left(I_k^{\text{Sobel}} | \, \boldsymbol{x}_k^{(i)*}\right) \cdot p_e\left(I_k^{\text{Distanz}} | \, \boldsymbol{x}_k^{(i)*}\right). \tag{5.78}$$

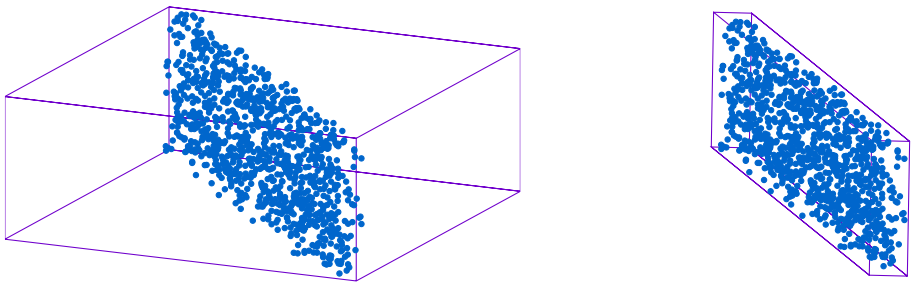
Das genannte Systemmodell und Messmodell beschreiben damit das PF, das zur visuellen Verfolgung der Kalibriertafel Anwendung findet. Mit Hilfe von $M = 1000$ Partikeln ist es damit möglich, die Position der Kalibriertafel in allen 1300 Bildern zu erkennen. Die Pose der Kalibriertafel $\mathbf{P}_{\text{Kamera},k}^{\text{Schachbrett}}$ entspricht dabei in jedem Zeitschritt dem Maximum a-posteriori, gebildet aus $\boldsymbol{x}_k^{\text{MMSE-0,2}}$. Auf Basis dieser Pose und der bekannten Größe der Kalibriertafel lassen sich die für die Hybride Metrik nötigen Eckpunkte und Sehstrahlen berechnen.

5.8.2 Detektion des Kalibrierkörpers in LiDAR-Punktwolken

Der folgende Abschnitt beschreibt die Detektion der Kalibriertafel in den Punktwolken des LiDAR. Da der Kalibrierkörper auf einer weitgehend ebenen Fläche steht, ist das von Himmelsbach et al. [2010] beschriebene Segmentierungsverfahren für 3D-Punktwolken anwendbar. Es bestimmt zunächst die Grundfläche, also Aufstandsfläche, und segmentiert die übrigen Punkte anhand ihrer Distanz zueinander in einzelne Gruppen. Die Menge aller LiDAR-Punkte $\mathcal{P}_{\text{LiDAR},k}$ einer Umdrehung wird damit in die Menge der Grundflächenpunkte $\mathcal{P}_{\text{LiDAR},k}^{\text{Grundfläche}}$ und N_k weitere Untermengen unterteilt, von denen jede ein geometrisch separiertes Objekt repräsentiert

$$\mathcal{P}_{\text{LiDAR},k} = \bigcup_{n=1}^{N_k} \mathcal{P}_{\text{LiDAR},k}^{(n)} \cup \mathcal{P}_{\text{LiDAR},k}^{\text{Grundfläche}}. \tag{5.79}$$

Die Aufgabe besteht darin, diejenige Untermenge $\mathcal{P}_{\text{LiDAR},k}^{(n)}$ zu finden, die dem Schachbrett entspricht. Ein Anhaltspunkt dafür ist die Geometrie einer jeden Untermenge. Jede Punktmenge $\mathcal{P}_{\text{LiDAR},k}^{(n)}$ muss in ihren Abmessungen grob der realen Kalibriertafel entsprechen. Da die Kalibriertafel bzgl. des Koordinatensystems $\mathbf{S}_{\text{LiDAR}}$ möglicherweise verdreht ist, können zum Vergleich nicht einfach die Differenzen der minimalen und maximalen x-, y- oder z-Koordinaten der Punkte verwendet werden. Stattdessen wird die Dimension eines jeden Schachbrettkandidaten entlang der Hauptrichtungen des Punktwolkensegments gesucht. Dieser Unterschied ist in Abb. 5.29 dargestellt.



(a) Ausdehnung bzgl. Koordinatenursprung.

(b) Ausdehnung bzgl. der Hauptachsen.

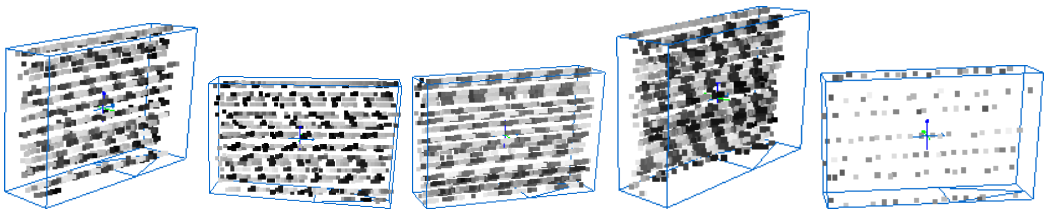
Abbildung 5.29:
Vergleich der Ausdehnung einer Punktwolke bzgl. des Koordinatenursprungs (links) und bzgl. der Hauptachsen der Punktwolke (rechts).

Die Hauptrichtungen entsprechen den Eigenvektoren der Kovarianzmatrix $\boldsymbol{\Sigma}_k^{(n)}$ der Punktwolke um deren Mittelwert $\boldsymbol{\mu}_k^{(n)}$. Die Eigenvektoren spannen eine orthogonale Basis auf, bezüglich der sich die maximalen und minimalen x-, y- und z-Koordinaten durch Hauptachsentransformation der Punkte bestimmen lassen. Die Differenzen zwischen maximalen und minimalen Koordinaten entsprechen schließlich der Ausdehnung der Punktwolke in Länge, Breite und Höhe. Unter Berücksichtigung der realen Abmessungen und zusätzlicher Toleranz dürfen Länge und Breite um nicht mehr als 0,5 m von den entsprechenden Größen der tatsächlichen Kalibriertafel abweichen. In Kombination mit dem nachfolgenden Kriterium, ist diese vereinfachte Art der Klassifikation auf dem zugrunde liegenden Datensatz ausreichend.

Ein weiteres Kriterium ist die Anzahl der 3D-Punkte. Ein Schachbrettkandidat $\mathcal{P}_{\text{LiDAR},k}^{(n)}$ wird verworfen, wenn die Anzahl seiner Punkte kleiner als 300 ist. Dadurch werden vor allem weit entfernte Kandidaten verworfen. Eine Mindestzahl an Punkten ist auch für das dritte Kriterium erforderlich, welches auf optischen Eigenschaften beruht.

Denn wie im visuellen Fall ist auch hier das Schachbrettmuster das deutlichste Differenzierungsmerkmal. Obwohl LiDAR systembedingt farbenblind sind, liefert der hier verwendete Sensor die Reflektivität, d.h. die Intensität des reflektierten Laserlichts. Die weißen Flächen des Schachbretts reflektieren deutlich mehr des ausgesandten Lichts als die schwarzen Flächen. Diese Eigenschaft stellt Abb. 5.30 in Form von Grauwerten dar.

Ausgehend von einer bimodalen Verteilung der Reflektivität derartiger Schachbrett-Punktwolken wird das eine Maximum im Bereich der Reflektivität schwarzer Kacheln und das andere Maximum im Bereich weißer Kacheln erwartet. Um dies zu belegen erfolgt eine testweise Auswertung einzelner Aufnahmen der Kalibriertafel. Dazu



(a) Kalibriertafeln im Nahbereich.

(b) Kalibriertafel in der Ferne.

Abbildung 5.30:
Visualisierung einzelner Punktwolkensegmente, die ein Schachbrett zeigen. Die Reflektivität ist in Form von Grauwerten dargestellt. (b) zeigt eine Kalibriertafel in weiterer Entfernung, wodurch die Punktdichte merklich abnimmt und das Schachbrettmuster nur noch schwer zu erkennen ist.

werden manuell 100 Schachbrett-Punktwolken ausgewählt und die Reflektivitäten aller Punkte dieser Schachbrett-Punktwolken in der Reflektivitätsmenge $\mathcal{R}^{\text{Schachbrett}} = \{r^{(1)}, \ldots, r^{(N)}\}$ zusammengefasst. Abbildung 5.31 zeigt das Histogramm dieser Reflektivitätsmenge.

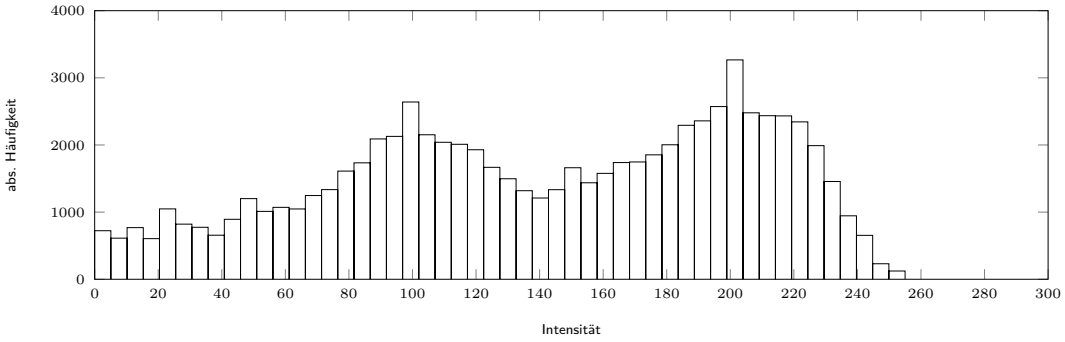


Abbildung 5.31:
Histogramm der Reflektivitäten aller 3D-Punkte der 100 manuell gewählten Schachbrett-Punktwolken.

Das Histogramm zeigt deutliche Extrema für die Reflektivitäten um die Intensitäten 100 und 200 und bestätigt damit die Vermutung einer bimodalen Verteilung. Ein zweikomponentiges Gauß'sches Mischmodell (GMM) mit einer Komponente für jedes der beiden Extrema dient als Modell für den späteren Vergleich mit der Reflektivität eines Kalibrierkörperkandidaten

$$p\left(x\right) = \sum_{l=1}^{2} w^{(l)} \mathcal{N}\left(\mu^{(l)}, \sigma^{2(l)}\right). \tag{5.80}$$

Die beiden Gaußverteilungen sind eindimensional, wodurch die Mischkoeffizienten $w^{(l)}$, die Mittelwerte $\mu^{(l)}$ und die Varianzen $\sigma^{2(l)}$ ebenfalls eindimensional sind. Durch die Mischung zweier Gaußverteilungen wird das Modell durch insgesamt sechs Parameter beschrieben. Für die Mischkoeffizienten gilt außerdem $\sum_{l=1}^{2} w^{(l)} = 1$ und $0 \leq w^{(l)} \leq 1$. Gesucht sind daher die sechs Parameter, die die Wahrscheinlichkeit der Reflektivitätsmessungen maximieren. Dies ist mit Hilfe des EM-Algorithmus (Expectation Maximization) möglich und liefert die in Tabelle 5.3 beschriebenen Werte. Abbildung 5.32 zeigt das dem bereits bekannten Histogramm überlagerte zugehörige Gauß'sche Mischmodell $\text{GMM}^{\text{Schachbrett}}$.

		$w^{(l)}$	$\mu^{(l)}$	$\Sigma^{(l)}$
l	1	0,54	91,04	1787,24
	2	0,46	197,32	652,96

Tabelle 5.3:
Ergebnis des EM-Algorithmus zur Schätzung der Parameter des $\text{GMM}^{\text{Schachbrett}}$.

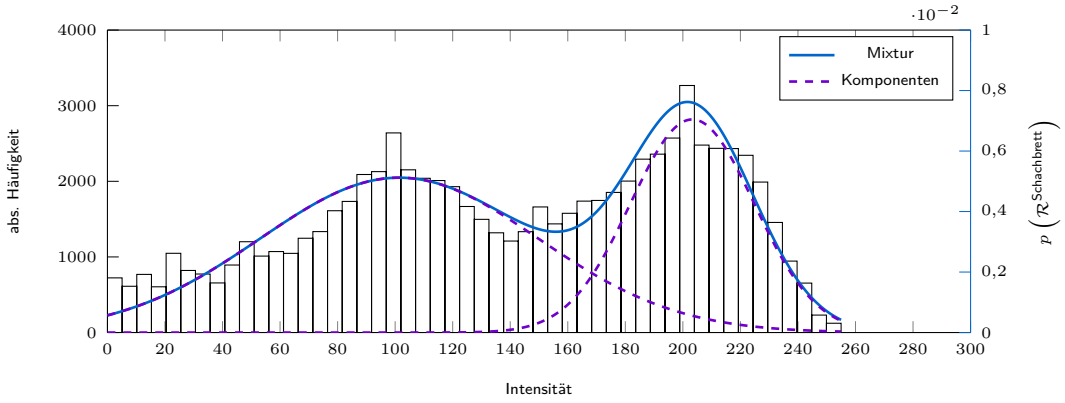


Abbildung 5.32: Histogramm aus Abb. 5.31 überlagert mit dem GMM.

Für jede Punktwolke $\mathcal{P}^{(n)}_{\text{LiDAR},k}$, die den geometrischen Test besteht und mindestens 300 Punkte enthält, wird das $\text{GMM}^{(n)}_{k}$ aus den Reflektivitäten analog zu $\text{GMM}^{\text{Schachbrett}}$ berechnet. Die Punktwolke $\mathcal{P}^{(n)}_{\text{LiDAR},k}$, deren Gauß'sches Mischmodell $\text{GMM}^{(n)}_{k}$ dem $\text{GMM}^{\text{Schachbrett}}$ am ähnlichsten ist, repräsentiert im aktuellen Zeitschritt die Schachbrett-Punktwolke $\mathcal{P}^{\text{Schachbrett}}_{\text{LiDAR},k}$. Analog zu Goldberger et al. [2003]

und Jensen et al. [2007] beruht der Vergleich zweier GMM auf der Kullback-Leibler Divergenz (KL)

$$D^{\text{GMM}}\left(\text{GMM}^{(1)}, \text{GMM}^{(2)}\right) = \sum_{l=1}^{2} w^{(1,l)} \min_{m} \left(KL\left(\mathcal{N}^{(1,l)} || \, \mathcal{N}^{(2,m)}\right) + \log \frac{w^{(1,l)}}{w^{(2,m)}} \right), \tag{5.81}$$

$$\text{mit } \text{GMM}^{(1)} = \left(w^{(1,1)}, w^{(1,2)}, \mu^{(1,1)}, \mu^{(1,2)}, \Sigma^{(1,1)}, \Sigma^{(1,2)}\right)^T, \tag{5.82}$$

$$\text{GMM}^{(2)} = \left(w^{(2,1)}, w^{(2,2)}, \mu^{(2,1)}, \mu^{(2,2)}, \Sigma^{(2,1)}, \Sigma^{(2,2)}\right)^T, \tag{5.83}$$

$$\mathcal{N}^{(1,l)} = \mathcal{N}\left(\mu^{(1,l)}, \Sigma^{(1,l)}\right) \tag{5.84}$$

$$\text{und } \mathcal{N}^{(2,m)} = \mathcal{N}\left(\mu^{(2,m)}, \Sigma^{(2,m)}\right). \tag{5.85}$$

Analog zur Pose des Kalibrierkörpers bzgl. der Kamera $\mathbf{P}^{\text{Schachbrett}}_{\text{Kamera},k}$ ist zu jedem Zeitschritt die Pose bzgl. des Laserscanners $\mathbf{P}^{\text{Schachbrett}}_{\text{LiDAR},k}$ notwendig. Die zugehörige HTM lässt sich aus dem Mittelwert $\boldsymbol{\mu}^{\text{Schachbrett}}_{k}$ und der zugehörigen Kovarianzmatrix $\boldsymbol{\Sigma}^{\text{Schachbrett}}_{k}$, die bereits aus der Berechnung des GMM bekannt ist, wie folgt bestimmen

$${}^{\text{LiDAR}}\mathbf{H}_{\text{Schachbrett},k} = \begin{bmatrix} & & & 0 \\ \boldsymbol{\Sigma}^{\text{Schachbrett}}_{k} & & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 & -\boldsymbol{\mu}^{\text{Schachbrett}}_{[x]\,k} \\ 0 & 1 & 0 & -\boldsymbol{\mu}^{\text{Schachbrett}}_{[y]\,k} \\ 0 & 0 & 1 & -\boldsymbol{\mu}^{\text{Schachbrett}}_{[z]\,k} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.86}$$

Daraus ergibt sich $\mathbf{P}^{\text{Schachbrett}}_{\text{LiDAR},k}$ entsprechend der Gleichungen in Anhang C. Abbildung 5.33 zeigt beispielhaft eine der 1300 erkannten Schachbrett-Punktwolken mit der zugehörigen Pose.

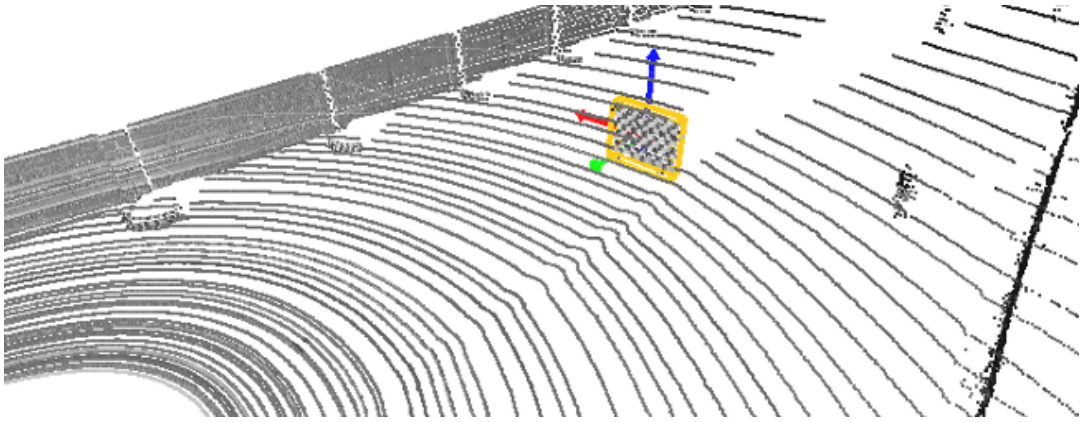


Abbildung 5.33:
Beispiel einer Schachbrettwand, die in der Punktwolke des LiDAR detektiert wurde.

Die Ergebnisse der Kalibrierung sind beispielhaft anhand eingefärbter Punktwolken in Abschnitt 6.1.1 dargestellt.

5.9 Odometrie-basierte extrinsische Kalibrierung

Die bislang vorgestellten Kalibrierungsverfahren setzen alle in der ein oder anderen Form auf real-weltige Korrespondenzen, wie Ecken, Kanten oder spezielle Kalibrierkörper. Neben Metriken, wie in Abschnitt 5.6 beschrieben, die spezielle Sensorcharakteristika berücksichtigen, wird ebenfalls versucht, die Korrespondenzen selbst sensorcharakteristisch zu beschreiben. Ein Beispiel hierfür sind die zahllosen Schlüsselpunkt-basierten Verfahren zur Extraktion und Beschreibung von Ecken in Bildern, wie Kanade Lucas Tomasi Feature Tracker (KLT) [Lucas und Kanade, 1981], Harris-Ecken [Harris und Stephens, 1988], Scale-Invariant Feature Transform (SIFT) [Lowe et al., 2004], Speeded Up Robust Features (SURF) [Bay et al., 2006], Features from Accelerated Segment Test (FAST) [Rosten und Drummond, 2006], SidCell [Schweitzer und Wuensche, 2009], Binary Robust Independent Elementary Features (BRIEF) [Calonder et al., 2010] oder Oriented FAST and Rotated BRIEF (ORB) [Rublee et al., 2011]. Für Tiefenbilder eines LiDAR wurden derartige Algorithmen ebenfalls angewandt [Toth et al., 2010]. Gleiches gilt für LiDAR-Intensitätsbilder [McManus et al., 2011].

In einigen Fällen ist es allerdings nicht ohne weiteres möglich, real-weltige Korrespondenzen zwischen Sensoren zu finden. Als Beispiel seien hier Sensoren mit nicht-überlappenden Sichtfeldern genannt. Zwar gibt es Ansätze, wie den von Li et al. [2013], die mit einem Kalibrierkörper den Bezug zwischen den Sichtfeldern herstellen. Doch einerseits handelt es sich dabei um ein sog. Batch-Verfahren, bei dem die Kalibrierung nicht kontinuierlich verbessert wird, sondern in einzelnen Iterationen rückwirkend berechnet wird. Zum anderen erwartet das Verfahren, dass der Kalibrierkörper von beiden zu kalibrierenden Sensoren aus gleichzeitig sichtbar ist, was besonders für entgegengesetzt gerichtete Kameras schwierig ist. Dem letzteren Problem begegnen Heng et al. [2013] durch den Verzicht auf einen Kalibrierkörper. Dennoch handelt es sich auch hier um ein Batch-Verfahren.

Ein weiteres Beispiel für eine schwer kalibrierbare Sensorkonfigurationen sind Kamera und Inertial Navigation System (INS). Ein INS, wie in Abschnitt 3.2 beschrieben, ist kein bildgebender Sensor. Daher kann eine Korrespondenz nur basierend auf Beschleunigungen, Positionen oder daraus abgeleiteten Größen getroffen werden.

Das entsprechende Verfahren für bildbasierte Sensoren nennt sich Visuelle Odometrie. Dabei wird versucht, die Bewegung einer Kamera oder eines Roboters rein kamerabasiert zu bestimmen. Der Begriff wurde als erstes durch Nistér et al. [2004] geprägt und innerhalb der letzten Jahre u.a. von Scaramuzza und Siegwart [2008] und Engel et al. [2013] weiterentwickelt. Parallel dazu wurden die Methoden der Visuellen Odometrie auch auf andere Sensortypen, wie LiDAR, übertragen [Dong und Barfoot, 2014]. Auch die Entwicklung von *online*-SLAM-Verfahren (Simultaneous Localization and Mapping) verhalf zu der Möglichkeit, die Bewegung verschiedener Sensoren zeitdiskret in Form von Posen zu beschreiben.

Vor diesem Hintergrund stellt sich die Frage, ob die extrinsische Kalibrierung verschiedener Sensoren auch rein Odometrie-basiert durchgeführt werden kann. Diese Frage wird in den folgenden Kapiteln untersucht und basiert dabei auf der Publikation von Schneider et al. [2013].

5.9.1 Sensorkonfiguration und Delta-Posen

Ausgangspunkt für die folgende Untersuchung ist die in Abb. 5.34 dargestellte Sensorkonfiguration eines Fahrzeugs mit INS und Kamera. Die gesuchte extrinsische Kalibrierung zwischen beiden Sensoren ist hier durch die homogene Transformationsmatrix ${}^{\text{Kamera}}\mathbf{H}_{\text{INS}}$ dargestellt. Das Fahrzeug in diesem Beispiel bewegt sich zwischen den beiden Zeitpunkten $k-1$ und k. Die Bewegung der beiden Sensoren wird durch die homogenen Transformationsmatrizen ${}^{\text{INS},k}\mathbf{H}_{\text{INS},k-1}$ und ${}^{\text{Kamera},k}\mathbf{H}_{\text{Kamera},k-1}$ beschrieben.

Jeder Sensor beobachtet seine eigene Bewegung individuell. Eine in Fahrtrichtung gerichtete Kamera berechnet bei Geradeausfahrt eine Bewegung in x-Richtung. Für eine im Uhrzeigersinn um 90° gegierte Kamera entspricht die Geradeausfahrt des Fahrzeugs einer Sensorbewegung in y-Richtung. Nur wenn es möglich wäre, beide Sensoren räumlich identisch zu positionieren und zu orientieren, würde auch von beiden Sensoren die Bewegung identisch erfasst. In diesem Fall gilt ${}^{\text{INS},k}\mathbf{H}_{\text{INS},k-1} = {}^{\text{Kamera},k}\mathbf{H}_{\text{Kamera},k-1}$. Andernfalls gilt ${}^{\text{INS},k}\mathbf{H}_{\text{INS},k-1} \neq {}^{\text{Kamera},k}\mathbf{H}_{\text{Kamera},k-1}$. Die zu beiden homogenen Transformationsmatrizen äquivalenten Posen ${}^{\text{INS},k}\mathbf{P}_{\text{INS},k-1}$ und ${}^{\text{Kamera},k}\mathbf{P}_{\text{Kamera},k-1}$ werden als Delta-Posen der beiden Sensoren zwischen den Zeitpunkten $k-1$ und k bezeichnet und werden von nun an als Messungen betrachtet.

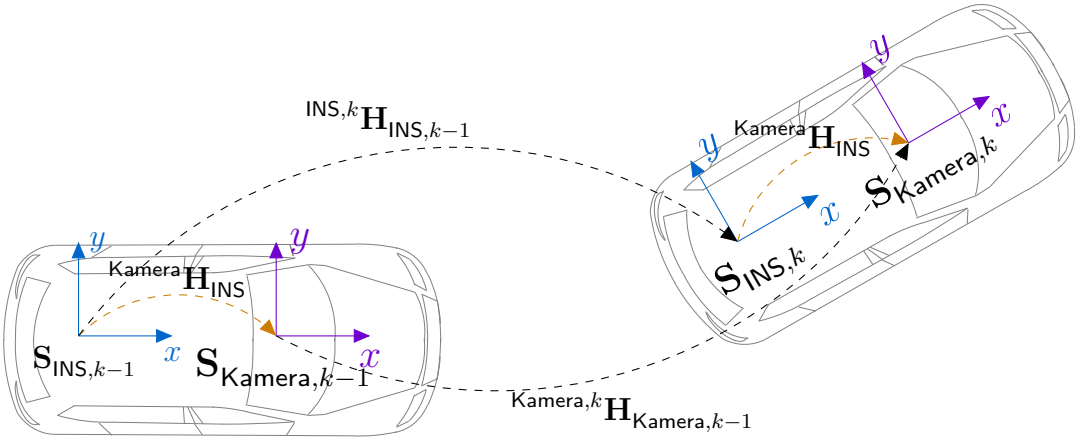


Abbildung 5.34:
Beispielhafte Sensorkonfiguration eines Fahrzeugs mit INS und Kamera zu zwei Zeitpunkten $k-1$ und k.

5.9.2 Schätzung der extrinsischen Kalibrierung auf Basis von Delta-Posen

Die Schätzung der Relativpose zwischen INS und Kamera basiert auf dem in Abschnitt 4.5 vorgestellten Unscented Kalmanfilter (UKF). Die folgende Untersuchung basiert auf drei aufeinander aufbauenden Varianten, mit dem Ziel der Schätzung der Relativpose zwischen INS und Kamera.

Variante 1 Beide Sensoren sind fest am Fahrzeug verbaut, so dass die gesuchte extrinsische Kalibrierung ${}^{\text{Kamera}}\mathbf{H}_{\text{INS}}$ als nahezu konstant über der Zeit angesehen werden kann. Filtervariante 1 entspricht damit einem Parameterschätzer, der die zu ${}^{\text{Kamera}}\mathbf{H}_{\text{INS}}$ äquivalenten sechs Posenparameter schätzt. Der Zustandsvektor $\boldsymbol{x}$ besteht demnach aus eben diesen sechs Posenparametern

$$\boldsymbol{x} = (x, y, z, \Phi, \Theta, \Psi)^T . \tag{5.87}$$

Die einzige Veränderung an $\boldsymbol{x}$ ergibt sich durch die Schätzung, innerhalb der sich der Zustand, ausgehend von seinem Initialwert, der wahren gesuchten Relativpose annähert. Das Systemmodell $f(\cdot)$ benötigt daher keinen Steuervektor $\boldsymbol{u}_{k-1}$ und ist ebenfalls zeitunabhängig. Ein solches Systemmodell wird in der Literatur auch Identität genannt

$$\boldsymbol{x}_k^* = f\left(\boldsymbol{x}_{k-1}\right) = \boldsymbol{x}_{k-1}. \tag{5.88}$$

Wie eingangs beschrieben sind beide Sensoren in der Lage, zwischen zwei diskreten Zeitpunkten ihre Delta-Posen ${}^{\text{INS},k}\mathbf{P}_{\text{INS},k-1}$ und ${}^{\text{Kamera},k}\mathbf{P}_{\text{Kamera},k-1}$ zu bestimmen. Dies sind die Messungen des vorgestellten Systems.

Die Aufgabe des Messmodells ist nun, anhand von $\boldsymbol{x}_k^*$ diese Messungen vorherzusagen. Allerdings kann auf Basis von $\boldsymbol{x}_k^*$ allein, keine der Messungen direkt vorhergesagt werden. Bei Betrachtung von Abb. 5.34 fällt jedoch auf, dass jeweils eine der Messungen auf Basis von $\boldsymbol{x}_k^*$ bzw. dazu korrespondierend ${}^{\text{Kamera}}\mathbf{H}_{\text{INS},k}^*$ und der jeweils anderen Messung vorhergesagt werden kann.

Es ist zu erwarten, dass die Messungen des INS, d.h. ${}^{\text{INS},k}\mathbf{P}_{\text{INS},k-1}$, im Gegensatz zu denen der Kamera, d.h. ${}^{\text{Kamera},k}\mathbf{P}_{\text{Kamera},k-1}$, weniger stark verrauscht sind. Deshalb betrachtet diese erste Filtervariante ${}^{\text{INS},k}\mathbf{P}_{\text{INS},k-1}$ als konstanten, rauschfreien Parametervektor $\boldsymbol{p}_k$ des Messmodells. Damit lässt sich die folgende Messgleichung aufstellen, die Anhand von $\boldsymbol{x}_k^*$ und $\boldsymbol{p}_k = {}^{\text{INS},k}\mathbf{P}_{\text{INS},k-1}$ die Messung ${}^{\text{Kamera},k}\mathbf{P}_{\text{Kamera},k-1}^*$ vorhersagt:

$$g\left(\boldsymbol{x}_k^*, \boldsymbol{p}_k\right) = \mathit{pose}\left(\mathit{htm}\left(\boldsymbol{x}_k^*\right) \cdot \mathit{htm}\left(\boldsymbol{p}_k\right) \cdot \mathit{htm}\left(\boldsymbol{x}_k^*\right)^{-1}\right). \tag{5.89}$$

Auch wenn das Rauschen von ${}^{\text{INS},k}\mathbf{P}_{\text{INS},k-1}$ ignoriert wird, bietet das UKF an dieser Stelle natürlich weiterhin die Möglichkeit, ein Messrauschen $\mathbf{R}_k$ zu setzen. Um damit das Rauschen von ${}^{\text{INS},k}\mathbf{P}_{\text{INS},k-1}$ zumindest ein Stück weit zu berücksichtigen, kann $\mathbf{R}_k$ bspw. größer gewählt werden. Im Abschnitt 5.9.3 wird das Messrauschen nochmals behandelt.

Variante 2 Die Filtervariante 2 verfolgt das Ziel, das Rauschen von ${}^{\text{INS},k}\mathbf{P}_{\text{INS},k-1}$ besser miteinzubeziehen. Anders ausgedrückt, sollen die System- und Messgleichungen der Variante 1 derart verändert werden, sodass das Rauschen des INS berücksichtigt wird. Weiterhin besteht das Problem, dass sich die vorhergesagte Visuelle Odometrie ${}^{\text{Kamera},k}\mathbf{P}_{\text{Kamera},k-1}^*$ nur mit Hilfe von ${}^{\text{INS},k}\mathbf{P}_{\text{INS},k-1}$ und der geschätzten extrinsischen Kalibrierung, d.h. den Zustandsgrößen $\boldsymbol{x}$ aus Variante 1, berechnen lässt.

Zur Lösung dieses Problems sollen die Posenparameter von ${}^{\text{INS},k}\mathbf{P}_{\text{INS},k-1}$ von nun an als Steuergrößen $\boldsymbol{u}_{k-1}$ interpretiert werden. Damit ist ${}^{\text{INS},k}\mathbf{P}_{\text{INS},k-1}$ bereits im Prozessmodell verfügbar. Gleichzeitig wird der Zustandsvektor $\boldsymbol{x}$ um weitere Zustandsgrößen erweitert, die den Posenparametern der Visuellen Odometrie entsprechen. Der Zu-

standsvektor der Variante 2 besteht damit aus einem extrinsischen Anteil $\boldsymbol{x}^{\text{extr}}$ und einem Anteil für die Visuelle Odometrie $\boldsymbol{x}^{\text{odom}}$,

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{x}^{\text{odom}} \\ \boldsymbol{x}^{\text{extr}} \end{pmatrix} \text{ mit} \tag{5.90}$$

$$\boldsymbol{x}^{\text{odom}} = \left(x^{\text{odom}}, y^{\text{odom}}, z^{\text{odom}}, \Phi^{\text{odom}}, \Theta^{\text{odom}}, \Psi^{\text{odom}}\right)^T \text{ und} \tag{5.91}$$

$$\boldsymbol{x}^{\text{extr}} = \left(x^{\text{extr}}, y^{\text{extr}}, z^{\text{extr}}, \Phi^{\text{extr}}, \Theta^{\text{extr}}, \Psi^{\text{extr}}\right)^T. \tag{5.92}$$

Das Systemmodell für $\boldsymbol{x}^{\text{extr}}$ bleibt, wie schon in Variante 1, die Identität

$$f^{\text{extr}}\left(\boldsymbol{x}_{k-1}^{\text{extr}}\right) = \boldsymbol{x}_{k-1}^{\text{extr}}. \tag{5.93}$$

Das Systemmodell für $\boldsymbol{x}^{\text{odom}}$ beschreibt, wie sich die Parameter der Visuellen Odometrie in Form von Zustandsgrößen aus $\boldsymbol{x}^{\text{extr}}$ und ${}^{\text{INS},k}\mathbf{P}_{\text{INS},k-1}$ berechnen lassen

$$\boldsymbol{u}_{k-1} = {}^{\text{INS},k}\mathbf{P}_{\text{INS},k-1} \text{ und} \tag{5.94}$$

$$f^{\text{odom}}\left(\boldsymbol{x}_{k-1}^{\text{extr}}, \boldsymbol{u}_{k-1}\right) = \mathit{pose}\left(\mathit{htm}\left(\boldsymbol{x}_{k-1}^{\text{extr}}\right) \cdot \mathit{htm}\left(\boldsymbol{u}_{k-1}\right) \cdot \mathit{htm}\left(\boldsymbol{x}_{k-1}^{\text{extr}}\right)^{-1}\right). \tag{5.95}$$

Zu bemerken ist hierbei, dass die Visuelle Odometrie nun in zweierlei Form vorkommt. Einerseits ist sie, wie schon in Variante 1, weiterhin eine Messung mit entsprechendem Messrauschen. Andererseits ist sie nun auch Teil des Zustands und wird im Rahmen der Systemgleichungen u.a. auf Basis der INS-Odometrie berechnet. Der Vorteil dieser Darstellung ergibt sich bei genauerer Betrachtung des Prozessrauschens, welches jetzt 12-dimensional ist.

Für $\boldsymbol{x}^{\text{extr}}$ handelt es sich bei Variante 2 weiterhin um einen Parameterschätzer, d.h. die entsprechenden Kovarianzen der Prozessstörungskovarianzmatrix sind besonders klein gewählt. Die Berechnung von $\boldsymbol{x}^{\text{odom}}$ hingegen hängt nun einerseits von den quasi-konstanten und nur geringfügig verrauschten Zustandsgrößen $\boldsymbol{x}^{\text{extr}}$ ab, und der im Vergleich stärker verrauschten INS-Odometrie ${}^{\text{INS},k}\mathbf{P}_{\text{INS},k-1}$. Entsprechend ermöglicht uns die Prozessstörungskovarianzmatrix über die zu $\boldsymbol{x}^{\text{odom}}$ korrespondierenden Kovarianzen die Modellierung des Rauschens von ${}^{\text{INS},k}\mathbf{P}_{\text{INS},k-1}$. Sei $\mathbf{Q}_{k-1}^{\text{extr}}$ das Prozessrauschen der extrinsischen Parameter $\boldsymbol{x}^{\text{extr}}$ und $\mathbf{Q}_{k-1}^{\text{odom}}$ das aus dem

Rauschen von ${}^{\mathsf{INS},k}\mathbf{P}_{\mathsf{INS},k-1}$ abgeleitete Prozessrauschen von $\boldsymbol{x}^{\mathsf{odom}}$, so ergibt sich die vollständige Prozessstörungskovarianzmatrix der Variante 2 als

$$\mathbf{Q}_{k-1} = \begin{bmatrix} \mathbf{Q}_{k-1}^{\mathsf{odom}} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_{k-1}^{\mathsf{extr}} \end{bmatrix}. \tag{5.96}$$

Darüber hinaus vereinfacht die Aufnahme der visuallen Odometrie in den Zustandsvektor die Messgleichungen, da die prädizierte Messung nun Teil des prädizierten Zustands, also $\boldsymbol{x}^{\mathsf{odom}\,*}$, ist:

$$g\left(\boldsymbol{x}_k^*\right) = \boldsymbol{x}_k^{\mathsf{odom}\,*} \tag{5.97}$$

Im Vergleich zur Variante 1 kann allerdings auch in dieser Variante 2 das Rauschen der INS-Messung nicht direkt angegeben werden. Allerdings stellt sie einen wichtigen Schritt hin zur dritten und letzten Variante dar.

Variante 3 In Variante 2 wird das Rauschen der INS-Messung nicht direkt gesetzt, sondern nur in abgeleiteter Form bzgl. der im Zustand enthaltenen Visuellen Odometrie $\boldsymbol{x}^{\mathsf{odom}}$. Zur Verdeutlichung dieses Sachverhalts wird noch einmal auf das Eingangsbeispiel der beiden 90°-ausgerichteten Sensoren verwiesen. Angenommen einer der beiden Sensoren unterläge starkem Rauschen entlang seiner x-Richtung. Nach der Prädiktion der Odometrie entspräche dies aufgrund der Rotation einem besonders starken Rauschen in y-Richtung bzgl. des zweiten Sensors. Gesucht ist daher ein Weg, um das Rauschen der INS-Messung in Prozessrauschen bzgl. $\boldsymbol{x}^{\mathsf{odom}}$ transformieren zu können.

Das Prozessrauschen der Variante 2 wird, wie in Gleichung (5.96) gezeigt, aufgrund der Dimension des Zustandsvektors durch eine 12×12-Matrix definiert. Das Rauschen des Odometrieanteils des Zustands $\boldsymbol{x}^{\mathsf{odom}}$ ist dabei im Wesentlichen nur vom linken, oberen 6×6-Abschnitt des Prozessrauschens bestimmt. Analog zum Steuervektor $\boldsymbol{u}_{k-1}$, der der Delta-Pose des INS entspricht, sei $\mathbf{R}_{k-1}^{\mathsf{INS}}$ die zugehörige 6×6-Kovarianzmatrix des INS. Auch hier wird konventionsbedingt der Zeitindex $k-1$ verwendet, auch wenn an dieser Stelle vom Sensorrauschen zur aktuellen INS-Messung die Rede ist.

Die Frage ist nun, wie die Kovarianzmatrix $\mathbf{R}_{k-1}^{\mathsf{INS}}$ korrekt durch das Prozessmodell $f^{\mathsf{odom}}\left(\cdot\right)$ in Prozessrauschen überführt werden kann. Wie sich beim Blick auf die Unscented Transform (UT) des UKF aus Gleichung (4.39) bzw. deren Ver-

wendung in Gleichung (4.50) zeigt, ist die Lösung eines solchen Problems bereits bekannt. Zusammengefasst werden im Rahmen der UT auf Basis der Filter-internen Schätzfehlerkovarianzmatrix $\mathbf{P}_{k-1}$ die sog. Sigmapunkte um den alten korrigierten Zustand $\hat{\boldsymbol{x}}_{k-1}$ berechnet, durch das Prozessmodell transformiert und anschließend zum prädizierten Zustand und, unter additiver Beaufschlagung des Prozessrauschens, zu einer prädizierten internen Schätzfehlerkovarianzmatrix verrechnet. Die Steuergrößen $\boldsymbol{u}_{k-1}$ bleiben dabei für jeden Sigmapunkt fix.

Wie sich herausstellt, kann die UT ebenfalls zur Überführung des INS-Rauschens in Prozessrauschen verwendet werden. Dazu wird im UKF-Algorithmus vor Gleichung (4.50) der folgende Aufruf eingefügt

$$\left\{\cdot, \hat{\mathbf{Q}}_{k-1}, \cdot, \cdot\right\} = ut\left(\boldsymbol{u}_{k-1}, \mathbf{R}_{k-1}^{\mathsf{INS}}, f_{k-1}\left(\hat{\boldsymbol{x}}_{k-1}, \cdot\right), \mathbf{D}_{k-1}\,\mathbf{Q}_{k-1}\,\mathbf{D}_{k-1}^{T}\right) \tag{5.98}$$

und das Prozessrauschen in Gleichung (4.50) durch das derart gewonnene neue Prozessrauschen $\hat{\mathbf{Q}}_{k-1}$ ersetzt:

$$\{\boldsymbol{x}_k^*, \mathbf{P}_k^*, \cdot, \cdot\} = ut\left(\hat{\boldsymbol{x}}_{k-1}, \mathbf{P}_{k-1}, f_{k-1}\left(\cdot, \boldsymbol{u}_{k-1}\right), \hat{\mathbf{Q}}_{k-1}\right) \tag{5.99}$$

Letztendlich findet in Gleichung (5.98) ein Rollentausch zwischen der Zustandsvariablen $\hat{\boldsymbol{x}}_{k-1}$ mit Schätzfehlerkovarianzmatrix $\mathbf{P}_{k-1}$ und den Steuergrößen $\boldsymbol{u}_{k-1}$ mit zugehöriger Kovarianzmatrix $\mathbf{R}_{k-1}^{\mathsf{INS}}$ statt. Im Rahmen dieses Funktionsaufrufs werden die Sigmapunkte abhängig von $\mathbf{R}_{k-1}^{\mathsf{INS}}$ nun um den Vektor der Steuergrößen $\boldsymbol{u}_{k-1}$ statt um den Zustand berechnet und durch das Systemmodell transformiert. Auch wenn die Sigmapunkte sechsdimensional sind, ergeben sich bei dieser Transformation prädizierte Sigmapunkte $\boldsymbol{\mu}^{(i)\mathsf{y}}$ mit der Dimension 12 aufgrund des Systemmodells. Bei genauer Betrachtung des Systemmodells fällt dabei auf, dass der extrinsische Teil $\boldsymbol{x}^{\mathsf{extr}}$ nicht von den Steuergrößen abhängt. Die Berechnung der Kovarianzmatrix in Gleichung (4.49) führt daher, vor der Addition von $\boldsymbol{\Sigma}^{\mathsf{gg}}$, d.h. $\mathbf{D}_{k-1}\,\mathbf{Q}_{k-1}\,\mathbf{D}_{k-1}^{T}$, zu einer 12×12-Matrix mit Rang sechs, in der lediglich der linke, obere 6×6-Bereich von 0 verschiedene Werte aufweist. Dieser Bereich allerdings enthält das gesuchte transformierte Rauschen der Steuergrößen $\hat{\mathbf{R}}_{k-1}^{\mathsf{INS}}$. Zusammen mit $\boldsymbol{\Sigma}^{\mathsf{gg}}$ wird daraus $\hat{\mathbf{Q}}_{k-1}$, das im weiteren Ablauf der Filtergleichungen an die Stelle des Prozessrauschens tritt.

Im Folgenden sind die einzelnen Schritte des Aufrufs von Gleichung (5.98) nochmals genauer beleuchtet. Zunächst findet die Berechnung der Sigmapunkte $\boldsymbol{\mu}^{(i)\times}$ um die gemessene Delta-Pose der INS-Messung statt. Die Streuung der Sigmapunkte basiert

auf der Kovarianzmatrix $\mathbf{R}_{k-1}^{\mathsf{INS}}$, die dem Rauschen des INS entspricht. Die Prädiktion des vorangegangen Zustands $\hat{\boldsymbol{x}}_{k-1}$ mit diesen Sigmapunkten in Form von Steuergrößen in Gleichung (4.43) liefert nun 12-dimensionale prädizierte Sigmapunkte $\boldsymbol{\mu}^{(i)\,\mathsf{y}}$. Jeder Sigmapunkt $\boldsymbol{\mu}^{(i)\,\mathsf{y}}$ basiert auf der gleichen Zustandsvariablen und der gleichen Abbildungsfunktion. Letztlich sind die Unterschiede zwischen diesen Sigmapunkten nur durch $\mathbf{R}_{k-1}^{\mathsf{INS}}$ verursacht. Durch Berechnung der Kovarianzmatrix $\hat{\mathbf{R}}_{k-1}^{\mathsf{INS}}$ über alle $\boldsymbol{\mu}^{(i)\,\mathsf{y}}$ findet eine Approximation der Gaußverteilung mit Mittelwert $\boldsymbol{\mu}^{\mathsf{y}}$ auf Basis der prädizierten Sigmapunkte $\boldsymbol{\mu}^{(i)\,\mathsf{y}}$ statt. Unter der Annahme, dass $\boldsymbol{\mu}_{[0:5]}^{\mathsf{y}} = \boldsymbol{x}_k^{\mathsf{odom}\,*}$ gilt, entspricht $\hat{\mathbf{R}}_{k-1}^{\mathsf{INS}}$ dem gesuchten transformierten Rauschen der INS-Messung.

Vor dem experimentellen Vergleich der drei Varianten wird an dieser Stelle noch näher auf den Term $\mathbf{D}_{k-1}\ \mathbf{Q}_{k-1}\ \mathbf{D}_{k-1}^T$ eingegangen. $\mathbf{Q}_{k-1}$ ist in diesem Zusammenhang eine Prozessstörungskovarianzmatrix, die die Unsicherheit im extrinsischen Teil des Systemmodells angibt. Das entspricht dem in Gleichung (5.93) beschriebenen Teil des Prozessmodells. Damit ist $\mathbf{Q}_{k-1}$ ebenfalls sechsdimensional. Die Transformationsmatrix $\mathbf{D}_{k-1}$ wird an dieser Stelle dazu verwendet, $\mathbf{Q}_{k-1}$ in eine zwölfdimensionale Matrix zu transformieren, die, wie zuvor beschrieben, mit der transformierten Steuergrößenkovarianzmatrix additiv gekoppelt wird. $\mathbf{D}_{k-1}$ wird daher derart gewählt, dass sich nach dieser Kopplung die folgende Struktur für die Kovarianzmatrix $\hat{\mathbf{Q}}_{k-1}$ ergibt

$$\hat{\mathbf{Q}}_{k-1} = \begin{bmatrix} \hat{\mathbf{R}}_{k-1}^{\mathsf{INS}} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_{k-1} \end{bmatrix}, \text{ sowie} \tag{5.100}$$

$$\mathbf{D}_{k-1}^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.101}$$

5.9.3 Vergleich der Varianten durch Simulation

Dieser Abschnitt zeigt einen simulativen Vergleich der im vorherigen Kapitel vorgestellten drei Varianten zur extrinsischen Kalibrierung zweier Sensoren auf Basis ihrer Eigenbewegungen.

Grundlage aller Simulationen ist eine extrinsische Kalibrierung, die als Ground-Truth mit den folgenden Werten festgelegt wird

$$\boldsymbol{x}^{\text{Ground-Truth,extr}} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0{,}1 \\ 0{,}1 \\ 0{,}1 \end{pmatrix}. \tag{5.102}$$

Die Werte der Ground-Truth sind dabei nicht einem bestimmten Fahrzeugaufbau nachempfunden, sondern stellen exemplarisch eine mögliche Relativeinbaulage mit vergleichbaren Größenordnungen für Positionen und Winkel dar.

Damit stehen die beiden Sensoren in jeder Richtung 1 m voneinander entfernt, also insgesamt $\sqrt{3}$ m, und sind um 0,1 rad $\approx 5{,}73°$ um jede der drei Koordinatenachsen verdreht. Die Analyse basiert auf einem Datensatz, der auf Grundlage eines virtuellen Fahrzeugs, angenähert durch ein kinematisches Einspurmodell, mit 3,5 m Achsabstand berechnet wird und einer Slalombewegung des Fahrzeugs entspricht.

In diesem Slalom-Datensatz bewegt sich das Fahrzeug mit einer konstanten Bahngeschwindigkeit $v = 5\,\text{m/s}$. Die Bewegung wird durch einen sinusförmigen Lenkwinkelverlauf hervorgerufen. Die Lenkwinkelamplitude entspricht $\lambda^{\text{max}} = 10°$ bei einer Frequenz von $f^{\lambda} = 0{,}1\,\text{Hz}$. Der daraus resultierende Rollwinkel hat eine Amplitude von $\Phi^{\text{max}} = 3°$ bei einer Frequenz von ebenfalls $f^{\Phi} = 0{,}1\,\text{Hz}$. Die Größen sind so gewählt, dass sie auch einer Slalomfahrt unter realen Bedingungen entsprechen könnten. Die simulierten Werte entsprechen den Delta-Posen von INS und Kamera. Zur Veranschaulichung zeigt Abb. 5.35 die räumliche Integration der ersten 1000 Datenpunkte der Fahrzeugbewegung im Slalom-Datensatz. In seiner Gesamtheit besteht der Datensatz aus der mehrfachen Aneinanderreihung derart generierter Datenpunkte und hat einem Gesamtlänge von 8980 Datenpunkten. Die Länge entspricht dem später vorgestellten Real-Datensatz und dient der besseren Vergleichbarkeit von Simulation und realen Daten.

Simulierte Messungen der Odometrie beider Sensoren werden alle 0,1 s, entsprechend alle 0,5 m, genommen. Dabei befindet sich der erste der beiden Sensoren im Referenzpunkt, d.h. Mitte der hinteren Achse, des Fahrzeugs. Damit entspricht seine zu messende Bewegung genau der des Fahrzeugs. Der zweite Sensor befindet sich entsprechend um $\boldsymbol{x}^{\text{Ground-Truth,extr}}$ bzgl. des Referenzpunktes verschoben.

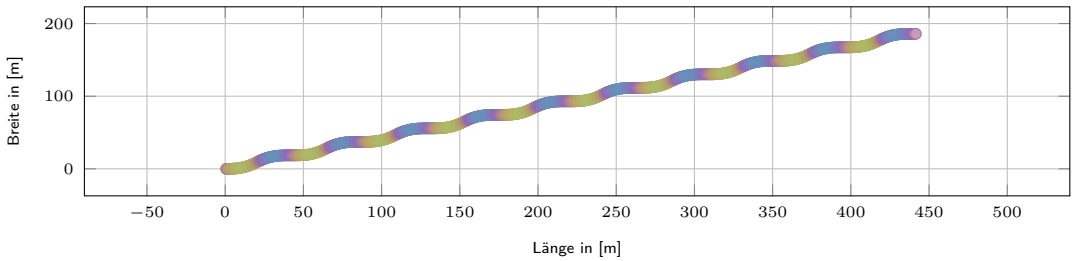


Abbildung 5.35:
Die ersten 1000 Datenpunkte des Slalom-Datensatzes. Der Rollwinkel des Fahrzeugs ist farblich kodiert. ■ entspricht einem Rollwinkel von −3°, während ■ einem Rollwinkel von 3° entspricht. Rollwinkel um 0° werden in ■ dargestellt.

Um seine Messung zu simulieren, wird die Delta-Pose des ersten Sensors in das Koordinatensystem des zweiten transformiert.

Das anschließende Beaufschlagen von mittelwertfreiem Gauß'schen Rauschen auf die derart erhaltenen Delta-Posen dient der Simulation des realen Sensorrauschens. Für die Positionskomponenten wird von einer Varianz in Höhe von $\sigma^{\text{pos}2} = 1 \times 10^{-5}\,\text{m}^2$, d.h. $\sigma^{\text{pos}} \approx 3\,\text{mm}$ ausgegangen, für die Winkelkomponenten $\sigma^{\text{ang}2} = 3 \times 10^{-6}\,\text{rad}^2$, d.h. $\sigma^{\text{ang}} \approx 0{,}1°$. Beide Werte sind an die Genauigkeit des in den Versuchsträgern verbauten INS angelehnt.

Zum Vergleich der drei Filtervarianten aus Abschnitt 5.9.2 wird jedes Filter in der Nähe von $\boldsymbol{x}^{\text{Ground-Truth,extr}}$ mit $\boldsymbol{x}_0 = (0{,}9, 0{,}9, 0{,}9, 0{,}09, 0{,}09, 0{,}09)^T$ initialisiert. Die Kovarianzmatrizen entsprechen vereinfacht Diagonalmatrizen. Tabelle 5.4 zeigt die Diagonalwerte für jede der drei Matrizen und für jedes der drei Filter. Zum Vergleich der drei Varianten betrachtet Abb. 5.36 zunächst, wie sich die Positionskomponenten des extrinsischen Teils der Zustandsvariablen, d.h. x, y und z, über der Zeit entwickeln. Anhand der Abbildung ist erkennbar, dass die Varianten unterschiedlich schnell auf die korrekten Werte konvergieren. Variante 1 scheint zunächst schneller zu konvergieren, erreicht innerhalb der simulierten Sequenz jedoch nicht exakt den Zielwert in der z-Komponente. Die Varianten 2 und 3 konvergieren langsamer, erreichen aber nach ca. 7000 Datenpunkten den gesuchten Wert der Ground-Truth. Überhaupt ist zwischen letzteren beiden Varianten visuell kaum ein Unterschied erkennbar. Der Unterschied zur Variante 1 ist erklärbar dadurch, dass in den beiden anderen Varianten das Rauschen der Sensoren besser modellierbar ist. So können die Messungen der beiden Sensoren besser gewichtet werden.

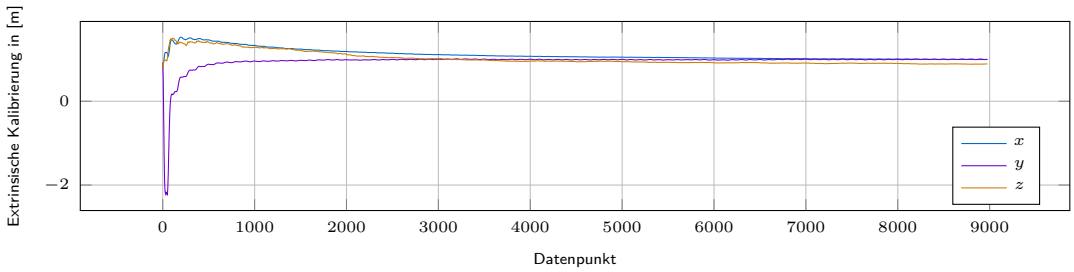
Darüber hinaus fällt auf, dass die z-Komponente insbesondere bei Varianten 2 und 3 am langsamsten konvergiert. Bei Betrachtung des Versuchsaufbaus wird

Variante	Matrix	$(\cdot)_{[x,x]}$	$(\cdot)_{[y,y]}$	$(\cdot)_{[z,z]}$	$(\cdot)_{[\Phi,\Phi]}$	$(\cdot)_{[\Theta,\Theta]}$	$(\cdot)_{[\Psi,\Psi]}$
1	$\mathrm{diag}(\mathbf{P})$	3,3	3,3	3,3	0,9	0,9	0,9
	$\mathrm{diag}(\mathbf{Q})$	0,01	0,01	0,01	0,002	0,007	0,009
	$\mathrm{diag}(\mathbf{R})$	0,2	0,6	0,9	0,2	0,2	0,2
2	$\mathrm{diag}(\mathbf{P})$	1 3,3	1 3,3	1 3,3	1 0,9	1 0,9	1 0,9
	$\mathrm{diag}(\mathbf{Q})$	1 0,01	1 0,01	1 0,01	0,9 0,002	0,9 0,007	0,9 0,009
	$\mathrm{diag}(\mathbf{R})$	0,2	0,6	0,9	0,2	0,2	0,2
3	$\mathrm{diag}(\mathbf{P})$	1 3,3	1 3,3	1 3,3	1 0,9	1 0,9	1 0,9
	$\mathrm{diag}(\mathbf{Q})$	1 0,01	1 0,01	1 0,01	0,9 0,002	0,9 0,007	0,9 0,009
	$\mathrm{diag}(\mathbf{R})$	0,2	0,6	0,9	0,2	0,2	0,2
	$\mathrm{diag}\left(\mathbf{R}^{\mathsf{INS}}\right)$	0,6	0,6	0,6	0,2	0,2	0,2

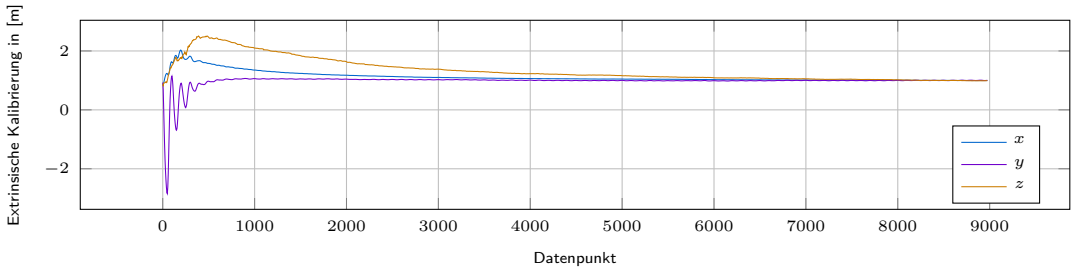
Tabelle 5.4:
Werte der Diagonalelemente für die Kovarianzmatrizen der drei Filtervarianten in der Simulation. Bei mehrzeilig dargestellten Diagonalelementen beziehen sich die obersten Elemente auf $\boldsymbol{x}^{\mathsf{odom}}$ und die unteren auf $\boldsymbol{x}^{\mathsf{extr}}$.

der Grund dafür ersichtlich. Um die z-Komponente schätzen zu können, muss das Fahrzeug eine Bewegung vollführen, die von beiden Sensoren nicht nur unterschiedlich wahrgenommen wird, sondern diese unterschiedliche Wahrnehmung muss auch auf eine unterschiedliche z-Komponente zurückführbar sein.

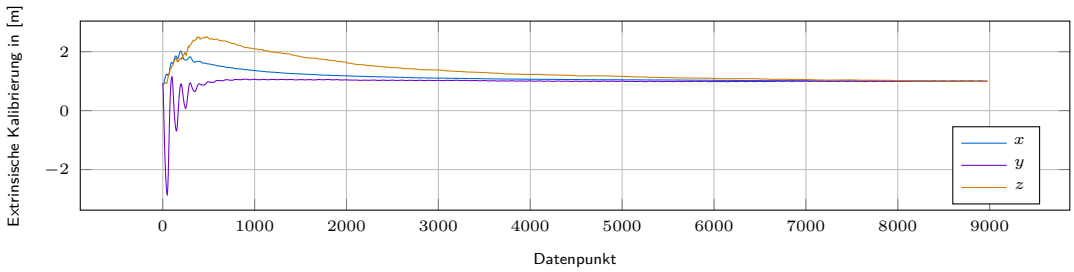
Bewegungen, für die dies nicht gilt, sind reine translatorische Bewegungen und Rotationen um die Gierachse des Fahrzeugs. Es ist leicht vorstellbar, dass bei derartigen Bewegungen die Wahrnehmung durch die Sensoren unabhängig von ihrem relativem Abstand in z-Richtung zueinander ist. Führt das Fahrzeug hingegen Roll- oder Nickbewegungen aus, führt der vertikale Abstand zwischen den Sensoren zu einem stärkeren Wanken des oberen Sensors. Da in unserem Fall das Fahrzeug allerdings keinerlei Nickbewegungen durchführt und zusätzlich die Amplitude des Rollwinkels verhältnismäßig klein ist, wird die z-Komponente in Folge dessen nur schwer beobachtbar und konvergiert daher langsamer. Insgesamt lässt sich also festhalten, dass die Miteinbeziehung der Visuellen Odometrie in den Zustandsvektor für die Positionsschätzung vorteilhaft ist.



(a) Variante 1



(b) Variante 2

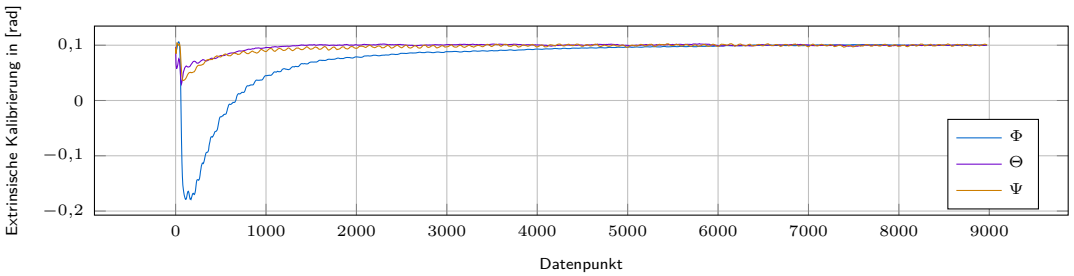


(c) Variante 3

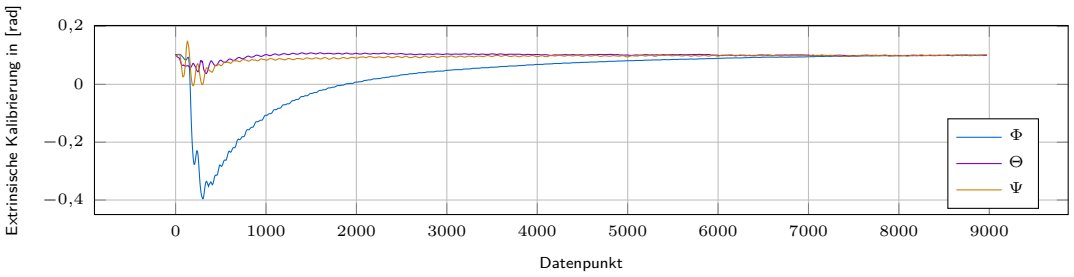
Abbildung 5.36:
Zeitliche Entwicklung der extrinsischen Positionskomponenten aller drei Filtervarianten im Slalom-Datensatz.

Bei der Betrachtung des zeitlichen Verlaufs der Winkelschätzung fällt auf, wie in Abb. 5.37 gezeigt, dass Variante 1 wieder am schnellsten zu konvergieren scheint. Bereits nach etwa 5000 Datenpunkten erreicht Variante 1 den gesuchten Zielwert. Im Gegensatz dazu konvergieren die Varianten 2 und 3 langsamer und erreichen nach erst etwa 7000 Datenpunkten den gesuchten Wert. Bei allen drei Varianten konvergiert der Rollwinkel Φ am langsamsten, was ebenfalls wieder auf den vergleichsweise geringen Rollwinkel in den simulierten Odometriedaten zurückführbar ist. Wie auch schon bei der Betrachtung der Positionskomponenten ist zwischen den Varianten 2 und 3 kaum ein Unterschied erkennbar.

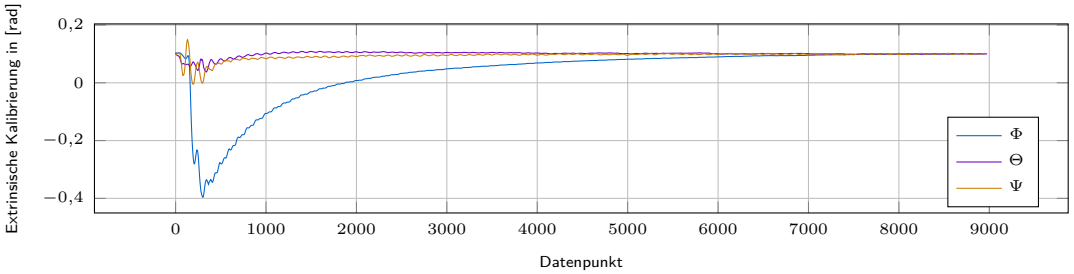
Zum quantitativen Vergleich wird angenommen, dass alle Filter spätestens bis zum 7980-ten Datenpunkt eingeschwungen sind. Die folgende Betrachtung stützt sich



(a) Variante 1



(b) Variante 2



(c) Variante 3

Abbildung 5.37:
Zeitliche Entwicklung der extrinsischen Winkelkomponenten aller drei Filtervarianten im Slalom-Datensatz.

daher lediglich auf die Filterergebnisse der letzten 1000 Datenpunkte der Sequenz. Untersucht wird, inwieweit sich die drei Filtervarianten hinsichtlich der Stabilität ihrer Schätzungen unterscheiden. Dazu zeigt Tabelle 5.5 die Standardabweichungen der drei Filter über die besagten, letzten 1000 Datenpunkte.

Anhand der Tabelle wird deutlich, dass alle drei Varianten ähnlich stabile Schätzwerte liefern. Die Unterschiede liegen, mit Ausnahme von $\sigma_{[z]}$ im Submillimeter- bzw. Submilliradianbereich. Für $\sigma_{[z]}$ liefert Variante 1 um nur wenige Millimeter stabilere Schätzungen. Insgesamt sind die Unterschiede jedoch vernachlässigbar gering.

Da es sich hierbei um eine Simulation handelt, sind die korrekten Werte, auf die die Filter konvergieren sollen, bekannt. Tabelle 5.6 zeigt einen MSE-Vergleich (Mean

Variante	$\sigma_{[x]}$	$\sigma_{[y]}$	$\sigma_{[z]}$	$\sigma_{[\Phi]}$	$\sigma_{[\Theta]}$	$\sigma_{[\Psi]}$
1	0,0033	0,0038	0,0046	0,0002	0,0003	0,0011
2	0,0030	0,0038	0,0092	0,0007	0,0003	0,0011
3	0,0031	0,0038	0,0091	0,0007	0,0003	0,0011

Tabelle 5.5:
Standardabweichungen über die letzten 1000 Datenpunkte des Slalom-Datensatzes nach Filtervariante. Die besten und schlechtesten Werte einer Spalte sind jeweils farbig (■ bzw. ■) dargestellt.

Squared Error) aller Varianten über die letzten 1000 Datenpunkte. Zur besseren Darstellung sind die Werte entsprechend der Skalierung-Spalte mit Skalierungsfaktor angegeben.

Bereits der Skalierungsfaktor verrät, dass die Genauigkeitsunterschiede der drei Filtervarianten marginal sind. Einzig $\text{MSE}_{[z]}$ zeigt einen deutlich höheren Fehler in der Schätzung der z-Komponente bei Variante 1. Dieser Fehler deckt sich mit der Beobachtung in Abbildung 5.36.

Variante	Skalierung	$\text{MSE}_{[x]}$	$\text{MSE}_{[y]}$	$\text{MSE}_{[z]}$	$\text{MSE}_{[\Phi]}$	$\text{MSE}_{[\Theta]}$	$\text{MSE}_{[\Psi]}$
1	10^{-6}	10,94	14,52	12 026,78	0,05	0,1	1,28
2	10^{-6}	9,04	14,47	84,34	0,46	0,1	1,25
3	10^{-6}	9,34	14,45	83,65	0,46	0,1	1,25

Tabelle 5.6:
MSE über die letzten 1000 Datenpunkte des Slalom-Datensatzes. Die besten und schlechtesten Werte einer Spalte sind jeweils farbig (■ bzw. ■) dargestellt.

Abschließend ergibt sich daraus, dass Varianten 2 und 3 auf den hier simulierten Daten vergleichbare Ergebnisse liefern. Die genauere Modellierung des Fehlers mit Hilfe der UT führt nicht, wie ursprünglich erwartet, zur einer schnelleren Konvergenz oder genaueren Schätzung der Zustandsgrößen. Während Variante 1 sowohl im Bereich der Position als auch der Orientierung schneller konvergiert, zeigt die Variante eine Schwäche in der Genauigkeit der Schätzung der z-Komponente.

5.9.4 Extrinsische Kalibrierung von Inertialsensorik und einer Stereokamera

Nachdem sich die Varianten 2 und 3 in der Evaluierung auf Simulationsdaten als der Variante 1 leicht überlegen dargestellt haben, kommen beide nun zur Kalibrierung eines Oxford Technical Solutions (OxTS) RT3003 INS und einer Bumblebee XB3 auf Basis von Realdaten zum Einsatz. Beide Sensoren sind fest an dem Versuchsträger

MuCAR-4 montiert. Zur Datenaufzeichnung findet eine Kalibrierfahrt, die in Abb. 5.38 dargestellt ist, statt. Dabei wurde das Fahrzeug manuell in eine Slalomfahrt versetzt, um einen möglichst großen Rollwinkel des Fahrzeugs zu erzeugen. Das INS ist dabei mittig im Fahrzeug unter der Armlehne zwischen den Vordersitzen, d.h. nahe dem Schwerpunkt des Fahrzeugs, verbaut. Die Bumblebee XB3 befindet sich an der Windschutzscheibe unterhalb des Rückspiegels. Da die Montagegewinde der Bumblebee XB3 an der Unterseite der Kamera platziert sind, ist die Kamera auf dem Kopf stehend mit Blick entlang der x-Achse des Fahrzeugs verbaut.

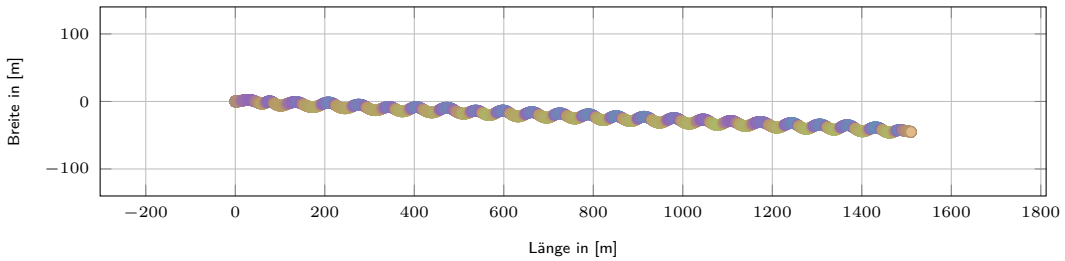


Abbildung 5.38:
Darstellung des Realwelt-Datensatzes. Der Rollwinkel des Fahrzeugs ist farblich kodiert. ■ entspricht einem Rollwinkel von ca. −4,37° während ■ einem Rollwinkel von ca. 3,24° entspricht.

Wie in Anhang B.1 beschrieben, handelt es sich bei der Bumblebee XB3 um ein Kamerasystem mit 3 Kameras. Die Kameras sind paarweise zueinander ab Werk kalibriert, so dass zwischen einem Stereobetrieb mit einem Basisabstand von 12 cm oder 24 cm gewählt werden kann. Vereinfacht gesagt, führt eine größere Stereobasis zu einer höheren Tiefenauflösung. Daher wird die Kamera für diese Anwendung mit der breiteren Basis von 24 cm betrieben. Es werden somit die beiden äußeren Kameras verwendet.

Um die Frage, wie die Eigenbewegung eines Roboters nur anhand von Kamerabildern erfasst werden kann, hat sich in den letzten Jahren ein eigener Forschungsschwerpunkt, genannt Visuelle Odometrie, entwickelt. Daraus entstanden ist die VISO2-C++-Bibliothek von Geiger et al. [2011], die sich als effizient und robust herausgestellt hat. Daher kommt die Bibliothek auch in dieser Anwendung zum Einsatz. Für zwei Stereobildpaare zu aufeinanderfolgenden Zeitpunkten liefert die Bibliothek die Delta-Pose bzgl. der linken Kamera. Diese Delta-Pose lässt sich direkt als Messung verwenden.

Das OxTS RT3003 INS liefert ebenfalls Delta-Posen, die direkt zur Kalibrierung verwenden werden können. Diese Delta-Posen sind allerdings nicht zeitlich synchron zu den Delta-Posen des Stereosystems. Die Bumblebee XB3 zeichnet die Bilder mit

16 Hz, d.h. alle 62,5 ms, auf, während die Delta-Posen des INS mit 100 Hz, d.h. alle 10 ms, eintreffen. Eine Möglichkeit, zueinander passende Delta-Posen von beiden Sensoren zu finden, bietet das bereits in Abschnitt 3.4.2 beschriebene `tas::app`-Framework. Aufgrund der höheren Übertragungsdauer der Kamerabilder eignen sich die Bildpaare als *auslösendes* Objekt. Allerdings findet das Framework in diesem Fall nur zeitlich nahe beieinander liegende Paare von Delta-Posen. Das Filter setzt allerdings zeitlich synchrone Posen-Paaren voraus.

Zur Lösung dieses Problems wird eine lineare Interpolation zwischen den Posen des INS durchgeführt. Angenommen $\mathbf{P}_0^{\text{INS}}$ und $\mathbf{P}_1^{\text{INS}}$ beschreiben die globalen Posen des INS zu zwei direkt aufeinanderfolgenden Zeitpunkten t_0^{INS} und t_1^{INS}. Wenn t_k^{Kamera} dem Datenzeitstempel der Kamerabilder enspricht und außerdem gilt $t_0^{\text{INS}} \leq t_k^{\text{Kamera}} \leq t_1^{\text{INS}}$, lässt sich die absolute Pose $\mathbf{P}_k^{\text{INS}}$ wie folgt berechnen

$$\mathbf{P}_k^{\text{INS}} = s \cdot \mathbf{P}_0^{\text{INS}} + (1 - s) \cdot \mathbf{P}_1^{\text{INS}} \text{ wobei} \tag{5.103}$$

$$s = 1 - \frac{t_k^{\text{Kamera}} - t_0^{\text{INS}}}{t_1^{\text{INS}} - t_0^{\text{INS}}}. \tag{5.104}$$

Die Addition und Skalierung von Posen ist in Anhang C definiert. Für die Verwendung im Filter kommen schließlich nur die Delta-Posen zwischen derart auf die Bildzeitstempel interpolierten globalen Posen zum Einsatz. Um diese Interpolation in der Praxis durchführen zu können, muss bei Eintreffen der Kamerabilder die neuere Pose, in diesem Fall $\mathbf{P}_1^{\text{INS}}$, bereits verfügbar sein. Anders ausgedrückt: Der Rechner erhält die Posen vom INS alle 10 ms. Damit zum Übergabezeitpunkt der Kamerabilder an die KogMo-RTDB die neuere Pose bereits bekannt ist, müssen zwischen dem mittlerem Aufnahmezeitpunkt, d.h. dem Datenzeitpunkt und dem Übergabezeitpunkt ebenfalls mindestens 10 ms liegen.

Im Stereomodus liefert die Bumblebee XB3 zwei Graubilder mit einer Bayer-Matrix in einer Auflösung von 1280 pix $\times$ 960 pix. Das entspricht einer Datenmenge von $2 \cdot (1280 \cdot 960)$ B = 2,4576 MB. Der FireWire-Bus, über den die Bumblebee XB3 angeschlossen ist, verfügt über eine Übertragungsgeschwindigkeit von theoretisch maximalen 786 432 Mbit/s = 98,304 MB/s. Damit dauert die Übertragung eines Stereobildpaares mindestens 2,4576 MB/98,304 MB/s = 25 ms. Hinzu kommen die Umrechnung von Bayer-Matrix in Farbbilder und die eigentliche Übergabe an die KogMo-RTDB. Damit ist sichergestellt, dass eine `tas::app`-Anwendung, die auf die Kamerabilder als *auslösendes* Objekt wartet, stets auch die beiden notwendigen INS-Posen zur Interpolation zur Verfügung hat.

Aus der in Abb. 5.38 gezeigten Fahrt über ca. 1500 m ergeben sich 8980 Paare von Delta-Posen. Zur besseren Initialisierung des Filters wird die Positionskomponente grob manuell ausgemessen und die umgekehrte Einbaulage der Bumblebee XB3 ebenfalls berücksichtigt. Die Initialisierungswerte für $\boldsymbol{x}_0$ und die Filterkovarianzmatrizen sind in Tabelle 5.7 angegeben.

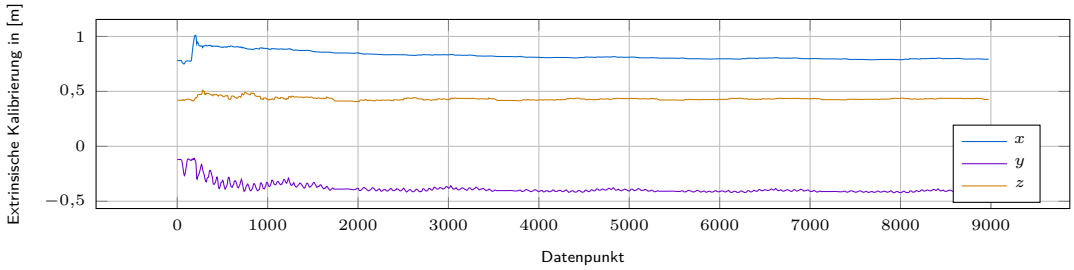
Vector	$(\cdot)_{[x]}$	$(\cdot)_{[y]}$	$(\cdot)_{[z]}$	$(\cdot)_{[\Phi]}$	$(\cdot)_{[\Theta]}$	$(\cdot)_{[\Psi]}$
$\boldsymbol{x}_0$	0	0	0	0	0	0
	0,78	−0,12	0,42	−3,142	0	0
Matrix	$(\cdot)_{[x,x]}$	$(\cdot)_{[y,y]}$	$(\cdot)_{[z,z]}$	$(\cdot)_{[\Phi,\Phi]}$	$(\cdot)_{[\Theta,\Theta]}$	$(\cdot)_{[\Psi,\Psi]}$
$\mathrm{diag}\,(\mathbf{P})$	1	1	1	1	1	1
	3,3	3,3	3,3	0,9	0,9	0,9
$\mathrm{diag}\,(\mathbf{Q})$	1	1	1	0,9	0,9	0,9
	0,01	0,01	0,01	0,002	0,007	0,009
$\mathrm{diag}\,(\mathbf{R})$	0,9	0,3	0,3	0,2	0,2	0,2
$\mathrm{diag}\left(\mathbf{R}^{\mathsf{INS}}\right)$	0,6	0,6	0,6	0,2	0,2	0,2

Tabelle 5.7:
Werte für $\boldsymbol{x}_0$ und die Diagonalelemente für die Kovarianzmatrizen der Filtervarianten 2 und 3 auf dem Realwelt-Datensatz. Bei mehrzeilig dargestellten Diagonalelementen beziehen sich die obersten Elemente auf $\boldsymbol{x}^{\mathsf{odom}}$ und die unteren auf $\boldsymbol{x}^{\mathsf{extr}}$.

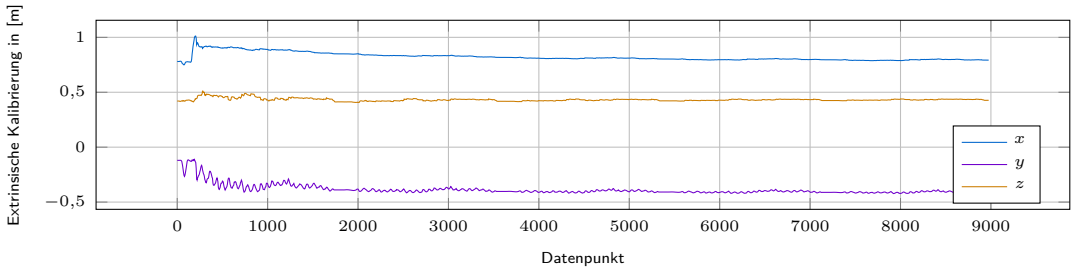
Bei Vergleich der Kovarianzen mit denen aus Tabelle 5.4 fällt als einziger Unterschied ein geändertes Messrauschen $\mathbf{R}$ auf. Statt simulierter Messungen entstammen die Messungen des Realwelt-Datensatzes der berechneten Visuellen Odometrie. Die Slalomfahrt zur Aufzeichnung des Datensatzes hat auf weitestgehend freiem Gelände stattgefunden. Die Bildkorrespondenzen, auf deren Grundlage die Visuelle Odometrie mit Hilfe von Geiger et al. [2011] berechnet wird, entsprechen damit häufig weit entfernten Punkten. Aufgrund der im Vergleich dazu verhältnismäßig kleinen Stereobasis von 24 cm ist ein höheres Messrauschen in Blickrichtung und damit in der x-Komponente von $\mathbf{R}$ nachvollziehbar.

Abbildung 5.39 zeigt die zeitliche Entwicklung der Positionskomponenten des geschätzten Zustandsvektors über alle Datenpunkte des Realwelt-Datensatzes. In Abb. 5.40 sind entsprechend die zeitlichen Verläufe der Winkelkomponenten gegenübergestellt.

Wie auch in der Simulation zeigt sich für beide Varianten ein nahezu identischer zeitlicher Verlauf. Ebenfalls ist nach etwa 7000 Datenpunkten kaum noch eine Änderung der Zustandsgrößen erkennbar. Analog zu Tabelle 5.5 zeigt Tabelle 5.8



(a) Variante 2



(b) Variante 3

Abbildung 5.39:
Zeitliche Entwicklung der extrinsischen Positionskomponenten der Filtervarianten 2 und 3 über die 8980 Datenpunkte des Realwelt-Datensatzes.

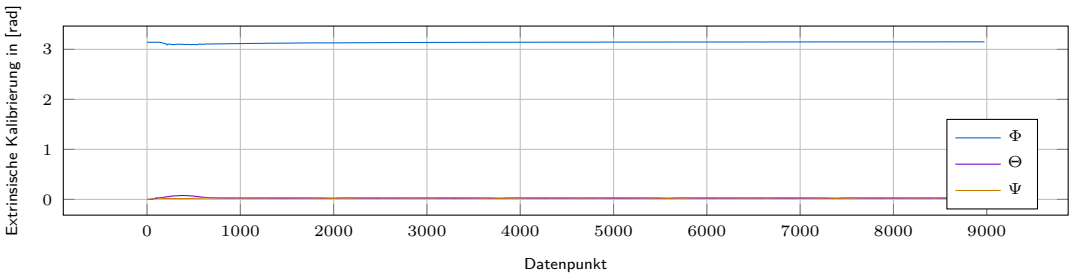
eine Analyse der Standardabweichungen der Zustandsgrößen auf Basis der letzten 1000 Datenpunkte des Datensatzes.

Variante	$\sigma_{[x]}$	$\sigma_{[y]}$	$\sigma_{[z]}$	$\sigma_{[\Phi]}$	$\sigma_{[\Theta]}$	$\sigma_{[\Psi]}$
2	0,0033	0,0073	0,0032	0,0003	0,0009	0,0015
3	0,0033	0,0073	0,0032	0,0003	0,0009	0,0015

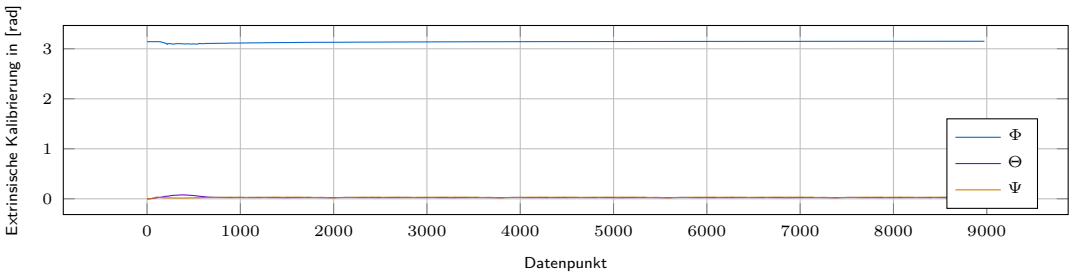
Tabelle 5.8:
Standardabweichungen über die letzten 1000 Datenpunkte des Realwelt-Datensatzes für die Varianten 2 und 3.

Dabei fällt auf, dass die berechneten Standardabweichungen für beide Filtervarianten identisch sind. Damit bestätigt sich die Beobachtung aus der Simulation, dass die genauere Repräsentation des INS-Rauschens auf den vorliegenden Daten keinen nennenswerten Vorteil bringt. Die gezeigten Standardabweichungen liegen zudem in der Größenordnung derer aus Tabelle 5.5. Die beiden Filtervarianten erreichen eine Genauigkeit der Positionskomponenten von $< 0{,}5\,\text{cm}$ und der Winkelkomponenten von $< 0{,}1°$.

Die hier untersuchten Filtervarianten eignen sich daher zur Schätzung der relativen Orientierung und Position zwischen zwei Sensoren. Aufgrund ihrer rekursiven Aus-



(a) Variante 2



(b) Variante 3

Abbildung 5.40:
Zeitliche Entwicklung der extrinsischen Winkelkomponenten der Filtervarianten 2 und 3 über die 8980 Datenpunkte des Realwelt-Datensatzes.

legung in Form eines UKF verfügen sie über eine nahezu konstante Laufzeit und eignen sich dadurch zur *online*-Kalibrierung im fahrenden Fahrzeug.

6 Sensorfusion

Sind die Sensoren eines Fahrzeugs kalibriert, besteht die Möglichkeit, die Daten verschiedener Sensoren miteinander zu fusionieren. Ein gutes Beispiel hierfür sind die Daten von Kamera und Light Detection and Ranging (LiDAR). Während Kameras aufgrund der projektiven Abbildung keine Tiefe sehen oder nur durch Structure from Motion (SfM) oder Stereo rekonstruieren können, liefern LiDAR-Sensoren vergleichsweise genaue Tiefenmessungen. Andererseits verfügen die in dieser Arbeit verwendeten Kameras über eine relativ hohe laterale und vertikale Auflösung. In diesem Punkt sind sie den LiDAR-Sensoren deutlich überlegen.

Wie schon in Kämpchen [2007] beschrieben, lassen sich Sensorfusionsverfahren grob in die drei Kategorien Fusion auf Sensorebene, Merkmalebene und Objekt- bzw. Szenenebene unterteilen. Dank der in Kapitel 5 vorgestellten Verfahren verfügt der Versuchsträger über mehrere intrinsisch kalibrierte Kameras, einen LiDAR und ein Inertial Navigation System (INS). Alle Sensoren sind extrinsisch zueinander kalibriert, so dass die Umrechnung von Koordinaten zwischen den Sensoren möglich ist. Ausgehend von dieser Sensorkonfiguration beschreibt dieses Kapitel nun einige Anwendungsfälle zur Fusion der Sensoren. Dabei liegt der Fokus vorwiegend auf der Fusion innerhalb der Sensor- und der Merkmalebene.

6.1 Fusion auf Sensorebene

Auf der Sensorebene behandelt dieser Abschnitt die Fusion von Sensorrohdaten. Die beiden Hauptsensoren zur Umfelderfassung auf beiden Versuchsträgern sind einerseits Kameras und andererseits ein LiDAR. Das folgende Kapitel beschreibt eine Rohdatenfusion beider Sensortypen

6.1.1 Rohdatenfusion von Kamera und LiDAR

Dank der Kalibrierung der beiden Sensoren mit Hilfe der Hybriden Metrik aus Abschnitt 5.8 sind Kamera und LiDAR zueinander kalibriert. Zur qualitativen Bewertung der Kalibrierung werden die 3D-Punkte des LiDAR in das Koordinatensystem $\mathbf{S}_{\text{Kamera}}$ der Kamera transformiert. Durch Projektion der 3D-Punkte in das Kamerabild ergibt sich für jeden Punkt die zugehörige Farbe aus dem getroffenen Bildpixel

$$\mathcal{P}_{\ulcorner,k}^{\text{Schachbrett}} = \mathsf{p}\left({}^{\text{Kamera}}\mathbf{P}_{\text{LiDAR}} \cdot \mathcal{P}_{\text{LiDAR},k}^{\text{Schachbrett}}\right). \tag{6.1}$$

Abbildung 6.1 zeigt Beispiele der eingefärbten Punktwolken. Punkte, die außerhalb des Sichtbereichs der Kamera liegen, erhalten einen Grauwert entsprechend ihrer Reflektivität.

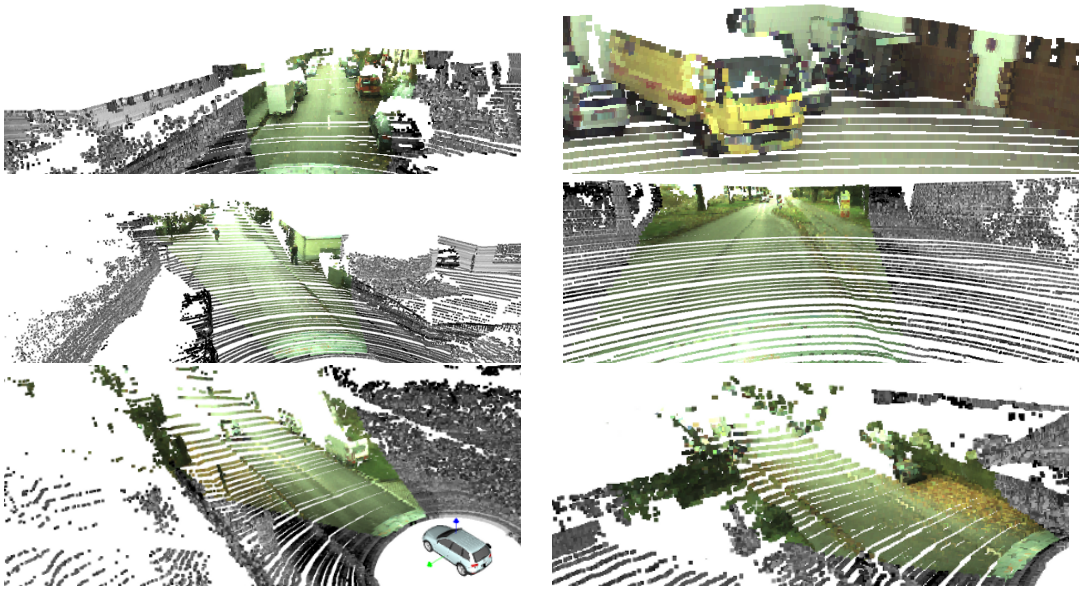


Abbildung 6.1:
Durch Projektion eingefärbte Punktwolken. 3D-Punkte außerhalb des Sichtbereichs der Kamera sind als Grauwerte entsprechend ihrer Reflektivität dargestellt.

Auf der anderen Seite ermöglicht die Projektion der 3D-Punkte ins Kamerabild auch die Bereitstellung von Tiefeninformationen für einzelne Pixel des Kamerabildes. Die Tiefe entspricht in diesem Fall der Entfernung des 3D-Punkts zum Koordinatensystem der Kamera $\mathbf{S}_{\text{Kamera}}$. Abbildung 6.2 zeigt die Projektion der 3D-Punkte einer Umdrehung des LiDAR in das Bild einer Kamera. Hierbei ist die in Abschnitt 3.1.3 beschriebene unterschiedliche Abtastrate der oberen 32 Laser im Vergleich zu den unteren 32 Lasern sichtbar.

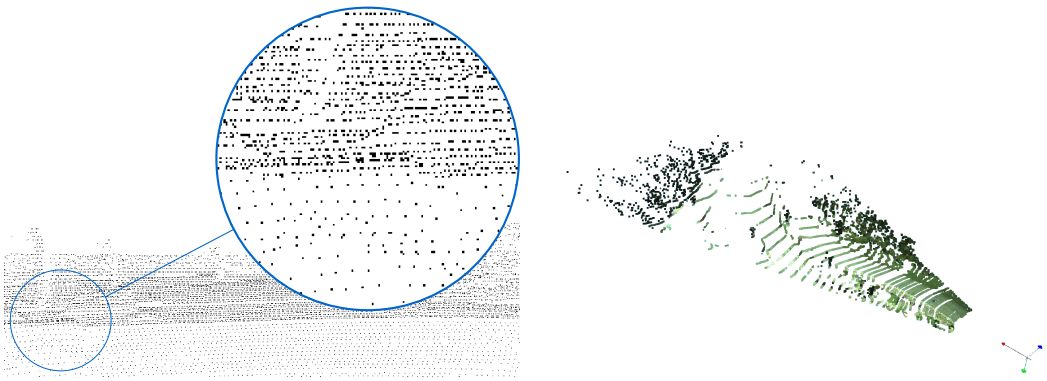


Abbildung 6.2:
Projektion von 3D-Punkten in das Bild einer Kamera (links) und zugehörige, eingefärbte Punktwolke (rechts).

6.1.2 Berücksichtigung von Verdeckungen aufgrund unterschiedlicher Sensoreinbaulagen

Die Projektion der 3D-Punkte in das Kamerabild mit Hilfe von Gleichung (6.1) lässt außer Acht, dass aufgrund der unterschiedlichen Einbaulagen der beiden Sensoren einige 3D-Punkte, wie in Abbildung 6.3 gezeigt, für die Kamera verdeckt sein können. Im vorliegenden Fall wird der für den LiDAR sichtbare 3D-Punkt auf dem Hindernis 2 auf das Kamerabild projiziert, obwohl der Punkt durch Hindernis 1 für die Kamera verdeckt liegt. Dadurch enthält einerseits der Punkt einen falschen Farbwert und andererseits das entsprechende Pixel den falschen Tiefenwert.

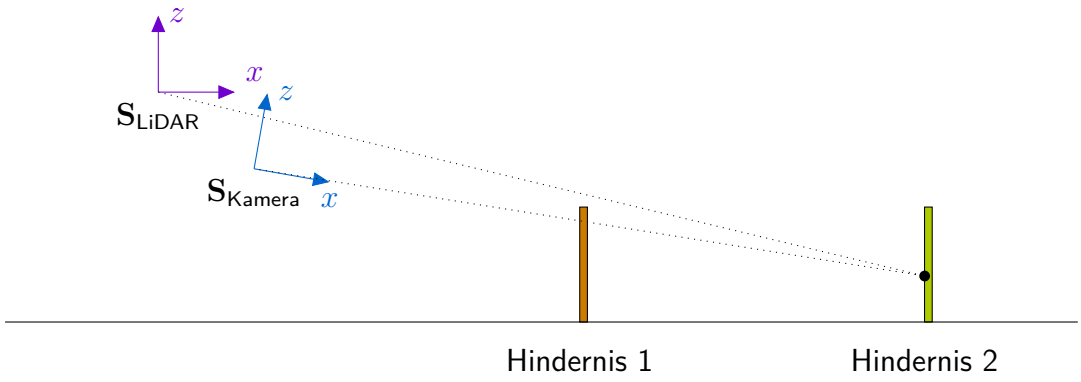


Abbildung 6.3:
Verdeckung aufgrund unterschiedlicher Einbaulagen der Sensoren. Der vom LiDAR aus sichtbare Punkt auf dem Hindernis 2 ist von der Kamera aus nicht sichtbar.

Zur Lösung des Problems müssen mehrere Fälle betrachtet werden. In dem Fall, dass ein weiterer Laserstrahl das Hindernis 1 an der gleichen Stelle trifft, durch die der erste Laserstrahl hindurchprojiziert wird, ist die Lösung einfach. Da beide

3D-Punkte auf das gleiche Pixel projiziert werden, entspricht die korrekte Tiefe der des am nächsten liegenden Punktes. Infolgedessen ist auch lediglich dieser Punkt einzufärben. Dieser Fall beschriebt allerdings einen Sonderfall. Viel häufiger kommt es vor, dass Punkte von Hindernis 2 im Kamerabild neben Punkte von Hindernis 1 projiziert werden.

In diesem Fall kommt das in Schneider et al. [2010] entwickelte Verfahren zur Verdeckungsrechnung zur Anwendung. Die Idee dahinter basiert darauf, einzelne Objekte in der Szene in Abhängigkeit ihres Abstands zur Kamera zu sortieren, um letztendlich nur diejenigen Pixel einzufärben und ihre Tiefe zu übernehmen, die von keinem anderen Objekt verdeckt werden. In ersten Schritt segmentiert dazu das Segmentierungsverfahren von Himmelsbach et al. [2010] die 3D-Punktwolke in Grundfläche und einzelne Objekte, analog zu Gleichung (5.79). Nach Transformation aller 3D-Punkte in das Kamerakoordinatensystem $\mathbf{S}_{\text{Kamera}}$ wird für jede der Punktwolken außer der Grundfläche der mittlere x-Abstand zur Kamera bestimmt. Nach der Projektion in das Kamerabild und der anschließenden Gruppierung derjenigen Punkte, die zum selben Punktwolkensegment gehören, wird für jede Gruppe die 2D konvexe Hülle in der Bildfläche berechnet. Durch Sortieren entlang der mittleren Entfernung zur Kamera ergibt sich damit eine geordnete Liste an 2D konvexen Hüllen, wie in Abb. 6.4 dargestellt. Bei der anschließenden Projektion dieser konvexen Hüllen der Reihe nach von nah nach fern auf das Kamerabild ergibt sich für jedes Pixel eine Segment-ID, die der Segment-ID des ursprünglichen Punktwolkensegments entspricht. Wurde dabei einem Pixel bereits eine Segment-ID zugewiesen, wird diese nicht nochmals aktualisiert.

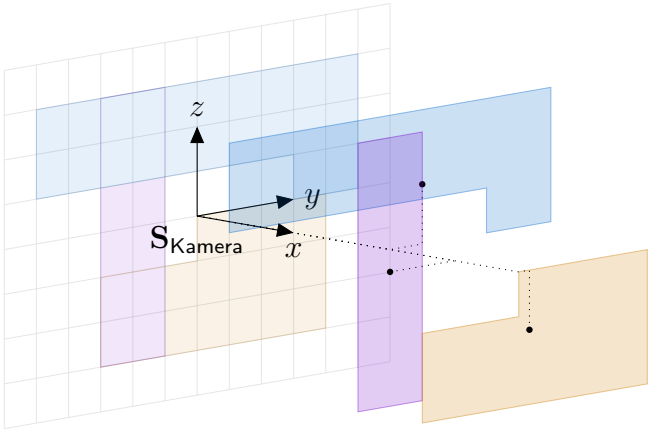


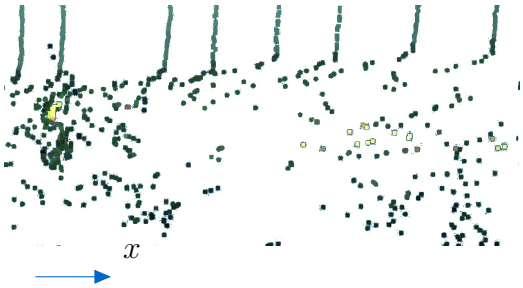
Abbildung 6.4:
Auflösung des Verdeckungsproblems durch 2D konvexe Hüllen, die mit aufsteigender Entfernung auf das Kamerabild projiziert werden.

Um zu entscheiden, ob ein projizierter 3D-Punkt von der Kamera aus sichtbar ist, muss nun nur noch ein Vergleich der Segment-IDs von 3D-Punkt und projiziertem Pixel durchgeführt werden. Nur wenn beide gleich sind, findet ein Einfärben des 3D-Punktes und die Übernahme der Tiefe für den entsprechenden Pixel statt.

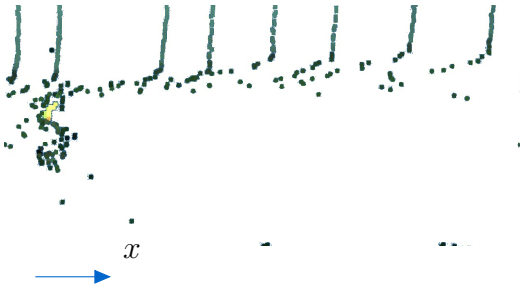
Dass dieses Konzept in realen Sequenzen ebenfalls funktioniert, zeigt Abb. 6.5. Dargestellt sind hier projizierte 3D-Punkte eines stark reflektierenden, gelben Verkehrszeichens, das in einer Wiese steht. Ohne Verdeckungsrechnung werden weniger stark reflektierende 3D-Punkte des dahinterliegenden Grases zwischen die stark reflektierenden Schild-Punkte projiziert. In diesem Fall werden fälschlicherweise die entsprechenden Gras-Punkte auch mit der Farbe des Schildes eingefärbt. Mit Hilfe der hier vorgestellten Verdeckungsrechnung werden die Gras-Punkte erfolgreich herausgefiltert.

(a)
Bildausschnitt eines Schildes ohne Verdeckungsrechnung.

(b)
Bildausschnitt eines Schildes mit Verdeckungsrechnung.



(c)
Eingefärbte Punktwolke aus der Vogelperspektive ohne Verdeckungsrechnung.



(d)
Eingefärbte Punktwolke aus der Vogelperspektive mit Verdeckungsrechnung.

Abbildung 6.5:
Projektion von 3D-Punkten und deren Einfärbung ohne Verdeckungsrechnung (links) und mit Verdeckungsrechnung (rechts). (a) und (b) zeigen die ins Bild projizierte Reflektivitätsinformation der 3D-Punkte. Helle Bildpunkte korrespondieren mit stark reflektierenden 3D-Punkten. (c) zeigt falsch eingefärbte Gras-Punkte, die durch die Verdeckungsrechnung in (d) eliminiert wurden.

Ein weiterer Versuch zeigt den Effekt der Verdeckungsrechnung bei Annäherung des Fahrzeugs an ein Hindernis. Der Aufbau entspricht dem der vorangegangenen

Auswertung mit einem gelben, stark reflektierenden Schild in einer Wiese. Das Versuchsfahrzeug bewegt sich auf das Schild zu. Anfangs ist das Fahrzeug derart weit entfernt, dass kaum Punkte hinter dem Schild vom LiDAR erfasst werden. Je näher das Fahrzeug dem Schild kommt, desto mehr blickt der LiDAR hinter das Schild und dementsprechend mehr Punkte werden durch das Schild hindurch projiziert und ohne Verdeckungsrechnung eingefärbt. Für diesen Versuch wurden die falsch gefärbten 3D-Punkte manuell markiert und in jedem Bild gezählt. Die Sequenz besteht aus 24 Bildern, die mit einer Frequenz von 100 Hz aufgenommen wurden. Abbildung 6.6 zeigt, wie die Zahl der falsch eingefärbten Punkte ohne Verdeckungsrechnung ansteigt, während sie mit Verdeckungsrechnung annähernd konstant bleibt.

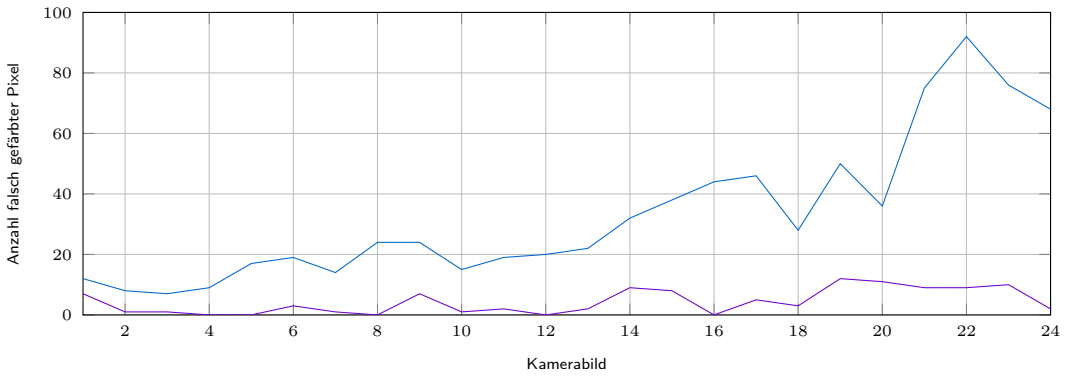


Abbildung 6.6:
Zahl der falsch eingefärbten 3D-Punkte bei Annäherung an das Schild mit und ohne Verdeckungsrechnung.

Das hier beschriebene Verfahren zur Verdeckungsrechnung reduziert zuverlässig die Zahl der falsch eingefärbten 3D-Punkte und filtert damit auch falsch projizierte 3D-Punkte aus dem Kamerabild. Dennoch beruht das Verfahren auf zwei Annahmen, die an dieser Stelle genauer beschrieben werden.

Zum einen basiert die Ordnung der konvexen Hüllen auf dem mittleren Abstand des zugehörigen Punktwolkensegments zur Kamera. Dies führt dann zu Fehlern, wenn verschiedene Punktwolkensegmente ineinander verzahnt sind, was in der Realität erfahrungsgemäß nur selten vorkommt, da in diesen Fällen das Segmentierungsverfahren meist ohnehin nur ein zusammenhängendes Objekt erstellt. Die zweite Annahme besteht darin, dass die konvexen Hüllen die Silhouetten der realen Objekte beschreiben. Diese Annahme ist insofern eine Vereinfachung, da die konvexen Hüllen nur auf Basis der 3D-Punkte des LiDAR berechnet werden. Der LiDAR trifft insbesondere bei entfernten Objekten, aufgrund der groben lateralen Auflösung, selten

genau die Objektränder. Das kann dazu führen, dass verdeckte Punkte genau in den Bereich zwischen konvexer Hülle und realer Objektsilhouette im Bild projiziert werden. Eine mögliche Lösung des Problems stellt die Kombination des Verfahrens zur Verdeckungsberechnung mit einem Bildsegmentierungsverfahren, wie dem von Felzenszwalb und Huttenlocher [2004], dar. Da sich das hier vorgestellte Verfahren für die Anwendungsfälle dieser Arbeit als genau genug herausgestellt hat, wird die angesprochene Erweiterung hier nicht weiter verfolgt.

6.2 Fusion auf Merkmalsebene

Basierend auf einer Rohdatenfusion wie der in Abschnitt 6.1, gibt es verschiedene Möglichkeiten zur Weiterverarbeitung der fusionierten Daten. Zhan et al. [2009] und Strom et al. [2010] zeigen Segmentierungsverfahren basierend auf derartigen farbigen Punktwolken mit dem Ziel der Objekterkennung und -klassifikation. Garnett et al. [2017] generiert auf Basis von kalibrierten LiDAR-Punktwolken die Ground-Truth zum Trainieren eines Neuronalen Netzes zur Erkennung, Klassifikation und Segmentierung von Objekten. Manz et al. [2009, 2011a] und Manz [2013] zeigen verschiedene Wege zum Verfolgen einer Fahrbahn auf Basis von Kamerabildern, LiDAR-Punktwolken und Odometrie eines INS.

Diese vorliegende Arbeit behandelt das autonome Fahren auf Basis von Fusion eines INS mit einem Stereokamerasystem und in einem weiteren Schritt ebenfalls mit LiDAR-Punktwolken.

6.2.1 Hindernisvermeidung mit Hilfe von Stereo und INS

Noch vor der Klassifikation von anderen Verkehrsteilnehmern und Hindernissen beim autonomen Fahren ist die Hindernisvermeidung eine Schlüsselfähigkeit des autonomen Fahrens. In der Robotik wird diese Aufgabe mit unterschiedlichen Sensoren behandelt. Seit Ende der 80-er Jahre des letzten Jahrhunderts haben sich u.a. Borenstein und Koren [1988, 1991] dem Problem mit Hilfe von Ultraschallsensoren genähert. Auch heute kommen noch Ultraschallsensoren bei modernen Fahrzeugen zum Einsatz, etwa als Einparkhilfe. Etwa ein Jahrzehnt später haben Chakravarthy und Ghose [1998] und andere damit begonnen, auch Radio Detection and Ranging (RaDAR) zur Hindernisvermeidung einzusetzen. Gegenüber den Ultraschallsensoren verfügen RaDAR-Sensoren über eine deutlich höhere Reichweite und werden

deshalb auch in heutigen Fahrzeugen zum Erkennen anderer Verkehrsteilnehmer beispielsweise in Form eines Abstandsregeltempomaten (ACC) verwendet. Besonders durch die Grand Challenge beflügelt, werden seit diesem Jahrhundert vermehrt auch LiDAR zur Hindernisvermeidung eingesetzt. Dies hat auch zur Entwicklung des Velodyne HDL-64E geführt, der auch auf Munich Cognitive Autonomous Robot Car 3rd Generation (MuCAR-3) und Munich Cognitive Autonomous Robot Car 4th Generation (MuCAR-4) zum Einsatz kommt. Nennenswerte Veröffentlichungen zur LiDAR-Hindernisvermeidung stammen von von Hundelshausen et al. [2008] und Himmelsbach et al. [2011a]. Eine Kombination aus RaDAR und LiDAR beschreibt Thrun [2010]. Bedingt durch die steigende Rechenleistung für mobile Roboterplattformen finden seit diesem Jahrhundert zunehmend Kamera-basierte Algorithmen Anwendung zur Hindernisvermeidung. Hier sei auf das Buch von Hartley und Zisserman [2003] verwiesen, welches die Grundlagen für SfM und stereoskopisches Sehen zusammenfasst. Seit dieser Zeit entstehen immer mehr Algorithmen, die aus Stereoaufnahmen eines Fahrzeugs die Tiefe rekonstruieren. Einige dieser Verfahren sind Blockmatching (BM) [Chen et al., 2001], Semiglobal Matching (SGM) [Hirschmüller, 2005, Hirschmüller, 2008] oder Efficient Large-scale Stereo Matching (ELAS) [Geiger et al., 2010].

Eine wesentliche Gemeinsamkeit dieser drei Verfahren ist, dass jedes Stereobildpaar individuell betrachtet wird. Dabei wird bereits bekanntes Wissen über die Umgebung aus vorangegangenen Bildern ignoriert. In der Regel berechnen diese Verfahren Disparitätsbilder, die anschließend in eine 3D-Punktwolke transformiert werden. Diese Punktewolkte wird letztlich verwendet, um u.a. die Höhenwerte einer Belegungskarte zu aktualisieren [Malartre et al., 2009].

6.2.2 Disparität und Tiefe

Um das Problem bei dieser Umwandlung zu verstehen, stellt sich zunächst die Frage, was ein Disparitätenbild ist. Dazu zeigt Abb. 6.7 die Draufsicht auf ein Stereokamerasystem, das einen beliebigen Punkt $\boldsymbol{A}$ im Raum beobachtet. Das hier dargestellte System zeigt zwei Lochkameras mit den Projektionszentren in den Punkten $\boldsymbol{B}$ und $\boldsymbol{C}$. Die Projektion von $\boldsymbol{A}$ fällt auf die Kamerachips beider Kameras. Für die linke Kamera ist dies der Punkt mit der bildzentrierten Koordinate u_l, für die rechte Kamera wird die Koordinate mit u_r bezeichnet. Für den Moment gilt die Annahme, dass alle Punkte in einer Ebene liegen und somit keinerlei Höhenunterschiede

auftreten. Die Arbeit geht später darauf ein, unter welchen Voraussetzungen diese Vereinfachung gerechtfertigt ist.

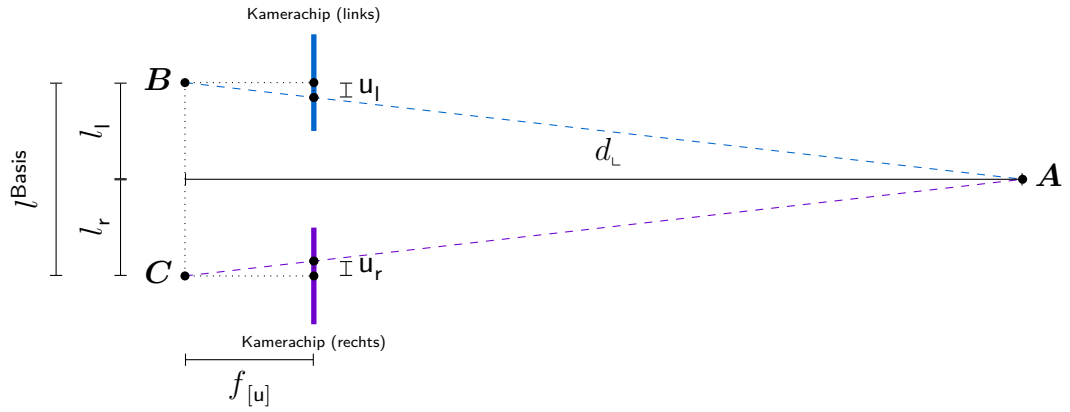


Abbildung 6.7:
Schematische Darstellung eines idealisierten Stereokamerasystems zur Herleitung der Disparität.

Die Disparität beschreibt den Abstand in Bildkoordinaten zwischen den zwei korrespondierenden Projektionen eines Punktes in einem Stereokamerasystem. In dem hier dargestellten Beispiel entspricht das $\mathrm{d}^{\boldsymbol{A}} = \mathrm{u_l} + \mathrm{u_r}$. Bei Betrachtung von Abb. 6.7 fällt die Ähnlichkeit der Dreiecke auf. Darum gilt

$$\frac{d_{\llcorner}}{l_{\mathrm{l}}} = \frac{f_{[\mathrm{u}]}}{\mathrm{u_l}} \qquad \text{und analog} \qquad \frac{d_{\llcorner}}{l_{\mathrm{r}}} = \frac{f_{[\mathrm{u}]}}{\mathrm{u_r}}$$
$$\mathrm{u_l} = \frac{f_{[\mathrm{u}]} \cdot l_{\mathrm{l}}}{d_{\llcorner}} \qquad \mathrm{u_r} = \frac{f_{[\mathrm{u}]} \cdot l_{\mathrm{r}}}{d_{\llcorner}}.$$

Durch Einsetzen erhalten wir

$$\mathrm{d}^{\boldsymbol{A}} = \frac{f_{[\mathrm{u}]} \cdot l_{\mathrm{l}}}{d_{\llcorner}} + \frac{f_{[\mathrm{u}]} \cdot l_{\mathrm{r}}}{d_{\llcorner}} \tag{6.2}$$
$$\mathrm{d}^{\boldsymbol{A}} = \frac{f_{[\mathrm{u}]} \cdot l_{\mathrm{l}} + f_{[\mathrm{u}]} \cdot l_{\mathrm{r}}}{d_{\llcorner}}. \tag{6.3}$$

Wie bereits erwähnt stellt Abb. 6.7 ein vereinfachtes Stereokamerasystem dar. Einerseits handelt es sich bei den Kameras um Lochkameras, d.h. es findet keine Linsenverzeichnung statt und die Projektion von 3D-Punkten auf das Bild der jeweiligen Kamera entspricht den Gleichungen (5.12) und (5.13). Des weiteren sind die Kameras koplanar ausgerichtet. Damit liegen die Bildebenen beider Kameras in derselben Ebene. Hinzu kommt die Annahme der gleichen Bildweite beider Kameras $f_{[\mathrm{u}]}$.

In realen Szenarien ist es unwahrscheinlich, ein solches System vorzufinden, allerdings bietet die OpenCV-Bibliothek (Opensource Computer Vision) eine Funktion zum Rektifizieren zweier Kamerabilder. Dabei werden die Bilder zweier zueinander kalibrierter Kameras derart korrigiert, dass sie dem in Abb. 6.7 dargestellten idealisierten Stereosystem entsprechen. Die gleiche Funktionalität bietet das Software Development Kit (SDK) der Bumblebee XB3, so dass auch für deren rektifizierte Bilder die Annahmen aus Abb. 6.7 zutreffen. Damit lässt sich Gleichung (6.3) vereinfachen und es folgt

$$\mathrm{d}^{\boldsymbol{A}} = \frac{f_{[\mathrm{u}]} \cdot l_{\mathrm{l}} + f_{[\mathrm{u}]} \cdot l_{\mathrm{r}}}{d_{\llcorner}} = f_{[\mathrm{u}]} \cdot \frac{l_{\mathrm{l}} + l_{\mathrm{r}}}{d_{\llcorner}}. \tag{6.4}$$

Bei genauer Betrachtung fällt auf, dass $l_{\mathrm{l}} + l_{\mathrm{r}} = l^{\mathrm{Basis}}$ der Basislänge zwischen beiden Kameras entspricht. Außerdem beschreibt $d_{\llcorner}$ die Orthogonaldistanz zwischen den Projektionszentren $\boldsymbol{B}$ und $\boldsymbol{C}$ der beiden Kameras und dem Punkt $\boldsymbol{A}$. Damit ergibt sich die Abhängigkeit von Disparität und Orthogonaldistanz des Punktes $\boldsymbol{A}$ zu

$$\mathrm{d}^{\boldsymbol{A}} = f_{[\mathrm{u}]} \cdot \frac{l^{\mathrm{Basis}}}{d_{\llcorner}} \tag{6.5}$$

$$\text{und } d_{\llcorner} = f_{[\mathrm{u}]} \cdot \frac{l^{\mathrm{Basis}}}{\mathrm{d}^{\boldsymbol{A}}}. \tag{6.6}$$

Da $f_{[\mathrm{u}]}$ und l^{Basis} für ein gegebenes Stereokamerasystem konstant sind, hängt die Disparität letztlich nur von der Orthogonaldistanz $d_{\llcorner}$ ab. Beide sind zueinander antiproportional, d.h. die Disparität wird kleiner, je weiter entfernt ein Punkt vom Kamerasystem liegt. Das Problem kleiner Disparitäten wird anhand von Gleichung (6.6) deutlich. Wenn $\mathrm{d}^{\boldsymbol{A}}$ gegen 0 läuft, geht $d_{\llcorner}$, aufgrund der Diskretisierung, in immer größer werdenden Schritten gegen ∞. Diesem Problem kann nur durch eine größere Brennweite einerseits, vor allem aber durch eine größere Stereobasis l^{Basis} andererseits begegnet werden. Im Gegenzug verkleinert sich der Überlappungsbereich der Sichtfelder beider Kameras im Nahbereich, wenn sich der Abstand beider Kameras erhöht. Für eine Anwendung in der Robotik sollte sich die Stereobasis beider Kameras demnach nach der Anforderung nach Nah- oder Fernbereichstereo richten.

Die Verwendung der Orthogonaldistanz in den obigen Formeln hat den Vorteil, dass sie eine von der Bildzeile unabhängige Beschreibung der Disparität eines Pixels ermöglicht. Die Formeln behalten daher ihre Gültigkeit für beliebige Bildkoordinaten einer rektifizierten Kamera. Bei der Rückprojektion in den 3D-Raum muss allerdings die Länge des Vektors zwischen Projektionszentrum und Bildpixel einbezogen werden.

Allerdings sind Bildweite und Bildkoordinaten in Pixeln, während die Orthogonaldistanz metrisch angegeben ist. Um dennoch einen metrischen 3D-Vektor zu dem 3D-Punkt zu erhalten, wird in die Betrachtung eine virtuelle Kamera mit einer metrischen Bildweite von 1 m, wie in Abb. 6.8 gezeigt, eingeführt.

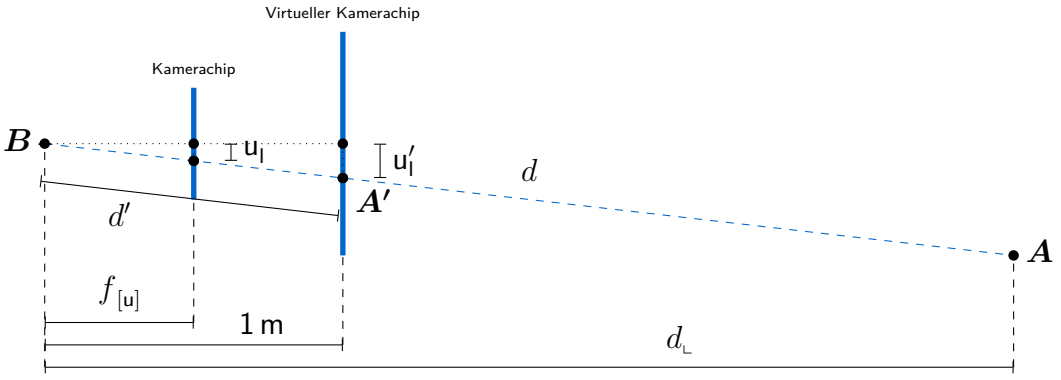


Abbildung 6.8: Rückprojektion in 3D mit Hilfe einer virtuellen metrischen Kamera.

Mit Hilfe des Strahlensatzes ist leicht zu erkennen, dass gilt

$$\frac{d}{d'} = \frac{d_{\llcorner}}{1}. \tag{6.7}$$

Im Folgenden bezeichnet $\boldsymbol{f'}$ den Vektor von $\boldsymbol{B}$ nach $\boldsymbol{A'}$. Zu klären ist nun, wie für beliebige Pixelkoordinaten u,v der entsprechende Vektor $\boldsymbol{f'}$ mit Hilfe der intrinsischen Kameraparameter errechnet werden kann. Aufgrund der Rektifizierung handelt es sich hier um ein Kamerasystem nach dem Lochkameramodell. Damit sind die einzigen Parameter, wie in Abschnitt 5.1.1 beschrieben, die horizontalen und vertikalen Bildweiten $f_{[u]}$ bzw. $f_{[v]}$ sowie die Hauptpunktverschiebungen $c_{[u]}$ und $c_{[v]}$. Die Pixelkoordinaten liegen im Regelfall bzgl. der linken oberen Bildecke vor. Im ersten Schritt findet eine Umwandlung durch Translation und Rotation in bildzentrierte Koordinaten statt

$$\begin{pmatrix} 1 \\ u_l \\ v_l \end{pmatrix} = \begin{pmatrix} 1 \\ u \\ v \end{pmatrix}_{\boxdot} = \begin{bmatrix} 1 & 0 & 0 \\ c_{[u]} & -1 & 0 \\ c_{[v]} & 0 & -1 \end{bmatrix} \cdot \begin{pmatrix} 1 \\ u \\ v \end{pmatrix}_{\ulcorner}. \tag{6.8}$$

Nach dem Strahlensatz ergibt sich

$$\frac{u_{\boxdot}}{f_{[u]}} = \frac{u'_l}{1} \qquad \text{und analog} \qquad \frac{v_{\boxdot}}{f_{[v]}} = \frac{v'_l}{1},$$

wobei u'_{l} die Projektion von u_{l} auf den virtuellen Kamerachip gemäß Abb. 6.8 darstellt und v'_{l} für den vertikalen Fall analog gebildet wird. Damit ergeben sich die y- und z-Komponenten von $\boldsymbol{f'}$ durch Division der bildzentrierten Pixelkoordinaten durch die jeweilige Bildweite, d.h.

$$\boldsymbol{f'} = \begin{pmatrix} 1 \\ \frac{\mathsf{u}_{\boxdot}}{f_{[\mathsf{u}]}} \\ \frac{\mathsf{v}_{\boxdot}}{f_{[\mathsf{v}]}} \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{c_{[\mathsf{u}]}}{f_{[\mathsf{u}]}} & \frac{-1}{f_{[\mathsf{u}]}} & 0 \\ \frac{c_{[\mathsf{v}]}}{f_{[\mathsf{v}]}} & 0 & \frac{-1}{f_{[\mathsf{v}]}} \end{bmatrix} \cdot \begin{pmatrix} 1 \\ \mathsf{u} \\ \mathsf{v} \end{pmatrix}_{\ulcorner} . \tag{6.9}$$

Zusätzlich zu Gleichung (5.34) wird, basierend auf den Gleichungen (6.6) und (6.9), eine weitere Funktion zur Umwandlung von Pixelkoordinaten in 3D-Punkte bzw. 4D-homogene Koordinaten als

$$\mathsf{p}^{-1}\left(\mathsf{p}_{\ulcorner}, \mathsf{d}^{\boldsymbol{p}}, l^{\mathsf{Basis}}\right) = \begin{pmatrix} f_{[\mathsf{u}]} \cdot \frac{l^{\mathsf{Basis}}}{\mathsf{d}^{\boldsymbol{p}}} \\ \mathsf{p}_{[\mathsf{u}]\,\boxdot} \cdot \frac{l^{\mathsf{Basis}}}{\mathsf{d}^{\boldsymbol{p}}} \\ \mathsf{p}_{[\mathsf{v}]\,\boxdot} \cdot \frac{l^{\mathsf{Basis}}}{\mathsf{d}^{\boldsymbol{p}}} \cdot \frac{f_{[\mathsf{u}]}}{f_{[\mathsf{v}]}} \end{pmatrix} = \boldsymbol{p} \tag{6.10}$$

definiert, wobei $\mathsf{d}^{\boldsymbol{p}}$ die Disparität des entsprechenden Bildpunktes ist und l^{Basis} den bereits bekannten Basisabstand des Stereokamerasystems beschreibt. Zur Vereinfachung wird hier die Umwandlung von $\mathbf{S}_{\ulcorner}$ in $\mathbf{S}_{\boxdot}$ gemäß Gleichung (6.8) nur implizit durch den Variablenmodifikator ausgedrückt.

Gängige Stereoverfahren, wie BM, SGM und ELAS, versuchen, für jeden Bildpunkt die Disparität zu bestimmen, um möglichst dichte Disparitätenbilder zu erzeugen. Die in der Gleichung (6.10) dargestellte Funktion, wird, wie eingangs erwähnt, dazu verwendet, die Disparitätenbilder in 3D-Punktwolken umzuwandeln. Wie im folgenden Abschnitt zu sehen ist, verfolgt diese Arbeit einen anderen Ansatz zur Berechnung einer dreidimensionalen Abbildung der Umgebung.

6.2.3 Rekursive Darstellung der Fahrzeugumgebung

Wie bereits erwähnt betrachten die meisten Stereoverfahren jedes Stereobildpaar individuell, wodurch kein Wissenstransfer von einer Messung bzw. von einem Zeitschritt zum nächsten stattfindet. In der Literatur finden sich dazu nur wenige Ausnahmen.

Broggi et al. [2011] beschreiben ein dem BM und SGM ähnliches Verfahren zur Berechnung von Disparitäten, die anschließend in 3D-Punktwolken transformiert werden. Die Besonderheit dieses Verfahrens liegt in einem eingebauten Verifika-

tionsschritt der berechneten 3D-Punkte. Sowohl BM und SGM basieren auf der Suche nach der Disparität auf korrespondierenden Bildpunkten in einem Paar von Stereobildern. Kommt es bei dieser Korrespondenzsuche zu einer falschen Zuordnung, resultiert dies in einer falschen Disparität und damit in einem falschen 3D-Punkt. Mit Hilfe der Sensorik an Bord ihrer Fahrzeuge sind Broggi et al. [2011] in der Lage, auch die 6-DOF-Bewegung (Degrees of Freedom) des Fahrzeugs zwischen zwei Bildaufzeichnungen zu bestimmen. Damit ist es ihnen möglich, die 3D-Punkte in das Tiefenbild der vorherigen Aufnahme zu transformieren. Dort findet ein Vergleich mit der lokalen Pixelnachbarschaft statt. Stimmt die Tiefe des projizierten Punktes nicht mit der seiner Nachbarschaft überein, wird die Tiefenmessung und damit der 3D-Punkt für das aktuelle Stereobildpaar verworfen.

Dieses Verfahren dient lediglich der Filterung der Messungen. Es berücksichtigt nicht alle Grenzfälle und behandelt Unsicherheiten nur ansatzweise. Beispielsweise führt die Verifikation an Tiefenkanten häufig zur Aussortierung korrekter Messungen. Außerdem wird die Größe der Nachbarschaft fest gewählt. Sie ist damit unabhängig von der Entfernung des 3D-Punktes und nicht durch die Unsicherheit in die Eigenbewegung des Fahrzeugs bestimmt. Außerdem basiert die Filterung auf der Tiefe, die, wie in Gleichung (6.6) gezeigt, gerade bei weit entfernten Punkten stärker rauscht als bei nahen Punkten.

Bei dem zweiten Verfahren handelt es sich um das sog. Stixel-Verfahren [Badino et al., 2009, Pfeiffer und Franke, 2010, Dickmann et al., 2013]. Bei einem Stixel handelt es sich um ein rechteckiges Bildelement, welches dem überdeckten Bildbereich eine bestimmte Höhe über einer als Ebene modellierten Grundfläche zuweist. Die Stixel werden dabei rekursiv in Position und Geschwindigkeit geschätzt, so dass sich auch bewegte Objekte über Stixel modellieren lassen. Typischerweise werden dabei mehrere benachbarte Stixel zu Gruppen mit ähnlicher Bewegungsrichtung zusammengefasst. Der Hauptnachteil des Stixel-Verfahrens ist die Annahme einer ebenen Grundfläche, die in vielen Fällen nicht getroffen werden kann.

Diese Arbeit stellt ein neues Verfahren vor, genannt Rekursives Automotives Stereo (RAS) [Schneider et al., 2014]. Abb. 6.9 zeigt eine schematische Darstellung des Verfahrens. Als Eingang dient dazu ein Stereobildpaar und die Eigenbewegung des Fahrzeugs. Mit Hilfe eines Stereo-Verfahrens wird daraus ein Disparitätenbild bestimmt. Zusammen mit der Eigenbewegung dient es als Messung für den Filterblock, der wiederum eine 3D-Punktwolke ausgibt.

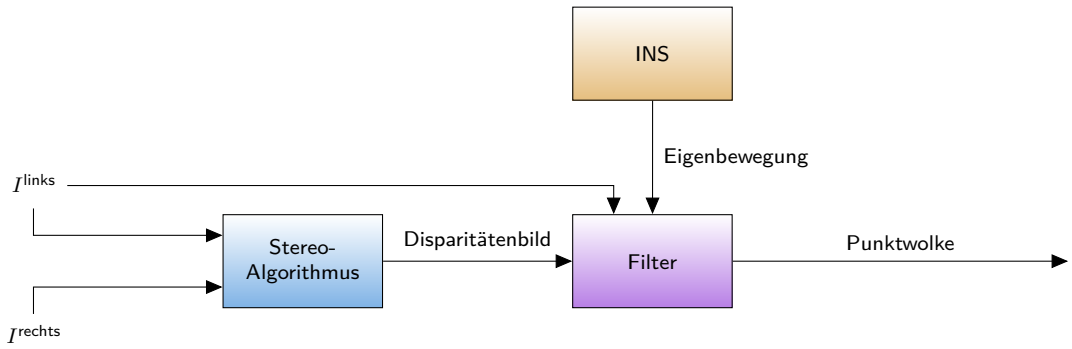


Abbildung 6.9: Schematische Darstellung von RAS.

Das Verfahren stellt keinerlei Anforderungen an das verwendete Stereo-Verfahren, solange es als Ausgabe ein Disparitätenbild bereitstellt. Die Kameras, die beide Bilder I^{links} und I^{rechts} bereitstellen, müssen allerdings intrinsisch und extrinsisch kalibriert sein, da im Filterblock eine Projektion von 3D-Punkten in das Bild der linken Kamera stattfindet. Zu den Bildern synchron wird außerdem die Eigenbewegung des Fahrzeugs benötigt, da mit dieser die Prädiktion der systeminternen 3D-Punktwolke stattfindet. Die Synchronisation von Bildern und Eigenbewegung entspricht dem in Abschnitt 5.9.4 vorgestellten Verfahren.

Zustandsvektor Der Einfachheit halber wird das System zunächst für eine statische Fahrzeugumgebung beschrieben, d.h. jegliche Bewegung in der Szene wird als ausschließlich von dem Versuchsträger selbst verursacht angenommen. Eine Liste von 3D-Punkten repräsentiert dabei die Fahrzeugumgebung bzgl. des Fahrzeugkoordinatensystems $\mathbf{S}_{\text{ego}}$. Jeder dieser Punkte wird individuell mit Hilfe eines Unscented Kalmanfilter (UKF) gefiltert. Der entsprechende Zustandsvektor pro Punkt lautet damit

$$\boldsymbol{x} = (x, y, z)^T . \tag{6.11}$$

Prozessmodell Es gilt die Annahme, dass ein INS die Bewegung ${}^{\text{ego},k}\mathbf{P}_{\text{ego},k-1}$ des Fahrzeugs von einem Filterschritt zum nächsten, d.h. zwischen zwei Stereobildaufzeichnungen, bereitstellt. Aufgrund der Annahme einer statischen Welt muss der zum Zustand gehörende Punkt daher nur in das neue Koordinatensystem trans-

formiert werden. Dazu wird der Steuervektor $\boldsymbol{u}_{k-1} = {}^{\text{ego},k}\mathbf{P}_{\text{ego},k-1}$ definiert. Das Systemmodell lautet damit

$$f\left(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}\right) = htm\left(\boldsymbol{u}_{k-1}\right) \cdot \boldsymbol{x}_{k-1}. \tag{6.12}$$

Messmodell 1 Wie eingangs erwähnt finden sich in der Literatur eine Vielzahl von Verfahren, die ein Disparitätenbild in eine 3D-Punktwolke umwandeln und bei denen darauf aufbauend die Weiterverarbeitung auf Basis dieser Punktwolke stattfindet. Ein solches Vorgehen wäre auch hier denkbar, allerdings erfordert das UKF im Hinblick auf Gleichung (6.6) eine Quantifizierung des Messrauschens. Tiefe und Disparität sind zueinander antiproportional. Würde ebenfalls in 3D gemessen, müsste damit auch das Messrauschen metrisch angegeben werden. Allerdings ist das Messrauschen abhängig von der Entfernung der 3D Punkte zur Kamera, d.h. weit entfernte Punkte unterliegem stärkeren Messrauschen als nahe Punkte.

Das Messen der Disparität hingegen gestaltet sich wesentlich einfacher. Wird davon ausgegangen, dass die Disparität in jedem Bereich des Bildes und unabhängig von ihrer Größe mit der gleichen Ungenauigkeit gemessen werden kann, vereinfacht sich ebenfalls die Bestimmung des Messrauschens, da es in diesem Fall nicht mehr vom Messwert selbst abhängt. Zur Überprüfung dieser Annahme findet eine Aufzeichnung von 100 Stereobildern einer statischen Szene statt. Der BM-Algorithmus bestimmt nun für jedes Bildpaar das entsprechende Disparitätenbild. Davon ausgehend lassen sich für jeden Bildpunkt der 100 Bilder anschließend Mittelwert und Standardabweichung bestimmen. Abb. 6.10 zeigt die mittlere Disparität der gemessenen Pixel in Form eines Disparitätenbildes.

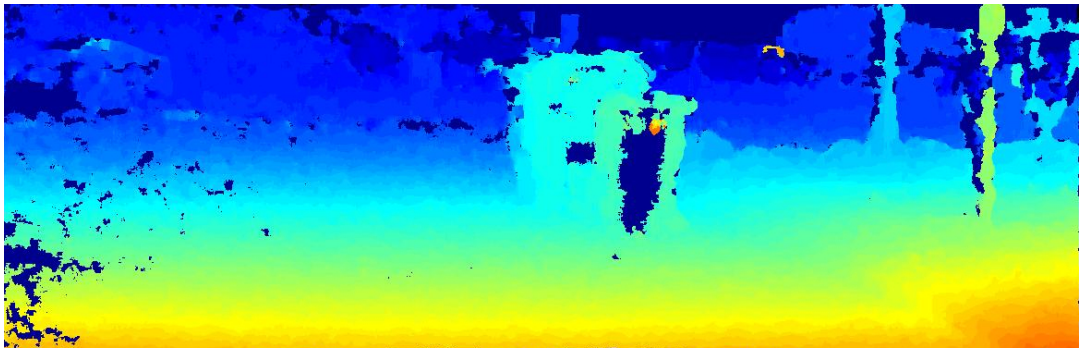


Abbildung 6.10:
Mittlere Disparität über 100 Disparitätenbilder einer statischen Szene. ■ entspricht Pixeln im Nahbereich mit hoher Disparität. Pixel mit geringer Disparität sind in ■ dargestellt.

Abbildung 6.11 zeigt ein Histogramm über die Standardabweichungen der Disparität von jedem Pixel der 100 Disparitätenbilder. Es gibt Aufschluss darüber, dass die Standardabweichung der Disparität für 99,43 % aller Pixel kleiner ist als 2 pix und für 97,11 % aller Pixel sogar kleiner ist als 1 pix. Damit ist das Messrauschen zwar nicht konstant, kann allerdings mit einem Rauschen von $1\sigma \approx 1\,\text{pix}$ angenähert werden.

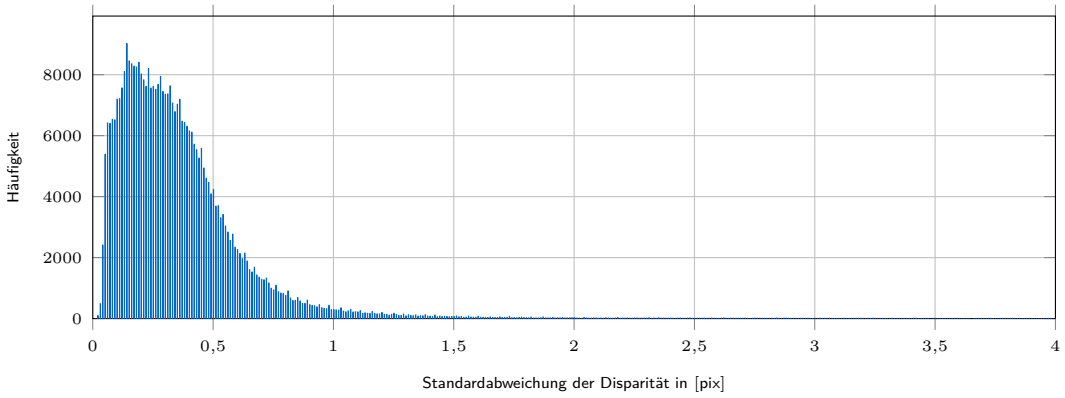


Abbildung 6.11:
Histogram über die Standardabweichungen der Disparität von jedem Pixel der 100 Disparitätenbilder.

Die Aufgabe des Messmodells ist es nun, für einen prädizierten Zustand $\boldsymbol{x}_k^*$ die zu erwartende Disparität zu berechnen. Das Filtern der 3D-Koordinaten des Punktes arbeitet bzgl. des Referenzkoordinatensystems $\mathbf{S}_{\text{ego}}$. Darum müssen die Koordinaten zunächst in das Koordinatensystem der zum Disparitätenbild korrespondierenden Kamera projiziert werden. Dabei ist es vom Stereoverfahren abhängig, ob es sich dabei um die linke oder rechte Kamera handelt. Das hier beschriebene Verfahren arbeitet unabhängig von der gewählten Kamera, weshalb hier nur vom Kamerakoordinatensystem $\mathbf{S}_{\text{Kamera}}$ die Rede ist. Es gilt

$$\begin{pmatrix} \boldsymbol{x}^*_{\text{Kamera},k} \\ 1 \end{pmatrix} = {}^{\text{Kamera},k}\mathbf{H}_{\text{ego},k} \cdot \begin{pmatrix} \boldsymbol{x}^*_k \\ 1 \end{pmatrix}. \tag{6.13}$$

Bei ${}^{\text{Kamera},k}\mathbf{H}_{\text{ego},k}$ handelt es sich um die homogene Transformationsmatrix, die entsprechend der extrinsischen Kalibrierung der Disparitätenkamera Punkte von $\mathbf{S}_{\text{ego}}$ in das Kamerakoordinatensystem transformiert. ${}^{\text{Kamera},k}\mathbf{H}_{\text{ego},k}$ ist, wie am Zeitindex k erkennbar, zeitabhängig. Dadurch ist es möglich, auf Änderungen der Einbaulage der Kamera einzugehen. Derartige Änderungen können u.a. durch eine Drehung der Kameraplattform oder durch eine *online*-Kalibrierung der Kamera auftreten.

Zur Berechnung der Disparität eines Punktes genügt gemäß Gleichung (6.5) die Orthogonaldistanz des Punktes zum Projektionszentrum der Kamera. In diesem Fall entspricht die Orthogonaldistanz der x-Komponente des nach $\mathbf{S}_{\mathsf{Kamera}}$ transformierten Zustandvektors $\boldsymbol{x}^*_{[x]\,\mathsf{Kamera},k}$. Durch Einsetzen von $\boldsymbol{x}^*_{[x]\,\mathsf{Kamera},k}$ in Gleichung (6.5) ergibt sich die erwartete Disparität

$$\boldsymbol{y}_k^{\mathsf{Disparität}\,*} = f_{\mathsf{[u]}} \cdot \frac{l^{\mathsf{Basis}}}{\boldsymbol{x}^*_{[x]\,\mathsf{Kamera},k}}. \tag{6.14}$$

Demgegenüber steht die eigentliche Messung der Disparität. Mit Hilfe von Gleichung (5.33) errechnet sich die Projektion von $\boldsymbol{x}^*_{\mathsf{Kamera},k}$ in das Disparitätenbild. Daraus folgend entspricht die Messung, sofern vorhanden, der Disparität an der sich projizierten Pixelposition

$$\boldsymbol{y}_k^{\mathsf{Disparität}} = I_k^{\mathsf{Disparität}}\left(\mathsf{x}_k^*\right), \tag{6.15}$$

$$\text{wobei } \mathsf{x}_k^* = \mathsf{p}\left(\boldsymbol{x}^*_{\mathsf{Kamera},k}\right). \tag{6.16}$$

In der Praxis kann es passieren, dass einzelne Disparitätenmessungen aufgrund falscher Korrespondenzbestimmung stark von der prädizierten Messung abweichen. Aus diesem Grund werden solche Disparitätsmessungen aussortiert, die weiter als ein bestimmter Abstand von der prädizierten Messung abweichen. Hierzu wäre ein fester Schwellwert denkbar. Damit würde allerdings vernachlässigt, dass das UKF die Kovarianz $\mathbf{P}$ des eigenen Zustands kennt und somit ein Maß dafür existiert, wie zuverlässig das Filter bei der aktuellen Schätzung ist. Allerdings ist $\mathbf{P}$ bzgl. des Zustandsraums definiert, in diesem Fall kartesische 3D-Koordinaten. Im Rahmen der Unscented Transform (UT), aufgerufen durch Gleichung (4.51), wird in Gleichung (4.49) die Innovationskovarianz $\mathbf{S}_k$ berechnet. Die Innovationskovarianzmatrix ist eine Repräsentation der Zustandskovarianz im Messraum zuzüglich des Messrauschens. Im hier vorliegenden Messmodell ist die Innovationskovarianz eindimensional und gib die Varianz der prädizierten Disparität $\boldsymbol{y}_k^{\mathsf{Disparität}\,*}$ an. Daraus lässt sich ein dynamischer Schwellwert für die Gültigkeitsprüfung einer Disparitätsmessung ableiten. Sei $\sigma^{2\mathsf{d}}$ die Innovationsvarianz der prädizierten Disparität. Eine Messung $\boldsymbol{y}_k^{\mathsf{Disparität}}$ wird nur akzeptiert, wenn der Betrag der Differenz von prädizierter Disparität und gemessener Disparität kleiner ist als die zweifache Standardabweichung der prädizierten Disparität

$$\left|\boldsymbol{y}_k^{\mathsf{Disparität}\,*} - \boldsymbol{y}_k^{\mathsf{Disparität}}\right| < 2 \cdot \sqrt{\sigma^{2\mathsf{d}}}. \tag{6.17}$$

Damit akzeptiert das Filter bei hoher Kovarianz auch weiter entfernt liegende Disparitätsmessungen. Je mehr das Filter konvergiert und die Kovarianz schrumpft, desto besser müssen die Disparitätsmessungen zur prädizierten Messung passen, damit sie nicht verworfen werden.

Auch wenn das UKF mit diesem Messmodell prinzipiell arbeitet, stellt sich die Frage, ob das Filter aus der Disparität allein genug Informationen extrahieren kann, um alle Zustandsgrößen schätzen zu können. Das UKF überführt im Innovationsschritt die Differenz zwischen erwarteter Disparität und gemessener Disparität in eine Korrektur des Zustands. Da die Disparität nur in Zusammenhang mit der Entfernung des Punktes steht und nicht mit dessen lateraler Lage, kann die Korrektur nur in der Distanz zum Punkt wirken. Der Punkt wird dadurch lediglich entlang des Sehstrahls vom Projektionszentrum der Kamera zum Punkt selber verschoben. Da die Filterung selbst bzgl. $\mathbf{S}_{\text{ego}}$ stattfindet und daher noch eine Transformation zwischen $\mathbf{S}_{\text{ego}}$ und $\mathbf{S}_{\text{Kamera}}$ involviert ist, bedeutet das nicht automatisch, dass $\boldsymbol{x}_{[y]}$ bzw. $\boldsymbol{x}_{[z]}$ nicht beobachtbar sind. Im Filter lässt sich lediglich eine wachsende Kovarianz lateral zur Orthogonaldistanz zur Kamera feststellen. In dieser Dimension ist der Punkt mit dem vorgestellten Messmodell daher nur schwer lokalisierbar.

Messmodell 2 Um diesem Problem zu begegnen, muss eine Stabilisierung auch lateral zum Sehstrahl durchgeführt werden. Aufgrund der hohen lateralen Auflösung der Kameras kommt ein weiteres, Bild-basiertes Messmodell zur Ausführung. Auch wenn an dieser Stelle die Wahl der verwendeten Kamera beliebig ist, basiert auch dieses Messmodell, analog zu vorher, auf der in $\mathbf{S}_{\text{Kamera}}$ verbauten Kamera. Ein Zwischenergebnis des Disparitätenmessmodells ist die zum prädizierten Punkt korrespondierende Projektion in das Kamerabild einer der beiden Stereokameras. Diese Position wird auch im zweiten, Deskriptor-basierten Messmodell, verwendet. Unsere prädizierte Messung entspricht im Grunde Gleichung (6.16)

$$\boldsymbol{y}_k^{\text{Deskriptor}\,*} = \begin{pmatrix} \mathsf{u}_k^* \\ \mathsf{v}_k^* \end{pmatrix} = \mathsf{x}_k^* = \mathsf{p}\left(\boldsymbol{x}_{\text{Kamera},k}^*\right). \tag{6.18}$$

Die Frage ist nun, wie die Position des zu $\boldsymbol{x}_k$ gehörenden Bildpunktes gemessen wird, um damit $\boldsymbol{y}_k^{\text{Deskriptor}}$ zu bestimmen. Abschnitt 5.9 listet bereits eine Reihe von Schlüsselpunkt-basierten Verfahren zur Korrespondenzbestimmung auf. Diese Verfahren bestehen meistens aus zwei Schritten: Extraktion von Schlüsselpunkten und Beschreibung der Schlüsselpunkte. Auch wenn die Deskriptoren zur Beschreibung

der Schlüsselpunkte darauf ausgelegt sind, die vom dazu passenden Verfahren bereitgestellten Schlüsselpunkte zu beschreiben, eignen sich die Deskriptoren in lokalen Bildbereichen häufig ebenfalls zur Beschreibung der Nachbarschaft eines beliebigen Bildpunktes.

Ein solches Verfahren ist Binary Robust Independent Elementary Features (BRIEF) [Calonder et al., 2010], das hier aufgrund des geringen Rechenbedarfs verwendet wird. Für jeden gefilterten 3D-Punkt wird ein 16-dimensionaler BRIEF-Deskriptor hinterlegt. Im Rahmen der Filterinnovation findet eine Berechnung der Deskriptoren aller Bildpunkte in einem 10 pix × 10 pix großen, um $\boldsymbol{y}_k^{\text{Deskriptor}\,*}$ zentrierten Fenster statt. Anschließend erfolgt ein Vergleich des hinterlegten Deskriptors mit den 100 benachbarten Deskriptoren. Als Distanzmaß kommt die Kosinus-Distanz zum Einsatz

$$D^{\text{Kosinus}}\left(\boldsymbol{a}, \boldsymbol{b}\right) = \left| 1 - \frac{\langle \boldsymbol{a}, \boldsymbol{b} \rangle}{||\boldsymbol{a}|| \cdot ||\boldsymbol{b}||} \right|. \tag{6.19}$$

Die Kosinus-Distanz hat einen Wertebereich von [0,1] und ist 0, wenn beide Vektoren linear abhängig sind und damit in die gleiche Richtung zeigen. Die Distanz wird zu 1, wenn beide Vektoren rechtwinklig zueinander stehen. Von allen 100 Bildpunkten wählt RAS denjenigen als Messung $\boldsymbol{y}_k^{\text{Deskriptor}}$, bei dem die Kosinus-Distanz minimal wird und gleichzeitig unter einem Wert von 0,25 liegt. Anschließend aktualisiert RAS den im Filter hinterlegten Deskriptor mit dem ähnlichsten aus der Nachbarschaft. Liegt die minimale Distanz über 0,25, wird die Messung verworfen. Der Schwellwert wurde dabei empirisch ermittelt und dient der Behandlung von Fällen, in denen bspw. aufgrund von Verdeckung keine korrekte Korrespondenz gefunden werden kann.

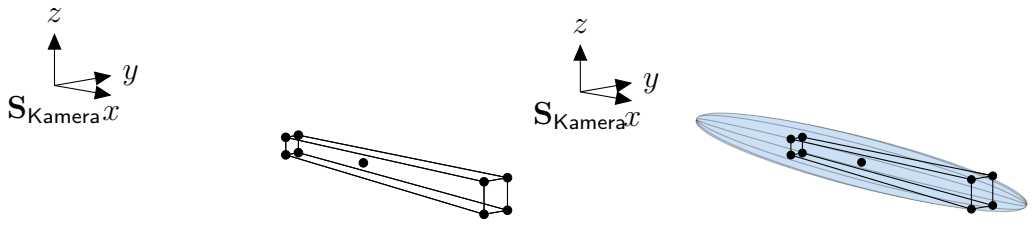
Wie in der Beschreibung beider Messmodelle angedeutet, besteht bei beiden Messmodellen die Möglichkeit, dass keine gültige Messung gefunden wird. Einerseits liefern Verfahren wie BM, SGM oder ELAS nicht für jeden Bildpunkt eine Disparität, und andererseits findet das zweite Messmodell aufgrund der Ähnlichkeitsbedingung möglicherweise auch keinen passenden Deskriptor. Aus diesem Grund verwendet RAS eine sequentielle Version des UKF und filtert zunächst die Deskriptor-basierte Messung $\boldsymbol{y}_k^{\text{Deskriptor}}$ und anschließend die Disparitätsmessung $\boldsymbol{y}_k^{\text{Disparität}}$. Das sequentielle Filtern hat den Vorteil, dass beide Messungen unabhängig vom Vorhandensein der anderen zur Verbesserung der Schätzung herangezogen werden. Für den Fall, dass keine der Messungen gültig ist, findet keine Innovation statt. Das macht RAS robust gegenüber fehlenden Messungen, die beispielsweise auch durch Blendungen der Kamera verursacht werden.

Initialisierung Neben der Beschreibung von Systemmodell und Messmodellen bleibt noch offen, wie die Initialisierung eines Filters durchgeführt und wie $\boldsymbol{x}_0$ und $\mathbf{P}_0$ definiert werden. Wie im nächsten Abschnitt gezeigt wird, geht die Initialisierung eines Filters von Bildkoordinaten aus. Daher lässt sich der Zustandsvektor $\boldsymbol{x}_0$ mit der bereits bekannten Rückprojektion aus Gleichung (6.10) initialisieren. Schwieriger hingegen ist die Initialisierung von $\mathbf{P}_0$. Für ein kalibriertes Stereokamerasystem hängt die Rückprojektion hauptsächlich von drei Parametern ab: Den Pixelkoordinaten $\mathsf{p}_{[\mathsf{u}]\,\ulcorner}$, $\mathsf{p}_{[\mathsf{v}]\,\ulcorner}$ und der gemessenen Disparität d^p. Die initiale Filterkovarianzmatrix beschreibt allerdings die Kovarianz in den drei Koordinatenachsen. Dazu wird zunächst analysiert, wie sich die Unsicherheiten in $\mathsf{p}_{[\mathsf{u}]\,\ulcorner}$, $\mathsf{p}_{[\mathsf{v}]\,\ulcorner}$ und d^p auf die 3D-Koordinaten auswirken.

Durch partielle Ableitung der Gleichung (6.10) ergibt sich

$$\frac{\partial\, \boldsymbol{p}}{\partial\, \mathsf{p}_{[\mathsf{u}]\,\boxdot}} = \begin{pmatrix} 0 \\ \frac{l^{\mathsf{Basis}}}{\mathsf{d}^p} \\ 0 \end{pmatrix}, \quad \frac{\partial\, \boldsymbol{p}}{\partial\, \mathsf{p}_{[\mathsf{v}]\,\boxdot}} = \begin{pmatrix} 0 \\ 0 \\ \frac{l^{\mathsf{Basis}}}{\mathsf{d}^p} \end{pmatrix} \quad \text{und} \quad \frac{\partial\, \boldsymbol{p}}{\partial\, \mathsf{d}^p} = \begin{pmatrix} -f_{[\mathsf{u}]} \cdot \frac{l^{\mathsf{Basis}}}{\mathsf{d}^{p\,2}} \\ -\mathsf{p}_{[\mathsf{u}]\,\boxdot} \cdot \frac{l^{\mathsf{Basis}}}{\mathsf{d}^{p\,2}} \\ -\mathsf{p}_{[\mathsf{v}]\,\boxdot} \cdot \frac{l^{\mathsf{Basis}}}{\mathsf{d}^{p\,2}} \end{pmatrix}. \quad (6.20)$$

Es ist erkennbar, dass Änderungen an den Pixelkoordinaten nur linear auf die y- bzw. z-Koordinaten des Punktes $\boldsymbol{p}$ eingehen. Die Stärke dieser Auswirkung ist dabei antiproportional zur Disparität. Änderungen an der Disparität d^p hingegen beeinflussen alle drei Raumkoordinaten gleichermaßen. Für ein Messtupel $\boldsymbol{y} = \left(\mathsf{p}_{[\mathsf{u}]\,\boxdot}, \mathsf{p}_{[\mathsf{v}]\,\boxdot}, \mathsf{d}^p\right)^T$ mit den Standardabweichungen $\sigma_{[\mathsf{u}]}$, $\sigma_{[\mathsf{v}]}$ und $\sigma_{[\mathsf{d}]}$ spannt der 1σ-Abstand ein Paralleltrapez um den zugehörigen 3D-Punkt auf. Dieses Paralleltrapez ist in Abb. 6.12a dargestellt.



(a) 1σ-Abstand als Paralleltrapez. **(b)** 1σ-Abstand als Ellipsoid.

Abbildung 6.12: Initialisierung des Filters.

Die Aufgabe besteht darin, dass Paralleltrapez in eine entsprechende Kovarianzmatrix umzuwandeln. Im 3D-Raum beschreiben Kovarianzmatrizen allerdings kein Trapez, sondern, wie in Abb. 6.12b dargestellt, einen Ellipsoid. Daher muss nun eine 3D-Kovarianzmatrix gefunden werden, die dem Messrauschen mit den Stan-

dardabweichungen $\sigma_{[\mathsf{u}]}$, $\sigma_{[\mathsf{v}]}$ und $\sigma_{[\mathsf{d}]}$ entspricht. Sei $\mathbf{R}^{\boldsymbol{y}}$ die Kovarianzmatrix zur Messung $\boldsymbol{y}$ mit

$$\mathbf{R}^{\boldsymbol{y}} = \begin{bmatrix} {\sigma^2}_{[\mathsf{u}]} & 0 & 0 \\ 0 & {\sigma^2}_{[\mathsf{v}]} & 0 \\ 0 & 0 & {\sigma^2}_{[\mathsf{d}]} \end{bmatrix}, \tag{6.21}$$

dann lässt sich auch hier die UT aus Abschnitt 4.5 zunutze machen und $\hat{\mathbf{P}}_0$ wie folgt approximieren

$$\left\{ \hat{\boldsymbol{x}}_0, \hat{\mathbf{P}}_0, \cdot, \cdot \right\} = ut\left(\boldsymbol{y}, \mathbf{R}^{\boldsymbol{y}}, \mathsf{p}_{\mathsf{ego}}^{-1}\left(\cdot, \cdot, l^{\mathsf{Basis}} \right), \mathbf{0} \right), \tag{6.22}$$

wobei $\mathsf{p}_{\mathsf{ego}}^{-1}(\cdot)$ die Rückprojektion in $\mathbf{S}_{\mathsf{Kamera}}$ mit anschließender Koordinatentransformation in $\mathbf{S}_{\mathsf{ego}}$ beschreibt. Nach Gleichung (6.22) liefert die UT neben einer Schätzung für die initiale Filterkovarianzmatrix $\hat{\mathbf{P}}_0$ auch eine Schätzung $\hat{\boldsymbol{x}}_0$ für den initialen Zustandsvektor. Mit dieser Schätzung $\hat{\boldsymbol{x}}_0$ wird das Filter initialisiert, da sie der Streuung der Messung gerechter wird als $\boldsymbol{x}_0$. Auch wenn diese Art der Initialisierung mit $\hat{\boldsymbol{x}}_0$ und $\hat{\mathbf{P}}_0$ letztlich eine Approximation darstellt, bleibt zu erwähnen, dass diese Umwandlung nur in dem Fall notwendig ist, wenn ein neuer 3D-Punkt hinzugefügt wird. Ab diesem Zeitpunkt laufen Messung und Prädiktion in den jeweils passenden Domänen ab. Messungen stammen ausschließlich aus Kamerabildern, wo das Messrauschen entsprechen quantifizierbar ist. Die Prädiktion geschieht in 3D, wo später die Erweiterung auf bewegte Punkte leichter fällt.

6.2.4 Analyse des Filterverhaltens anhand von Realdaten

Um zu untersuchen, wie sich das beschriebene Filter auf Basis realer Daten verhält, dient der folgende Versuch mit dem Versuchsträger MuCAR-4. Zu Beginn des Versuchs steht das Fahrzeug in einem Abstand von ca. 24 m vor einer Wand. Anschließend beschleunigt es bis zu einer Geschwindigkeit von etwa 4 m/s und bremst schließlich ab, so dass es ca. 4 m vor dieser Wand zum Stehen kommt.

Als Kamerasystem kommt das Stereokamerasystem des Versuchsträgers zum Einsatz. Die Disparitätenbilder werden mit Hilfe des BM-Algorithmus berechnet. Die Wand ist rauh verputzt, so dass das BM-Verfahren dichte Disparitäten berechnen kann. Die Initialisierung des Filters basiert auf der Bildmitte der linken Kamera, dessen korrespondierender Punkt auf der Wand liegt. Der Versuch soll zeigen, wie sich Schätzung und deren Varianz im Laufe des Versuchs ändern. Als Referenz dienen

uns dabei die Punktmessungen eines LiDAR. Dazu werden zu jedem Kamerabild diejenigen 3D-Punkte des LiDAR bestimmt, die nach Projektion in das Kamerabild auf der Wand liegen. Diese Punkte wurden manuell nach der Aufnahme markiert. Aus diesen Punkten ergeben sich Mittelwert und Standardabweichung in x-Richtung.

Zum Vergleich von prädizierter und realer Messung, die als Disparitäten definiert sind, mit der Referenz als Tiefe werden die Disparitäten und die Innovationskovarianz in Tiefen gemäß Gleichung (6.10) transformiert. Die genannten Werte sind in Abb. 6.13 dargestellt.

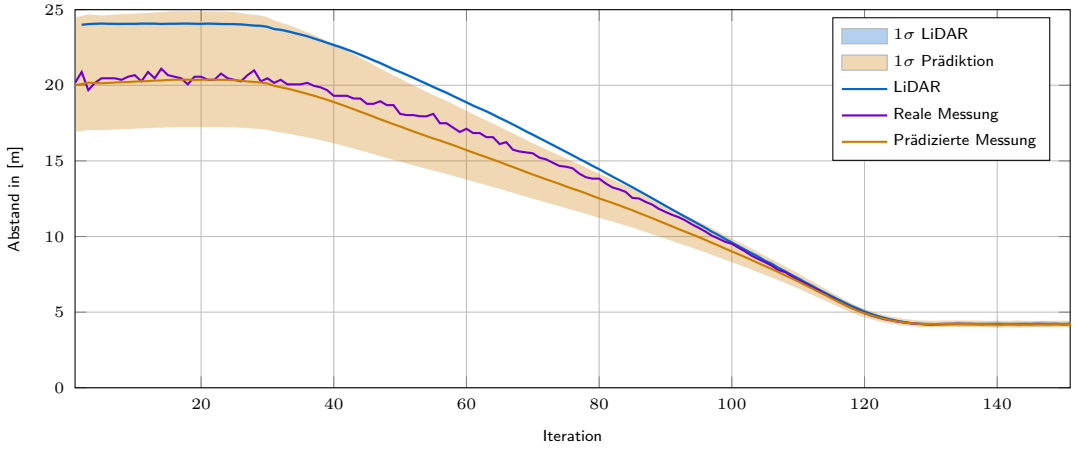


Abbildung 6.13: Annäherung an ein stehendes Hindernis.

Die Standardabweichung der LiDAR-Referenzpunkte liegt im Bereich weniger Zentimeter. Durch die Darstellung als 1σ-Abstand, ist diese kaum erkennbar. Zu Beginn des Versuchs bis Iteration 30 fällt auf, dass die zur gemessenen Disparität entsprechende Entfernung um etwa 4 m von der durch den LiDAR gemessenen Referenz entfernt liegt. Bei der verwendeten Kamera und dem gegebenen Versuchsaufbau entspricht dies allerdings nur einem Fehler in der Disparität von 2 pix und liegt damit noch innerhalb des hier ebenfalls dargestellten 1σ-Abstands des Prädiktionsfehlers. Dennoch wird der zugehörige 3D-Punkt zunächst zu nahe am Fahrzeug initialisiert. Während der Bewegung in Richtung Wand ist das Rauschen der Messung in 3D erkennbar. Je näher das Fahrzeug der Wand kommt, desto stärker nimmt das Rauschen in 3D ab. In Pixeln bleibt das Rauschen unverändert, durch die Projektion von Disparität in Entfernung äußert es sich bei größerer Disparität entsprechend weniger. Letztlich konvergiert das Filter auf die finale Position und damit auch die Messung des Referenzsensors. Im Bereich der Iterationen 40 bis 100 fällt auf, dass die prädizierte Messung stets unterhalb der wahren Messung bleibt. Der Grund dafür liegt im zweiten Messmodell, dem Deskriptor-Messmodell. Dieses hält das

Filter auf der initial zu nah geschätzten Position und nähert sich erst im Verlauf der Filterung der wahren Distanz an. Es verhindert gleichzeitig, dass verrauschte Disparitätenmessungen die geschätzte Position des 3D-Punktes springen lassen.

Das Deskriptor-Messmodell dient, neben dem Prozessmodell, auch dazu, die Schätzung des 3D-Punktes im Falle fehlender Disparitätsmessungen zu stützen. Um dies zu zeigen, wird der vorhergehende Versuch wiederholt. Diesmal findet die Initialisierung des Filters allerdings an einer Bildposition statt, für die, aufgrund fehlender Texturierung, die Disparitätsbestimmung mit Hilfe des BM-Algorithmus zeitweise fehlschlägt oder Messungen aufgrund der Gültigkeitsprüfung über die Innovationskovarianz verworfen werden. Abbildung 6.14 zeigt den Verlauf der Schätzung dieses Punktes.

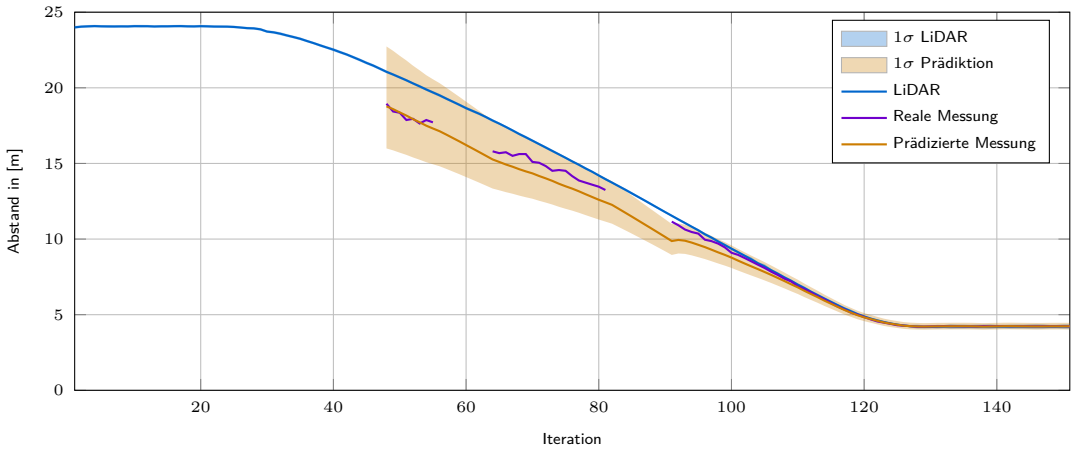


Abbildung 6.14:
Annäherung an ein stehendes Hindernis mit unterbrochener Disparitätsmessung.

Auch in diesem Fall ist die durch den LiDAR bereitgestellte Referenz eingeblendet. Das Filtern beginnt mit der ersten Disparitätsmessung etwa bei Iteration 50. Auch diesmal wird die Wand initial zu nah am Fahrzeug geschätzt, was sich im weiteren Verlauf der Filterung ausgleicht. Erkennbar sind die fehlenden Disparitätsmessungen um die Iterationen 60 und 85. Während dieser Phasen stützen jedoch weiterhin das Deskriptor-Messmodell und das Prozessmodell die Schätzung, was am nahezu parallelen Verlauf von Schätzung und Referenz und des sich deutlich weniger verringerenden Rauschens der Prädiktion erkennbar ist. Wenn zur Iteration 90 plötzlich erneut Disparitätsmessungen zur Verfügung stehen, die nun auch nahe an der Referenz liegen, gleicht das Filter das Delta zügig aus, da die Diskrepanz zwischen erwarteter und gemessener Disparität durch die Nähe zur Wand deutlich größer ist. Letztlich

findet auch hier die Schätzung den korrekten Abstand zum Fahrzeug, wie auch schon im ersten Versuch.

Wie bei jedem Filter sind auch hier die entscheidenden Parameter zur Beeinflussung des Schätzverlaufs die Kovarianzmatrizen der Messung und der Prädiktion. In den hier gezeigten Fällen wurde dem Prozessmodell und dem Deskriptor-Messmodell deutlich stärker vertraut, so dass die rauschenden Disparitätsmessungen, abgesehen von der Initialisierung, weniger stark in die Schätzung eingegangen sind. Wie die Kovarianzen aufeinander abgestimmt werden müssen, hängt in der Realität zum einen von der Auflösung und vom Bildrauschen der Kamera und zum anderen vom verwendeten Disparitätsverfahren ab.

6.2.5 Verwaltung mehrerer Punktfilter

Die bisherige Beschreibung umfasst lediglich die Filterung eines einzelnen 3D-Punktes. Wie eingangs erwähnt, bildet das Verfahren intern eine auf mehreren 3D-Punkten basierende Repräsentation der Umgebung. Das Disparitätenbild, das in jedem Zeitschritt berechnet wird, bietet die Möglichkeit, derartige gefilterte 3D-Punkte zu jedem Zeitschritt zu initialisieren. Potentielle Kandidaten für eine Initialisierung sind alle Pixel, für die eine Disparität berechnet werden konnte und die nicht bereits mit einem gefilterten 3D-Punkt korrespondieren. Bei einer beispielhaften Auflösung von $1280 \times 960\,\mathrm{pix}$ ergibt sich, auch unter der Berücksichtigung, dass für einige Punkte keine Disparität berechnet wurde, eine Punktmenge, die die Central Processing Unit (CPU) selbst aktueller Rechner an den Rand der Berechenbarkeit im Bildtakt oder gar darüber hinaus bringt.

Um den Ressourcen verschiedener Plattformen gerecht zu werden, wird ein Konzept für den Lebenszyklus eines gefilterten 3D-Punktes definiert, das den Kompromiss zwischen Detailierungsgrad der Umgebungsrepräsentation und verfügbaren Rechenressourcen erlaubt. Dazu erfolgt eine Unterteilung des Disparitätenbilds des Stereokamerasystems in ein gleichmäßiges Raster mit einstellbarer Zellgröße. Ziel ist es, in jeder der Zellen mindestens einen gefilterten 3D-Punkt zu haben, um den gesamten Sichtbereich der Kamera abzudecken. Kleinere Zellengrößen führen damit automatisch zu einer höheren Anzahl gefilterter 3D-Punkte und damit zu höherem Rechenaufwand. Entsprechend verringert sich der Rechenaufwand durch eine Vergrößerung der Zellen.

Prinzipiell muss das hinterlegte Gitter nicht regelmäßig sein. Bei Bedarf lassen sich bestimmte Bildbereiche auch dynamisch höher oder niedriger auflösen. Ein gleichmäßiges Gitter hat für eine in Fahrtrichtung gerichtete Kamera allerdings die Eigenschaft, dass es implizit durch die perspektivische Abbildung die weiter entfernte Umgebung gröber abtastet als die nahe Umgebung. Für ein System zur Hindernisvermeidung ist dies ein intuitiver Ansatz, da die Hindernisvermeidung vor allem im Nahbereich um das Fahrzeug eine genaue und dichte Abtastung der Umgebung benötigt, während die weitere Entfernung lediglich zur groben Hinderniserkennung dient und damit auch nur gröber abgetastet werden muss.

Der Lebenszyklus eines gefilterten 3D-Punkts besteht aus drei Phasen: Initialisierung, Filterung und Deinitialisierung. Demgegenüber besteht jede Iteration aus ebenfalls drei Schritten: Prädiktion und Innovation der gefilterten 3D-Punkte sowie Filterverwaltung. Das Zusammenspiel beider ist in Abb. 6.15 dargestellt.

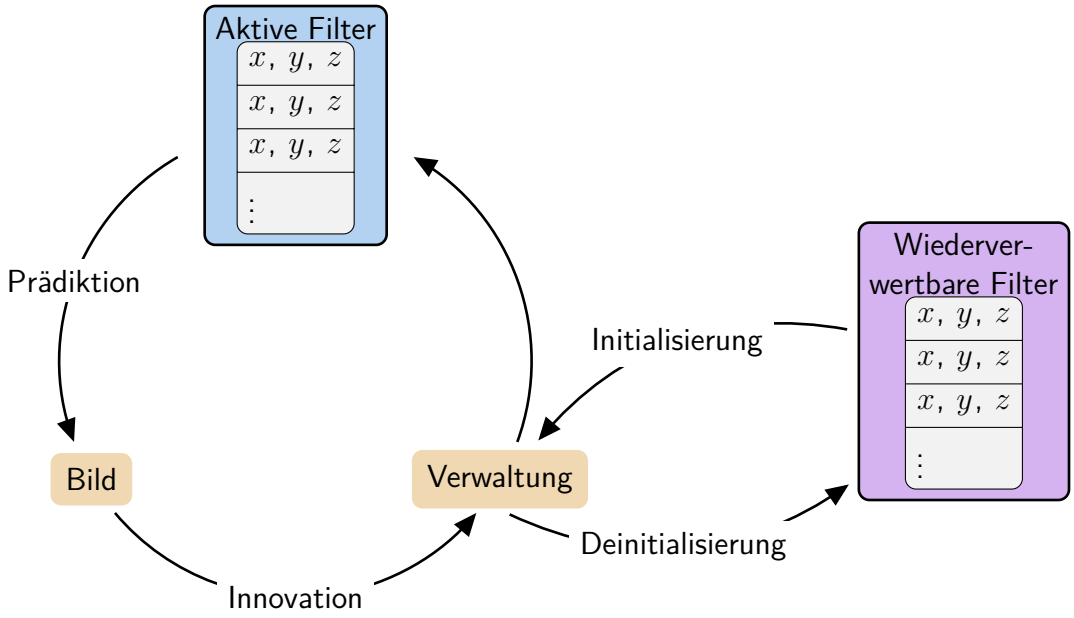


Abbildung 6.15: Darstellung einer Iteration von RAS.

Während der Prädiktion und der Innovation erfolgt die Aktualisierung der Punktpositionen gemäß der Beschreibung aus dem vorhergehenden Abschnitt für jeden gefilterten 3D-Punkt individuell. Die Verwaltungsphase besteht aus der Deinitialisierung nicht mehr benötigter 3D-Filter und der Initialisierung neuer 3D-Filter. Die Deinitialisierung überführt den 3D-Filter in einen Pool nicht benötigter Filter, aus dem sich bei der Initialisierung eines neuen 3D-Filters bedient wird. Durch dieses Vorgehen verringert sich die Fluktuation im Speicherverbrauch zur Laufzeit des

Verfahrens. Ein 3D-Filter wird immer dann deinitialisiert, wenn er innerhalb der letzten 25 Bilder keine gültige Messung erfahren hat. Ein Grund dafür kann sein, dass der 3D-Punkt nicht mehr im Sichtbereich der Kameras liegt, da das Fahrzeug daran vorbeigefahren ist. Es kann auch sein, dass der Punkt durch eine falsche Disparität an einer falschen Position initialisiert wurde und somit keine Deskriptorkorrespondenz gefunden wird. Bei einer Bildrate von 10 Hz entspricht dieser Schwellwert einer Zeit von 2,5 s. Wird ein Filter deinitialisiert, wird er als solcher markiert und bis zu einer potentiellen erneuten Initialisierung nicht mehr als Teil der Umgebungsrepräsentation angesehen.

Für jedes aktive Filter markiert RAS die korrespondierende Bildzelle als belegt. Anschließend iteriert RAS über alle nicht belegten Bildzellen und initialisiert einen Punktfilter an der der Zellenmitte entsprechenden 3D-Position, sofern für diese Stelle die Disparität bekannt ist. Bei einer solchen Initialisierung verwendet RAS, falls möglich, zunächst ein deinitialisiertes Filter aus dem Pool der wiederverwertbaren Filter. Nur wenn dieser Pool leer ist, wird ein neues Filter erzeugt. Durch die Initialisierung geht ein solches Filter in die Liste der aktiven Filter über und wird damit in der nächsten Iteration gleichwertig zu allen anderen aktiven Filtern behandelt.

Je nach Größe der Bildzellen ist während der Innovation eine Überlappung der Suchbereiche für das Deskriptormessmodell mehr oder weniger wahrscheinlich. Dadurch werden Deskriptoren für einige Pixel mehrfach berechnet. Um dies zu umgehen, werden berechnete Deskriptoren bei RAS gecacht. Dabei speichert RAS sämtliche berechnete Deskriptoren während einer Iteration mit ihrer Bildposition ab. Vor dem Berechnen eines Deskriptors findet dann eine Überprüfung statt, ob für die entsprechende Bildposition bereits ein Deskriptor berechnet wurde. Falls kein Deskriptor existiert, berechnet RAS den Deskriptor und trägt ihn in den Cache ein. Dadurch ist sichergestellt, dass jeder benötigte Deskriptor höchstens einmal berechnet wird.

6.2.6 Erweiterung auf dynamische Umgebungen

Nachdem bislang nur statische Umgebungen betrachtet wurden, wird nun eine Erweiterung auf dynamische Objekte vorgestellt. Dazu wird der Zustandsvektor aus Gleichung (6.11) um eine Geschwindigkeit in x- und y-Richtung erweitert. Die Geschwindigkeit in z-Richtung wird bewusst vernachlässigt, da im automotiven Einsatz kaum Vertikalgeschwindigkeiten erwartet werden und auf diese Weise Mehrdeutigkei-

ten in der Interpretation der Messungen verringert werden. Der neue Zustandsvektor lautet demnach

$$\boldsymbol{x} = \left(x, y, z, v_{[x]}, v_{[y]}\right)^T. \tag{6.23}$$

Infolgedessen ist es ebenfalls erforderlich, das Prozessmodell anzupassen. Auch hier steht der Steuervektor $\boldsymbol{u}_{k-1} = {}^{\text{ego},k}\mathbf{P}_{\text{ego},k-1}$, der auf Basis von INS-Messungen die Bewegung des Versuchsträgers zwischen zwei Filteriterationen beschreibt, zur Verfügung. In dynamischen Systemen spielt ebenfalls die Zeit eine wesentliche Rolle. Um den Weg zu berechnen, um den sich ein Punkt zwischen zwei Iterationen des Filters bewegt hat, ist die Zeitdifferenz ${}^{k}T_{k-1}$ zwischen der letzten und der aktuellen Filteriteration notwendig.

Das Systemmodell, das im statischen Fall lediglich die Transformation des 3D-Punktes anhand der zu $\boldsymbol{u}_{k-1}$ korrespondierenden homogenen Transformationsmatrix berechnet hat, muss nun vor der Transformation die Bewegung des Punktes berücksichtigen. Da die Bewegung relativ zum Fahrzeug beschrieben ist, muss diese Relativbewegung ebenfalls zu der neuen Fahrzeugpose transformiert werden. Das Prozessmodell lautet

$$f\left(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, {}^{k}T_{k-1}\right) = \mathbf{A} \cdot \mathbf{B} \cdot \mathbf{C} \cdot \begin{pmatrix} \boldsymbol{x}_{k-1} \\ 1 \end{pmatrix}, \tag{6.24}$$

$$\text{wobei } \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & {}^{k}T_{k-1} & 0 & 0 \\ 0 & 1 & 0 & 0 & {}^{k}T_{k-1} & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{6.25}$$

$$\mathbf{B} = \begin{bmatrix} \mathit{htm}\left(\boldsymbol{u}_{k-1}\right) & \mathbf{0} \\ \mathbf{0} & \mathit{htm}\left(\boldsymbol{u}_{k-1}\right) \end{bmatrix}, \tag{6.26}$$

$$\text{und } \mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{{}^{k}T_{k-1}} & 0 & 0 & 0 & \frac{-1}{{}^{k}T_{k-1}} & 0 & 0 & 0 \\ 0 & \frac{1}{{}^{k}T_{k-1}} & 0 & 0 & 0 & \frac{-1}{{}^{k}T_{k-1}} & 0 & 0 \end{bmatrix}. \tag{6.27}$$

Dabei dient Matrix $\mathbf{C}$ zwei Zielen: Zum einen berechnet sie die neue Position des Punktes abhängig von seiner Eigenbewegung durch Multiplikation seiner Geschwindigkeit mit der Zeitdifferenz und anschließender Addition der Selbigen auf die Zustandsgrößen der Position. Allerdings ändert sich zeitgleich auch die Position des Ego-Fahrzeugs. Gesucht ist daher die Relativbewegung des Punktes bzgl. der neuen Fahrzeugposition. Zu diesem Zweck enthält $\mathbf{C}$ ebenfalls einen Anteil, der als Identität die alte Position des Punktes mit in das Ergebnis der Multiplikation von $\mathbf{C}$ und $\left(\boldsymbol{x}_{k-1}, 1\right)^T$ aufnimmt.

Beide Positionen werden anschließend mit Hilfe von $\mathbf{B}$ individuell in das neue Fahrzeugkoordinatensystem transformiert. Die prädizierte Position des Punktes kann dann direkt aus den ersten drei Koordinaten extrahiert werden. Dies spiegelt sich in den ersten drei Zeilen der Matrix $\mathbf{A}$ wider. Die prädizierte Ego-relative Geschwindigkeit des Punktes bzgl. der neuen Fahrzeugposition ergibt sich durch Subtraktion des transformierten unbewegten Punktes vom transformierten bewegten Punkt und anschließender Division durch ${}^{k}T_{k-1}$. Dies äußert sich in den Zeilen vier und fünf der Matrix $\mathbf{A}$.

Im Gegensatz zum Prozessmodell ist das Messmodell unabhängig von der Bewegung eines Punktes. Wie zuvor besteht die Herausforderung darin, aus dem Zustand die erwartete Disparität und die erwarteten Pixelkoordinaten zu berechnen. Im Fall einer statischen Umgebung erfolgt dies durch die Gleichungen (6.13), (6.14) und (6.16). Alle drei Gleichungen basieren nur auf der prädizierten Position des Punktes, d.h. den ersten drei Komponenten $\boldsymbol{x}^*_{[1:3]\,k}$ des zugehörigen prädizierten Zustands $\boldsymbol{x}^*_k$. Durch Ersetzen von $\boldsymbol{x}^*_k$ und $\boldsymbol{x}^*_{\text{Kamera},k}$ durch $\boldsymbol{x}^*_{[1:3]\,k}$ und $\boldsymbol{x}^*_{[1:3]\,\text{Kamera},k}$ in den genannten drei Gleichungen ist das Messmodell auf dynamische Umgebungen angepasst.

Zur Initialisierung der internen Filterkovarianzmatrix $\mathbf{P}_0$ kommt im Fall einer statischen Umgebung, wie in Gleichung (6.22) beschrieben, die UT zum Einsatz. Die Rückprojektionsfunktion $\mathrm{p}^{-1}_{\text{ego}}(\cdot)$ ist nicht in der Lage, eine Geschwindigkeitskompo-

nente für den berechneten 3D-Punkt zu bestimmen, da die Geschwindigkeit nicht aus der gemessenen Bildposition und Disparität hervorgeht. Unter der Annahme, dass jeder Punkt zunächst statisch ist, wird $\mathsf{p}_{\mathsf{ego}}^{-1}(\cdot)$ derart modifiziert, dass die Geschwindigkeit in x- und y-Richtung auf 0 gesetzt wird. Damit erhält die von der UT geschätzte Kovarianzmatrix $\mathbf{P}_0^*$ keinerlei Kovarianz in den der Geschwindigkeit entsprechenden Zeilen und Spalten. Die Position des Punktes und seine Geschwindigkeit werden als unkorreliert betrachtet, so dass lediglich ein Rauschen für die 2×2-Submatrix $\mathbf{P}^*_{[4:5,4:5]\,0}$ zu definieren ist. Dieses Rauschen wird situationsabhängig festgelegt. In Parkplatzszenarien, d.h. bei langsamen Geschwindigkeiten entlang beider Richtungen, werden kleinere Standardabweichungen zu Grunde gelegt, wohingegen in Autobahnszenarien hauptsächlich Längsgeschwindigkeiten erwartet werden und somit das Rauschen in y-Richtung deutlich kleiner bemessen werden kann als das in x-Richtung. Auch wenn die Initialisierung der Geschwindigkeit kein triviales Problem darstellt, ist das Filter trotzdem in der Lage, die korrekte Geschwindigkeit des Punktes über der Zeit zu schätzen.

Neben der Ungewissheit über die Geschwindigkeit müssen in dynamischen Szenarien auch die anderen bisher getroffenen Annahmen überprüft werden. Beispielsweise kann das Suchfenster für das Deskriptor-Messmodell vergrößert werden, um unvorhergesehenen Querbewegungen zu begegnen. Aufgrund des Cachings vergrößert sich dabei der zusätzliche Rechenbedarf nur unwesentlich. Aber auch die Filterung der Disparitätsmessungen anhand der Innovationskovarianz muss möglicherweise toleranter gestaltet werden, um schnell nahekommende Punkte nicht als Fehlmessungen zu klassifizieren. Gerade in Hinblick auf entgegenkommende Verkehrsteilnehmer ist eine gute und schnelle Schätzung der Geschwindigkeit erforderlich. Da die Disparitätsmessungen für weiter entfernte Objekte zu einer hohen Entfernungsunsicherheit führen, ist die Kombination mit einem weiteren Sensor zu empfehlen.

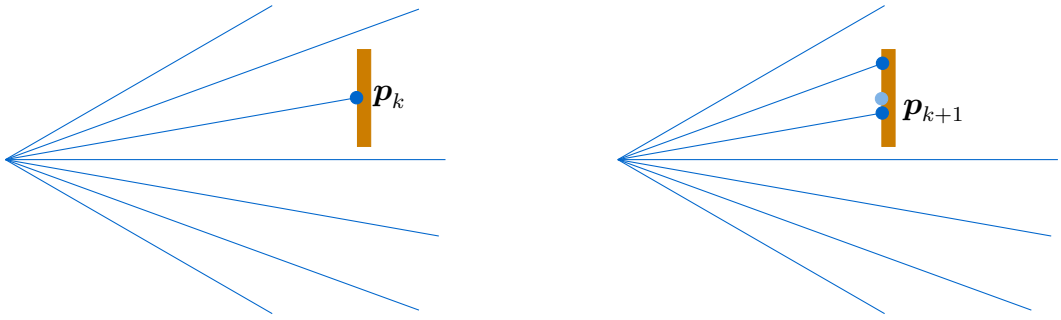
Nicht zuletzt aus diesem Grund werden im Automobilbereich Entfernungsmessungen zu anderen Fahrzeugen hauptsächlich mit RaDAR-Sensoren vorgenommen. Über den Dopplereffekt erlauben RaDAR-Sensoren zudem direkt die Messung der Geschwindigkeit. Steht kein RaDAR zur Verfügung, muss die Geschwindigkeit aus möglichst genauen Messungen der Entfernung nach obigen Gleichungen inferiert werden. Dazu eignen sich bspw. LiDAR-Sensoren, oder zu gewissem Grad auch Stereokameras mit möglichst großem Basisabstand. In Kombination mit Algorithmen zur Objekt- und Situationserkennung können Annahmen über die Geschwindigkeiten bei der Initialisierung auch kontextbezogen getroffen werden. So können bspw. 3D-Punkte auf als vorausfahrend erkannten Fahrzeugen mit einer Geschwindigkeit ähnlich der des Ego-Fahrzeugs initialisiert werden, während Gebäude oder die Fahrbahn als

statisch betrachtet werden. Derartige Erweiterungen werden in dieser Arbeit jedoch nicht genauer untersucht und bieten potential für zukünftige Arbeiten.

6.2.7 Einbeziehung der Messungen eines LiDAR

Besonders genaue Entfernungsmessungen liefern die auf beiden Versuchsträgern verbauten Velodyne HDL-64E. Bislang wird dieser Sensor als Referenzsensor verwendet. Durch die in dieser Arbeit hergeleitete extrinsische Kalibrierung des Sensors ist eine Einbeziehung der im LiDAR gemessenen Entfernungen zur genaueren Bestimmung des Abstands möglich.

Ein großer Nachteil des Messprinzips des Velodyne HDL-64E ist dabei allerdings, dass in dynamischen Umgebungen kaum derselbe 3D-Punkt zweimal vom LiDAR vermessen wird. Um diesen Effekt zu beobachten, reicht bereits die Eigenbewegung des Fahrzeugs aus, was anhand von Abb. 6.16 verdeutlicht wird. Der LiDAR misst in äquidistanten Zeitschritten die Entfernung eines ausgesandten Laserstrahls. Bei konstanter Rotationsbewegung erfolgt die Messung in konstanten Winkelinkrementen. Abbildung 6.16a zeigt 7 Laserstrahlen mit äquidistantem Winkelabstand, von denen einer das Hindernis mittig im Punkt $\boldsymbol{p}_k$ trifft. In Abbildung 6.16b befindet sich das Objekt 1 m näher am Sensor. In diesem Fall trifft der LiDAR das Objekt im Punkt $\boldsymbol{p}_{k+1}$. Zum Vergleich ist die Trefferstelle aus dem ersten Fall als • dargestellt.



(a)
Messung eines Objekts in ca. 4 m Abstand.

(b)
Messung eines Objekts in ca. 3 m Abstand.

Abbildung 6.16:
Messung eines Objekts in verschiedenen Abständen mit einem Velodyne HDL-64E.

Das Problem der Zuordnung von Messungen ist als das Korrespondenzproblem bekannt. In reinen Bildverarbeitungsverfahren wird es häufig durch die schon ge-

nannten und auch bei RAS eingesetzten Schlüsselpunkt- und Deskriptor-Verfahren gelöst. In der Kombination aus Kamera und LiDAR ist es bereits aus Abschnitt 5.6 bekannt, wo die Korrespondenz mit Hilfe eines bekannten Kalibrierkörpers und dessen Abmessungen hergestellt wurde. Ein solcher Kalibrierkörper ist hier jedoch nicht vorhanden. Die Korrespondenzfindung zwischen Punktwolken, die in der Literatur auch als Punktwolkenregistrierung bezeichnet wird, ist dennoch möglich, wenn auf Basis der Punktwolken zunächst eine Merkmalsextraktion durchgeführt wird und die Korrespondenzfindung dann auf Basis der extrahierten Merkmale durchgeführt wird. Typische Merkmale sind dabei bspw. Ebenen, die mit Hilfe von Random Sample Consensus (RANSAC) in den Punktwolken bzw. Teilen der Punktwolke extrahiert werden.

Auch für die Erweiterung von RAS wären derartige geometrische Modelle eine Möglichkeit. Es könnte durch die Annahme lokaler Planizität die Entfernung eines 3D-Punktes durch Interpolation benachbarter 3D-Messungen des LiDAR bestimmt werden. An Tiefenkanten ist diese Approximation allerdings ungenau und der entstehende Fehler nur schwer quantifizierbar. Typischerweise ist die horizontale und vertikale räumliche Auflösung eines Kamerabilds deutlich größer als die eines Velodyne HDL-64E. Anhand von Abb. 6.2 ist erkennbar, dass die Punktdichte der in ein Kamerabild projizierten 3D-Punkte eines Velodyne HDL-64E vergleichsweise hoch ist. Damit steigt die Wahrscheinlichkeit, dass ein beliebiger von RAS gefilterter Punkt zeitweise mit einem Pixel korrespondiert, für das auch Tiefeninformationen aus einer direkten LiDAR-Messung bereitstehen.

Messmodell 3 Daher wird RAS um ein drittes Messmodell erweitert, das LiDAR-Messmodell. Die Messung entspricht der Entfernung des Punktes zum Sensorkoordinatensystem $\mathbf{S}_{\text{LiDAR}}$. RAS filtert die Punkte allerdings bzgl. des Fahrzeugkoordinatensystems $\mathbf{S}_{\text{ego}}$. Die prädizierte Messung ergibt sich daher durch Transformation der Positionskoordinaten $\boldsymbol{x}^*_{[1:3]}$ des prädizierten Zustands in $\mathbf{S}_{\text{LiDAR}}$ und anschließender Berechnung der Vektornorm

$$\boldsymbol{y}^*_{\text{LiDAR, }\mathbf{S},k} = \left\| {}^{\text{LiDAR},k}\mathbf{H}_{\text{ego},k} \cdot \begin{pmatrix} \boldsymbol{x}^*_{[1:3]\ k} \\ 1 \end{pmatrix} \right\|. \tag{6.28}$$

Das Messrauschen dieses Messmodells ist in [m] anzugeben. Gerade für weiter entfernt liegende Punkte wird die Position und damit letztlich auch die Geschwindigkeitsschätzung genauer als mit rein bildbasierten Methoden.

Der Zugriff auf genaue Entfernungsmessungen ermöglicht darüber hinaus auch eine deutlich genauere Initialisierung der 3D-Position eines Punktes. Dazu wird die Initialisierung neuer Filter wie folgt angepasst. Gemäß Abschnitt 6.2.5 erfolgt die Initialisierung neuer Filter bislang nur in der Mitte von Bildzellen, innerhalb derer im aktuellen Bild kein Filter existiert. Für das LiDAR-Messmodell werden alle 3D-Punkte im Sichtbereich der Kamera auf ihre korrespondierenden Bildpixel projiziert. Während dieses Prozesses findet zusätzlich eine Speicherung statt, in welcher Zelle und an welche Pixelposition einer der LiDAR-Punkte fällt. Wenn nun für eine Zelle ein neues Filter initialisiert wird, findet die Initialisierung statt in der Zellmitte an der der Zellmitte am nächsten liegenden Pixelposition, für die durch den LiDAR die Entfernung bekannt ist, statt. Existiert eine solche Position innerhalb der Zelle nicht, verläuft die Initialisierung wie zuvor pixel- bzw. disparitätsbasiert in der Zellmitte.

Bei der Initialisierung eines Punktfilters durch LiDAR-Messungen gelten andere Genauigkeiten als bei der Initialisierung durch Disparitäten. Die Entfernungsungenauigkeit des Velodyne HDL-64E ist aus dem Datenblatt in Anhang B.2 entnehmbar. Ein Velodyne HDL-64E misst 3D-Punkte in Polarkoordinaten, d.h. horizontalem und vertikalem Raumwinkel und Entfernung. Zusammen beschreiben die drei Unsicherheiten in 3D eine Art Schirm, der aufgrund der Schätzung in kartesische Koordinaten bzgl. $\mathbf{S}_{\text{ego}}$, transformiert werden muss.

Die Messung eines 3D-Punktes in Polarkoordinaten wird beschrieben als Messtupel $\boldsymbol{y}_{\text{LiDAR}} = (\Theta, \Psi, r)^T$. Analog zu Gleichung (6.21) ergibt sich das Messrauschen eines Punktes in Polarkoordinaten bzgl. $\mathbf{T}_{\text{LiDAR}}$ als Diagonalmatrix

$$\mathbf{R}^{\boldsymbol{y},\text{LiDAR}} = \begin{bmatrix} {\sigma^2}_{[\Theta]} & 0 & 0 \\ 0 & {\sigma^2}_{[\Psi]} & 0 \\ 0 & 0 & {\sigma^2}_{[r]} \end{bmatrix}. \tag{6.29}$$

Auch diesmal werden die Unsicherheiten mit Hilfe der UT transformiert. Dazu ist eine Funktion zur Umrechnung von Polarkoordinaten in kartesische Koordinaten mit anschließender Koordinatentransformation von $\mathbf{S}_{\text{LiDAR}}$ nach $\mathbf{S}_{\text{ego}}$ notwendig. Basierend auf Gleichung (5.54) wird daher ${}^{\text{ego}}f_{\text{LiDAR}}\left(\boldsymbol{p}_{\text{LiDAR}}\right)$ definiert als

$${}^{\text{ego}}f_{\text{LiDAR}}\left(\boldsymbol{p}_{\text{LiDAR}}\right) = {}^{\text{ego}}\mathbf{H}_{\text{LiDAR}} \cdot \begin{pmatrix} \boldsymbol{p}_{[r]} \cdot \cos\left(\boldsymbol{p}_{[\Theta]}\right) \cdot \cos\left(\boldsymbol{p}_{[\Psi]}\right) \\ \boldsymbol{p}_{[r]} \cdot \cos\left(\boldsymbol{p}_{[\Theta]}\right) \cdot \sin\left(\boldsymbol{p}_{[\Psi]}\right) \\ \boldsymbol{p}_{[r]} \cdot \sin\left(\boldsymbol{p}_{[\Theta]}\right) \\ 1 \end{pmatrix}. \tag{6.30}$$

Damit lässt sich die initiale Filterkovarianzmatrix analog zu Gleichung (6.22) und den Modifikationen aus Abschnitt 6.2.6 wie folgt abschätzen:

$$\left\{\hat{\boldsymbol{x}}_0^{\text{LiDAR}}, \hat{\mathbf{P}}_0^{\text{LiDAR}}, \cdot, \cdot\right\} = ut\left(\boldsymbol{y}_{\text{LiDAR}}, \mathbf{R}^{\boldsymbol{y},\text{LiDAR}}, {}^{\text{ego}}f_{\text{LiDAR}}\left(\cdot\right), \mathbf{0}\right) \tag{6.31}$$

Filter, die auf diese Weise mit LiDAR-Messungen initialisiert werden, verfügen über wesentlich genauere Schätzungen der Entfernung, wodurch über die Filteriterationen eine etwaige Geschwindigkeit schneller geschätzt wird, als dies mit einer rein Disparität-basierten Initialisierung möglich ist.

7 Autonome Navigation auf Basis fusionierter Sensorik

Die Kalibrierung von Inertial Navigation System (INS), Light Detection and Ranging (LiDAR) und Stereokamerasystem eröffnet neue Möglichkeiten für die autonome Navigation. Durch Fusion dieser Sensoren lassen sich hochgenaue Eigenbewegungs- mit genauer Entfernungsmessung und hoher lateraler Auflösung kombinieren. Die zeitliche Synchronität der Sensoren erlaubt zudem die zeitliche Aggregation und Filterung der Messungen aller Sensoren. Dieses Kapitel gibt einen Überblick darüber, wie die in dieser Arbeit entwickelten Methoden zur autonomen Navigation verwendet werden. Es stellt dabei nur einen ersten Schritt dar und zeigt Entwicklungspotenziale für zukünftige Weiterentwicklungen auf.

7.1 Belegungskarten und Umgebungskarten

Damit ein autonomes Fahrzeug oder Roboter gefahrlos navigieren kann, benötigt es eine Karte der Umgebung. Eine solche Karte kann global, aber auch lokal vorliegen und enthält Informationen über Hindernisse in der lokalen Umgebung. Neben festen Hindernissen wie Gebäuden, Leitplanken oder Verkehrssignalen muss auch mit bewegten, dynamischen Hindernissen wie anderen Verkehrsteilnehmern gerechnet werden. Da hinterlegte Karten zum einen keine dynamischen Hindernisse enthalten und zum anderen die Aktualität der Karten nicht gewährleistet ist, reichen solche Karten zur Navigation alleine nicht aus. Das Fahrzeug muss mit Hilfe seiner Sensoren die statischen und dynamischen Hindernisse erfassen und in einer eigenen Karte integrieren, damit die Navigationsplanung eine kollisionsfreie Bewegung berechnen kann.

Der Raum um ein Fahrzeug erstreckt sich kontinuierlich entlang aller drei Raumdimensionen. Ein kontinuierliches Modell hätte damit unendliche Dimensionalität und wäre nicht mehr berechenbar. Aus diesem Grund haben sich in den letzten Jahren Belegungskarten, *engl. occupancy grid maps* [Thrun et al., 2005, Himmelsbach, 2015, Tanzmeister, 2016, Steyer, 2021], zu einem Standardverfahren zur Approxima-

tion der Roboterumgebung durchgesetzt. In einer Belegungskarte wird der Raum zunächst beschränkt, d.h. die maximale Ausdehnung entlang der x- und y-Achsen wird festgelegt. Der verbleibende Raum wird in Zellen mit definierter Kantenlänge unterteilt. Sind alle Zellen gleich groß, ergibt sich ein regelmäßiges 2D-Gitter. Diese Art der Diskretisierung der Umgebung vereinfacht im Gegensatz zum kontinuierlichen Raum die Bahnplanung eines Fahrzeugs, wie in Fassbender et al. [2014] gezeigt wird.

7.1.1 Höheninformationen in Belegungskarten

Die Bahnplanung sucht auf einer Belegungskarte den Weg für das autonome Fahrzeug auf Basis verschiedener Umgebungseigenschaften, die durch die Sensoren erfasst und in die Belegungskarte eingetragen werden. Eine dieser Eigenschaften ist die Höhe bezogen auf die aktuelle Aufstandsfläche des Fahrzeugs.

Die beiden Versuchsträger verfügen über LiDAR-Sensoren. Diese Sensoren vermessen die Fahrzeugumgebung und liefern eine 3D-Punktwolke im Fahrzeugkoordinatensystem $\mathbf{S}_{\text{ego}}$. Die Fahrzeuge verfügen außerdem über ein Stereokamerasystem, durch das wie in Abschnitt 6.2 beschrieben ebenfalls 3D-Punktwolken bzgl. $\mathbf{S}_{\text{Kamera}}$ und $\mathbf{S}_{\text{ego}}$ generiert werden können. Nach Transformation dieser Punktwolken in das von $\mathbf{S}_{\text{ego}}$ aus senkrecht nach unten in die Aufstandsfläche verschobene $\mathbf{S}_{\text{ego,p}}$ entspricht die gesuchte Höhe eines Punktes über Grund genau seiner z-Koordinate. Durch Überlagerung der Belegungskarte mit der transformierten Punktwolke lässt sich jeder Punkt exakt einer Zelle zuordnen.

Dabei ist es durchaus möglich, dass mehrere Punkte in die gleiche Zelle fallen. Um nun die Höhe einer Zelle zu bestimmen, bietet sich das Maximum aller z-Komponenten aller Punkte der Zelle an. Auch wenn diese Annahme in den meisten Fällen zutrifft, führt sie bei überhängenden Strukturen wie Ästen oder auch Brücken zu Problemen. Es ist daher notwendig, derartige Strukturen als solche zu erkennen und bei der Höhenberechnung einer Zelle zu ignorieren. An dieser Stelle kommt das Verfahren von Himmelsbach et al. [2010] zur Segmentierung der überhängenden Objekte zum Einsatz. Nach diesem Verfahren werden überhängende Objekte segmentiert und die zugehörigen 3D-Punkte vor der Eintragung in die Karte herausgefiltert.

Je nach Sensor und Auflösung gibt es wie in Abb. 7.1 dargestellt einige Zellen, die bei Auswertung von nur einer Sensoraufnahme ganz ohne Höheninformation bleiben.

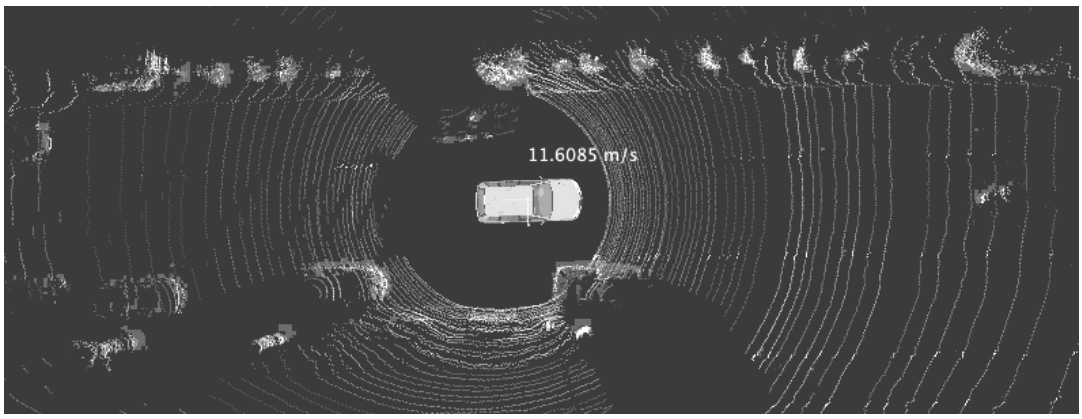


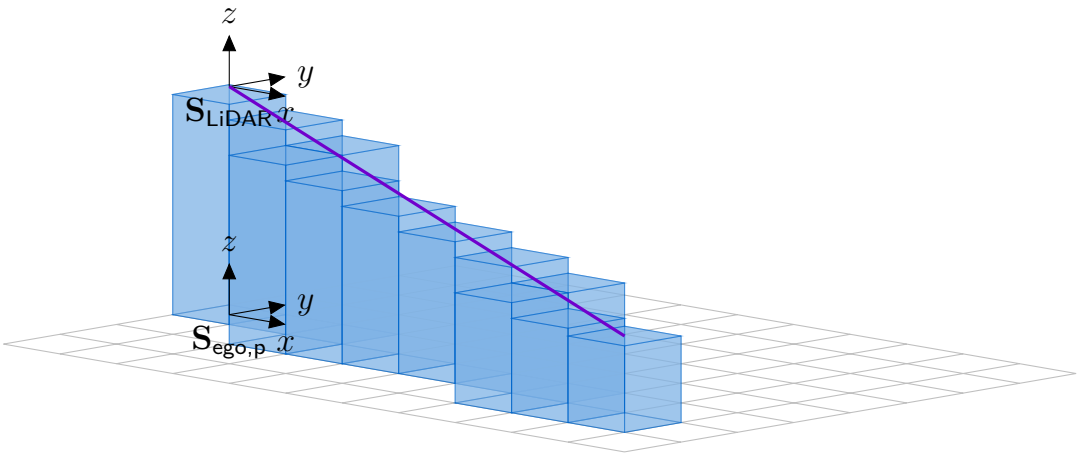
Abbildung 7.1:
Belegungskarte nach Eintragen einer der Messungen einer einzelnen LiDAR-Umdrehung. Zwischen den Scanlinien der Laser existiert keine Höheninformation.

Für eine verlässliche Bahnplanung ist diese Information nicht ausreichend, da in jeder der nicht vermessenen Zellen ein unüberfahrbares Hindernis liegen könnte. Zum Füllen dieser Informationslücken existieren mehrere Möglichkeiten: Geometrie, zeitliche Akkumulation oder Interpolation.

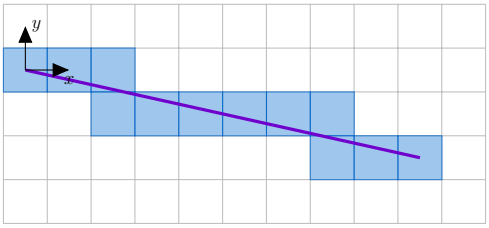
Geometrische Lösung Die geometrische Lösung basiert auf der Beobachtung, dass eine Zelle nur dann von einem Laser- oder Sichtstrahl getroffen werden kann, wenn der Messstrahl nicht zuvor ein Hindernis in einer anderen Zelle getroffen hat. Wird also eine Zelle in einer bestimmten Höhe getroffen, ist klar, dass die Höhe vorhergehender Zellen unterhalb der Höhe des Strahls liegen. Dieser Zusammenhang ist in Abb. 7.2 bildhaft dargestellt.

Zu Beachten ist dabei, dass jede Zelle die hier dargestellte maximale Höhe haben kann, allerdings nicht haben muss. Genauso muss damit gerechnet werden, dass unterhalb des Strahls auch plötzliche Höhensprünge bis zur jeweiligen Strahlhöhe auftreten können. Verwertbar ist dieses Kriterium daher nur, wenn die derart ermittelte Zellhöhe noch überfahrbar bleibt. Aufgrund des Messwinkels, hevorgerufen durch die Einbaulage der Velodyne HDL-64E auf den Versuchsträgern, kommt dieses Kriterium daher hauptsächlich für weiter entfernt liegende Zellen zum Tragen.

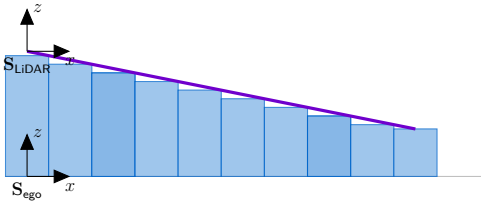
Zeitliche Akkumulation Die zeitliche Akkumulation von Messungen bietet sich als Alternative zur Geometrie an. Sie basiert auf dem Effekt, dass durch eine Bewegung des Versuchsträgers die Umgebung schrittweise abgetastet wird. Die



(a) Perspektivische Darstellung



(b) Vogelperspektive



(c) Seitenansicht

Abbildung 7.2:
Geometrisch hergeleitete maximale Zellhöhen. Die Höhen der vom Messstrahl durchquerten Zellen liegen immer unterhalb des Messstrahls.

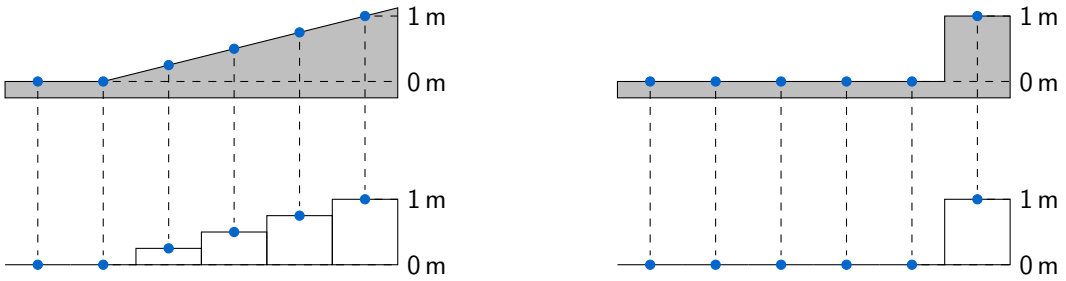
ringförmigen Scanlinien eines LiDAR beispielsweise werden dabei über noch nicht vermessene Zellen bewegt und generieren daher Messungen in zuvor nicht vermessenen Zellen. Wesentlich hierbei ist, dass die einer Zelle zugeordnete Höhe auch nach Abschluss der Sensoraufnahme erhalten bleibt. Dadurch trifft eine Belegungskarte implizit die Annahme, dass die modellierte Umgebung statisch ist. Dementsprechend führen bewegte Objekte im Erfassungsbereich der Sensoren, die zur Erstellung der Belegungskarte beitragen, zu Artefakten. Messungen, die zu bewegten Objekten gehören, müssen daher vor der Eintragung in eine solche Höhenkarte herausgefiltert und bei der Navigation, wie später gezeigt, gesondert betrachtet werden. Dies kann beispielsweise durch einen Schwellwert auf die Geschwindigkeitskomponente von Rekursives Automotives Stereo (RAS) erfolgen. Zur Behandlung dynamischer Objekte besteht auch die Möglichkeit, die Objekte über die Zeit zu verfolgen, wie von Himmelsbach und Wuensche [2012], Manz [2013], Fries und Wuensche [2014], Fries [2019] beschreiben, und die korrespondierenden 3D-Punkte bei der Eintragung in die Karte herauszufiltern. Eine dritte Möglichkeit zum Umgang mit dynamischen Umgebungen besteht in einer Gewichtung der Messungen gemäß ihres Alters. Ältere Messungen werden entsprechend weniger stark gewichtet als neuere. Dadurch ver-

blassen die Artefakte dynamischer Objekte schneller. Bei der zeitlichen Akkumulation durch ein bewegtes Fahrzeug muss darüber hinaus die Fahrzeugbewegung bekannt sein, was durch das verbaute INS gewährleistet ist (vgl. Abschnitt 5.9.4).

Interpolation Neben der Geometrie und der Akkumulation von Messungen besteht auch durch die Interpolation von Zellhöhen die Möglichkeit, Lücken in der Karte zu füllen. Manz [2013] verwendet zu diesem Zweck *Image Inpainting* [Bertalmio et al., 2001], ein aus der Bildverarbeitung stammendes Verfahren zur Rekonstruktion fehlender Bildteile. Da die Bahnplanung allerdings mit kleineren Lücken umgehen kann, wird die Interpolation in dieser Arbeit nicht verwendet.

7.1.2 Höhe und Hinderniswahrscheinlichkeit

Auch wenn die Höhe einer Zelle eine wertvolle Umgebungsinformation darstellt, so ist für ein autonomes Fahrzeug die Frage nach der Befahrbarkeit einer Zelle eine weitaus wichtigere Eigenschaft. Der Grund dafür ist in der Abb. 7.3 dargestellt. Die Abbildung zeigt einen Querschnitt durch zwei Geländeprofile und darunter die Diskretisierung in Zellen. Im Fall (a) zeigt das Bild eine Rampe von ca. 1 m Höhe, die durchweg befahrbar ist. In Fall (b) ist ein Hindernis mit einer Höhe von 1 m dargestellt, das nicht befahrbar ist. In beiden Fällen ist die Höhe der rechten Zellen identisch. Die Höhe allein ist daher als Maß für die Befahrbarkeit nicht ausreichend.



(a) Befahrbare Rampe **(b)** Hindernis

Abbildung 7.3:
Diskretisierung des Höhenprofils einer Rampe im Vergleich zu einem Hindernis. Obwohl die rechte Zelle in beiden Fällen gleich hoch ist, ist sie nur im Falle der Rampe befahrbar.

Die Befahrbarkeit richtet sich daher nicht nach der absoluten Höhe, sondern nach der Höhendifferenz zu den benachbarten Zellen oder auch innerhalb derselben Zelle,

sofern mehrere Messungen in dieselbe Zelle fallen. Wie groß diese Höhendifferenz sein darf, richtet sich u.a. nach der Fahrzeuggeometrie wie Raddurchmesser, Überhang des Fahrzeugs über die Achsen, Achsabstand und der Bodenfreiheit und ist damit eine fahrzeugspezifische Größe.

Da die Messwerte der Sensoren fehlerbehaftet sind, bietet sich statt der Speicherung von Höhe oder maximaler Höhendifferenz innerhalb einer Zelle eine probabilistische Repräsentation der Hinderniswahrscheinlichkeit an. Die Berechnung der Hinderniswahrscheinlichkeit basiert für jede Zelle $\mathbf{O}_{[x,y]}$ einer Belegungskarte $\mathbf{O}$ auf allen vorherigen Höhenmessungen der Zelle und ihrer Nachbarzellen. Zu einem Zeitpunkt ist es möglich, dass mehrere Höhenmessungen in dieselbe Zelle fallen. Für $\boldsymbol{y}_{[x,y]\ 1:k}$ als Menge aller Höhenmessungen einer Zelle seit Beginn der Messung bis zum aktuellen Zeitpunkt k beschreibt $y^{\mathsf{max}}_{[x,y]\ 1:k}$ die Menge der maximalen Höhenmessungen über der Zeit. Analog beschreibt $y^{\mathsf{min}}_{[x,y]\ 1:k}$ die Menge der minimalen Höhenmessungen über der Zeit unter Einbeziehung der Messungen von bis zu acht direkten Nachbarzellen. Durch die Berücksichtigung der Nachbarzellen können die in Abb. 7.3 gezeigten Fälle unterschieden werden, da die $y^{\mathsf{min}}_{[x,y]\ i}$ der Nachbarzellen in folgende Berechnungen miteinfließen.

Durch Vergleich der Differenzen von $y^{\mathsf{max}}_{[x,y]\ i}$ und $y^{\mathsf{min}}_{[x,y]\ i}$ mit der maximal erlaubten befahrbaren Höhendifferenz $h^{\mathsf{befahrbar}}$ ergibt sich die Anzahl der als befahrbar und unbefahrbar klassifizierten Messzeitpunkte wie folgt:

$$N^{\mathsf{befahrbar}}_{[x,y]\ 1:k} = \sum_{i=1}^{k} \begin{cases} 1, & \text{falls } \exists\, \boldsymbol{y}^{\mathsf{max}}_{[x,y]\ i} \wedge \exists\, \boldsymbol{y}^{\mathsf{min}}_{[x,y]\ i} \wedge \left(\boldsymbol{y}^{\mathsf{max}}_{[x,y]\ i} - \boldsymbol{y}^{\mathsf{min}}_{[x,y]\ i}\right) \leq h^{\mathsf{befahrbar}} \\ 0, & \text{sonst,} \end{cases} \tag{7.1}$$

$$N^{\mathsf{unbefahrbar}}_{[x,y]\ 1:k} = \sum_{i=1}^{k} \begin{cases} 1, & \text{falls } \exists\, \boldsymbol{y}^{\mathsf{max}}_{[x,y]\ i} \wedge \exists\, \boldsymbol{y}^{\mathsf{min}}_{[x,y]\ i} \wedge \left(\boldsymbol{y}^{\mathsf{max}}_{[x,y]\ i} - \boldsymbol{y}^{\mathsf{min}}_{[x,y]\ i}\right) > h^{\mathsf{befahrbar}} \\ 0, & \text{sonst.} \end{cases} \tag{7.2}$$

Jede Zelle hat somit eine Art Gedächtnis, das zählt, wie oft eine Zelle als befahrbar oder unbefahrbar klassifiziert wurde. Die Hinderniswahrscheinlichkeit ergibt sich für jede Zelle damit aus der Anzahl der befahrbaren Klassifizierungen durch die Summe aus befahrbaren und unbefahrbaren Klassifizierungen

$$p\left(\mathbf{O}_{[x,y]\ k} \middle|\, \boldsymbol{y}_{[x,y]\ 1:k}\right) = \begin{cases} \frac{N^{\mathsf{unbefahrbar}}_{[x,y]\ 1:k}}{N^{\mathsf{unbefahrbar}}_{[x,y]\ 1:k} + N^{\mathsf{befahrbar}}_{[x,y]\ 1:k}}, & \text{falls } N^{\mathsf{unbefahrbar}}_{[x,y]\ 1:k} + N^{\mathsf{befahrbar}}_{[x,y]\ 1:k} > 0 \\ \frac{1}{2}, & \text{sonst.} \end{cases} \tag{7.3}$$

Eine so berechnete Belegungskarte ist in Abb. 7.4 dargestellt. Die Wahrscheinlichkeiten sind dabei durch Grauwerte dargestellt. Bereiche mit hoher Hinderniswahrscheinlichkeit erscheinen weiß, Bereiche mit geringer Hinderniswahrscheinlichkeit dunkelgrau. Für die Mehrzahl der hier abgebildeten Zellen existieren keine Messungen. Sie werden daher grau dargestellt, was einer Hinderniswahrscheinlichkeit von 50 % entspricht.

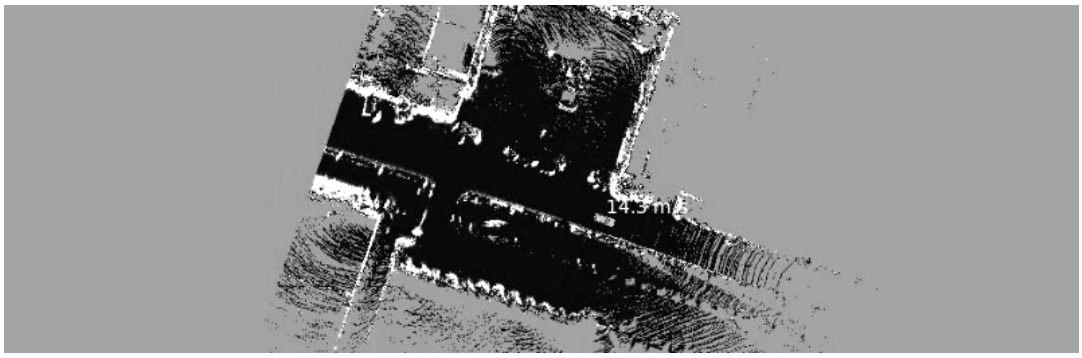


Abbildung 7.4:
Beispiel einer Belegungskarte auf Basis von LiDAR-Punktwolken. Hohe Hinderniswahrscheinlichkeiten sind weiß dargestellt, niedrige hingegen dunkelgrau.

Die hier dargestellte Belegungskarte stammt aus akkumulierten LiDAR-Messungen eines bewegten Fahrzeugs. Durch die Bewegung in Verbindung mit der Akkumulation der Messungen werden zum einen deutlich mehr Zellen vermessen als ohne Bewegung. Zum anderen erhöht sich die Dichte der Information. Ohne Bewegung würde die Belegungskarte entsprechend dem Sensor lediglich aus Hinderniswahrscheinlichkeiten in Form konzentrischer Kreise um das Fahrzeug bestehen.

Neben Höhe, Hinderniswahrscheinlichkeit und Bewegung gibt es noch eine Vielzahl von weiteren Zellattributen, die in eine Karte eingetragen werden können. Manz [2013] überlagert die Höheninformationen mit den Farben eines Kamerabilds. Andere berechnen darüber hinaus auch die Steigung innerhalb einer Zelle [Himmelsbach, 2015]. Da solche Karten weitaus mehr Informationen als nur die Belegung alleine bereitstellen, werden sie häufig auch Umgebungskarten genannt.

7.2 Hindernisvermeidung mit Belegungskarten

Mit Hilfe von Belegungskarten und dem Tentakel-Ansatz [von Hundelshausen et al., 2008, Himmelsbach et al., 2011a] kann eine kollisionsfreie Navigation eines autonom

fahrenden Fahrzeugs in unbekanntem Gelände realisiert werden. Die dazu notwendige Kalibrierung der Sensoren liefert diese Arbeit. Darüber hinaus kann auch das in Abschnitt 6.2.6 beschriebene Verfahren RAS als Datenquelle für eine Belegungskarte verwendet werden. Dieses Kapitel zeigt die Hindernisvermeidung in einem Versuchsträger. Die Demonstration basiert dabei einmal auf LiDAR-Punktwolken, einmal auf Punktwolken berechnet aus Disparitätenbildern und zuletzt auf RAS.

7.2.1 Szenario

Ausgangspunkt des Demonstrationsszenarios ist ein autonomes Fahrzeug mit der Aufgabe, von einem beliebigen Punkt $\boldsymbol{A}$ zum einem Punkt $\boldsymbol{B}$ zu fahren. Die Position des Fahrzeugs sowie der Start- und Endpunkt sind dem Fahrzeug in Form von GPS-Koordinaten bekannt. Außerdem verfügt das Fahrzeug über Daten eines groben Straßennetzes in Form von GPS-Wegpunkten einer bekannten Karte. Das Fahrzeug soll, dem Straßennetz folgend, vom Startpunkt $\boldsymbol{A}$ zum Zielpunkt $\boldsymbol{B}$ fahren, ohne dabei mit etwaigen Hindernissen zu kollidieren. Als Hindernisse dienen ein Straßenschild und ein geparktes Fahrzeug. Das Szenario wird auf einer relativ ebenen, straßenartigen Fläche durchgeführt.

7.2.2 Globale Planung

Im Laufe der globalen Planung werden zunächst die eigene Fahrzeugposition sowie die beiden Punkte $\boldsymbol{A}$ und $\boldsymbol{B}$ gemäß dem Verfahren von Luettel et al. [2011] auf das Straßennetz abgebildet. Die Position des Fahrzeugs stimmt zu Beginn mit $\boldsymbol{A}$ überein. Der A*-Algorithmus [Hart et al., 1968] sucht nun den kürzesten Weg zwischen $\boldsymbol{A}$ und $\boldsymbol{B}$ auf dem Straßennetz. Dieser Weg entspricht der Route, die das Fahrzeug zum Zielpunkt $\boldsymbol{B}$ abfahren soll. Die beiden Hindernisse sind derart aufgestellt, dass das Fahrzeug bei mittiger Spurfolge der geplanten GPS-Spur mit beiden Hindernissen kollidieren würde.

7.2.3 Lokale Planung

Die lokale Planung basiert auf einer Belegungskarte, für deren Zellen kontinuierlich die Hinderniswahrscheinlichkeiten berechnet werden und dem Tentakel-Verfahren [von Hundelshausen et al., 2008]. Die Belegungskarten werden auf drei verschiedene

Weisen generiert. Im ersten Fall stammen die Daten direkt aus der inertial korrigierten 3D-Punktwolke eines LiDAR. Im zweiten Fall werden ausschließlich kamerabasierte 3D-Punkte verwendet. Die Berechnung dieser Punkte basiert auf der Anwendung des BM-Verfahrens (Blockmatching) auf die kalibrierten Bilder der beiden äußeren Kameras der im Fahrzeug verbauten Bumblebee XB3. Im dritten Fall stammt die Punktwolke zur Eintragung in die Belegungskarte aus dem in Abschnitt 6.2.6 vorgestellten Verfahren RAS.

7.2.4 Beobachtungen

Auch wenn die Fahrt mit den drei dargestellten Eingangsdaten für die Berechnung der Belegungskarte kollisionsfrei möglich ist, müssen einzelne Parameter angepasst werden. Als Basis des Vergleichs dient die Verwendung der LiDAR-Daten, da aufgrund der Vorarbeiten [von Hundelshausen et al., 2008, Himmelsbach et al., 2011a] hier die meisten Erfahrungen bestehen.

Im Vergleich zu den LiDAR-Punkten liefert das Stereo-Verfahren BM deutlich mehr 3D-Punkte. Während eine LiDAR-Messung pro Umdrehung bis zu 133 312 Punkte erfasst, ergibt die Umwandlung eines dichten Disparitätenbilds der Bumblebee XB3 bei voller Auflösung bis zu 1 228 800 Punkte, die zusätzlich noch über einen deutlich kleineren lateralen Öffnungswinkel verteilt sind. Den Vergleich zeigen Abb. 7.5 und Abb. 7.6.

Die höhere Punktdichte führt zu deutlich mehr 3D-Punkten für die Zellen der Belegungskarte und damit augenscheinlich zu einer höheren statistischen Aussagekraft der Hinderniswahrscheinlichkeit. Andererseits unterliegen die 3D-Punkte deutlich stärkerem Rauschen, insbesondere an Tiefensprüngen und Diskretisierungseffekten, wie Abb. 7.7 zeigt.

Diese Effekte führen zu deutlich mehr falsch positiven Hindernisklassifizierungen, d.h. Zellen werden deutlich häufiger unbefahrbar klassifiziert, auch wenn sie eigentlich befahrbar sind, was zu einer erhöhten Hinderniswahrscheinlichkeit führt. Infolgedessen betrachtet die Belegungskarte große Bereiche der Umgebung als unbefahrbar, wodurch das Fahrzeug zum Stehen kommt. Zur Reduzierung der falsch-positiv-Rate muss der Hindernisschwellwert $h^{\text{befahrbar}}$ von 0,2 m auf 0,8 m angehoben werden. Dementsprechend werden Objekte erst dann als Hindernis klassifiziert, wenn sie mindestens 80 cm hoch sind. Kleinere Objekte führen damit unweigerlich zu Kollisionen.

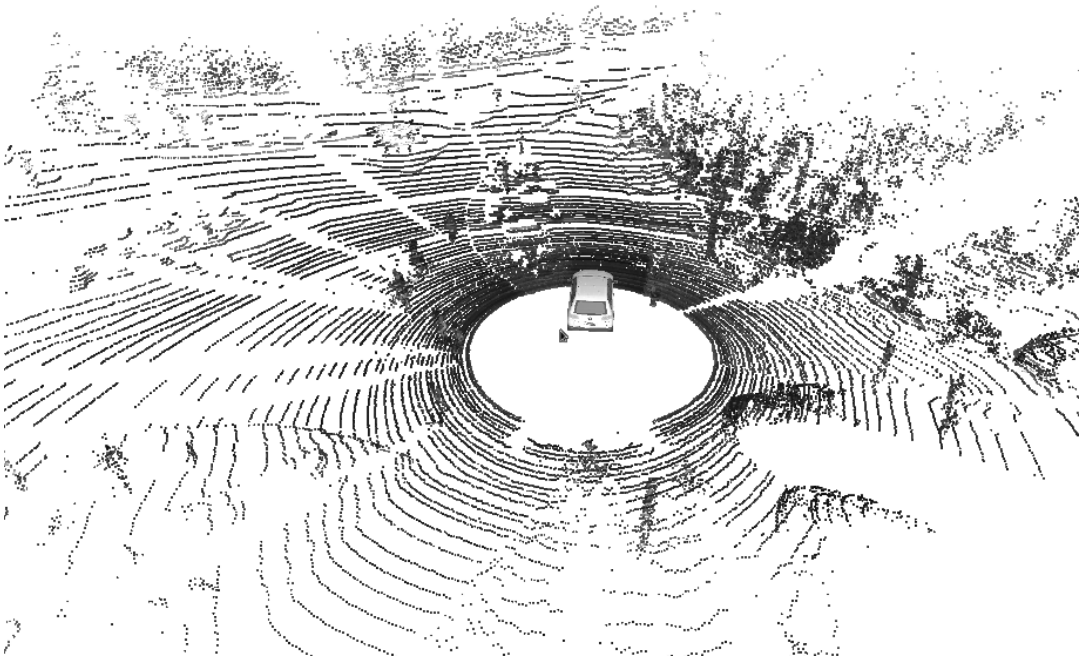


Abbildung 7.5:
Punktwolke eines Velodyne HDL-64E. Die Grauwerte repräsentieren die Intensität des reflektierten Laserlichts.

Im Vergleich dazu treten bei der Betrachtung der RAS-Punktwolke, wie in Abb. 7.8 gezeigt, keine derartigen Diskretisierungs- oder Ausfransungserscheinungen auf. Das Verfahren liefert stabile 3D-Punkte, die auch mit einem Befahrbarkeitsschwellwert von $h^{\text{befahrbar}} = 0{,}2\,\text{m}$ verlässliche Hinderniswahrscheinlichkeiten liefern. Da die Punkte außerdem zeitlich verfolgt werden, ist die zeitliche Akkumulation durch die Belegungskarte unnötig und wird deaktiviert. Jede Zelle wird durch einen Höhenvergleich mit $h^{\text{befahrbar}}$ in jeder Iteration als befahrbar oder unbefahrbar klassifiziert.

Aus Performanzgründen muss das 2D-Gitter von RAS auf eine Kantenlänge von 15 pix gesetzt werden, um alle Berechnungen noch innerhalb eines Messzyklus' von 100 ms durchführen zu können. Dadurch sinkt die Zahl der 3D-Punkte in den Bereich einiger Tausend. Durch die geringe Anzahl von 3D-Punkten wird nur ein kleiner Teil der Zellen der Belegungskarte klassifiziert. Der überwiegende Teil der Zellen erhält nach Gleichung (7.3) eine Hinderniswahrscheinlichkeit von 50 %. Diese Zellen erschweren die Suche nach einem kollisionsfreien Pfad. Dieses Problem kann durch die folgenden Maßnahmen gelöst werden: Verlängern der Zykluszeit, Anpassen der Zellgrößen und Interpolation.

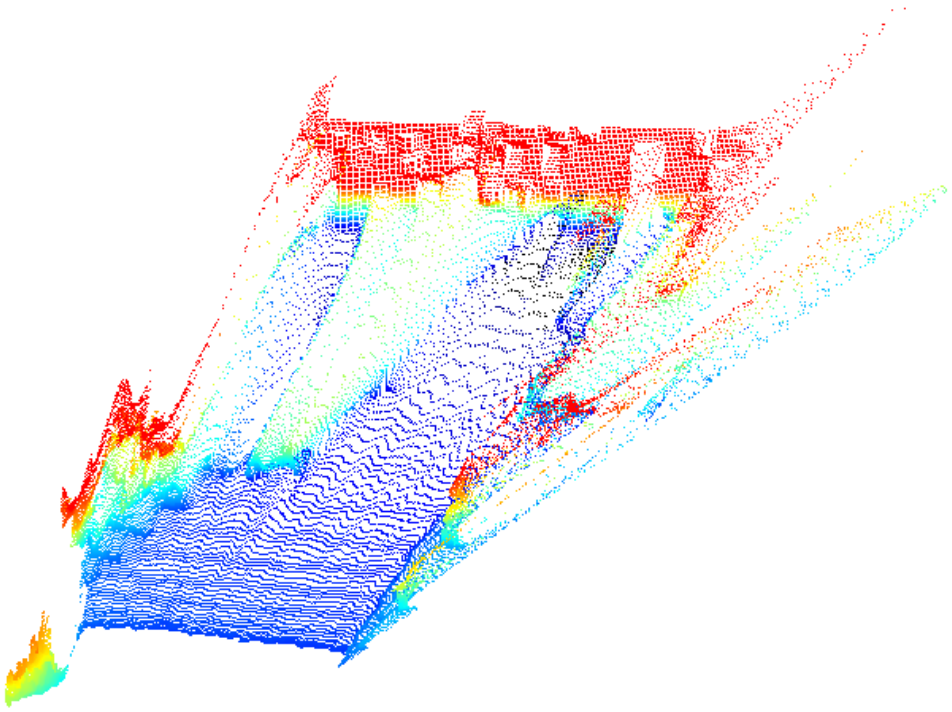
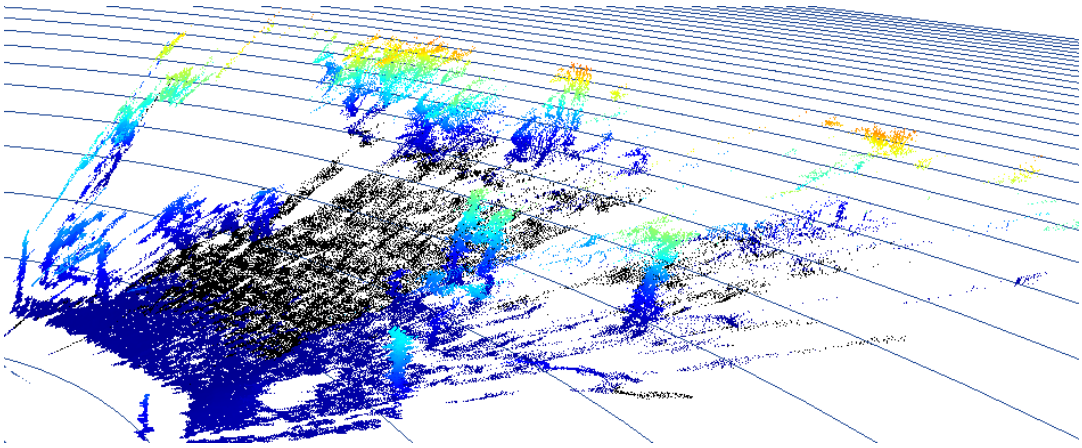


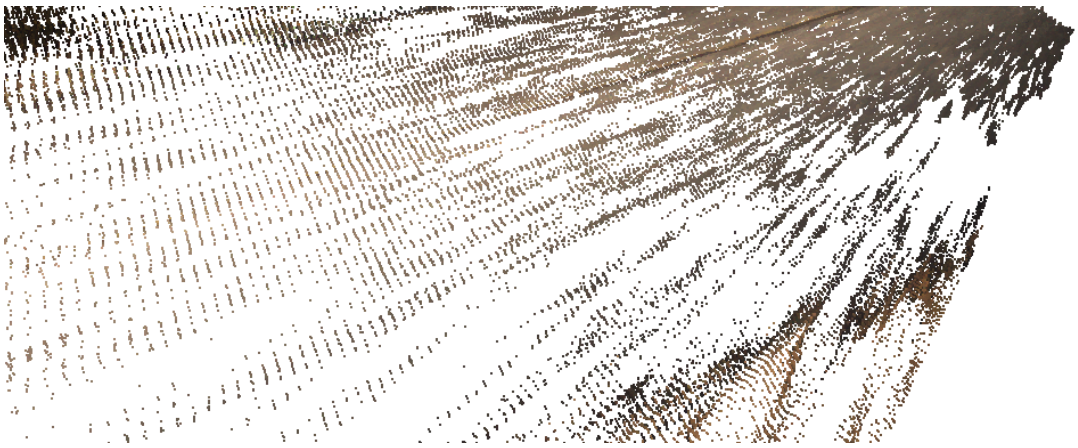
Abbildung 7.6: 3D-Punktwolke auf Basis von BM. Die Farben repräsentieren die Höhe.

Verlängern der Zykluszeit Das Verlängern der Zykluszeit gibt dem Algorithmus mehr Zeit für die Berechnungen und erlaubt dadurch die zeitliche Filterung von mehr 3D-Punkten, wodurch das 2D-Gitter enger gewählt werden kann. Allerdings stehen die Messergebnisse erst später zur Verfügung. Bei einer angenommenen Geschwindigkeit von 10 m/s und einer Verdopplung der Zykluszeit von 100 ms auf 200 ms legt das Fahrzeug zwischen zwei Messzeitpunkten statt 1 m damit 2 m zurück, fast eine halbe Fahrzeuglänge.

Anpassen der Zellgrößen Eine weitere Möglichkeit, die Lücken in der Belegungskarte zu schließen, ist die Vergrößerung der Zellgrößen der Belegungskarte. Für LiDAR-Punktwolken wird mit einer Zellgröße von 0,2 m mal 0,2 m gearbeitet. Allerdings wird bei RAS der Nahbereich durch die perspektivische Projektion dichter abgetastet als der Fernbereich. Eine Vergrößerung der Zellen würde damit hauptsächlich zum Schließen der Lücken im Nahbereich beitragen. Um auch im Fernbereich die Lücken zu schließen, müssten die Zellen größer als für den Nahbereich notwendig gewählt werden, was im Widerspruch zu einer möglichst feinen Abtastung der Umgebung im Nahbereich stehen kann.



(a) Ausgefranste Objektkanten an Tiefensprüngen.



(b) Diskretisierungseffekte, dargestellt an einer eingefärbten Stereo-Punktwolke.

Abbildung 7.7: Typische Probleme mit Punktwolken aus dem BM-Verfahren.

Interpolation Die implementierte Lösung besteht aus der Interpolation zwischen einzelnen RAS-Punkten zur Schließung der Lücken. Für jede Zelle wird dabei die Höhe aus der Interpolation zwischen den drei am nächsten liegenden RAS-Punkten berechnet. Das Ergebnis ist in Abb. 7.8 gezeigt. Sowohl die Nachbarschaftssuche als auch die Interpolation können effizient berechnet werden. Zur Wahl der Interpolation gegenüber den zuvor genannten Möglichkeiten gibt es noch weitere Gründe. Das Verlängern der Zykluszeit erhöht die Zeit, in der das Fahrzeug blind fährt, und ist somit eine kritische Größe. Sie führt außerdem zu einer Verzögerung der nachgelagerten Planung und Regelung des Fahrzeugs, die auf die bestehende Zykluszeit ausgelegt ist. Eine Verlängerung der Zykluszeit hat damit auch Auswirkungen auf die Fahrzeugregelung. Größere Zellen verschlechtern die Auflösung der Belegungskarte im Nahbereich. Bis zum Füllen der Lücken im Nahbereich ist diese Maßnahme sinnvoll, darüber hinaus führt sie durch die stärkere Quantifizierung zu einem Informationsverlust im Nahbereich. Die Beibehaltung der Zellgröße erlaubt zudem die Einbeziehung der

LiDAR-Messungen oder anderer Sensorik, sofern nicht schon in RAS integriert, unter der bekannten Parametrierung.

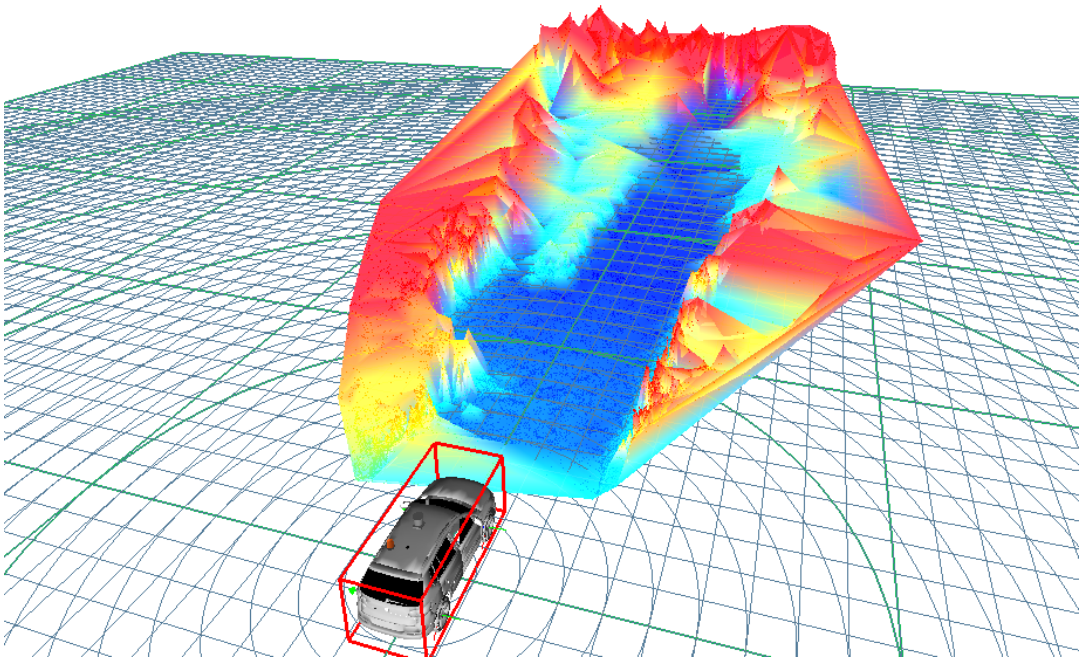


Abbildung 7.8:
3D-Oberflächenmodell auf Basis von RAS. Die Farben repräsentieren die Höhe.

8 Zusammenfassung und Ausblick

8.1 Schlussbetrachtung

An diesem Punkt ist es möglich, ein autonomes Fahrzeug kollisionsfrei durch unbekanntes Terrain zu bewegen. Die dazu nötigen Teildisziplinen unterteilen sich in Sensorwahrnehmung, Sensorsynchronisation, Sensorkalibrierung, Sensorfusion und -verarbeitung sowie Navigation. Diese Arbeit trägt zu allen genannten Bereichen bei, hauptsächlich zu den Themen Synchronisation, Kalibrierung und Fusion. Die einzelnen Beiträge sind hier nochmal zusammenfassend aufgeführt, bevor im Anschluss ein Ausblick auf weitere Erweiterungen gegeben wird.

8.1.1 Sensorsynchronisation

Diese Arbeit beschreibt Algorithmen zum autonomen Fahren, die auf einem Multi-Sensorsystem beruhen. Insbesondere steht die Kombination von Kamera und Light Detection and Ranging (LiDAR) im Fokus der Arbeit. Um die Daten beider Sensoren miteinander zu fusionieren, wird eine Hardware/Software Lösung zur Synchronisation beider Sensoren vorgestellt. Die Aufgabe wird insbesondere dadurch erschwert, dass das Kamerasystem um die Vertikalachse rotierbar ist, d.h. der Blickwinkel der Kamera als variabel angesehen werden muss. Die Lösung beruht auf dem LiDAR als Taktgeber, d.h. jede Rotation des Sensors wird als ein neuer Zyklus interpretiert. Ein Echtzeitsystem, basierend auf einer dSPACE MicroAutoBox, bestimmt nicht nur die Rotationsfrequenz des LiDAR, sondern misst zeitgleich die Belichtungszeit des Kamerasystems. Unter Berücksichtigung des Kamerablickwinkels und der zuvor ermittelten Belichtungszeit ermittelt das System den Triggerzeitpunkt der Kameras mit dem Ziel, die Überlappung der Sichtbereiche beider Sensoren während der Aufnahme zu maximieren.

8.1.2 Datensynchronisation

Auch wenn die Aufnahmezeitpunkte beider Sensoren somit synchronisiert sind, erlaubt erst die Kombination mit dem Konzept der Datenquellen des auf der KogniMobil Real Time Data Base (KogMo-RTDB) basierenden `tas::app`-Framework die Kompensation der unterschiedlichen Erfassungswege beider Sensoren. Zur Erinnerung: Während die einzelnen Winkelabschnitte des durch den LiDAR erfassten Sichtbereichs kontinuierlich in Form kleiner Netzwerkpakete den Verarbeitungsrechner erreichen und somit fortwährend intertial korrigiert werden können, erreicht das zugehörige Kamerabild den Rechner erst nach Vollendung der Aufnahme inklusive einem Systembus-bedingten Zeitversatz. Die Unterscheidung zwischen Datenzeitstempel und Übertragungszeitstempel ermöglicht eine semi-automatische Synchronisation der Daten beider Sensoren. Das `tas::app`-Framework reduziert dabei das Synchronisationsproblem lediglich auf die Frage der Reihenfolge und Frequenz der eintreffenden Sensordaten. Ebenfalls zu den Sensordaten synchronisiert wird die Eigenbewegung des Fahrzeugs, die von einem Inertial Navigation System (INS) gemessen wird. Durch die im Vergleich zu LiDAR und Kamera hohe Abtastfrequenz und geringe Übertragungslatenz ist eine Interpolation der Eigenbewegung hin zum Aufnahmezeitpunkt möglich.

8.1.3 Sensorkalibrierung

Im Bereich Sensorkalibrierung analysiert diese Arbeit zunächst die Sensorcharakteristika von Kamera und LiDAR und entwickelt ein Kalibrierungsverfahren, welches durch die Berücksichtigung dieser Sensoreigenschaften schnellere und genauere Kalibrierungsergebnisse liefert. Anstatt zu versuchen, den Kalibrierkörper bzgl. einer Liste an korrespondierenden 3D-Punkten bzgl. beider Sensoren darzustellen, beruht das Verfahren auf zwei Annahmen. Durch die im Vergleich zur Kamera schlechtere laterale Auflösung bei deutlich höherer Entfernungsauflösung lässt sich die Schachbrett-artige Kalibriertafel bzgl. des LiDAR besser in Form einer Ebene durch die Hesse'sche Normalform beschreiben. Da die Kamera die Entfernung zu den relevanten 3D-Punkten nicht ermitteln sondern allenfalls schätzen kann, wird das Schachbrett bzgl. der Kamera besser durch eine Reihe an Sehstrahlen vom Kamerazentrum durch die entsprechenden Merkmalpunkte des Schachbretts repräsentiert. Die relative Kalibrierung der beiden Sensoren zueinander wird dann mit Hilfe numerischer Optimierung gefunden, wobei die Optimierungsfunktion darauf ausgelegt ist, dass die Schnittmenge aus 3D-Ebene und Sehstrahlen mit den Abmessungen des Schachbretts korrespondiert.

Nach individueller Analyse des traditionellen Punkt zu Punkt-Ansatzes und des Ebene zu Strahl-Ansatzes, führt letztlich die Kombination beider Ansätze zur schnelleren Konvergenz des Optimierungsverfahrens bei gleichzeitig genauerer Schätzung der Relativpose zwischen den Sensoren.

Darüber hinaus wird in dieser Arbeit ein weiteres deutlich generischeres Kalibrierungsverfahren vorgestellt, welches auf der Fähigkeit zur Eigenbewegungsermittlung der beteiligten Sensoren basiert. Ausgehend von der Beobachtung, dass die Bewegung eines Fahrzeugs von unterschiedlichen Sensorposen im Fahrzeug ebenfalls unterschiedlich wahrgenommen wird, wurde ein UKF-basiertes (Unscented Kalmanfilter) Verfahren entwickelt, dass aus diesen Bewegungunterschieden zwischen aufeinanderfolgenden Zeitpunkten die extrinsische Kalibrierung der messenden Sensoren bestimmt. Damit konnte in Simulation und auf Realdaten ein Stereokamerasystem zu einem INS *online* kalibriert werden.

8.1.4 Sensorfusion

Die in dieser Arbeit untersuchte Sensorfusion von Kamera und LiDAR betrachtet zwei Ebenen der Sensorfusion: Fusion auf Rohdaten und Fusion auf Merkmalen. Ausgehend von einer relativ einfachen Rohdatenfusion, mit deren Hilfe einerseits 3D-Punkte eingefärbt werden und andererseits einzelnen Bildpixeln Tiefeninformationen zur Verfügung gestellt wird, erfolgt eine Erweiterung mit einem Algorithmus zur Verdeckungsrechnung. Grund für die Verdeckungsrechnung ist hierbei die unterschiedliche Einbauhöhe von Kamera und LiDAR, wodurch der LiDAR in der Lage ist, hinter Hindernisse zu schauen, während dieser Bereich für die Kamera verdeckt bleibt.

Darauf aufbauend wird ein Stereo-Algorithmus entwickelt, der basierend auf existierenden Verfahren zur Berechnung von Disparitätenbildern unter Zuhilfenahme der Eigenbewegung des Fahrzeugs ein rekursiv geschätztes, punktbasiertes Umgebungsmodell berechnet. Das Verfahren bestimmt relevante Bildpunkte, die anhand der Eigenbewegung des Fahrzeugs prädiziert und mit Hilfe der Disparität und Bilddeskriptormethoden korrigiert werden. Während das Verfahren anfangs noch mit Hilfe von LiDAR-Messungen evaluiert wird, integriert eine Erweiterung die gemessenen 3D-LiDAR-Punkte in das Messmodell des Filters, um die Genauigkeit der geschätzten Entfernungen zu verbessern. Eine zusätzliche Erweiterung befähigt das Verfahren, auch in dynamischen Umgebungen zu agieren, indem die 3D-Punkte um Geschwindigkeitskomponenten erweitert werden. Der Nutzen dieser Umgebungsdarstellung wird

anschließend an Beispiel autonomer Hindernisvermeidung dargestellt. Das Fahrzeug ist in der Lage, kollisionsfrei in unbekannter Umgebung einer groben Bahnplanung zu folgen und bei Bedarf Hindernissen auszuweichen.

8.2 Weiterführende Arbeiten

Zur weiteren Vertiefung und Verbesserung der in dieser Arbeit vorgestellten und entwickelten Verfahren gibt es mehrere Ansätze für zukünftige, weiterführende Arbeiten. Diese sind im Folgenden thematisch gruppiert aufgeführt.

8.2.1 Sensorsynchronisation

Die Sensorsynchronisation von Kamera und LiDAR basiert auf einem Echtzeitsystem und liefert somit erwartungsgemäß gute Synchronisationsergebnisse der beiden Sensoren. Während die Anbindung an die Kameras durch direkte Verbindungen zum Auslösen der Belichtung und Messen der Belichtungszeit ideale Voraussetzungen zur Synchronisation liefert, führt der Signalweg des LiDAR, welcher den Start einer neuen Sensorrotation transportiert, über einen nicht echtzeitfähigen Verarbeitungsrechner. Die Verzögerungen durch den UDP-Datentransfer (User Datagram Protocol) vom Sensor zum Rechner sowie die Verarbeitungszeit auf dem Rechner sind unbekannt und somit nur abgeschätzt bekannt. Eine Verbesserungsmöglichkeit wäre eine direkte Signalverbindung zwischen Echtzeitrechner und LiDAR, die durch ein logisches Signal den Start einer Rotation des Sensors verzögerungsfrei kommuniziert. Die verwendeten Sensoren des Typs Velodyne HDL-64E verfügen allerdings nicht über einen derartigen Signalausgang. Eine Untersuchung müsste zeigen, ob eine derartige Erweiterung möglich ist. Eine weitere Möglichkeit wäre die Verwendung eines externen Zeitgebers. Neuere Versionen der Velodyne HDL-64E-Sensoren können die einzelnen Datenpakete mit Hilfe eines GPS-Moduls (Global Positioning System) auf eine globale Zeitbasis zeitstempeln. Eine Erweiterung des Echtzeitrechners um ein ähnliches Empfangsmodul der globalen Zeitbasis wäre ausreichend, um das aktuell unbekannt verzögerte Rotationssignal des LiDAR zeitlich korrekt mit dem Auslösesignal der Kameras in Bezug zu setzen.

8.2.2 Datensynchronisation

Das `tas::app`-Framework vereinfacht das Auffinden zeitlich zueinander passender Eingangsdaten. Wie beschrieben ist dazu Wissen über die Taktraten der Datenquellen und die zeitliche Verzögerung der Daten zwischen Aufnahmezeitpunkt und ihrer Verfügbarkeit notwendig. Durch die starke Modularisierung der einzelnen Verarbeitungsschritte (z.B. Bildeinzug, Farbraumkonvertierung, Segmentierung, Extraktion von Bildmerkmalen, ...) ist diese Zeitverzögerung nicht immer leicht zu bestimmen, vom Anwendungsfall abhängig und je nach Wahl der Verarbeitungsalgorithmen auch nicht notwendigerweise konstant. Es ist daher zu untersuchen, ob die Festlegung auf ein einzelnes *auslösendes* Objekt ausreichend ist. Unter der Annahme konstanter Taktraten aller Komponenten in der Verarbeitungskette wäre ein Lösungsansatz, die Taktraten der jeweiligen Datenquellen einer jeden Komponente über der Zeit zu bestimmen und basierend auf Vorhersagen die Berechnungszeitpunkte bzw. Taktrate einer jeden Komponente festzulegen. Im Allgemeinen kann allerdings nicht von konstanten Taktraten aller beteiligten Komponenten ausgegangen werden. Daher ist je nach Anwendung zu untersuchen, ob asynchrone Dateneingänge beispielsweise durch Prädiktion der einzelnen Größen auf einen gemeinsamen Zeitpunkt oder die Anpassung des Verarbeitungsalgorithmus auf sequentielle Verarbeitung möglich ist.

8.2.3 Sensorkalibrierung

Die extrinsische Kalibrierung von Kamera und LiDAR unter Berücksichtigung der Sensorcharakteristika zeigt Vorteile gegenüber einer traditionellen Kalibrierung. Dennoch basiert das Verfahren auf der Verwendung eines Kalibrierkörpers, welcher im täglichen Gebrauch gewöhnlich nicht zur Verfügung steht. Eine Erweiterung des Verfahrens auf natürlich vorkommende Merkmale, wie Ecken und Kanten in Bildern oder geometrische Primitive in 3D-Punktwolken, würde die Einsatzmöglichkeiten des Verfahrens steigern. Außerdem basiert das Verfahren auf numerischer Optimierung, die vorab die Aufzeichnung eines Kalibrierungsdatensatzes erfordert. Damit ist das Verfahren nicht zur kontinuierlichen Kalibrierung und Re-Kalibrierung der Sensorik geeignet, was insbesondere in Hinblick auf die drehbare Kameraplattform von Vorteil wäre. Ein auf einem Kalmanfilter (KF) basierender Ansatz könnte hier Abhilfe schaffen, erfordert allerdings ein genaueres Augenmerk auf die Initialisierung des Filters sowie die erwarteten Unsicherheiten.

Das rein auf der Sensorbewegung basierende, generische Kalibrierungsverfahren, welches in dieser Arbeit entwickelt wurde, arbeitet intern mit einem UKF und neben kartesischen Koordinaten mit Eulerwinkeln zur Repräsentation der relativen Orientierung der beiden Sensoren. Bei der Verwendung von Winkelgrößen und Unsicherheiten gilt es zu beachten, dass Winkelgrößen zyklisch sind, d.h. 0° entsprechen 360°, 10° entsprechen 370° usw. Wenn die Unsicherheiten zu groß werden und im Rahmen der Unscented Transform (UT) die Sigmapunkte berechnet werden, muss ein Überlauf verhindert werden, da ansonsten die Rekonstruktion der Gaußverteilung nach Transformation der Sigmapunkte nicht korrekt ist. Durch sog. Pivotisierung kann diese Gefahr verringert, allerdings nicht ausgeschlossen werden. Die Verwendung von Eulerwinkeln birgt zudem ebenfalls das Risiko eines sog. *Gimbal Locks*, d.h. des Verlustes eines rotatorischen Freiheitsgrades. Daher empfiehlt sich die Untersuchung, ob durch die Verwendung von Quaternionen oder Lie Algebra anstelle von Eulerwinkeln die Schätzung insgesamt robuster wird, da ein *Gimbal Lock* in beiden Ansätzen mathematisch ausgeschlossen wird.

Darüber hinaus konzentriert sich diese Arbeit auf das Thema extrinsische Sensorkalibrierung. Eine Einbeziehung der intrinsischen Sensorkalibrierung stellt ebenfalls ein weiteres interessantes Forschungsthema dar. Interessant wäre diesbezüglich die Verwendung des in Uhlig und Heizmann [2021] vorgestellten generalisierten Kameramodels und Kalibrierungsverfahrens auch für den Velodyne HDL-64E.

8.2.4 Sensorfusion

Wie bereits angesprochen basiert das Verfahren zur Verdeckungsrechnung beim Einfärben von LiDAR-Punktwolken auf der Annahme, dass die konvexe Hülle aller 3D-Punkte eines Segments der realen Objektkontur entspricht. Jeder einzelne Laser eines Velodyne HDL-64E tastet die Umgebung in äquidistanten Winkelinkrementen ab. Liegt die reale Objektkontur zwischen zwei Abtastwinkeln, basiert die Berechnung der konvexen Hülle auf einem näher der Objektmitte liegenden Punkt. Dementsprechend wird die konvexe Hülle im Allgemeinen kleiner als das eigentliche Objekt sein. Das kann dazu führen, dass weiter entfernt liegende Punkte, die zwischen konvexe Hülle und realer Objektkontur im Bild projiziert werden, fälschlicherweise mit der Objektfarbe eingefärbt werden. Der Effekt tritt theoretisch häufiger auf, je weiter das Objekt vom LiDAR entfernt ist, da in diesem Fall der Abstand zwischen zwei Abtastpunkten eines Lasers größer ist. Auf der anderen Seite blickt bei der hier vorliegenden Sensorkonfiguration der LiDAR erst bei nahe liegenden Objekten hinter

das von der Kamera gesehene Objekt, was den Fehler zum Teil wieder kompensiert. Zur korrekten Lösung des Problems sollte die Berechnung der konvexen Hülle um ein Bildsegmentierungsverfahren wie [Achanta et al., 2012] ergänzt werden. Mit Hilfe eines *Region Growing*-Ansatzes kann eine bestehende konvexe Hülle auf die umgebenden Konturen der SLIC-Segmente (Simple Linear Iterative Clustering) ausgeweitet werden. Damit würde die Objektkontur deutlich besser nachgebildet.

Das RAS-Verfahren (Rekursives Automotives Stereo) liefert im Wesentlichen zwei Stellen, an denen weiterführende Arbeiten anknüpfen können. Zum einen basiert die 3D-Repräsentation der Umgebung aktuell nur aus einer Triangulation der gefilterten 3D-Punkte. Die Triangulation wird umso genauer, je dichter die Umgebung repräsentiert wird, d.h. je engmaschiger das 2D-Gitter zur Erzeugung neuer Punktfilter gewählt wird. Allerdings wird dieser Parameter durch die zur Verfügung stehende Rechenleistung beschränkt. Eine realistischere Abbildung des Umgebungsbereichs zwischen den gefilterten 3D-Punkten könnte durch eine Funktion höherer Ordnung, wie Splines, approximiert werden. Denkbar wäre eine Art Spline-Gitter, zu dem die gefilterten 3D-Punkte als Kontrollpunkte beitragen. Auf der anderen Seite basiert RAS selbst auf den Ergebnissen bereits existierender Stereo- bzw. Disparitätenverfahren. Die meisten dieser Verfahren widersprechen der grundlegenden Annahme von RAS, dass sich die Umgebung zwischen zwei aufeinanderfolgenden Zeitschritten nur unwesentlich ändert. Statt jedes Stereobildpaar als individuelles Problem zu betrachten, sollten die einmal erhaltenen Informationen aus einem Stereobildpaar zur Berechnung des bzw. der folgenden Disparitätenbilder miteinbezogen werden. In der Konsequenz sollten die gefilterten RAS-Punkte, bzw. auch die ggf. interpolierten Informationen über die Zwischenräume, direkt in die Disparitätenberechnung einfließen. Das ELAS-Verfahren (Efficient Large-scale Stereo Matching) [Geiger et al., 2010] arbeitet im Hintergrund ebenfalls mit einzelnen Punkten und Triangulation. Daher würde sich dieses Verfahren voraussichtlich zur Erweiterung anbieten.

A Notation

A.1 Basistypen

Die Notation von Variablen in dieser Arbeit wird von einer Vielzahl an Parametern – wie bspw. Zeit oder Koordinatensystemen – beeinflusst. Um Mehrdeutigkeiten zu umgehen, wird an dieser Stelle die Notation von Variablen beschrieben. Dazu wurden einige Basistypen identifiziert, die in der folgenden Tabelle A.1 aufgelistet sind.

Name	**Beispiel**	**Beschreibung**
3D-Punkt	$\boldsymbol{p}$	fett, kursiv, Serifen, klein
Vektor	$\boldsymbol{v}$	wie 3D-Punkt
Vektor (normalisiert)	$\bar{\boldsymbol{v}}$	wie Vektor, Balken
Quaternion	q	Serifen, klein
Einheitsquaternion	$\bar{\mathrm{q}}$	wie Quaternion, Balken
Zustandsvektor	$\boldsymbol{x}$	wie Vektor
Matrix	$\mathbf{M}$	fett, Serifen, groß
Pose	$\mathbf{P}$	wie Matrix
Pixel	p	serifenlos, klein
Pixel (ohne Korrektur der Verzeichnung)	$\bar{\mathrm{p}}$	serifenlos, klein, Balken
Punktwolke	$\mathbf{P}$	fett, Serifen, groß
Skalar	s	kursiv, Serifen, klein
Menge	$\mathcal{S}$	kalligraphisch, groß

Tabelle A.1: Basistypen der verwendeten Notation.

A.2 Attribute

Da Mehrdeutigkeiten nicht immer vermeidbar sind, wurde darauf geachtet, dass Basistypen mit identischen Literalen in möglichst verschiedenen Kontexten verwendet werden. Zur genaueren Spezifikation ist es außerdem möglich, die Basistypen um weitere Attribute – wie Koordinatensysteme oder Zeitindizes – zu erweitern. Diese

Attribute sind in Tabelle A.2 beschrieben und können natürlich auch in Kombination auftreten.

Name	Beispiel	Beschreibung
Basiskoordinatensystem	$\mathbf{M}_{\text{ego}}$, $\boldsymbol{p}_{\text{ego}}$	Das Koordinatensystem, bzgl. dessen eine Transformationsmatrix oder ein Punkt angegeben sind.
Basiszeit	$\mathbf{M}_{k+1}$	Der Zeitpunkt eines bestimmten Wertes.
Zielkoordinatensystem	${}^{\text{Kamera}}\mathbf{M}_{\text{ego}}$	Nur in Verbindung mit dem Basiskoordinatensystem. Gibt das Zielkoordinatensystem einer Transformation an.
Zielzeit	${}^{\text{ego},k+1}\mathbf{M}_{\text{ego},k}$	Nur in Verbindung mit dem Basiskoordinatensystem und der Basiszeit. Gibt die Zielzeit einer Transformation an.
Komponente	$\boldsymbol{v}_{[x]}$, $\mathbf{M}_{[i,j]}$	Extrahiert eine Untermenge z.B. Skalar aus einem Vektor oder einer Matrix. Zugriff kann über Namen, Indizes oder Skalare geschehen.
Index	$\boldsymbol{v}^{(i)}$	Gibt das i-te Element einer Serie an. Nicht zu verwechseln mit $\boldsymbol{v}^i$, der i-ten Potenz von $\boldsymbol{v}$.
Text	$\boldsymbol{v}^{\text{Messung}}$	Zur textuellen Erweiterung einer Variablen.
Prädizierte Variable	$\boldsymbol{x}^*$	Zur Kenntlichmachung prädizierter Werte bspw. innerhalb von Filtergleichungen.
Korrigierte Variable	$\hat{\boldsymbol{x}}$	Zur Kenntlichmachung korrigierter Werte bspw. als Ergebnis einer zeitlichen Filterung.
Transponierte Variable	$\boldsymbol{v}^T$	Zur Darstellung transponierter Vektoren und Matrizen.
Invertierte Variable	$\mathbf{M}^{-1}$	Zur Darstellung einer invertierten Matrix.

Tabelle A.2: Liste der Attribute

Bei der Kombination mehrerer Attribute werden diese an den entsprechenden Ecken nach einer festen Reihenfolge konkateniert. Ein Beispiel dazu ist das Element in Zeile m und Spalte n der prädizierten i-ten homogenen Fahrzeug-Transformationsmatrix $\mathbf{H}$, die bzgl. des Basis- und Zielkoordinatensystems $\mathbf{S}_{\text{ego}}$ zwischen den Zeitpunkten k und $k+1$ transformiert:

$$ {}^{\text{ego},k+1}\mathbf{H}^{(i)\ \text{Fahrzeug}\,*}_{[m,n]\ \text{ego},k} \tag{A.1}$$

A.3 Koordinatensysteme

Im Rahmen der Arbeit werden sowohl kartesische als auch polare Koordinatensysteme verwendet. Auch wenn in der Regel kartesische Koordinaten verwendet werden, so finden an einigen Stellen Umrechnungen statt. Um an diesen Stellen deutlich zu machen, um welche Art von Koordinatensystem es sich handelt, wird die Symbolik aus Tabelle A.3 eingeführt.

Name	**Beispiel**	**Beschreibung**
Kartesisches Koordinatensystem	$\mathbf{S}$	Ein beliebiges kartesisches Koordinatensystem.
Kartesisches Koordinatensystem mit Namen	$\mathbf{S}_{\text{Kamera}}$	Ein kartesisches Koordinatensystem im Ursprung einer Kamera.
Polares Koordinatensystem	$\mathbf{T}$	Ein beliebiges polares Koordinatensystem.
Polares Koordinatensystem mit Namen	$\mathbf{T}_{\text{LiDAR}}$	Ein polares Koordinatensystem im Ursprung eines LiDAR.
HTM zur Transformation zwischen zwei kartesischen Koordinatensystemen	${}^{\text{LiDAR}}\mathbf{H}_{\text{Kamera}}$	Eine HTM zur Transformation zwischen den beiden kartesischen Koordinatensystemen einer Kamera und eines LiDAR.

Tabelle A.3: Symbolik von Koordinatensystemen.

B Datenblätter

B.1 Kameras

B.1.1 AVT Guppy F-036C

Die verwendete Allied Vision Technologies GmbH (AVT) Guppy F-036C ist eine Farbkamera mit einem HDRC-Chip (High Dynamic Range CMOS). Die wichtigsten Daten der Kamera sind aus dem Datenblatt [ALLIED Vision Technologies GmbH, 2006] entnommen.

Sensor	$\frac{1}{3}$	"	Micron MT9V022 Sensor
	752×480	pix	Pixel
	6	µm	Pixelgröße
	8	bit	Farbtiefe
	10	bit	A/D-Wandler
Kamera	100, 200 und 400	Mbit/s	Übertragungsrate
	≤ 60	fps	Bildrate
	50	g	Masse (ohne Objektiv)
	$48{,}3 \times 30 \times 30$	mm	Abmessungen (L $\times$ B $\times$ H)

B.1.2 Bumblebee XB3

Die verwendete Bumblebee XB3 ist eine tri-fokale Farbkamera mit insgesamt entsprechend drei vorkalibrierten bildgebenden CCD-Einheiten (Charge Coupled Device). Die wichtigsten Daten der Kamera sind aus dem Datenblatt [Point Grey Research, Inc., 2012] entnommen.

Sensor	$\frac{1}{3}$	"	Sony ICX445 CCD Sensor
	1280 × 960	pix	Pixel
	3,75	µm	Pixelgröße
	8	bit	Farbtiefe
	12	bit	A/D-Wandler
Kamera	400 und 800	Mbit/s	Übertragungsrate
	12 und 24	cm	Basislänge
	16	fps	Bildrate
	505	g	Masse (ohne Objektiv)
	288 × 37 × 41,8	mm	Abmessungen (L × B × H)

B.2 LiDAR

Die folgenden Tabellen geben einen Überblick über die wesentlichen Sensorparameter der beiden verwendeten Velodyne LiDAR. In dieser Arbeit wird ausschließlich eine Rotationsfrequenz von 10 Hz verwendet. Die übrigen Daten dienen ausschließlich der Information.

B.2.1 Velodyne HDL-64E S1

Die folgenden Daten sind dem Datenblatt [Velodyne Acoustics, Inc., 2008] entnommen.

Sensor	64		Laser und Laserdetektoren
	360	°	Laterales Blickfeld
	26,8	°	Vertikales Blickfeld ($-2°$ - 24,8°)
	$< 0{,}05$	m	Entfernungsgenauigkeit
	5 - 15	Hz	Rotationsfrequenz
	$> 1 \times 10^6$		Messungen pro Sekunde
Auflösung (oberer Block)	ca. 0,4	°	Vertikale Auflösung
	0,0576	°	Laterale Auflösung bei 5 Hz
	0,1152	°	Laterale Auflösung bei 10 Hz
	0,1728	°	Laterale Auflösung bei 15 Hz
Auflösung (unterer Block)	ca. 0,4	°	Vertikale Auflösung
	0,2304	°	Laterale Auflösung bei 5 Hz
	0,4608	°	Laterale Auflösung bei 10 Hz
	0,6912	°	Laterale Auflösung bei 15 Hz

B.2.2 Velodyne HDL-64E S2

Die folgenden Daten sind dem Datenblatt [Velodyne LiDAR, Inc., 2012] entnommen.

Sensor	64		Laser und Laserdetektoren
	360	°	Laterales Blickfeld
	26,33	°	Vertikales Blickfeld ($-2°$ - 24,33°)
	$< 0{,}02$	m	Entfernungsgenauigkeit
	5 - 20	Hz	Rotationsfrequenz
	$1{,}3 \times 10^6$		Messungen pro Sekunde
Auflösung	0,33	°	Vertikale Auflösung $-2°$ bis 8,33°
	0,5	°	Vertikale Auflösung 8,33° bis 24,33°
	0,0864	°	Laterale Auflösung bei 5 Hz
	0,1728	°	Laterale Auflösung bei 10 Hz
	0,2592	°	Laterale Auflösung bei 15 Hz
	0,3456	°	Laterale Auflösung bei 20 Hz

Im Unterschied zum Velodyne HDL-64E S1 messen die unteren und oberen 32 Laser des Velodyne HDL-64E S2 synchron, wodurch die laterale Auflösung in beiden

Bereichen identisch ist. Jedoch ist die vertikale Anordnung zwischen beiden Versionen unterschiedlich und die Entfernungsgenauigkeit bei Version S2 höher.

B.3 OxTS RT3003 INS

Die folgenden Daten sind dem Datenblatt [Oxford Technical Solutions Ltd., 2011] entnommen und beschreiben die wesentlichen Informationen des verwendeten Inertial Navigation System (INS) des Typs Oxford Technical Solutions (OxTS) RT3003.

Sensor	100	Hz	Ausgaberate
	3,5	ms	Latenzzeit
Genauigkeit	0,6	m	CEP SBAS Position
	0,02	m	DGPS Position (1σ)
	$1{,}39 \times 10^{-2}$	m/s	Geschwindigkeit (RMS)
	0,03	°	Nick- und Wankwinkel (1σ)
	0,1	°	Fahrtrichtung (1σ)

C Homogene Transformationsmatrizen und Posen

In vielen Anwendungen der Robotik ist es notwendig, Koordinaten zwischen verschiedenen Koordinatensystemen zu transformieren. Zu diesem Zweck haben sich in der Robotik drei Repräsentationen etabliert: Homogene Transformationsmatrix (HTM), Posen und Quaternionen. Der vorliegende Abschnitt gibt einen kurzen Überblick und Empfehlungen zur Verwendung. Anschließend werden Operationen im Zusammenhang mit den genannten Repräsentationen so definiert, wie sie in der vorliegenden Arbeit verwendet werden.

C.1 Homogene Transformationsmatrizen

In dieser Arbeit finden HTM ihre Anwendung hauptsächlich in der Transformation homogener Koordinaten zwischen zwei Koordinatensystemen. Die zur Transformation verwendeten HTM beschreiben neben der relativen Lage der Koordinatensysteme auch deren räumliche Orientierung zueinander. Jede HTM, die die Transformation zwischen zwei Koordinatensystemen beschreibt, lässt sich in vier einzelne Matrizen zerlegen. Drei der vier Matrizen beschreiben jeweils die Rotation um die jeweiligen Koordinatenachsen. Die vierte beschreibt letztlich die Translation zwischen beiden Koordinatensystemen. Allgemein folgen die vier genannten HTM der folgenden Definition.

Rotation um die z-Achse mit dem Gierwinkel Ψ

$$\mathbf{R}^{\Psi} = \begin{bmatrix} \cos(\Psi) & \sin(\Psi) & 0 & 0 \\ -\sin(\Psi) & \cos(\Psi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.1}$$

Rotation um die y-Achse mit dem Nickwinkel Θ

$$\mathbf{R}^{\Theta} = \begin{bmatrix} \cos(\Theta) & 0 & -\sin(\Theta) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\Theta) & 0 & \cos(\Theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.2}$$

Rotation um die x-Achse mit dem Rollwinkel Φ

$$\mathbf{R}^{\Phi} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\Phi) & \sin(\Phi) & 0 \\ 0 & -\sin(\Phi) & \cos(\Phi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.3}$$

Translation des Koordinatenursprungs um $\boldsymbol{v} = (x, y, z)^T$

$$\mathbf{T}^{\boldsymbol{v}} = \begin{bmatrix} 1 & 0 & 0 & -x \\ 0 & 1 & 0 & -y \\ 0 & 0 & 1 & -z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.4}$$

C.1.1 Konkatenation von Matrixmultiplikationen

Da Matrixmultiplikationen nicht kommutativ sind, spielt die Reihenfolge der Multiplikationen eine Rolle. Die Definition für diese Arbeit wird an folgendem Beispiel deutlich. Gegeben sind zwei Koordinatensysteme $\mathbf{S}_{\text{ego}}$ und $\mathbf{S}_{\text{Kamera}}$. Das Kamerakoordinatensystem ist gegenüber $\mathbf{S}_{\text{ego}}$ um den Vektor $\boldsymbol{v} = (x, y, z)^T$ verschoben und gleichzeitig um Ψ, Θ und Φ rotiert. Die entsprechende HTM ${}^{\text{Kamera}}\mathbf{H}_{\text{ego}}$ ergibt sich dann zu

$${}^{\text{Kamera}}\mathbf{H}_{\text{ego}} = \mathbf{R}^{\Phi} \cdot \mathbf{R}^{\Theta} \cdot \mathbf{R}^{\Psi} \cdot \mathbf{T}^{\boldsymbol{v}}. \tag{C.5}$$

Damit führt eine Multiplikation eines beliebigen Vektors $\boldsymbol{x}_{\text{ego}}$ mit ${}^{\text{Kamera}}\mathbf{H}_{\text{ego}}$ zu $\boldsymbol{x}_{\text{Kamera}}$, d.h. die Multiplikation eines Vektors bzgl. $\mathbf{S}_{\text{ego}}$ mit ${}^{\text{Kamera}}\mathbf{H}_{\text{ego}}$ transformiert den Vektor in das Koordinatensystem $\mathbf{S}_{\text{Kamera}}$

$$\boldsymbol{x}_{\text{Kamera}} = {}^{\text{Kamera}}\mathbf{H}_{\text{ego}} \cdot \boldsymbol{x}_{\text{ego}}. \tag{C.6}$$

Allgemein ergibt sich damit durch Ausmultiplizieren von Gleichung (C.5) die übergreifende Beschreibung einer HTM zu

$$\begin{aligned}\mathbf{H} = &\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos(\Phi) & \sin(\Phi) & 0\\ 0 & -\sin(\Phi) & \cos(\Phi) & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\cos(\Theta) & 0 & -\sin(\Theta) & 0\\ 0 & 1 & 0 & 0\\ \sin(\Theta) & 0 & \cos(\Theta) & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\\ &\cdot\begin{bmatrix}\cos(\Psi) & \sin(\Psi) & 0 & 0\\ -\sin(\Psi) & \cos(\Psi) & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & -x\\ 0 & 1 & 0 & -y\\ 0 & 0 & 1 & -z\\ 0 & 0 & 0 & 1\end{bmatrix}\end{aligned} \tag{C.7}$$

$$\Rightarrow \mathbf{H} = \begin{bmatrix} c_\Psi c_\Theta & s_\Psi c_\Theta & -s_\Theta & z s_\Theta + (-x c_\Psi - y s_\Psi) c_\Theta \\ s_\Phi c_\Psi s_\Theta - c_\Phi s_\Psi & s_\Phi s_\Psi s_\Theta + c_\Phi c_\Psi & s_\Phi c_\Theta & \begin{matrix}(-x s_\Phi c_\Psi - y s_\Phi s_\Psi) s_\Theta \\ -z s_\Phi c_\Theta + x c_\Phi s_\Psi - y c_\Phi c_\Psi\end{matrix} \\ c_\Phi c_\Psi s_\Theta + s_\Phi s_\Psi & c_\Phi s_\Psi s_\Theta - s_\Phi c_\Psi & c_\Phi c_\Theta & \begin{matrix}(-x c_\Phi c_\Psi - y c_\Phi s_\Psi) s_\Theta \\ -z c_\Phi c_\Theta - x s_\Phi s_\Psi + y s_\Phi c_\Psi\end{matrix} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{C.8}$$

mit $c_\Psi = \cos(\Psi)$, $c_\Theta = \cos(\Theta)$, $c_\Phi = \cos(\Phi)$ und $s_\Psi = \sin(\Psi)$, $s_\Theta = \sin(\Theta)$, $s_\Phi = \sin(\Phi)$. Die Menge aller möglichen HTM wird bezeichnet als $\mathcal{H}$.

C.1.2 Zusätzliche Matrixoperationen mit Homogenen Transformationsmatrizen

Als bekannt vorausgesetzt wird die Multiplikation einer HTM mit 4×1 Vektoren, deren Ergebnis ebenfalls 4×1 Vektoren sind

$$\cdot : \mathbb{R}^{4\times4} \times \mathbb{R}^{4\times1} \mapsto \mathbb{R}^{4\times1}, (\mathbf{H}, \boldsymbol{p}) \mapsto \boldsymbol{p'} = \mathbf{H} \cdot \boldsymbol{p}. \tag{C.9}$$

Basierend auf dieser Matrix-Vektor-Multiplikation werden die folgenden zusätzlichen Operationen eingeführt, die im Rahmen der Arbeit vereinzelt zur Erhöhung der Lesbarkeit verwendet werden.

Multiplikation mit transponierten Vektoren Sollte der Vektor nur in transponierter Form vorliegen, wird dieser zunächst transponiert

$$\cdot : \mathbb{R}^{4\times4} \times \mathbb{R}^{1\times4} \mapsto \mathbb{R}^{4\times1}, (\mathbf{H}, \boldsymbol{p}) \mapsto \boldsymbol{p'} = \mathbf{H} \cdot \boldsymbol{p^T}. \tag{C.10}$$

Multiplikation mit nicht-homogenen Vektoren Sollte der Vektor nicht-homogen sein, wird diesem die vierte Komponente in Form einer 1 hinzugefügt

$$\cdot : \mathbb{R}^{4\times4} \times \mathbb{R}^{3\times1} \mapsto \mathbb{R}^{4\times1}, (\mathbf{H}, \boldsymbol{p}) \mapsto \boldsymbol{p'} = \mathbf{H} \cdot \begin{pmatrix} \boldsymbol{p} \\ 1 \end{pmatrix}. \tag{C.11}$$

Multiplikationsergebnis als nicht-homogener Vektor Sollte das Multiplikationsergebnis in Form eines nicht-homogenen Vektors erwartet werden, findet nach der Matrix-Vektor-Multiplikation eine entsprechende Division durch die vierte Komponente des Vektors statt, bevor diese anschließend entfernt wird

$$\cdot : \mathbb{R}^{4\times4} \times \mathbb{R}^{4\times1} \mapsto \mathbb{R}^{3\times1}, (\mathbf{H}, \boldsymbol{p}) \mapsto \boldsymbol{p'} = \left((\mathbf{H} \cdot \boldsymbol{p}) \cdot \frac{1}{(\mathbf{H} \cdot \boldsymbol{p})_{[4]}} \right)_{[1:3]}. \tag{C.12}$$

C.2 Posen

Auch wenn sich HTM zur Beschreibung der Relativlage zweier Koordinatensysteme und der Transformation von Koordinaten zwischen zwei Systemen eignen, ist die Lesbarkeit dieser 4×4 Matrixen nur erschwert möglich. Wie Gleichung (C.8) zeigt, sind die Winkel, d.h. die Orientierung der beiden Koordinatensysteme, nicht direkt ablesbar, da sie nur in Kombination und als Ergebnis trigonometrischer Funktionen erscheinen. Zwar ist der Translationsanteil ablesbar, d.h. der Abstand beider Koordinatensysteme entspricht der Vektornorm der ersten drei Parameter der vierten Spalte. Die Translationskoordinaten sind jedoch bereits bzgl. des Zielkoordinatensystems orientiert, so dass die Richtung des Abstands nicht ohne weiteres ablesbar ist.

Aus diesen Gründen kommen in dieser Arbeit Posen zum Einsatz. Eine Pose beschreibt die Translation sowie die anschließenden Rotationen in leicht verständlicher Form, durch sechs Parameter: $x, y, z, \Phi, \Theta, \Psi$. Zur Koordinatentransformation muss eine

Pose aber stets in die entsprechende HTM umgerechnet werden. Zu diesem Zweck existiert die Funktion *htm*$(\cdot)$, definiert als

$$\mathit{htm} : \mathcal{P} \mapsto \mathcal{H}, {}^{\mathsf{b}}\mathbf{P}_{\mathsf{a}} \mapsto {}^{\mathsf{b}}\mathbf{H}_{\mathsf{a}} = \mathbf{R}^{\mathbf{P}_{[\Phi]}} \cdot \mathbf{R}^{\mathbf{P}_{[\Theta]}} \cdot \mathbf{R}^{\mathbf{P}_{[\Psi]}} \cdot \mathbf{T}^{\mathbf{P}_{[x,y,z]}}. \tag{C.13}$$

Dabei beschreibt $\mathcal{P}$ die Menge aller Posen und $\mathbf{R}^{\mathbf{P}_{[\Phi]}}$, $\mathbf{R}^{\mathbf{P}_{[\Theta]}}$ und $\mathbf{R}^{\mathbf{P}_{[\Psi]}}$ die Rotationsmatrizen um die jeweiligen Winkel der Pose ${}^{\mathsf{b}}\mathbf{P}_{\mathsf{a}}$. Die Funktion *htm*$(\cdot)$ existiert zusätzlich in einer weiteren Variante, in der aus einem 6×1 Vektor bestehend aus den sechs Parametern $x, y, z, \Phi, \Theta, \Psi$ die entsprechende HTM berechnet wird:

$$\mathit{htm} : \mathbb{R}^{6\times 1} \mapsto \mathcal{H}, (x, y, z, \Phi, \Theta, \Psi)^T \mapsto {}^{\mathsf{b}}\mathbf{M}_{\mathsf{a}} = \mathbf{R}^{\Phi} \cdot \mathbf{R}^{\Theta} \cdot \mathbf{R}^{\Psi} \cdot \mathbf{T}^{x,y,z} \tag{C.14}$$

Umgekehrt existiert die inverse Funktion *pose*$(\cdot)$ zur Umwandlung von HTM in Posen

$$\mathit{pose} : \mathcal{H} \mapsto \mathcal{P}, {}^{\mathsf{b}}\mathbf{H}_{\mathsf{a}} \mapsto {}^{\mathsf{b}}\mathbf{P}_{\mathsf{a}}. \tag{C.15}$$

Die Funktion *pose*$(\cdot)$ existiert zusätzlich in einer weiteren Variante, in der aus einem 6×1 Vektor bestehend aus den sechs Parametern $x, y, z, \Phi, \Theta, \Psi$ die entsprechende Pose berechnet wird

$$\mathit{pose} : \mathbb{R}^{6\times 1} \mapsto \mathcal{P}, (x, y, z, \Phi, \Theta, \Psi)^T \mapsto {}^{\mathsf{b}}\mathbf{P}_{\mathsf{a}} = \mathit{pose}\left(\mathit{htm}\left((x, y, z, \Phi, \Theta, \Psi)^T\right)\right). \tag{C.16}$$

Aufgrund dieser Möglichkeiten zur Umrechnung zwischen Posen und HTM werden beide, sofern der Kontext es zulässt, im Rahmen der Arbeit auch synonym verwendet.

C.2.1 Zusätzliche Operationen mit Posen

In Gleichung (5.103) wird zwischen zwei zeitlich aufeinanderfolgenden Posen interpoliert und dazu jede der Posen skaliert und anschließend miteinander addiert. Die folgenden zwei zusätzlichen Posen-Operationen definieren die elementweise Multiplikation und Addition. Anzumerken ist hierbei, dass insbesondere die Addition keine Verkettung von Transformationen darstellt und nur zum Zwecke der Interpolation defniert wird.

Skalarprodukt Die Multiplikation einer Pose mit einem Skalar führt zur Multiplikation eines jeden Posenparameters. Winkelgrößen werden immer auf das Intervall $(-\pi, \pi]$ zurückgerechnet

$$\cdot : \mathcal{P} \times \mathbb{R} \mapsto \mathcal{P}, {}^{\mathsf{b}}\mathbf{P}_{\mathsf{a}} \cdot s \mapsto {}^{\mathsf{b}}\mathbf{P'}_{\mathsf{a}} = \mathit{pose} \left(\begin{pmatrix} {}^{\mathsf{b}}\mathbf{P}_{[x]\,\mathsf{a}} \\ {}^{\mathsf{b}}\mathbf{P}_{[y]\,\mathsf{a}} \\ {}^{\mathsf{b}}\mathbf{P}_{[z]\,\mathsf{a}} \\ {}^{\mathsf{b}}\mathbf{P}_{[\Phi]\,\mathsf{a}} \\ {}^{\mathsf{b}}\mathbf{P}_{[\Theta]\,\mathsf{a}} \\ {}^{\mathsf{b}}\mathbf{P}_{[\Psi]\,\mathsf{a}} \end{pmatrix} \cdot s \right). \tag{C.17}$$

Addition Die Addition zweier Posen führt zur parameterweisen Addition der jeweiligen Parameter. Winkelgrößen werden immer auf das Intervall $(-\pi, \pi]$ zurückgerechnet

$$\begin{aligned} + : \mathcal{P} \times \mathcal{P} \mapsto \mathcal{P}, {}^{\mathsf{b}}\mathbf{P}_{\mathsf{a}} + {}^{\mathsf{d}}\mathbf{P}_{\mathsf{c}} \mapsto {}^{\mathsf{f}}\mathbf{P}_{\mathsf{e}} = \\ \mathit{pose} \Big(\Big({}^{\mathsf{b}}\mathbf{P}_{[x]\,\mathsf{a}}, {}^{\mathsf{b}}\mathbf{P}_{[y]\,\mathsf{a}}, {}^{\mathsf{b}}\mathbf{P}_{[z]\,\mathsf{a}}, {}^{\mathsf{b}}\mathbf{P}_{[\Phi]\,\mathsf{a}}, {}^{\mathsf{b}}\mathbf{P}_{[\Theta]\,\mathsf{a}}, {}^{\mathsf{b}}\mathbf{P}_{[\Psi]\,\mathsf{a}} \Big)^T \\ + \Big({}^{\mathsf{d}}\mathbf{P}_{[x]\,\mathsf{c}}, {}^{\mathsf{d}}\mathbf{P}_{[y]\,\mathsf{c}}, {}^{\mathsf{d}}\mathbf{P}_{[z]\,\mathsf{c}}, {}^{\mathsf{d}}\mathbf{P}_{[\Phi]\,\mathsf{c}}, {}^{\mathsf{d}}\mathbf{P}_{[\Theta]\,\mathsf{c}}, {}^{\mathsf{d}}\mathbf{P}_{[\Psi]\,\mathsf{c}} \Big)^T \Big). \end{aligned} \tag{C.18}$$

Literaturverzeichnis

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., und Süsstrunk, S. (2012). SLIC Superpixels Compared to State-of-the-art Superpixel Methods. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, volume 34, Seiten 2274–2282.

ALLIED Vision Technologies GmbH (2006). *AVT Cameras - GUPPY F-036B / F-036C*. URL: 'http://www.1stvision.com/cameras/AVT/dataman/GuppyF_036B_C_WEB.pdf', Datum: 28.03.2016.

Andert, F. (2009). Drawing stereo disparity images into occupancy grids: Measurement model and fast implementation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Seiten 5191–5197.

Arulampalam, M. S., Maskell, S., Gordon, N., und Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188.

Badino, H., Franke, U., und Pfeiffer, D. (2009). The stixel world - A compact medium level representation of the 3d-world. In *Deutsche Arbeitsgemeinschaft für Mustererkennung e.V. (DAGM) Symposium*, volume 5748 LNCS, Seiten 51–60. Springer.

Baidu (2017). *Apollo: an open autonomous driving platform*. URL: 'https://github.com/ApolloAuto/apollo', Datum: 30.12.2021.

Basarke, C., Berger, C., Berger, K., Cornelsen, K., Doering, M., Effertz, J., Form, T., Gülke, T., Graefe, F., Hecker, P., Homeier, K., Klose, F., Lipski, C., Magnor, M., Morgenroth, J., Nothdurft, T., Ohl, S., Rauskolb, F., Rumpe, B., Schumacher, W., Wille, j. M., und Wolf, L. (2008). 2007 DARPA Urban Challenge Team CarOLO. Technical report, Technische Universität Braunschweig.

Bay, H., Tuytelaars, T., und Gool, L. J. V. (2006). SURF: Speeded Up Robust Features. In *European Conference on Computer Vision (ECCV)*, Seiten 404–417.

Bertalmio, M., Bertozzi, A., und Sapiro, G. (2001). Navier-stokes, fluid dynamics, and image and video inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, Kauai, HI, USA.

Bileschi, S. (2009). Fully automatic calibration of LIDAR and video streams from a vehicle. In *IEEE International Conference on Computer Vision (ICCV), Workshops*, Seiten 1457–1464. IEEE.

Bok, Y., Choi, D.-G., Vasseur, P., und Kweon, I. S. (2014). Extrinsic calibration of non-overlapping camera-laser system using structured environment. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Seiten 436–443. IEEE.

Borenstein, J. und Koren, Y. (1988). Obstacle Avoidance With Ultrasonic Sensors. *IEEE Journal of Robotics and Automation*, 4(2):213–218.

Borenstein, J. und Koren, Y. (1991). The vector field histogram–Fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288.

Bouguet, J.-Y. (2003). *Camera Calibration Toolbox for Matlab*. URL: 'http://www.vision.caltech.edu/bouguetj/calib_doc/', Datum: 10.04.2013.

Brandao, M., Ferreira, R., Hashimoto, K., Santos-Victor, J., und Takanishi, A. (2013). Integrating the whole cost-curve of stereo into occupancy grids. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Seiten 4681–4686.

Broggi, A., Buzzoni, M., Felisa, M., und Zani, P. (2011). Stereo obstacle detection in challenging environments: The VIAC experience. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Seiten 1599–1604.

Broggi, A., Cappalunga, A., Caraffi, C., Cattani, S., Ghidoni, S., Grisleri, P., Porta, P. P., Posterli, M., und Zani, P. (2010). Terramax vision at the urban challenge 2007. *IEEE Transactions on Intelligent Transportation Systems*, 11(1):194–205.

Broggi, A., Cattani, S., Porta, P. P., und Zani, P. (2006). A laserscanner-vision fusion system implemented on the terraMax autonomous vehicle. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Seiten 111–116.

Broggi, A., Cerri, P., Debattisti, S., Laghi, M. C., Medici, P., Molinari, D., Panciroli, M., und Prioletti, A. (2015). Proud — public road urban driverless-car test. *IEEE Transactions on Intelligent Transportation Systems (ITS)*, 16(6):3508–3519.

Brown, D. C. (1971). Close-range Camera Calibration. *Journal of Photogrammetric Engineering*, 37(8):855–866.

Calonder, M., Lepetit, V., Strecha, C., und Fua, P. (2010). BRIEF: Binary robust independent elementary features. In *European Conference on Computer Vision (ECCV)*, volume 6314 LNCS, Seiten 778–792. Springer.

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698.

Cappalunga, A., Cattani, S., Broggi, A., McDaniel, M. S., und Dutta, S. (2010). Real time 3D terrain elevation mapping using ants optimization algorithm and stereo vision. In *IEEE Intelligent Vehicles Symposium (IV)*, Seiten 902–909.

Chakravarthy, A. und Ghose, D. (1998). Obstacle avoidance in a dynamic environment: A collision cone approach. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, 28(5):562–574.

Chen, Y. S., Hung, Y. P., und Fuh, C. S. (2001). Fast block matching algorithm based on the winner-update strategy. *IEEE Transactions on Image Processing*, 10(8):1212–1222.

Chilian, A. und Hirschmüller, H. (2009). Stereo camera based navigation of mobile robots on rough terrain. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Seiten 4571–4576.

Coue, C., Pradalier, C., Laugier, C., Fraichard, T., und Bessiere, P. (2006). Bayesian Occupancy Filtering for Multitarget Tracking: An Automotive Application. *The International Journal of Robotics Research*, 25(1):19–30.

Daily, M., Medasani, S., Behringer, R., und Trivedi, M. (2017). Self-driving cars. *Computer*, 50(12):18–23.

Dang, T. (2009). *Kontinuierliche Selbstkalibrierung von Stereokameras*. Dissertation, Universität Karlsruhe (TH), Karlsruhe, Deutschland.

Dang, T., Hoffmann, C., und Stiller, C. (2009). Continuous stereo self-calibration by camera parameter tracking. *IEEE Transactions on Image Processing*, 18(7):1536–1550.

Deng, F., Hu, M., und Guan, H. Y. (2008). Automatic registration between lidar and digital images. In *ISPRS International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Seite 487 ff.

Dickmann, J., Appenrodt, N., und Brenk, C. (2013). Making Bertha See. In *IEEE International Conference on Computer Vision (ICCV), Workshops*, Seiten 214–221.

Dickmanns, E. D. (2007). *Dynamic vision for perception and control of motion*. Springer.

Dickmanns, E. D. und Zapp, A. (1987a). A Curvature-based Scheme for Improving Road Vehicle Guidance by Computer Vision. In *SPIE Mobile Robots*, volume 0727, Seiten 161–168, Cambridge, MA, USA.

Dickmanns, E. D. und Zapp, A. (1987b). Autonomous high speed road vehicle guidance by computer vision. In *IFAC World Congress*, Seiten 221–226, München, Deutschland.

Dong, H. und Barfoot, T. D. (2014). Lighting-invariant visual odometry using lidar intensity imagery and pose interpolation. In *Field and Service Robotics*, volume 92, Seiten 327–342. Springer.

Engel, J., Sturm, J., und Cremers, D. (2013). Semi-dense visual odometry for a monocular camera. In *IEEE International Conference on Computer Vision (ICCV)*, Seiten 1449–1456. IEEE.

Fassbender, D., Mueller, A., und Wuensche, H.-J. (2014). Trajectory planning for car-like robots in unknown, unstructured environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Seiten 3630–3635. IEEE.

Faugeras, O., Luong, Q.-T., und Papadopoulou, T. (2001). *The Geometry of Multiple Images: The Laws That Govern the Formation of Multiple Images of a Scene and Some of Their Applications*. MIT Press, Cambridge, MA, USA.

Felzenszwalb, P. F. und Huttenlocher, D. P. (2004). Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision*, 59(2):167–181.

Fleps, M., Mair, E., Ruepp, O., Suppa, M., und Burschka, D. (2011). Optimization based IMU camera calibration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Seiten 3297–3304.

Frickenstein, A., Vemparala, M. R., Nagaraja, N. S., und Unger, C. (2020). Method for recognizing a drivable area in the surroundings of a vehicle with the aid of a binary artificial neural network, computing device and driver assistance system. Patent DE102020105070A1, Bayerische Motoren Werke AG.

Fries, C. (2019). *Modellbasierte Fahrzeugerkennung eines Fahrerassistenzsystems zum autonomen Folgen im Konvoi*. Dissertation, Universität der Bundeswehr München, Fakultät für Luft- und Raumfahrttechnik, Neubiberg, Deutschland.

Fries, C. und Wuensche, H.-J. (2014). Monocular Template-based Vehicle Tracking for Autonomous Convoy Driving. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Seiten 2727–2732, Chicago, IL, USA.

Frueh, C., Sammon, R., und Zakhor, A. (2004). Automated texture mapping of 3D city models with oblique aerial imagery. In *International Symposium on 3D Data Processing, Visualization and Transmission*, Seiten 396–403.

Fryer, J. G. und Brown, D. C. (1986). Lens distortion for close-range photogrammetry. *Photogrammetric Engineering and Remote Sensing*, 52(1):51–58.

Garnett, N., Silberstein, S., Oron, S., Fetaya, E., Verner, U., Ayash, A., Goldner, V., Cohen, R., Horn, K., und Levi, D. (2017). Real-time category-based and general obstacle detection for autonomous driving. In *IEEE International Conference on Computer Vision (ICCV), Workshops*, Seiten 198–205.

Geiger, A., Moosmann, F., Car, O., und Schuster, B. (2012). Automatic camera and range sensor calibration using a single shot. In *IEEE International Conference on Robotics and Automation (ICRA)*, Seiten 3936–3943.

Geiger, A., Roser, M., und Urtasun, R. (2010). Efficient Large-Scale Stereo Matching. In *Asian Conference on Computer Vision (ACCV)*.

Geiger, A., Ziegler, J., und Stiller, C. (2011). StereoScan: Dense 3d reconstruction in real-time. In *IEEE Intelligent Vehicles Symposium (IV)*, Seiten 963–968, Baden-Baden, Deutschland.

Goebl, M., Althoff, M., Buss, M., Färber, G., Hecker, F., Heissing, B., Kraus, S., Nagel, R., Puente León, F., Rattei, F., Russ, M., Schweitzer, M., Thuy, M., Wang, C., und Wuensche, H. J. (2008). Design and capabilities of the Munich Cognitive Automobile. In *IEEE Intelligent Vehicles Symposium (IV)*, Seiten 1101–1107, Eindhoven, The Netherlands.

Goldberger, J., Gordon, S., und Greenspan, H. (2003). An efficient image similarity measure based on approximations of KL-divergence between two gaussian mixtures. In *IEEE International Conference on Computer Vision (ICCV)*, volume 1, Seiten 487–493.

Guindel, C., Beltrán, J., Martín, D., und García, F. (2017). Automatic extrinsic calibration for lidar-stereo vehicle sensor setups. In *IEEE International Conference on Intelligent Transportation Systems (ITS)*, Seiten 1–6.

Guo, C., Mirzaei, F. M., Roumeliotis, S., et al. (2012). An analytical least-squares solution to the odometer-camera extrinsic calibration problem. In *IEEE International Conference on Robotics and Automation (ICRA)*, Seiten 3962–3968. IEEE.

Guo, C., Mita, S., und McAllester, D. (2009). Stereovision-based road boundary detection for intelligent vehicles in challenging scenarios. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Seiten 1723–1728.

Habib, a. F., Ghanma, M. S., und Tait, M. (2004). Integration of lidar and photogrammetry for close range applications. In *ISPRS Congress Geo-Imagery Bridging Continents*, volume 35, Seite 170, Istanbul, Türkei.

Harris, C. und Stephens, M. (1988). A Combined Corner and Edge Detector. In *Alvey Vision Conference*, Seiten 147–151.

Hart, P. E., Nilsson, N. J., und Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107.

Hartley, R. und Kang, S. B. (2007). Parameter-free radial distortion correction with center of distortion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(8):1309–1321.

Hartley, R. und Zisserman, A. (2003). *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition.

Heikkilä, J. und Silvén, O. (1997). A Four-step Camera Calibration Procedure with Implicit Image Correction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Seiten 1106–1112, San Juan, Puerto Rico. IEEE Computer Society.

Heng, L., Li, B., und Pollefeys, M. (2013). CamOdoCal: Automatic intrinsic and extrinsic calibration of a rig with multiple generic cameras and odometry. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Seiten 1793–1800, Tokyo, Japan.

Himmelsbach, M. (2015). Abschlussbericht Phase I – Umgebungsmodell. Technical Report UniBwM / LRT / TAS / TR 2015:10, Universität der Bundeswehr München, Neubiberg.

Himmelsbach, M., Hundelshausen, F., und Wuensche, H. (2010). Fast segmentation of 3D point clouds for ground vehicles. In *IEEE Intelligent Vehicles Symposium (IV)*, Seiten 560–565.

Himmelsbach, M., Luettel, T., Hecker, F., von Hundelshausen, F., und Wuensche, H.-J. (2011a). Autonomous Off-Road Navigation for MuCAR-3 – Improving the Tentacles Approach: Integral Structures for Sensing and Motion. *Künstliche Intelligenz*, 25(2):145–149.

Himmelsbach, M., Schneider, S., und Wuensche, H.-J. (2011b). A Comparison of Error Metrics for Extrinsic Calibration and Fusion of Camera and Multi-Layer LIDAR. In *Signal Processing, Pattern Recognition and Applications (SPPRA)*, Seiten 61–68, Innsbruck, Austria.

Himmelsbach, M. und Wuensche, H. J. (2012). Tracking and classification of arbitrary objects with bottom-up/top-down detection. In *IEEE Intelligent Vehicles Symposium (IV)*, Seiten 577–582, Alcalá de Henares, Spain.

Hirschmüller, H. (2005). Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, Seiten 807–814.

Hirschmüller, H. (2008). Stereo Processing by Semiglobal Matching and Mutual Information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341.

Homm, F., Kaempchen, N., Ota, J., und Burschka, D. (2010). Efficient occupancy grid computation on the GPU with lidar and radar for road boundary detection. In *IEEE Intelligent Vehicles Symposium (IV)*, Seiten 1006–1013.

Hong, Y., Ren, G., und Liu, E. (2015). Non-iterative method for camera calibration. *Optics Express*, 23(18):23992–24003.

Huang, L. und Barth, M. (2009). A novel multi-planar LIDAR and computer vision calibration procedure using 2D patterns for automated navigation. In *IEEE Intelligent Vehicles Symposium (IV)*, Seiten 117–122.

Jensen, J. H., Ellis, D. P. W., Christensen, M. G., und Jensen, S. H. (2007). Evaluation of Distance Measures between Gaussian Mixture Models of MFCCs. In *International Conference on Music Information Retrieval (ISMIR)*, Seiten 107–108, Vienna, Austria. Austrian Computer Society.

Julier, S. (2002). The scaled unscented transformation. In *IEEE American Control Conference*, volume 6, Seiten 4555–4559. IEEE.

Kalman, R. (1960). A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82:35–45.

Kammel, S., Ziegler, J., Pitzer, B., Werling, M., Gindele, T., Jagzent, D., Schröder, J., Thuy, M., Goebl, M., von Hundelshausen, F., Pink, O., Frese, C., und Stiller, C. (2008). Team AnnieWAY's autonomous system for the 2007 DARPA Urban Challenge. *Journal of Field Robotics*, 25(9):615–639.

Kim, S. und Kim, J. (2012). Building occupancy maps with a mixture of Gaussian processes. In *IEEE International Conference on Robotics and Automation (ICRA)*, Seiten 4756–4761.

Kitagawa, G. (1996). Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models. *Journal of Computational and Graphical Statistics*, 5(1):1–25.

Knorr, M., Niehsen, W., und Stiller, C. (2013). Online extrinsic multi-camera calibration using ground plane induced homographies. In *IEEE Intelligent Vehicles Symposium (IV)*, Seiten 236–241.

Kwak, K., Huber, D. F., Badino, H., und Kanade, T. (2011). Extrinsic calibration of a single line scanning lidar and a camera. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Seiten 3283–3289.

Kämpchen, N. (2007). *Feature level fusion of laser scanner and video data for advanced driver assistance systems*. Dissertation, Universität Ulm, Fakultät für Ingenieurwissenschaften und Informatik, Ulm, Deutschland.

Lategahn, H., Derendarz, W., Graf, T., Kitt, B., und Effertz, J. (2010). Occupancy grid computation from dense stereo and sparse structure and motion points for automotive applications. In *IEEE Intelligent Vehicles Symposium (IV)*, Seiten 819–824.

Lébraly, P., Royer, E., Ait-Aider, O., Deymier, C., und Dhome, M. (2011). Fast calibration of embedded non-overlapping cameras. In *IEEE International Conference on Robotics and Automation (ICRA)*, Seiten 221–227.

Leibe, B., Cornelis, N., Cornelis, K., und Gool, L. V. (2007). Dynamic 3D Scene Analysis from a Moving Vehicle. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 07, Seiten 1–8.

Li, B., Heng, L., Koser, K., und Pollefeys, M. (2013). A multiple-camera system calibration toolbox using a feature descriptor-based calibration pattern. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Seiten 1301–1307, Tokyo, Japan.

Li, G., Gao, H., Sun, X., und Wang, J. (2020). Cross-camera obstacle tracking method, apparatus, device, system and medium. Patent EP3848851A1, Beijing Baidu Netcom Science and Technology Co Ltd.

Liu, F., Sparbert, J., und Stiller, C. (2008). IMMPDA vehicle tracking system using asynchronous sensor fusion of radar and vision. In *IEEE Intelligent Vehicles Symposium (IV)*, Seiten 168–173.

Lowe, D. G., Keypoints, S.-i., und Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110.

Lucas, B. und Kanade, T. (1981). An Iterative Image Registration Technique with an Application to Stereo Vision. In *International Joint Conference on Artifical Intelligence (IJCAI)*, volume 81, Seiten 674–679.

Luettel, T., Himmelsbach, M., Hundelshausen, F., Manz, M., Mueller, A., und Wuensche, H. J. (2009). Autonomous Offroad Navigation Under Poor GPS Conditions. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Workshops*, Seite 67, St. Louis, MO, USA.

Luettel, T., Himmelsbach, M., Manz, M., Mueller, A., von Hundelshausen, F., und Wuensche, H.-J. (2011). Combining Multiple Robot Behaviors for Complex Off-Road Missions. In *IEEE International Conference on Intelligent Transportation Systems (ITS)*, Seiten 674–680, Washington, DC, USA.

Lynen, S., Achtelik, M. W., Weiss, S., Chli, M., und Siegwart, R. (2013). A robust and modular multi-sensor fusion approach applied to mav navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Seiten 3923–3929. IEEE.

Maddern, W., Harrison, A., und Newman, P. (2012). Lost in translation (and rotation): Rapid extrinsic calibration for 2D and 3D LIDARs. In *IEEE International Conference on Robotics and Automation (ICRA)*, Seiten 3096–3102.

Maddern, W., Pascoe, G., Linegar, C., und Newman, P. (2017). 1 year, 1000 km: The oxford robotcar dataset. *The International Journal of Robotics Research*, 36(1):3–15.

Mahler, R. P. S. (2007). *Statistical Multisource-Multitarget Information Fusion*. Artech House, Inc., Norwood, MA, USA.

Mählisch, M. (2009). *Filtersynthese zur simultanen Minimierung von Existenz-, Assoziations- und Zustandsunsicherheiten in der Fahrzeugumfelderfassung mit heterogenen Sensordaten*. Dissertation, Fakultät für Ingenieurwissenschaften und Informatik Universität Ulm.

Malartre, F., Feraud, T., Debain, C., und Chapuis, R. (2009). Digital elevation map estimation by vision-lidar fusion. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Seiten 523–528.

Manz, M. (2013). *Modellbasierte visuelle Wahrnehmung zur autonomen Fahrzeugführung*. Dissertation, Universität der Bundeswehr München, Fakultät für Luft- und Raumfahrttechnik, Neubiberg, Deutschland.

Manz, M., Himmelsbach, M., Luettel, T., und Wuensche, H.-J. (2009). Fusing LIDAR and Vision for Autonomous Dirt Road Following Approach. In *Autonome Mobile Systeme*, Seiten 17–24, Karlsruhe, Deutschland. Springer.

Manz, M., Himmelsbach, M., Luettel, T., und Wuensche, H. J. (2011a). Detection and tracking of road networks in rural terrain by fusing vision and LIDAR. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, Seiten 4562–4568, San Francisco, CA, USA.

Manz, M., Luettel, T., von Hundelshausen, F., und Wuensche, H.-J. (2011b). Monocular Model-Based 3D Vehicle Tracking for Autonomous Vehicles in Unstructured Environment. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China.

Martinec, D. und Pajdla, T. (2007). Robust rotation and translation estimation in multiview reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Mazzei, L., Medici, P., und Panciroli, M. (2012). A lasers and cameras calibration procedure for VIAC multi-sensorized vehicles. In *IEEE Intelligent Vehicles Symposium (IV)*, Seiten 548–553.

McManus, C., Furgale, P., und Barfoot, T. D. (2011). Towards appearance-based methods for lidar sensors. In *IEEE International Conference on Robotics and Automation (ICRA)*, Seiten 1930–1935. IEEE.

Miksch, M., Yang, B., und Zimmermann, K. (2010). Automatic extrinsic camera self-calibration based on homography and epipolar geometry. In *IEEE Intelligent Vehicles Symposium (IV)*, Seiten 832–839.

Milella, A., Reina, G., Underwood, J., und Douillard, B. (2011). Combining radar and vision for self-supervised ground segmentation in outdoor environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Seiten 255–260.

Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., Haehnel, D., Hilden, T., Hoffmann, G., Huhnke, B., Johnston, D., Klumpp, S., Langer, D., Levandowski, A., Levinson, J., Marcil, J., Orenstein, D., Paefgen, J., Penny, I., Petrovskaya, A., Pflueger, M., Stanek, G., Stavens, D., Vogt, A., und Thrun, S. (2008). Junior: The Stanford entry in the Urban Challenge. *Journal of Field Robotics*, 25(9):569–597.

Motooka, K., Sugimoto, S., Okutomi, M., und Shima, T. (2014). Robust ground surface map generation using vehicle-mounted stereo camera. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Seiten 2741–2748. IEEE.

Münz, M., Mählisch, M., Dickmann, J., und Dietmayer, K. (2010). Probabilistic modeling of sensor properties in generic fusion systems for modern driver assistance systems. In *IEEE Intelligent Vehicles Symposium (IV)*, Seiten 760–765.

Munz, M., Mählisch, M., und Dietmayer, K. (2009). Probabilistische sensorfusion und integrierte existenzschätzung für zukünftige fahrerassistenzsysteme. In *Workshop Fahrerassistenzsysteme*, Seiten 166–176.

Musicki, D. und Evans, R. (2004). Joint integrated probabilistic data association-JIPDA. *IEEE Transactions On Aerospace And Electronic Systems*, 2(I):1120–1125.

Naikal, N., Zakhor, A., und Kua, J. (2009). Image Augmented Laser Scan Matching for Indoor Localization. Technical Report UCB/EECS-2009-35, EECS Department, University of California, Berkeley.

Nistér, D., Naroditsky, O., und Bergen, J. (2004). Visual Odometry. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, Seiten I—-652. IEEE.

Oskiper, T., Zhu, Z., Samarasekera, S., und Kumar, R. (2007). Visual odometry system using multiple stereo cameras and inertial measurement unit. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Oxford Technical Solutions Ltd. (2011). *RT3000: Hochpräzise GPS-gestützte Inertialmesssysteme*. URL: 'http://www.oxts.com/Downloads/Products/RT3000/RT3000_DE.pdf', Datum: 28.03.2016.

Pagel, F. (2010). Calibration of non-overlapping cameras in vehicles. In *IEEE Intelligent Vehicles Symposium (IV)*, Seiten 1178–1183.

Peng, Z., Yang, J., Chen, T.-H., und Ma, L. (2020). A first look at the integration of machine learning models in complex autonomous driving systems: a case study on apollo. In *28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, Seiten 1240–1250.

Pennecot, G., Morriss, Z., Lenius, S., Iordache, D. I., Gruver, D., Droz, P.-Y., Wachter, L., Ulrich, D., McCann, W., Pardhan, R., Fidric, B., Levandowski, A., und Avram, P. (2015). Vehicle with multiple light detection and ranging devices (LIDARs). Patent US9625582B2, Waymo LLC.

Perrollaz, M., Spalanzani, A., und Aubert, D. (2010). Probabilistic representation of the uncertainty of stereo-vision and application to obstacle detection. In *IEEE Intelligent Vehicles Symposium (IV)*, Seiten 313–318. IEEE.

Pfeiffer, D. und Franke, U. (2010). Efficient representation of traffic scenes by means of dynamic stixels. In *IEEE Intelligent Vehicles Symposium (IV)*, Seiten 217–224.

Point Grey Research, Inc. (2012). *Bumblebee - Stereo Vision Camera Systems*. URL: 'https://www.ptgrey.com/support/downloads/10132', Datum: 28.03.2016.

Richter, E., Schubert, R., und Wanielik, G. (2008). Radar and vision based data fusion - Advanced filtering techniques for a multi object vehicle tracking system. In *IEEE Intelligent Vehicles Symposium (IV)*, Seiten 120–125.

Ristic, B., Arulampalam, S., und Gordon, N. J. (2004). *Beyond the Kalman filter: Particle Filters for Tracking Applications*. Artech House.

Rojo, J., Rojas, R., Gunnarsson, K., Simon, M., Wiesel, F., Ruff, F., Wolter, L., Zilly, F., Santrac, N., Ganjineh, T., Sarkohi, A., Ulbrich, F., Latozky, D., Jankovic, B., Hohl, G., Wisspeintner, T., May, S., Pervölz, K., Nowak, W., Maurelli, F., Dröschel, D., und Berlin, T. (2007). Spirit of Berlin: An Autonomous Car for the DARPA Urban Challenge Hardware and Software Architecture.

Rosten, E. und Drummond, T. (2006). Machine Learning for High-Speed Corner Detection. In *European Conference on Computer Vision (ECCV)*, Seiten 430–443. Springer, Graz, Österreich.

Rublee, E., Rabaud, V., Konolige, K., und Bradski, G. (2011). ORB: An Efficient Alternative to SIFT or SURF. In *IEEE International Conference on Computer Vision (ICCV)*, Seiten 2564–2571. IEEE.

Sagawa, R., Osawa, N., Echigo, T., und Yagi, Y. (2005). Real time 3d environment modeling for a mobile robot by aligning range image sequence. In *British Machine Vision Conference (BMCV)*.

Sameer, Agarwal amd Snavely, N., Ian, S., Seitz, S. M., und Szeliski, R. (2009). Building Rome in a Day. In *International Conference on Computer Vision (ICCV)*.

Scaramuzza, D. und Siegwart, R. (2008). Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles. *IEEE Transactions on Robotics*, 24(5):1015–1026.

Schmid, M. R. (2012). *Umgebungserfassung für Fahrerassistenzsysteme mit hierarchischen Belegungskarten*. Dissertation, Universität der Bundeswehr München, Fakultät für Luft- und Raumfahrttechnik, Neubiberg, Deutschland.

Schneider, S., Himmelsbach, M., Luettel, T., und Wuensche, H. J. (2010). Fusing vision and LIDAR - Synchronization, correction and occlusion reasoning. In *IEEE Intelligent Vehicles Symposium (IV)*, Seiten 388–393, Kalifornien, USA.

Schneider, S., Luettel, T., und Wuensche, H. J. (2013). Odometry-based online extrinsic sensor calibration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Seiten 1287–1292, Tokyo, Japan.

Schneider, S., Mueller, G. R., Kallwies, J., und Wuensche, H. J. (2014). RAS: Recursive automotive stereo. In *IEEE Intelligent Vehicles Symposium (IV)*, Seiten 1282–1287, Dearborn, MI, USA.

Schueler, K., Weiherer, T., Bouzouraa, E., und Hofmann, U. (2012). 360 Degree multi sensor fusion for static and dynamic obstacles. In *IEEE Intelligent Vehicles Symposium (IV)*, Seiten 692–697.

Schweitzer, M. und Wuensche, H. J. (2009). Efficient keypoint matching for robot vision using GPUs. In *IEEE International Conference on Computer Vision (ICCV), Workshops*, Seiten 808–815.

Shade, R. und Newman, P. (2011). Choosing where to go: Complete 3D exploration with Stereo. In *IEEE International Conference on Robotics and Automation (ICRA)*, Seiten 2806–2811.

Stamos, I. und Allen, P. K. (2002). Geometry and Texture Recovery of Scenes of Large Scale. *Computer Vision and Image Understanding*, 88(2):94–118.

Steyer, S. J. (2021). *Grid-based object tracking*. Dissertation, Technische Universität München, München.

Strand, R. und Hayman, E. (2005). Correcting Radial Distortion by Circle Fitting. In *British Machine Vision Conference (BMCV)*, Seiten 9.1–9.10.

Strom, J., Richardson, A., und Olson, E. (2010). Graph-based segmentation for colored 3D laser point clouds. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Seiten 2131–2136. IEEE.

Suzuki, S., Raksincharoensak, P., Shimizu, I., Nagai, M., und Adomat, R. (2010). Sensor fusion-based pedestrian collision warning system with crosswalk detection. In *IEEE Intelligent Vehicles Symposium (IV)*, Seiten 355–360.

Tang, H. und Liu, Y. (2018). Automatic Simultaneous Extrinsic-Odometric Calibration for Camera-Odometry System. *IEEE Sensors Journal*, 18(1):348–355.

Tanzmeister, G. (2016). *Grid-based Environment Estimation for Local Autonomous Vehicle Navigation*. Dissertation, Technische Universität München, München.

Taylor, Z., Nieto, J., und Johnson, D. (2013). Automatic calibration of multi-modal sensor systems using a gradient orientation measure. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Seiten 1293–1300. IEEE.

Thrun, S. (2010). Toward robotic cars. *Communications of the ACM*, 53(4):99.

Thrun, S., Burgard, W., und Fox, D. (2005). *Probabilistic Robotics*. The MIT Press.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., und U.a., G. H. (2006). Stanley: The Robot that Won the DARPA Grand Challenge. *Journal of Field Robotics*, 23(9):661–692.

Toth, C. K., Ju, H., und Grejner-Brzezinska, D. A. (2010). Experiences with using SIFT for Mulitple Image Domain Matching. In *ASPRS Opportunities for Emerging Geospatial Technologies*, San Diego, CA, USA.

Uhlig, D. und Heizmann, M. (2021). A calibration method for the generalized imaging model with uncertain calibration target coordinates. In *15th Asian Conference on Computer Vision (ACCV)*, Kyōto, Japan.

Unterholzner, A. (2006). Kameraplattform MarVEye-7 Hardware Dokumentation. Technical Report UniBwM / LRT / TAS / IB 2006:1, Universität der Bundeswehr München, Neubiberg.

Unterholzner, A. (2007). MarVEye 7: Kameraplattform und Blickrichtungssteuerung. Technical Report UniBwM / LRT / TAS / IB 2007:2, Universität der Bundeswehr München, Neubiberg.

Unterholzner, A., Rohland, M., Schweitzer, M., und Wuensche, H. J. (2010). Vision-based online-calibration of inertial gaze stabilization. In *IEEE Intelligent Vehicles Symposium (IV)*, Seiten 646–651, San Diego, CA, USA.

Unterholzner, A. und Wuensche, H.-J. (2010). Hybrid adaptive control of an active multi-focal vision system. In *IEEE Intelligent Vehicles Symposium (IV)*, Seiten 534–539, San Diego, CA, USA.

Unterholzner, A. und Wuensche, H. J. (2013). Selective Attention for Detection and Tracking of Road-Networks in Autonomous Driving. In *IEEE Intelligent Vehicles Symposium (IV)*, Gold Coast, Queensland, Australia.

Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M. N., Dolan, J., Duggins, D., Galatali, T., Geyer, C., Gittleman, M., Harbaugh, S., Hebert, M., Howard, T. M., Kolski, S., Kelly, A., Likhachev, M., McNaughton, M., Miller, N., Peterson, K., Pilnick, B., Rajkumar, R., Rybski, P., Salesky, B., Seo, Y.-W., Singh, S., Snider, J., Stentz, A., Whittaker, W. R., Wolkowicki, Z., Ziglar, J., Bae, H., Brown, T., Demitrish, D., Litkouhi, B., Nickolaou, J., Sadekar, V., Zhang, W., Struble, J., Taylor, M., Darms, M., und Ferguson, D. (2008). Autonomous Driving In Urban Environments: Boss and the Urban Challenge. *Journal of Field Robotics*, 25(1):425–466.

Van der Merwe, R. und Wan, E. A. (2002). *The Unscented Kalman Filter*. Wiley, New York, New York, USA.

Vatavu, A., Danescu, R., und Nedevschi, S. (2012). Real-time dynamic environment perception in driving scenarios using difference fronts. In *IEEE Intelligent Vehicles Symposium (IV)*, Seiten 717–722.

Velodyne Acoustics, Inc. (2008). *HDL-64E - User's Manual*. URL: 'http://www.velodynelidar.com/lidar/products/manual/HDL-64E Manual.pdf', Datum: 28.03.2016.

Velodyne LiDAR, Inc. (2012). *HDL-64E S2 and S2.1 - User's Manual and Programming Guide*. URL: 'http://velodynelidar.com/lidar/products/manual/63-HDL64ES2h HDL-64E S2 CD HDL-64E S2 Users Manual.pdf', Datum: 28.03.2016.

von Hundelshausen, F., Himmelsbach, M., Hecker, F., Mueller, A., und Wuensche, H.-J. (2008). Driving with tentacles: Integral structures for sensing and motion. *Journal of Field Robotics*, 56(9):640–673.

Wan, E. und Merwe, R. V. D. (2000). The unscented Kalman filter for nonlinear estimation. In *IEEE Adaptive Systems for Signal Processing, Communications and Control Symposium*, Seiten 153–158. IEEE.

Wanielik, G., Neubert, U., und Lindner, P. (2008). CARAI - The Concept Vehicle of Chemnitz University of Technology for Advanced Driver Assistance Systems.

In *International Symposium on Wireless Personal Multimedia Communications, Workshop*.

Whittaker, W., Others, Urmson, C., Ragusa, C., Ray, D., Anhalt, J., Bartz, D., Galatali, T., Gutierrez, E., Johnston, J., Clark, M., Koon, P., Mosher, A., und Struble, J. (2006). A robust approach to high-speed navigation for unrehearsed desert terrain. *Journal of Field Robotics*, 23(8):467–508.

Wille, J. M., Saust, F., und Maurer, M. (2010). Stadtpilot: Driving autonomously on braunschweig's inner ring road. In *IEEE Intelligent Vehicles Symposium (IV)*, Seiten 506–511. IEEE.

Wu, S., Decker, S., Chang, P., Camus, T., und Eledath, J. (2009). Collision sensing by stereo vision and radar sensor fusion. *IEEE Transactions on Intelligent Transportation Systems (ITS)*, 10(4):606–614.

Wünsche, H.-J. (1988). *Bewegungssteuerung durch Rechnersehen: Ein Verfahren zur Erfassung und Steuerung räumlicher Bewegungsvorgänge in Echtzeit*. Fachberichte Messen, Steuern, Regeln Band 20. Springer.

Zhan, Q., Yubin, L., und Xiao, Y. (2009). Color-Based Segmentation of Point Clouds. In *ISPRS International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume XXXVIII, Seiten 248–252.

Zhang, Z. (1999). Flexible camera calibration by viewing a plane from unknown orientations. In *IEEE International Conference on Computer Vision (ICCV)*, volume 1, Seiten 666–673, Corfu, Greek.