Institut für Informatik VII
Robotik und Telematik

Masterarbeit

Autonome Navigation auf Basis eines visuellen SLAM-Systems

Darius Deubert

Dezember 2021

Erstgutachter: Prof. Dr. Andreas Nüchter
Zweitgutachter: Prof. Dr. Klaus Schilling

Zusammenfassung

Autonome mobile Roboter benötigen ein zuverlässiges Lokalisierungssystem, das im Idealfall auch eine Karte zur Pfadplanung und Kollisionsvermeidung zur Verfügung stellt. Das Ziel der vorliegenden Arbeit ist es, ein derartiges System zu entwickeln und durch umfangreiche Experimente und Messreihen zu evaluieren. Hierfür werden zunächst monokulare visual SLAM-Algorithmen auf dem Stand der Technik theoretisch und praktisch verglichen, um einen Algorithmus als Grundlage für die aufbauenden Funktionalitäten zu wählen. Das Schlüsselelement des entstehenden Gesamtsystems ist ein neu entwickeltes Verfahren, durch das mit Hilfe von visuellen Markern absolute Skaleninformationen aus einem monokularen Visual-SLAM-Algorithmus gewonnen werden. In einer ausgiebigen Messreihe zeigt sich eine hohe Zuverlässigkeit und eine durchschnittliche Ungenauigkeit von deutlich unter 10 cm. Vergleichsexperimente mit Odometrie und AMCL machen deutlich, dass das entwickelte Visual-SLAM-System in seiner Lokalisierungsgenauigkeit die weit verbreitete AMCL-Methode bei den Anwendungen im Rahmen dieser Arbeit übertrifft und bezüglich diverser Aspekte Vorteile bringt. Zusätzlich wird in dieser Arbeit ein Algorithmus entwickelt, der aus dem Visual-SLAM-Algorithmus eine hochaufgelöste zweidimensionale Occupancy Grid Map bestimmt. Somit dient das System auch als Grundlage für Pfadplanungsalgorithmen, wodurch insgesamt diverse Anforderungen an autonome mobile Roboter auf Basis eines visuellen SLAM-Systems erfüllt werden.

Danksagung

Zunächst danke ich Prof. Dr. Andreas Nüchter für das Ermöglichen, Betreuen und Begutachten dieser Arbeit. Des Weiteren gilt mein Dank David Bohlig, der mich während der gesamten Arbeit mit hilfreichen Beiträgen und konstruktiver Kritik unterstützt und die praktischen Tätigkeiten am Zentrum für Telematik betreut hat. Außerdem möchte ich Jan Richter für das Korrekturlesen der Arbeit danken.

Diese Arbeit wurde in Zusammenarbeit mit dem Zentrum für Telematik e.V. angefertigt.

Inhaltsverzeichnis

1 Einleitung **1**
- 1.1 Motivation . . . 1
- 1.2 Problembeschreibung . . . 2
- 1.3 Aufbau der Arbeit . . . 3
- 1.4 Wissenschaftlicher Beitrag . . . 3

2 Stand der Technik **5**
- 2.1 Visual Simultaneous Localization and Mapping . . . 5
  - 2.1.1 ORB-SLAM . . . 6
  - 2.1.2 ORB-SLAM2 und ORB-SLAM3 . . . 9
  - 2.1.3 LSD-SLAM . . . 9
- 2.2 AprilTags . . . 13

3 Grundlagen und Theorie **15**
- 3.1 Kamera Parameter und Kalibrierung . . . 15
  - 3.1.1 Extrinsische und Intrinsische Kamera Parameter . . . 15
  - 3.1.2 Nichtlineare Korrekturen . . . 19
  - 3.1.3 Kalibrierung nach Zhang . . . 21
  - 3.1.4 Kalibrierung der nichtlinearen Parameter . . . 22
- 3.2 Visuelle Features . . . 23
  - 3.2.1 SIFT . . . 23
  - 3.2.2 SURF . . . 24
  - 3.2.3 FAST Detektor . . . 25
  - 3.2.4 BRIEF Deskriptor . . . 25
  - 3.2.5 Bags of Visual Words . . . 25
- 3.3 Bundle Adjustment . . . 26
- 3.4 Perspective-n-Point Problem und Lösungsansätze . . . 27
  - 3.4.1 P3P . . . 28
  - 3.4.2 EPnP . . . 28
  - 3.4.3 IPPE . . . 30
- 3.5 ICP Algorithmus . . . 30
- 3.6 RANSAC . . . 31
- 3.7 Bresenham Algorithmus . . . 32

3.8 Occupancy Grid Mapping . . . 33
3.9 Kalmanfilter . . . 34
3.10 Adaptive Monte Carlo Localization . . . 36
3.11 Hand-Eye Kalibrierung . . . 36

4 Implementierung **39**
4.1 Robot Operating System . . . 39
4.1.1 Nodes . . . 39
4.1.2 Topics . . . 40
4.1.3 Bags . . . 40
4.1.4 Master . . . 40
4.1.5 roslaunch . . . 40
4.2 Visuelle Lokalisierung und Kartierung . . . 40
4.2.1 Auswahl der Algorithmen . . . 41
4.2.2 LSD-SLAM . . . 44
4.2.3 Anpassungen – LSD-SLAM . . . 45
4.2.4 ORB-SLAM3 . . . 47
4.3 Skalenkalibrierung vom SLAM- ins Weltkoordinatensystem . . . 48
4.3.1 Übersicht . . . 49
4.3.2 Lokalisierung mit AprilTags . . . 49
4.3.3 Bestimmung der Transformation . . . 50
4.3.4 RANSAC-basierter Ausreißerfilter . . . 52
4.4 Occupancy Grid Mapping . . . 53
4.5 Schnittstellen zum Navigationssystem des ZfT . . . 58
4.6 Resultierendes Gesamtsystem zur autonomen Navigation . . . 59
4.7 Ergänzender Kalmanfilter zur Fusion mit AMCL . . . 61

5 Experimente und Evaluierung **65**
5.1 Vergleich der vSLAM-Algorithmen . . . 65
5.1.1 Hardware . . . 65
5.1.2 Kamera Kalibrierung . . . 66
5.1.3 Evaluierung und Ergebnisse . . . 68
5.2 Skalenrekonstruktion und absolute Lokalisierung . . . 75
5.2.1 Anwendung des Systems zur Lokalisierung . . . 75
5.2.2 Versuchsaufbau der Messreihe zur Auswertung . . . 75
5.2.3 Hand-Eye Kalibrierung . . . 77
5.2.4 Analyse der Genauigkeit . . . 78

5.3 Occupancy Grid Map zur freien Navigation 84
5.3.1 Stärken und Schwächen des Verfahrens zum Occupany Grid Mapping . . 84
5.3.2 Versuchsaufbau und Messreihe zur Navigation 86
5.3.3 Evaluierung 87
5.4 Vergleich mit bisherigen Lokalisierungsmethoden 90
5.4.1 Bisherige Lokalisierungsmethoden und Kalmanfilter 90
5.4.2 Beschreibung des Vergleichsexperiments 90
5.4.3 Ergebnisse des experimentellen Vergleichs 91

6 Zusammenfassende Diskussion und Ausblick **97**

Anhänge **101**

A Messreihe zur Auswertung der absoluten Lokalisierung **103**

B Messreihe zur Auswertung der freien Navigation **107**

C Messreihe zum Vergleich verschiedener Lokalisierungsmethoden **109**

D Inhalt des beigefügten Datenträgers **145**

Abkürzungsverzeichnis

AMCL Adaptive Monte Carlo Localization

BRIEF Binary Robust Independent Elementary Features

DLT Direct Linear Transform

DoG Difference of Gaussians

DSO Direct Sparse Odometry

FAST Features from Accelerated Segment Test

ICP Iterative Closest Point

IMU Inertial Measurement Unit

LSD-SLAM Large-Scale Direct Monocular SLAM

OGM Occupancy Grid Maps

ORB Oriented FAST and Rotated BRIEF

PnP Perspective-n-Point

RANSAC Random Sample Consensus

RMS Quadratisches Mittel

ROS Robot Operating System

SfM Structure from Motion

SIFT Scale Invariant Feature Transform

SLAM Simultaneous Localization and Mapping

SSH Secure Shell

SURF Speeded Up Robust Features

SVD Singular Value Decomposition (Singulärwertzerlegung)

TGZ Technologie und Gründerzentrum

vSLAM Visual Simultaneous Localization and Mapping

ZfT Zentrum für Telematik

Kapitel 1

Einleitung

1.1 Motivation

Mit wissenschaftlichem und technischem Fortschritt geht zunehmende Digitalisierung und Automatisierung in der Industrie einher. Bei einer Befragung deutscher Industrieunternehmen ab 100 Mitarbeitern gaben 62% der Befragten an, bereits Anwendungen für Industrie 4.0 zu nutzen, wobei weitere 21% den Einsatz spezieller Anwendungen für Industrie 4.0 planen (Bitkom Research, 2021)[1]. Ein Schlüsselelement für die Industrie 4.0, besonders in Hinsicht auf Fortschritte im Logistikbereich, sind mobile Roboter. Diese fördern die Individualisierung von Fertigungsprozessen, sind dabei ressourceneffizient und mindern dadurch die Produktionskosten. Die *International Federation of Robotics* prognostiziert deshalb im Whitepaper „A Mobile Revolution"(Juni 2021) eine jährliche Absatzsteigerung von autonomen mobilen Robotern um 31% zwischen 2020 und 2023. Essenziell für die Anwendung mobiler Roboter ist ein zuverlässiges Lokalisierungssystem und Technologien zur Pfadplanung und Hindernisvermeidung.

Verglichen mit der Lokalisierung im Freien, bei der häufig GPS verwendet wird, werden an Lokalisierungssysteme für mobile Roboter in Industrieumgebungen andere Anforderungen gestellt. Zum einen ist GPS in Innenräumen oft schwer zu empfangen und zum anderen benötigen die Roboter eine höhere Genauigkeit, um in der Umgebung zu navigieren. Weit verbreitet sind LIDAR basierte Simultaneous Localization and Mapping (SLAM) Systeme, die Laserscans nutzen, um eine Karte sukzessive aufzubauen und sich darin zu lokalisieren. Bekannte LIDAR basierte SLAM-Algorithmen sind Gmapping [24, 25] und Hector SLAM [28]. Mit den erstellten Karten arbeitet beispielsweise die Adaptive Monte Carlo Localization (AMCL). LIDAR basierte Systeme profitieren von der hohen Genauigkeit der Laserscanner und bauen gleichzeitig Occupancy Grid Maps (OGM) auf, die sich hervorragend zur Pfadplanung und Hindernisvermeidung eignen. Nachteilig ist, dass eine initiale Posenschätzung vorgegeben werden muss und eine einzigartige geometrische Umgebung zur eindeutigen Lokalisierung vorausgesetzt wird. Zudem sind Laserscanner vergleichsweise teuer. Funktechnik setzt auf fest installierte Sender mit Technolo-

[1] https://de.statista.com/statistik/daten/studie/830769/umfrage/bedeutung-von-industrie-40-in-deutschland/

gien wie WLAN, RFID oder Bluetooth. Die Lokalisierung über WLAN ist besonders günstig, da die Sender in den Industrieumgebungen meist ohnehin vorhanden sind. Eine passende Verteilung der Sender für die Position ist allerdings nicht immer gegeben. RFID ist ebenfalls kostengünstig und kann leicht in einer bestehenden Umgebung installiert werden. Die vielseitige Einsetzbarkeit wurde bereits in einigen wissenschaftlichen Veröffentlichungen [14, 34, 43] unter Beweis gestellt. Auch Projektpartner des Zentrum für Telematik nutzen die genannten Technologien bereits.

Für die Lokalisierung werden immer häufiger auch visuelle Systeme erprobt. Entweder tracken fest installierte Kameras die Bewegung von Robotern oder sogenannte Structure from Motion (SfM) Algorithmen verarbeiten die Bilder einer Kamera, die auf dem Roboter angebracht ist. Letztere System sind häufig rechenintensiv und deshalb nicht für den Online-Einsatz auf Onboard-PC der Roboter geeignet, da diese oft nur über eine reduzierte Rechenleistung verfügen. Leichtgewichtiger und besser für die Lokalisierung eignen sich Visual-SLAM (vSLAM) Algorithmen. Diese verwenden eine oder mehrere Kameras auf dem Roboter, häufig zusammen mit zusätzlicher Sensorik wie IMUs oder Odometrie, um die Bewegung des Roboters zu bestimmen und darauf aufbauend eine topologische Karte mit Keyframes aufzubauen. Hierbei wird zwischen Feature-basierten Verfahren, bei denen markante Stellen in Bildfolgen miteinander abgeglichen werden um die Posenänderung zu bestimmen (vgl. ORB-SLAM [37]), und direkten Verfahren, wie beispielsweise LSD-SLAM [13], unterschieden, bei denen die Pose direkt über die Intensitäten der Pixel bestimmt wird.

1.2 Problembeschreibung

Es existieren bereits viele vSLAM-Algorithmen, die eine hohe Genauigkeit versprechen. Eine Lokalisierung mit absoluten Skaleninformationen ist allerdings nur mit zusätzlicher Sensorik oder speziellen Stereo- oder Tiefenkameras möglich. Zusätzliche Sensorik einzubinden ist aufwändig und benötigt eine kompliziertere Systemarchitektur. Stereo- und Tiefenkameras sind deutlich teurer als herkömmliche monokulare Kameras. Hinzu kommt, dass monokulare Kameras bereits auf vielen Roboterplattformen vorhanden sind und einfach nachzurüsten sind.

Für die autonome Navigation ist neben der Lokalisierung auch ein System zur Pfadplanung und Kollisionsvermeidung notwendig. Die existierenden vSLAM-Algorithmen dienen bisher allerdings nur zur Lokalisierung, da die erstellten Karten topologischer Natur sind und für Pfadplanung und Kollisionsvermeidung geometrische Karten benötigt werden. Xu et al. [51] haben ein System entwickelt, das auf dem vSLAM-Algorithmus ORB-SLAM2 [38] basiert, und eine Occupancy Grid Map erstellt, die zur Pfadplanung und Kollisionsvermeidung verwendet wird. Die Methode verwendet virtuelle Laserscans, die aus den Punktwolken einer Microsoft Kinect Tiefenkamera berechnet werden. Ein vergleichbares System, das mit einer monokularen Kamera auskommt, ist nicht bekannt.

Ein System zur autonomen Navigation, das lediglich mit einer monokularen Kamera auskommt, ist kostengünstig, da keine teure Sensorik benötigt wird, kann bei bestehenden Roboterplattformen leicht nachgerüstet werden und steht damit in Konkurrenz zu LIDAR basierten Systemen und sonstigen Technologien zur Lokalisierung in Innenräumen. Wie im vorherigen Abschnitt

deutlich wird, ist die Lokalisierung eine Schlüsseltechnologie bei autonomen mobilen Robotern, wodurch die Relevanz genannter Systeme in Zeiten von Industrie 4.0 erheblich ist.

1.3 Aufbau der Arbeit

Diese Arbeit hat zum Ziel, ein System zur autonomen Navigation auf Basis eines monokularen vSLAM-Algorithmus zu entwickeln, dessen Genauigkeit und Tauglichkeit abzuschätzen und das entwickelte System mit bestehenden Lokalisierungsverfahren zu vergleichen.

In dieser Arbeit werden in Kapitel 2 zunächst bestehende Technologien, insbesondere vSLAM-Algorithmen, analysiert, die die Grundlage für das entwickelte System bilden. Kapitel 3 behandelt die theoretischen Grundlagen, auf deren Basis das entwickelte System aufbaut. Die Schwerpunkte liegen auf der Photogrammetrie und Punktwolkenverarbeitung.

In 4. Kapitel wird das vSLAM-System implementiert. Dafür werden zunächst geeignete vSLAM-Algorithmen ausgewählt. Diese werden auf die neueste Version des Betriebssystems Ubuntu mit Langzeitunterstützung und die aktuelle Version, des Robot Operating Systems (ROS) angepasst. Anschließend wird eine Methode entwickelt, durch die absolute Skaleninformationen aus den vSLAM-Posen gewonnen werden. Dieser Schritt ist für die praktische Anwendung des Lokalisierungssystems unerlässlich. Darauf baut ein Algorithmus auf, der aus der SLAM-Karte eine skalentreue, zweidimensionale Occupancy Grid Map erstellt, die zur Pfadplanung und Hindernisvermeidung eingesetzt wird. Schließlich wird das entwickelte vSLAM-System in das bestehende Navigationssystem des Zentrums für Telematik integriert, wodurch im Ergebnis ein ganzheitliches System zur autonomen Navigation mobiler Roboter entsteht.

Auf das Implementierungskapitel folgt ein Experimentalkapitel, dessen Abschnitte verschiedene Aspekte des entwickelten Systems praktisch testen und auswerten. Kapitel 5.1 vergleicht die zuvor ausgewählten vSLAM-Algorithmen empirisch, indem diese praktisch angewandt und die Ergebnisse gegenübergestellt werden. Eine umfassende Messreihe evaluiert die Genauigkeit der Skalenrekonstruktion und gibt Aufschluss über die Faktoren der Ungenauigkeit (Kapitel 5.2). Die Eignung der Occupancy Grid Map, sowie deren Stärken und Schwächen werden in Abschnitt 5.3 behandelt. Schließlich stellt Kapitel 5.4 das entwickelte vSLAM-System dem bestehenden Lokalisierungssystem durch ein Vergleichsexperiment gegenüber, bei dem markante Trajektorien abgefahren werden.

Abschließend wird das Ergebnis zusammengefasst und diskutiert. Zudem werden diverse Möglichkeiten zur Verbesserung, Erweiterung und Anwendung des entwickelten Systems beschrieben.

1.4 Wissenschaftlicher Beitrag

Visuelle Lokalisierung gewinnt in der Robotik zunehmend an Bedeutung. Der wissenschaftliche Beitrag dieser Arbeit liegt in mehreren Aspekten. Die Arbeit liefert einen praktischen Vergleich der populären vSLAM-Algorithmen LSD-SLAM und ORB-SLAM3. Zudem wird eine neue

Methode entwickelt und ausgiebig evaluiert, die es ermöglicht aus einem vSLAM-Algorithmus ohne absolute Skaleninformationen skalentreue Posenschätzungen zu erhalten. Diese Methode dient als Schlüsselelement, um monokulare vSLAM-Systeme zur absoluten Lokalisierung zu verwenden. Das System wird im Vergleich zu AMCL und Odometrie eingeordnet und um einen Algorithmus zur Erstellung einer zweidimensionalen Occupancy Grid Map als Grundlage für Pfadplanung und Kollisionsvermeidung ergänzt.

Kapitel 2

Stand der Technik

Das folgende Kapitel führt in die bestehenden vSLAM-Algorithmen ein und analysiert deren Funktionsweise. Anschließend werden die AprilTags vorgestellt, die als Grundlage zur Fiducial Marker Localization dienen.

2.1 Visual Simultaneous Localization and Mapping

Visual Simultaneous Localization and Mapping (vSLAM) hat das Ziel, aus einer Bildfolge die Kamera-Trajektorie zu schätzen und gleichzeitig eine Karte der Umgebung aufzubauen und eine Punktwolke der Umgebung zu erzeugen. Dabei hat sich gezeigt, dass image to image matching Techniken, bei denen Bilder bei unbekannter Kamerapose mit Bildern bei bekannter Kamerapose verglichen werden, bei vSLAM wesentlich besser funktionieren als sogenannte map to map oder image to map Methoden.

vSLAM wurde in der Robotik zu Beginn vor allem zusammen mit zusätzlicher Sensorik, wie IMUs, Radencodern oder Laserscannern verwendet. Mittlerweile sind die Methodiken allerdings so weit ausgereift, dass vSLAM ohne Unterstützung zusätzlicher Informationen eine hohe Genauigkeit erreicht.

Die vSLAM-Algorithmen werden danach unterschieden, ob eine Kamera (monokular) oder mehrere Kameras (stereo) verwendet werden. Monokulare Systeme haben den Vorteil, dass die Verarbeitung der Daten leichtgewichtiger ist. Die Verwendung von mehreren Kameras als Stereo-System ist komplizierter und entsprechend rechenaufwändiger. Allerdings ist es mit Stereokameras im Gegensatz zu monokularen Kameras möglich nach entsprechender Kalibrierung ohne Weiteres Tiefeninformationen und damit absolute geometrische Informationen der Umgebung zu berechnen. Bei monokularen Systemen ist das nur erschwert möglich.

Außerdem sind vSLAM-Algorithmen in Feature-basiert und direkt unterteilt. **Feature-basierte Methoden** extrahieren Features aus dem Bild und berechnen daraus die Kamerapose, sowie die Geometrie der Umgebung als Funktion ausschließlich über die Features. Das Problem wird so

deutlich vereinfacht. Dennoch gehen Limitierungen in der Anwendung damit einher, da nur Features als Bildinformationen verwendet werden. Beispielsweise enthalten kollineare Features keine ausreichende Tiefeninformation, treten aber in der Praxis nicht selten bei menschengemachten Objekten auf.

Direkte Methoden verwenden im Gegensatz dazu die gesamte Bildinformation, um die Geometrie der Umgebung abzuleiten. Zu Beginn wurde diese Technologie vor allem mit Tiefen- und Stereo-Kameras genutzt, wobei mittlerweile auch robuste und genaue Algorithmen, wie beispielsweise LSD-SLAM [13], für monokulare Systeme existieren. Diese sind allerdings nur leichtgewichtig genug für eine CPU, falls auch hier nur Teile des gesamten Bilds verwendet werden (semi-dense depth filtering).

Durch die sinkenden Kosten und steigende Popularität von RGB-D Kameras, wie der Microsoft Kinect, wurden in den vergangenen Jahren auch vermehrt Algorithmen entwickelt, die Bilder mit dichten Tiefeninformationen verwenden, um daraus die Umgebung zu rekonstruieren. RGB-D Kameras haben den Vorteil, dass sie absolute Tiefeninformation über das ganze Bild liefern. Allerdings sind sie größtenteils für den Innenraum ausgelegt und können aufgrund ihrer Funktionsweise nicht für große Entfernungen und bei Sonnenlicht verwendet werden.

Diese Arbeit beschränkt sich aufgrund der Verfügbarkeit und des geringeren Rechen- und Entwicklungsaufwands auf monokulare Systeme.

2.1.1 ORB-SLAM

ORB-SLAM beschreibt ein monokulares, Feature-basiertes SLAM-System, das von Raul Mur-Artal, José M. M. Montiel und Juan D. Tardos entwickelt wurde [37]. Das System unterstützt Tracking, Mapping, Relokalisierung und Loop-Closing in Echtzeit und eignet sich deshalb hervorragend für Anwendungen der Robotik. Das System umfasst hierfür drei parallele Threads (siehe Abbildung 2.1): Beim Tracking wird die Kamerapose ermittelt, der Local-Mapping-Thread fügt neue Keyframes hinzu und führt lokale Optimierungen durch und beim Loop-Closing werden Schleifen erkannt, im Covisibility-Graph eingefügt und entsprechende Korrekturen durchgeführt.

Folgende Datenstrukturen werden beim ORB-SLAM verwendet:

- **Map-Points** enthalten ihre 3D Position im globalen Koordinatensystem, ihre Blickrichtung (Mittel der Blickrichtungen in Keyframes, die den Punkt enthalten), sowie einen ORB-Deskriptor und den Distanzbereich in dem das ORB-Feature als solches erkannt werden kann (aufgrund begrenzter Skaleninvarianz).
- **Keyframes** enthalten die Kamerapose bei Aufnahme des Frames, intrinsische Parameter der Kamera und die ORB-Features des Frames (teilweise assoziiert mit einem Map-Point).
- Der **Covisibility-Graph** ist ein ungerichteter, gewichteter Graph, dessen Knoten Keyframes sind, zwischen denen eine Kante existiert, falls die Keyframes mindestens 15 gemeinsame Map-Points beobachten. Die Anzahl der gemeinsamen Map-Points entspricht dem Kantengewicht.

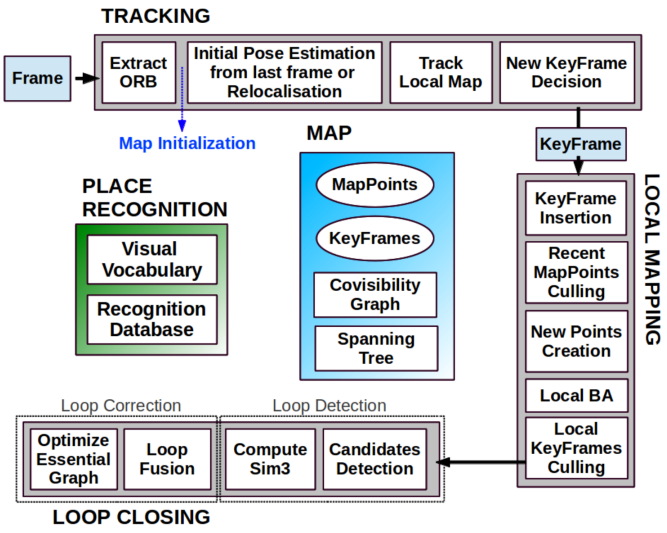


Abbildung 2.1: Überblick über die Threads und Funktionsweise von ORB-SLAM. [37]

- Zusätzlich wird der **Essential-Graph** verwendet, der alle Knoten und eine Teilmenge der Kanten des Covisibility-Graphs enthält. Er ermöglicht aufgrund der reduzierten Komplexität effizientere Optimierungen.

ORB-Feature Detektion

Für alle genannten Prozesse werden die selben Arten von Features, sogenannte ORB-Features (Oriented FAST and Rotated BRIEF), verwendet, die Echtzeitperformance ohne die Verwendung einer GPU garantieren sollen. Im Vergleich zu den weit verbreiteten SIFT (Abschnitt 3.2.1), SURF (Abschnitt 3.2.2) oder A-KAZE Features sind ORB-Features annähernd um zwei Größenordnungen schneller zu verarbeiten. Sie basieren auf dem FAST Detektor (Abschnitt 3.2.3) in Kombination mit dem BRIEF Deskriptor (Abschnitt 3.2.4). Hierbei werden skaleninvariante FAST Keypoints detektiert, deren Orientierung festgestellt und durch eine Abwandlung von BRIEF mit einem 256 Bit Deskriptor beschrieben. Hierdurch sind ORB-Features robust gegenüber der Beobachtungsperspektive und zudem schnell zu berechnen und zuzuordnen. [42]

Je nach Auflösung der Bilder werden bei ORB-SLAM 1000 bzw. 2000 FAST Punkte auf acht Skalierungsebenen extrahiert. Dabei werden die Bilder in Zellen aufgeteilt und in jeder Zelle der Schwellwert für die FAST Detektion so angepasst, dass fünf Punkte pro Zelle gefunden werden, zu denen anschließend die Deskriptoren berechnet werden.

Tracking

War das Tracking im letzten Frame erfolgreich, werden die Map-Points, die im letzten Frame beobachtet wurden, unter Zuhilfenahme eines Bewegungsmodells im neuen Frame gesucht. Auf

Grundlage der gefundenen Korrespondenzen mit bekannten Map-Points wird daraus die aktuelle Pose geschätzt. Die Initialisierung erfolgt hingegen durch die Berechnung der Homographie und der Fundamentalmatrix aus initial aufgenommenen Bildern. War das Tracking im letzten Frame nicht erfolgreich, wird das aktuelle Frame durch eine angepasste Version der Bags of Words Place Recognition [20] (siehe Abschnitt 3.2.5) global relokalisiert.

Sobald die erste Schätzung der neuen Pose erfolgt ist, wird eine lokale Karte herangezogen, die aus den Keyframes, die Map-Points des aktuellen Frames enthalten, sowie deren Nachbarn im Covisibility-Graph besteht. Alle in der lokalen Karte enthaltenen Map-Points werden im aktuellen Frame gesucht und miteinander assoziiert. Mit den gefundenen Map-Points wird die Pose zusätzlich durch Bundle Adjustment (vgl. Abschnitt 3.3) optimiert.

Im letzten Schritt des Tracking wird entschieden, ob das aktuelle Frame als Keyframe infrage kommt. Grundsätzlich sollen eher zu viele Frames als Keyframe aufgenommen werden, da überflüssige Keyframes später aus der Karte entfernt werden. Entscheidend ist eine gute Kenntnis der aktuellen Pose und eine passende Anzahl an gefundenen Map-Points.

Local Mapping

Wurde ein neues Keyframe gefunden, wird es zunächst in den Covisibility-Graph eingefügt und die Bags of Words Repräsentation berechnet. Durch Zuordnung der ORB-Features zu im Covisibility-Graph benachbarten Keyframes werden neue Map-Points gefunden und hinzugefügt. Um lokale Konsistenz zu erreichen, wird Bundle Adjustment (vgl. Abschnitt 3.3) mit den Nachbarn des neuen Keyframes im Covisibility-Graph ausgeführt und Ausreißer werden entfernt.

Map-Points werden entfernt, falls sie in weniger als drei weiteren Keyframes erkannt werden. Keyframes werden entfernt, falls 90% der enthaltenen Map-Points in mindestens drei anderen Keyframes enthalten sind. So wird versucht die Anzahl an redundanten Informationen gering zu halten.

Loop-Closing

Ein bedeutsamer Faktor für globale Konsistenz und die Korrektur von Drift ist Loop-Closing. Im Falle von ORB-SLAM werden hierfür für das letzte Keyframe mit der Bags of Words Repräsentation ähnliche Bilder als Kandidaten zur Loop-Fusion gesucht, die durch verschiedene Methoden verifiziert werden. Wurden Korrespondenzen gefunden, werden neue Kanten im Covisibility-Graph eingefügt und Map-Point Duplikate zusammengeführt. Schließlich wird die sogenannte Pose-Graph-Optimization durchgeführt, bei der die Posen der Keyframes und die Map-Points entsprechend der Abweichung bei der Schließung der Schleife korrigiert werden.

2.1.2 ORB-SLAM2 und ORB-SLAM3

ORB-SLAM2 ist die Nachfolgeversion von ORB-SLAM, die 2017 von R. Mur-Artal und J. D. Tardos veröffentlicht wurde [38]. Die Erweiterung umfasst die Unterstützung von Stereo- und RGB-D-SLAM und ein verbessertes Vokabular für die Bags of Words Place Recognition. Zudem wird bei ORB-SLAM2 neben dem lokalen Bundle Adjustement bei jeder Loop-Closure ein „Full“ Bundle Adjustment durchgeführt, bei dem alle Keyframes einbezogen werden.

Die Änderungen zu **ORB-SLAM3** [7] betreffen vor allem die Integration von inertialen Messungen (sog. visual inertial SLAM) auf Grundlage eines Maximum-a-Posteriori (MAP) Schätzers, die eine erhöhte Genauigkeit und absolute Skaleninformation verspricht. Eine zusätzliche Neuerung, die für diese Arbeit von weitaus größerer Bedeutung ist, ist die Einführung des Multi-Map-Systems ORB-SLAM Atlas [11], durch das im Falle von fehlgeschlagenem Tracking eine neue Teilkarte erstellt wird, die bei einer Loop-Closure in die bestehende Karte integriert wird. Bei ORB-SLAM oder alternativen vSLAM-Algorithmen wird die gesammelte Information bis zur erfolgreichen Relokalisierung nicht verwendet. Dieses System verbessert besonders die Kartierung in Umgebungen mit unzureichenden visuellen Informationen (z.B. große gleichfarbige Flächen). Weitere Neuerungen sind eine Verbesserung der Bags of Words Place Recognition und eine abstrakte Kamera Repräsentation, die es ermöglicht verschiedene Kameramodelle zu implementieren.

2.1.3 LSD-SLAM

Large-Scale Direct Monocular SLAM (LSD-SLAM) ist ein direkter SLAM-Algorithmus, der von Engel et al. [13] entwickelt wurde und wie auch ORB-SLAM in Echtzeit eine Karte bestehend aus Keyframes aufbaut. Die Keyframes sind hierbei durch Ähnlichkeitstransformationen der Lie-Gruppe $\mathfrak{sim}(3)$ verbunden, die eine Kombination aus Rotation, Translation und Skalierung beschreibt. Zusätzlich werden stochastische Informationen über die Tiefenschätzung miteinbezogen.

Der Algorithmus besteht insgesamt aus drei Komponenten (siehe Abbildung 2.2): Beim **Tracking** wird die aktuelle Pose relativ zum aktuellen Keyframe bestimmt. Die **Tiefenschätzung** ermittelt anhand von Pixel-basierten Vergleichen Tiefeninformationen, die entweder dazu verwendet werden, das aktuelle Keyframe weiter zu verbessern, oder ein neues Keyframe zu erstellen. Keyframes, die nicht weiter verfeinert werden, werden von der **Map-Optimization** Komponente in die globale Karte integriert. Außerdem erkennt die Komponente Schleifen und führt darauf basierend Optimierungen durch.

Die Karte wird als Pose-Graph von Keyframes dargestellt. Jedem Keyframe $\mathcal{K}_i$ ist ein Kamerabild $I_i : \Omega_i \to \mathbb{R}$ mit Intensitäten für jeden Pixel zugeordnet. Für eine Teilmenge Ω_{Di} aller Pixel, die in Bereichen mit ausreichend hohem Intensitätsgradienten liegen, werden zusätzlich die inverse Tiefe $D_i : \Omega_{Di} \to \mathbb{R}^+$ und deren Varianz $V_i : \Omega_{Di} \to \mathbb{R}^+$ gespeichert. Die Kanten ε_{ji} des Pose-Graphs enthalten die relative Ausrichtung $\xi_{ji} \in \mathfrak{sim}(3)$ der Keyframes zueinander und deren zugehörige Kovarianzmatrix Σ_{ji}.

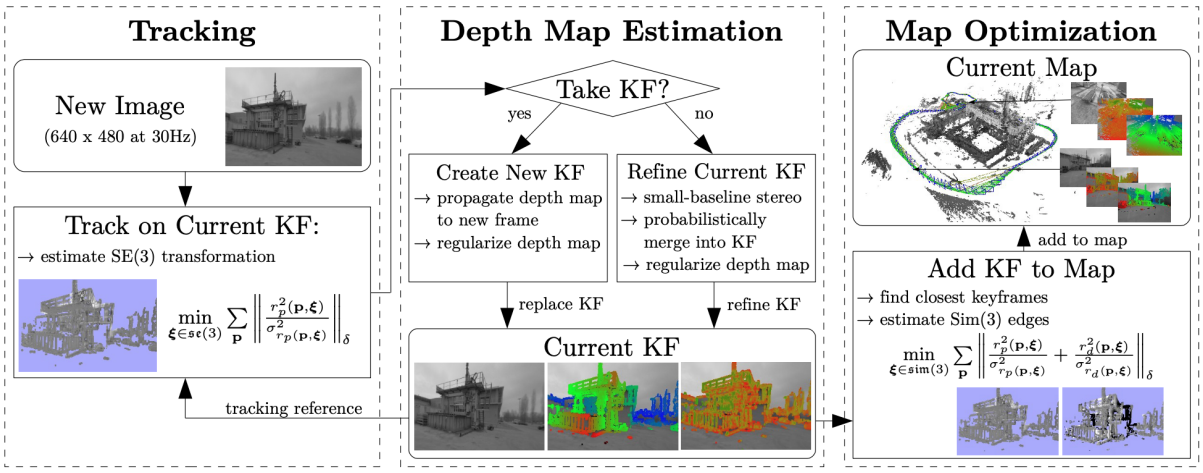


Abbildung 2.2: Überblick über die Komponenten und Funktionsweise von LSD-SLAM. [13]

Tracking

Die relative Pose $\xi_{ji} \in \mathfrak{se}(3)$ (Rotation und Translation, ohne Skalierung) eines neuen Bilds I_j zu einem Keyframe $\mathcal{K}_i = (I_i, D_i, V_i)$ wird bei LSD-SLAM über den photogrammetrischen Fehler E_p bestimmt, indem dieser nach ξ_{ji} minimiert wird:

$$E_p(\xi_{ji}) = \sum_{p \in \Omega_{D_i}} \left\| \frac{r_p^2(p, \xi_{ji})}{\sigma^2_{r_p(p,\xi_{ji})}} \right\|_\delta , \tag{2.1}$$

$$\text{mit } r_p(p, \xi_{ji}) := I_i(p) - I_j(\omega(p, D_i(p), \xi_{ji})) , \tag{2.2}$$

wobei $\omega(p, D_i(p), \xi)$ den Punkt p und seine inverse Tiefe in ein Kameraframe, dass mit ξ Transformiert wurde, projiziert und $\| \cdot \|_\delta$ die Huber Loss Funktion bezeichnet. Die Varianz $\sigma^2_{r_p(p,\xi_{ji})}$ wird unter Annahme eines Gauß-verteilten Rauschens σ_I^2 anhand der Varianz der inversen Tiefe im Bild V_i bestimmt:

$$\sigma^2_{r_p(p,\xi_{ji})} := 2\sigma_I^2 + \left(\frac{\partial r_p(p, \xi_{ji})}{\partial D_i(p)} \right)^2 V_i(p) . \tag{2.3}$$

Die Minimierung des photogrammetrischen Fehlers erfolgt durch ein gewichtetes Gauß-Newton Verfahren, das ein Maximum-Likelihood Schätzer für ξ unter der Annahme von unabhängig und gleich Gauß-verteilten Variablen ist. Die Grundidee des Verfahrens besteht darin, die Fehlerfunktion zu linearisieren und, um robust gegenüber Ausreißern (durch Reflexionen etc.) zu sein, die linearisierte Fehlerfunktion passend zu gewichten und zu minimieren.

Die Schätzung der Transformation ξ ohne Gewichtung wird berechnet, indem eine initiale Schätzung $\xi^{(0)}$ schrittweise mit einer Änderung $\delta\xi^{(n)}$ linksseitig konkateniert wird:

$$\xi^{(n+1)} = \delta\xi^{(n)} \circ \xi^{(n)} \tag{2.4}$$

Die Änderung $\delta\xi^{(n)}$ wird hierbei mit dem Gauß-Newton Verfahren bestimmt:

$$\delta\xi^{(n)} = -(J^T J)^{-1} J^T \mathbf{r}\left(\xi^{(n)}\right) \quad \text{, mit } J = \left.\frac{\partial \mathbf{r}\left(\epsilon \circ \xi^{(n)}\right)}{\partial \epsilon}\right|_{\epsilon=0} \text{ und } \mathbf{r} = (r_1, \ldots, r_n)^T \,. \tag{2.5}$$

Um Ausreißer in der Fehlerfunktion weniger stark einzubeziehen, wird in jeder Iteration eine Gewichtungsmatrix W berechnet, die Ausreißer weniger stark gewichtet.

$$\delta\xi^{(n)} = -(J^T W J)^{-1} J^T W \mathbf{r}\left(\xi^{(n)}\right) \tag{2.6}$$

Aus den Matrizen W und J der letzten Iteration wird die Kovarianzmatrix Σ_ξ geschätzt.

Man beachte, dass die Transformationen $\xi \in \mathfrak{sim}(3)$ bei LSD-SLAM linksseitig konkateniert werden. Konsistente rechtsseitige Multiplikation ist ebenfalls möglich, wobei jeweils auf die Vereinbarkeit von verschiedenen Implementierungen zu achten ist.

Eine elementare Verbesserung gegenüber vorherigen direkten Methoden liegt darin, die Varianz der Tiefeninformationen miteinzubeziehen und nur eine Teilmenge aller Pixel zu verwenden.

Tiefenschätzung

Die Wahl, ein neues Keyframe zu erstellen, erfolgt durch einen gewichteten Wert aus der Distanz und den Winkeln relativ zum aktuellen Keyframe:

$$\text{dist}(\xi_{ji}) := \xi_{ji}^T W \xi_{ji} \,, \tag{2.7}$$

wobei W eine Diagonalmatrix für die Gewichtung ist. Überschreitet $\text{dist}(\xi_{ji})$ einen bestimmten Schwellwert, wird aus dem zuletzt aufgenommenen und getrackten Frame ein neues Keyframe.

Hierzu wird die Tiefeninformation des neuen Keyframes, zunächst nur anhand des vorherigen Keyframes, initialisiert. Der Ansatz hierzu wurde bereits in einer früheren Veröffentlichung von Engel et al. [12] erläutert. Um Pixel des neuen Keyframes denen des vorherigen Keyframes zuzuordnen, wird entlang der epipolaren Linien nach der Intensität gesucht und anschließend eine genaue Sub-Pixel Lokalisierung durchgeführt. Das Suchintervall legt die Tiefeninformation des vorherigen Keyframes als $d \pm 2\sigma_d$ fest. Die Varianz der inversen Tiefen wird mit dem photogrammetrischen Fehler, dem geometrischen Fehler und einem Faktor, der von intrinsischen Parametern und der Translation der Kamera abhängig ist, abgeschätzt. Für jeden Pixel werden unabhängige Tiefenschätzungen erhalten.

Anschließend wird eine Iteration räumlicher Regularization durchgeführt, bei der die inversen Tiefen geglättet werden, indem für jeden Pixel der gewichtete Mittelwert (nach inverser Varianz) der benachbarten inversen Tiefen bestimmt wird. Um geometrische Kanten zu berücksichtigen, wird nicht geglättet falls sich zwei Tiefenwerte um mehr als 2σ unterscheiden. Zusätzlich werden Ausreißer gefiltert, indem für jeden Pixel eine Wahrscheinlichkeit betrachtet wird, zu der es sich um einen Ausreißer handelt. Diese Wahrscheinlichkeit wird jeweils erhöht, falls keine

Korrespondenz im Vergleichsframe gefunden wird, und bei jeder gefundenen Korrespondenz erniedrigt. Steigt die Wahrscheinlichkeit über einen bestimmten Schwellenwert, wird das Pixel zurückgesetzt.

Nachdem die Tiefeninformationen um einen Mittelwert von eins skaliert werden, wird das neue Keyframe zum Tracking verwendet. Die Skalierung wird in der zugeordneten Kamerapose $\xi \in \mathfrak{sim}(3)$ berücksichtigt.

Das Keyframe wird von den folgenden Frames, solange diese keine Keyframes werden, nach einem ähnlichen Verfahren verbessert (vgl. Engel et al. [12]). Dazu werden die Tiefenschätzungen der aktuellen Frames in die Tiefen im Keyframe umgerechnet, wobei die Rotation zwischen den Frames als klein angenommen wird und demnach nur die Translation entlang der optischen Achse einbezogen werden. Die Pixel werden entweder mit der beobachteten Tiefe und Varianz initialisiert oder, im Falle einer bestehenden Hypothese, entsprechend der vorherigen Schätzung und deren Varianz mit der neuen Schätzung verrechnet. Diese Fusion, bei der die Verteilungsfunktionen berücksichtigt werden, ist vergleichbar mit dem Korrekturschritt eines Kalmanfilters. Sind zwei Schätzungen statistisch nicht miteinander vereinbar, wird die niedrigere Tiefe und deren Varianz übernommen.

Image-Alignment in $\mathfrak{sim}(3)$

Ein Problem bei monokularem SLAM ist, dass keine absoluten Skaleninformationen abgeleitet werden können. Da deshalb nur eine relative Skalierung zwischen Frames bestimmt werden kann, führt das über lange Trajektorien zu einem Drift der Skalierung, was wiederum in Problemen bei Verfahren resultiert, die absolute Distanzen verwenden.

Wie zuvor beschrieben, wird die Skalierung in die Kameraposen $\xi \in \mathfrak{sim}(3)$ miteinbezogen, wodurch bei Loop-Closures der Skalendrift bestimmt und korrigiert wird. Dafür ist zunächst Image-Alignment mit relativen Posen $\xi \in \mathfrak{sim}(3)$ notwendig. LSD-SLAM verwendet hierzu - im Vergleich zum Image-Alignment in $\mathfrak{se}(3)$ beim Tracking - neben dem photogrammetrischen Fehler r_p auch den Tiefenfehler r_d, um die relative Skalierung zu schätzen. Die Fehlerfunktion wird dann zu

$$E(\xi_{ji}) = \sum_{p \in \Omega_{D_i}} \left\| \frac{r_p^2(p, \xi_{ji})}{\sigma^2_{r_p(p,\xi_{ji})}} + \frac{r_d^2(p, \xi_{ji})}{\sigma^2_{r_d(p,\xi_{ji})}} \right\|_\delta , \tag{2.8}$$

$$\text{mit } r_d(p, \xi_{ji}) := [p']_3 - D_j([p']_{1,2}) \tag{2.9}$$

$$\text{und } \sigma^2_{r_d(p,\xi_{ji})} := V_j([p']_{1,2}) \left(\frac{r_d(p, \xi_{ji})}{\partial D_j([p']_{1,2})} \right)^2 + V_i(p) \left(\frac{r_d(p, \xi_{ji})}{\partial D_i(p)} \right)^2 , \tag{2.10}$$

wobei $p' := \omega_s(p, D_i(p), \xi_{ji})$ den transformierten Punkt p bezeichnet. Der photogrammetrische Fehler r_p^2 und dessen Varianz $\sigma^2_{r_p}$ folgt den Gleichungen (2.2) und (2.3).

Die Minimierung der Fehlerfunktion nach ξ_{ji} erfolgt analog zum Image-Alignment in $\mathfrak{se}(3)$ durch das iterative Gauß-Newton Verfahren. Es ist zu beachten, dass $\mathfrak{sim}(3) \subset \mathbb{R}^7$ ist, wohingegen $\mathfrak{se}(3) \subset \mathbb{R}^6$ gilt. Der Rechenaufwand steigt hierdurch allerdings nur geringfügig.

Abbildung 2.3: Apriltag (ID: 0) der Familie 36h11. [1]

Abhängigkeitszuordnung (Constraint-Search)

Die nächsten zehn Frames, sowie Kandidaten aus einem appearance-basierten Algorithmus der Toolbox OpenFABMAP [23], werden als Kandidaten für Loop-Closures herangezogen. Die Kandidaten werden zusätzlich gefiltert, indem überprüft wird, ob deren gegenseitige relative Posen, die unabhängig bestimmt werden, statistisch miteinander vereinbar sind. Zu bemerken ist, dass OpenFABMAP Feature-basiert ist und LSD-SLAM deshalb SURF-Features verwendet und somit kein vollständig direkter Algorithmus ist.

Map-Optimization

Die Karte wird ständig durch Pose-Graph-Optimization mit folgender Fehlerfunktion minimiert:

$$E(\xi_{W1}, \ldots, \xi_{Wn}) := \sum_{(\xi_{ji}, \Sigma_{ji}) \in \mathcal{E}} \left(\xi_{ji} \circ \xi_{Wi}^{-1} \circ \xi_{Wj}\right)^T \Sigma_{ji}^{-1} \left(\xi_{ji} \circ \xi_{Wi}^{-1} \circ \xi_{Wj}\right) \tag{2.11}$$

wobei das Subskript Wi für die Pose des Keyframes i im globalen Koordinatensystem und $\mathcal{E}$ für die Menge aller Kanten zwischen den Keyframes steht. Zur Optimierung wird das G2o Framework von Kümmerle et al. [30] verwendet.

2.2 AprilTags

AprilTags sind zweidimensionale Barcodes, die an der University of Michigan entwickelt wurden [39, 50]. Im Gegensatz zu bekannten zweidimensionalen Barcodes, wie QR-Codes, enthalten die Codes nur wenige Bits an Informationen. AprilTags sind für eine robuste Detektion und hohe Lokalisierungsgenauigkeit ausgelegt. Deshalb eignen sie sich hervorragend, um auf deren Basis ein PnP Problem zur Ermittlung der Kamerapose zu formulieren.

Es existieren verschiedene Familien der AprilTags, die jeweils aus einer festen Menge an Tags mit jeweils einzigartiger ID bestehen. Ein beispielhaftes Tag, der in dieser Arbeit verwendeten Familie 36h11 ist in Abbildung 2.3 zu sehen. Die Tags der Familie 36h11 bestehen aus einem ein

[1] https://april.eecs.umich.edu/software/apriltag

Bit breitem weißen und einem ein Bit breitem schwarzen Rahmen. Darin befinden sich die 6×6 Datenbits, die die ID des Tags angeben. Da eine Hemmingdistanz von 11 gewahrt wird, sind IDs im Bereich 0 bis 586 verfügbar.

Um die relative Pose zu den Tags zu bestimmen, werden häufig sogenannte Bundles, das heißt Anordnungen mehrerer unterschiedlicher Tags einer Familie, verwendet. Das hat den Vorteil, dass nicht alle Tags erkannt werden müssen und dass die Genauigkeit der ermittelten Pose deutlich erhöht wird.

Kapitel 3

Grundlagen und Theorie

In diesem Kapitel werden die theoretischen Grundlagen, auf denen das entwickelte System aufbaut, zusammengefasst. Hierbei werden zunächst die wichtigsten Inhalte und Methodiken der Photogrammetrie, wie Kameramodelle, visuelle Features und Bundle Adjustment, behandelt. Hinzu kommen Methoden der Punktwolkenverarbeitung, algorithmische und mathematische Grundlagen zum Erstellen von Occupancy Grid Maps und Technologien der Robotik.

3.1 Kamera Parameter und Kalibrierung

Um aus den Informationen eines Bilds geometrische Informationen der Umgebung zu gewinnen, ist es notwendig, das Kamerasystem entsprechend zu modellieren. Die Parameter des gewählten Modells werden anschließend durch eine geeignete Kalibrierung ermittelt.

In den folgenden Abschnitten werden sogenannte Central Cameras betrachtet. Diese haben einen einzigen Blickpunkt und demnach ein zentrales Projektionsmodell. Central Cameras werden wiederum in Spherical Cameras und Perspective Cameras unterteilt. Erstere nutzen optische Systeme und sehen damit in mehrere/alle Richtungen. Perspective Cameras verwenden dagegen nur einen planaren Sensor und erhalten deshalb gerade Linien (engl.: *straight line preserving*). Reale Kameras sind oft von zusätzlichen Abweichungen betroffen, die durch die Modelle nicht exakt abgedeckt sind. Die angewandten Korrekturen minimieren die Abweichungen jedoch bereits deutlich.

3.1.1 Extrinsische und Intrinsische Kamera Parameter

Kameramodelle variieren in der Anzahl und Art ihrer Parameter. Letztendlich soll durch die Parameter ein Punkt aus dem Weltkoordinatensystem in das Sensorkoordinatensystem transformiert werden. Die Transformation kann durch verschiedene Zwischenschritte veranschaulicht werden (siehe Abbildung 3.1).

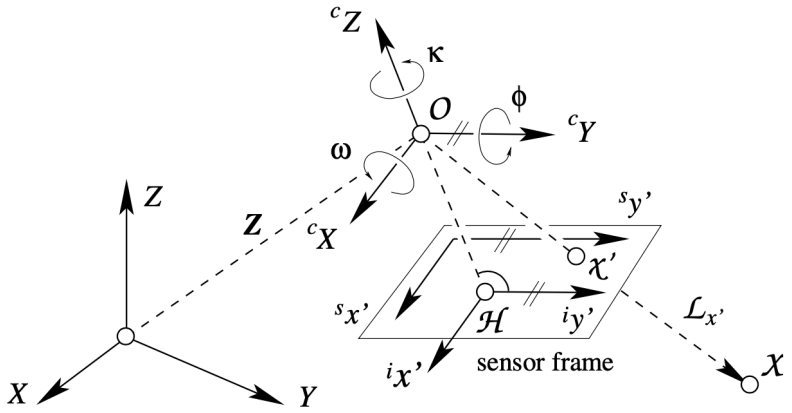


Abbildung 3.1: Überblick über die Transformationsschritte vom Weltkoordinatensystem S_o über das Kamerakoordinatensystem S_c und das Bildkoordinatensystem S_i in das Sensorkoordinatensystem S_s. [19]

Die Lösung für diese Transformation wird auch Direct Linear Transform (DLT) genannt. Auf das inverse Problem wird in diesem Kapitel nicht eingegangen. Für umfangreichere Definitionen und Erklärungen, sowie die Lösung des inversen Problems wird auf Förstner und Wrobel [19] verwiesen.

Zunächst werden die Koordinaten $\mathbf{X}$ aus dem **Weltkoordinatensystem** S_o in das **Kamerakoordinatensystem** S_c transformiert. Das geschieht in zwei Schritten: eine Translation zum Projektionszentrum der Kamera (Ursprung von S_c) und Rotation von S_o zu S_c. In homogenen Koordinaten lässt sich das wie folgt darstellen:

$$
\begin{aligned}
{}^c\mathbf{X} &= \begin{bmatrix} R & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I_3 & -\mathbf{Z} \\ 0 & 1 \end{bmatrix} \mathbf{X} && (3.1)\\
&= \begin{bmatrix} R & -R\mathbf{Z} \\ 0 & 1 \end{bmatrix} \mathbf{X}\,, && (3.2)
\end{aligned}
$$

wobei $\mathbf{Z} = [X_o, Y_o, Z_o]^T$ die Position des Projektionszentrums im Weltkoordinatensystem und die Rotationsmatrix R, beispielsweise parametriert durch die Euler-Winkel ω, ϕ, κ, die Rotation von S_o zu S_c darstellt. Die Variablen haben jeweils drei Freiheitsgrade, was in insgesamt sechs extrinsischen Parametern resultiert. Die extrinsischen Parameter werden auch äußere Orientierung genannt.

Die intrinsischen Parameter legen dagegen die Transformation vom Kamerakoordinatensystem ins Sensorkoordinatensystem, die innere Orientierung, fest. Dazu wird zunächst in das **Bildkoordinatensystem** S_i transformiert. Hierfür wird unter anderem angenommen, dass im optischen System keine Verzeichnung auftritt.

In Abbildung 3.2 sind die geometrischen Zusammenhänge bei der Abbildung durch eine um die optische Achse symmetrische Linse zu sehen. Für die Transformation werden mehrere Vereinfachungen getroffen. Die Linse wird als verzeichnungsfrei angenommen, der Sensor ist eben und

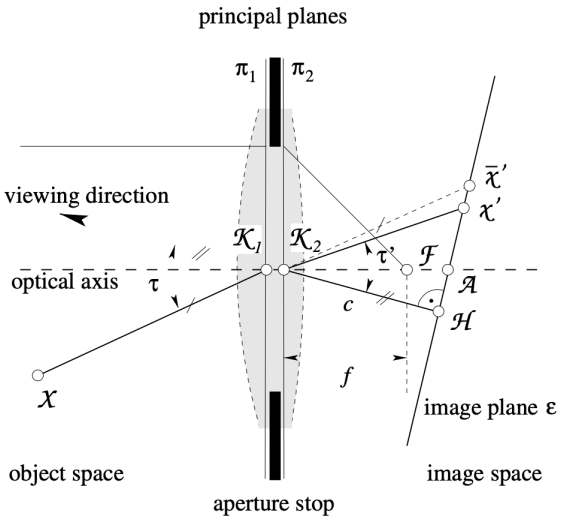


Abbildung 3.2: Geometrische Zusammenhänge bei der Abbildung durch eine Linse. [19]

der Ursprung des Bildkoordinatensystems liegt auf dem Hauptpunkt $\mathcal{H}$. Gelten diese Vereinfachungen, heißt das Modell Ideal Perspective Camera. Zusätzlich wird zunächst angenommen, dass der Hauptpunkt $\mathcal{H}$ auf der optischen Achse liegt und dass alle Projektionen $\mathcal{K}_1 = \mathcal{K}_2$ durchlaufen.

Mithilfe des Strahlensatzes werden die zweidimensionalen homogenen Bildkoordinaten ${}^i\mathbf{x}'$ aus den Koordinaten eines Punkts ${}^c\mathbf{x}' = [{}^cu', {}^cv', {}^cw']^T$ bestimmt (vgl. Abbildung 3.3). Die Distanz der Bildebene zum Projektionszentrum wird hierbei als c bezeichnet und entspricht ${}^cZ$.

$$ {}^ix' = c\frac{{}^cu'}{{}^cw'} \ , \quad {}^iy' = c\frac{{}^cv'}{{}^cw'} \ . \tag{3.3} $$

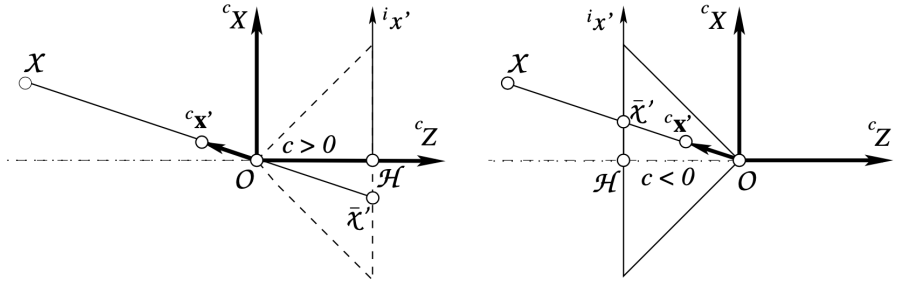


Abbildung 3.3: Kamerakoordinatensystem S_c, wobei das Bildkoordinatensystem durch ${}^cZ = c$ gegeben ist. Links ist das System mit $c > 0$ und rechts, wie häufig zur unkomplizierteren Veranschaulichung verwendet, mit $c < 0$. [19]

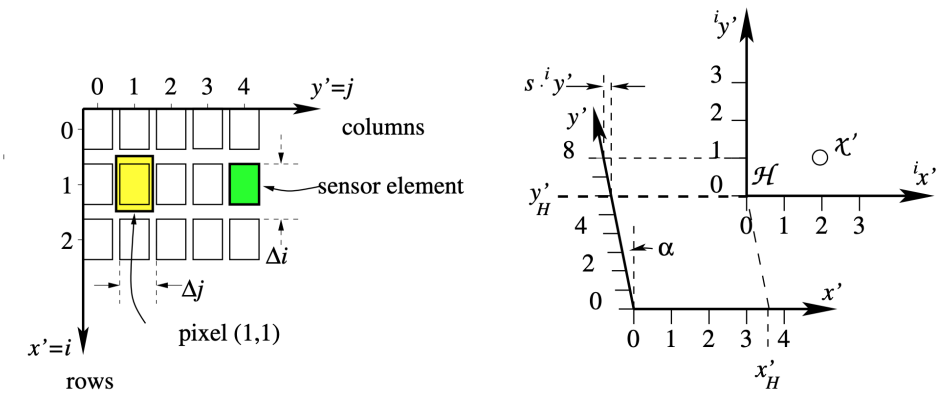


Abbildung 3.4: Sensorkoordinatensystem S_s (links) und Bildkoordinatensystem S_i in S_s (rechts) mit Skalendifferenz $m = \tan(\alpha)$, Scherung abhängig von ${}^{i}y'$ und Ursprung des Bildkoordinatensystems $\mathcal{H}$. [19]

In homogenen Koordinaten wird das durch die Kameramatrix ${}^{i}K$ ausgedrückt.

$$ {}^{i}\mathbf{x}' = {}^{i}K\ {}^{c}\mathbf{x}' = \begin{bmatrix} c & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & 1 \end{bmatrix} {}^{c}\mathbf{x}' \ . \tag{3.4} $$

Um zusätzlich Scherung und Streckung der Bilder zu berücksichtigen wird die Kameramatrix ${}^{i}K$ wie im Folgenden erweitert.

Intrinsische Parameter ändern sich mit der Pose der Kamera nicht und sind deshalb im Gegensatz zu den extrinsischen Parametern für eine Kamera konstant, solange das optische System unverändert bleibt. Die Erweiterung kann auch als zusätzliche Transformation in das **Sensorkoordinatensystem** S_s gesehen werden (siehe Abbildung 3.4). Diese besteht aus drei Schritten:

1. Translation vom Projektionszentrum zum Ursprung des Sensorkoordinatensystems $[x'_H, y'_H]^T$

2. Korrektur der y-Koordinaten um den Faktor $1+m$ zur Berücksichtigung unterschiedlicher Skalierung der Sensorelemente in x/y-Richtung

3. Scherung der ${}^{c}y$-Achse s (wird meist vernachlässigt)

In homogenen Koordinaten lässt sich das wie folgt ausdrücken:

$$\mathbf{x} = \begin{bmatrix} 1 & s & x'_H \\ 0 & 1+m & y'_H \\ 0 & 0 & 1 \end{bmatrix} {}^i\mathbf{x}' \tag{3.5}$$

$$= \begin{bmatrix} 1 & s & x'_H \\ 0 & 1+m & y'_H \\ 0 & 0 & 1 \end{bmatrix} {}^iK\ {}^c\mathbf{x}' \tag{3.6}$$

$$= \begin{bmatrix} c & cs & x'_H \\ 0 & c(1+m) & y'_H \\ 0 & 0 & 1 \end{bmatrix} {}^c\mathbf{x}' = K\ {}^c\mathbf{x}' \tag{3.7}$$

Insgesamt enthält die erweiterte Kameramatrix K somit fünf intrinsische Parameter. Insgesamt hat die DLT mit den sechs extrinsischen Parametern elf unabhängige Parameter.

Daraus ergeben sich folgende Modelle (absteigende Komplexität):

- **Perspective Camera**: Kamera mit planarem Sensor, Scherung s und Skalendifferenz m (alle fünf intrinsischen Parameter werden einbezogen)
- **Euklidische Kamera**: Perspective Camera, bei der $s = 0$ und $m = 0$ ist (Einfaches Modell für Pinhole Kameras, bei dem die Bildpunkte in einem beliebigen Koordinatensystem gemessen werden)
- **Ideale Kamera**: Euklidische Kamera, bei der der Ursprung des Bildkoordinatensystems auf dem Projektionszentrum liegt ($\rightarrow x'_H = 0,\ y'_H = 0$)
- **(Ideale) Einheitskamera**: Ideale Kamera mit $c = 1$ und somit $K = I_3$ (Basis für omnidirektionale Kameras / Spherical Cameras)
- **Normalisierte Kamera**: Ideale Einheitskamera, deren Ausrichtung der des Weltkoordinatensystems entspricht ($\rightarrow R = I_3$)

Eine Übersicht ist in Abbildung 3.5 zu sehen.

3.1.2 Nichtlineare Korrekturen

Um reale Kameras abzubilden, sind aufgrund nichtlinearer Fehler, wie radial-symmetrischer und tangentialer Verzeichnung durch das Objektiv, Unebenheit der Sensoroberfläche und Refraktion, zusätzliche Korrekturen notwendig.

Grundsätzlich wird zwischen physikalischen und phänomenologischen Modellen zur Korrektur unterschieden. Physikalische Modelle versuchen die physikalische Ursache für einen Fehler nachzubilden. Da sie oft kompliziert sind und es mit einem hohen Aufwand verbunden ist, geeignete Parameter zu finden, werden sie in der Praxis selten angewandt. Phänomenologische Modelle

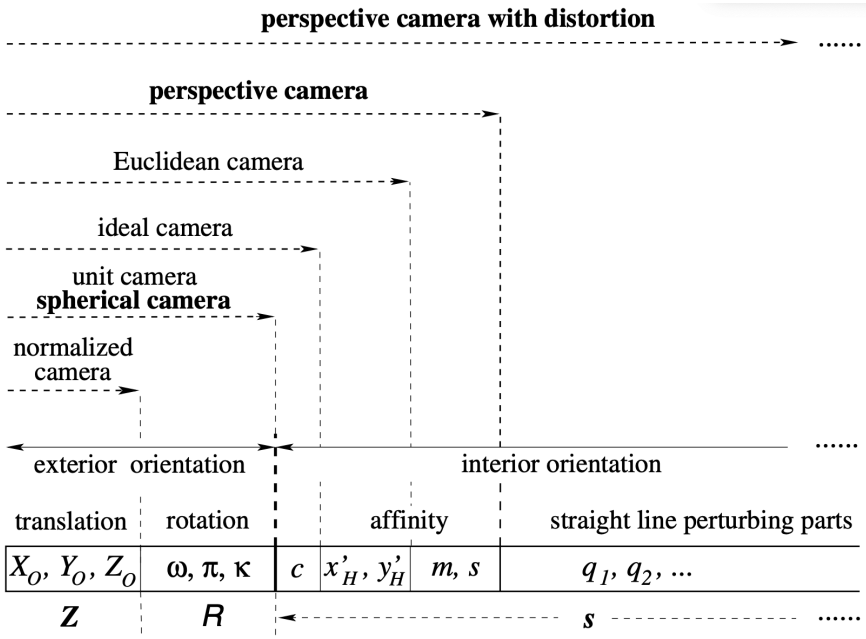


Abbildung 3.5: Überblick über verschiedene Kameramodelle. [19]

bilden dagegen die eigentlichen Effekte ab. Die wichtigsten Effekte sind die radial-symmetrische Verzeichnung (engl. *Barrel Distortion*)

$$x_{distorted} = x(1 + q_1 r^2 + q_2 r^4 + q_3 r^6) \tag{3.8}$$

$$y_{distorted} = y(1 + q_1 r^2 + q_2 r^4 + q_3 r^6) \tag{3.9}$$

und die tangentiale Verzeichnung

$$x_{distorted} = x + [2p_1 xy + p_2(r^2 + 2x^2)] \tag{3.10}$$

$$y_{distorted} = y + [p_1(r^2 + 2y^2) + 2p_2 xy] \,, \tag{3.11}$$

die daraus resultiert, dass die Bildebene nicht exakt parallel zum Objektiv angebracht ist. Die Gleichungen bilden einen unverzeichneten Pixel-Punkt (x, y) auf seine Pixel-Koordinaten nach der Verzeichnung $(x_{distorted}, y_{distorted})$ ab, wobei r den Abstand des Pixels zum optischen Zentrum / Projektionszentrum (x'_H, y'_H) bezeichnet. Die tangentiale Verzeichnung hat in der Regel einen wesentlich geringeren Einfluss als die radiale Verzeichnung.

Detailreichere Modelle für nichtlineare Korrekturen sind in Förstner und Wrobel [19] zu finden, spielen in der Praxis allerdings eine untergeordnete Rolle.

3.1.3 Kalibrierung nach Zhang

Für die Ermittlung der erläuterten intrinsischen Parameter existiert eine Vielzahl an Verfahren. Eine populäre Methode ist die Kalibrierung nach Zhang [52]. Wie bei vielen anderen Verfahren wird ein bekanntes Kalibrierobjekt verwendet. Im Falle von Zhang müssen Objektpunkte, die auf einer Ebene liegen, verwendet werden. Hierfür eignet sich ein Schachbrettmuster auf einer ebenen Oberfläche, da die Berührungspunkte der Felder durch ihre hohen Intensitätsgradienten leicht erkannt werden und deren Koordinaten bekannt sind. Die Methode wird im Folgenden erläutert.

Zunächst wird das Weltkoordinatensystem so gewählt, dass sein Ursprung in der Ecke des Schachbrettmusters liegt und die z-Achse senkrecht auf die Ebene des Schachbretts steht. Allgemein gilt:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} c & cs & x_H \\ 0 & c(1+m) & y_H \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.12}$$

Da die Z Komponente der Objektpunkte immer 0 ist, ergibt sich folgende Vereinfachung:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} c & cs & x_H \\ 0 & c(1+m) & y_H \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{3.13}$$

Zur Ermittlung der Parameter werden die innere und äußere Orientierung in eine Matrix zusammengefasst. Für jeden beobachteten Objektpunkt i wird eine Gleichung der Form

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = H_{3\times3} \begin{bmatrix} X_i \\ Y_i \\ 1 \end{bmatrix} \quad \text{mit } H_{3\times3} = \begin{bmatrix} c & cs & x_H \\ 0 & c(1+m) & y_H \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix} = K[r_1, r_2, t] \tag{3.14}$$

aufgestellt. Die Matrix $H_{3\times3}$ wird geschätzt, indem die Gleichungen (wie auch bei DLT) in ein homogenes Gleichungssystem der Form

$$a_{x_i}^T h = 0 \ , \quad a_{y_i}^T h = 0 \ , \tag{3.15}$$

mit

$$h = vec(H_{3\times3}^T) \ , \tag{3.16}$$

$$a_{x_i}^T = (-X_i, -Y_i, -1, 0, 0, 0, x_i X_i, x_i Y_i, x_i) \ \text{ und} \tag{3.17}$$

$$a_{y_i}^T = (0, 0, 0, -X_i, -Y_i, -1, y_i X_i, y_i Y_i, y_i) \tag{3.18}$$

gebracht und mittels Singulärwertzerlegung gelöst wird. Es werden mindestens vier Objektpunkte zur Lösung des Gleichungssystems benötigt.

Aus $H_{3\times3}$ wird nun die Matrix K bestimmt, die die eigentlichen zu ermittelnden Parameter enthält. Das erfolgt in vier Schritten:

1. Restriktionen bezüglich K, r_1 und r_2 ausnutzen
2. Matrix $B = K^{-T}K^{-1}$ definieren
3. B berechnen
4. B zerlegen

Zunächst wird sich die Eigenschaft zunutze gemacht, dass r_1 und r_2 zusammen mit r_3 eine Orthonormalbasis bilden, da sie Spaltenvektoren einer Rotationsmatrix sind. Daraus wird abgeleitet, dass

$$r_1^T r_2 = 0 \text{ und } ||r_1|| = ||r_2|| = 1 \tag{3.19}$$

gilt. Durch den Zusammenhang $H_{3\times3} = [h_1, h_2, h_3] = K[r_1, r_2, t]$, beziehungsweise $[r_1, r_2, t] = K^{-1}[h_1, h_2, h_3]$, eingesetzt in Gleichung (3.19), ergibt sich:

$$h_1^T K^{-T}K^{-1}h_2 = 0 \quad \text{und} \quad h_1^T K^{-T}K^{-1}h_1 = h_2^T K^{-T}K^{-1}h_2 \tag{3.20}$$

Durch die Zusammenfassung von $B = K^{-T}K^{-1}$ ergibt sich durch Umformung ein homogenes Gleichungssystem.

$$h_1^T B h_2 = 0 \tag{3.21}$$

$$h_1^T B h_1 - h_2^T B h_2 = 0 \tag{3.22}$$

Das Gleichungssystem wird analog zum Gleichungssystem (3.15) gelöst, indem die Symmetrie der Matrix B ausgenutzt wird und in Vektorform in ein entsprechendes Gleichungssystem umgewandelt wird, das durch Singulärwertzerlegung gelöst wird. Die Lösung dieses Gleichungssystems erfordert mindestens drei verschiedene Ansichten auf die Objektpunkte, um mit drei linear unabhängigen Gleichungen die sechs Freiheitsgrade der Matrix B zu bestimmen.

Die Matrix K resultiert aus der Cholesky-Zerlegung von B. Damit sind alle intrinsischen Parameter bestimmt.

3.1.4 Kalibrierung der nichtlinearen Parameter

Um zusätzlich die nichtlinearen Parameter zu schätzen, wird eine nichtlineare Fehlerfunktion $F(K, \mathbf{q}, R_n, t_n)$ über die Aufnahmen n minimiert.

$$F(K, \mathbf{q}, R_n, t_n) = \sum_i ||\mathbf{x}_{ni} - \hat{\mathbf{x}}(K, \mathbf{p}, \mathbf{q}, R_n, t_n, \mathbf{X}_{ni})||^2 \,, \tag{3.23}$$

wobei $\mathbf{x}_{ni}$ die Bildpunkte der Objektpunkte $\mathbf{X}_{ni}$ im Sensorkoordinatensystem und $\hat{\mathbf{x}}$ die Projektion der Objektpunkte in das Sensorkoordinatensystem, abhängig von der aktuellen Schätzung der Parameter, ist. Die nichtlinearen Verzeichnungsparameter werden in den Vektoren $\mathbf{p}$ und $\mathbf{q}$ zusammengefasst.

Die Fehlerfunktion kann mit dem Levenberg-Marquardt-Algorithmus unter Verwendung der Ergebnisse aus der Kalibrierung nach Zhang als Initialwerte nach den gesuchten Parametern minimiert werden. Das führt nicht nur zu einer Schätzung der nichtlinearen Parameter $\mathbf{p}$ und $\mathbf{q}$, sondern auch zu einer Verbesserung der Schätzung der intrinsischen Kameraparameter. [40]

3.2 Visuelle Features

Features sind bestimmte charakteristische Informationen von Bildinhalten. Allgemein handelt es sich hier um spezielle Strukturen, wie Punkte oder Kanten. Dieser Arbeit behandelt vor allem Punkt-Features, da diese eine hohe Relevanz für Methoden des Tracking, Mapping und der Lokalisierung haben.

Features werden meist aufgrund markanter Eigenschaften ihrer lokalen Umgebung (z.B. benachbarte Pixel) detektiert und anschließend mit eindeutigen Kennungen (Deskriptoren) versehen, anhand derer sie verglichen werden. Für die Anwendung in den Bereichen der Objekterkennung oder der Erfassung von dreidimensionalen Daten aus Kamerabildern ist es wichtig, markante Stellen der Umgebung zuverlässig auf mehreren Bildern zu erkennen und einander zuzuordnen. Hierfür existieren verschiedene Methoden zur Detektion von Features, die zusammen mit geeigneten Deskriptoren verwendet werden, um die Features zwischen den Bildern zuzuordnen.

3.2.1 SIFT

Scale Invariant Feature Transform (SIFT) wurde ursprünglich von David Lowe vorgeschlagen [33] und später verbessert [32]. Die Features in einem Bild werden in vier Schritten detektiert und mit Deskriptoren beschrieben:

1. **Scale-Space Extrema Detection**: Zunächst werden Keypoint Kandidaten identifiziert, indem sukzessive Gauss Filter (auch *Gaußsche Weichzeichner*) bei verschiedenen Skalierungen auf das Bild angewandt und die Differenz der entstehenden Bilder berechnet werden. Die Maxima der Difference of Gaussians (DoG) Bilder sind potentielle Keypoints, die invariant gegenüber Rotation und Skalierung sind.

2. **Keypoint Lokalisierung**: Die gefundenen Keypoint Kandidaten sind zu einem großen Anteil instabil. Deshalb werden im nächsten Schritt Keypoint Kandidaten mit schwachem Kontrast, die deshalb anfällig für Rauschen sind, verworfen. Außerdem werden Kandidaten, die aus Kanten resultieren, gefiltert. Bei Kanten ist die Hauptkrümmung über die Kante wesentlich höher als die Hauptkrümmung entlang der Kante. Demzufolge werden Keypoint Kandidaten abhängig vom Verhältnis der Eigenwerte der Hesse-Matrix zweiter Ordnung verworfen, da Kanten als Anhaltspunkte wesentlich schlechter geeignet sind als Ecken.
   Zudem werden in diesem Schritt die genauen Positionen der Keypoints bestimmt, indem um die Keypoints herum jeweils mit einer Taylorreihenentwicklung interpoliert wird und daraus die Maxima in Subpixelauflösung bestimmt werden.

3. **Orientation Assignment**: Um auch den Deskriptor unabhängig von der Orientierung zu vergeben, wird jedem Keypoint in diesem Schritt eine Orientierung zugeordnet. Die Orientierung und Magnitude $m(x, y)$ der benachbarten Pixel wird zunächst in Form eines Winkels $\theta(x, y)$ aus den benachbarten Pixeln berechnet:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \tag{3.24}$$

$$\theta(x,y) = \text{atan2}(L(x,y+1) - L(x,y-1),\ L(x+1,y) - L(x-1,y)) \tag{3.25}$$

   Die Orientierungen der benachbarten Pixel werden anschließend in einem Histogramm mit 36 Orientierungsklassen zusammengeführt und die höchsten Peaks darin werden als Orientierung zugewiesen. Kommen für einen Keypoint mehrere Peaks als Orientierung in Frage, wird ein zusätzlicher Keypoint mit der alternativen Orientierung an der selben Stelle eingefügt.

4. **Keypoint Deskriptor**: Für benachbarte Regionen der Keypoints werden wie im vorherigen Schritt Histogramme erstellt. Diese werden je nach Orientierung des Keypoints aneinandergehängt und als Vektor dargestellt. Insgesamt entstehen 16 Histogramme mit 8 Orientierungsklassen für jeden Keypoint. Die daraus resultierenden Deskriptoren sind demnach Vektoren mit 128 Einträgen, die für Robustheit normiert werden.

Das Ergebnis sind orientierungs- und skaleninvariante Features, die stark charakteristisch sind und in großen Zahlen auf Bildern gefunden werden können. Sie eignen sich hervorragend zum Feature-Matching durch die Berechnung der euklidischen Distanz, sind aber aufwändig zu berechnen.

3.2.2 SURF

Speeded Up Robust Features (SURF) basieren auf einem ähnlichen Ansatz wie SIFT. Sie erreichen eine ähnliche Unverwechselbarkeit, Robustheit und Reproduzierbarkeit und sind dabei deutlich schneller zu berechnen und zu vergleichen.

Bei SURF werden die DoG mit Box-Filtern approximiert, wodurch die Konvolution wesentlich schneller durchzuführen ist. Für die Detektion der Features werden daraus die Hesse-Matrizen berechnet, die als Kriterium für die Aufnahme als Keypoint verwendet werden. Auf zusätzliches Filtern der Keypoints (vgl. Schritt 2 bei SIFT) wird deshalb verzichtet.

Um die Orientierung zu berechnen, werden Haar wavelet responses (auch Haar-Features) in vertikaler und horizontaler Richtung für Pixel in einem festen Radius um die Keypoints berechnet und mit einer Gauss-Verteilung gewichtet. Ähnlich wie bei SIFT werden die Orientierungen der Pixel in Klassen eingeteilt und darin aufsummiert. Die stärkste Klasse wird als Orientierung zugewiesen.

Für den Deskriptor werden die Haar-Features in rechteckigen Regionen um die Keypoints herum berechnet und in einem Vektor dargestellt. Um SURF-Features einander effizienter zuzuordnen, wird jedem Feature ein zusätzlicher Wert abhängig vom Kontrast zugewiesen. Features werden nur verglichen, wenn dieser Wert in einem ähnlichen Bereich liegt, was zusätzlich Zeit spart. [3]

3.2.3 FAST Detektor

Der Features from Accelerated Segment Test (FAST) [41] ist ein effizientes Kriterium, um markante Stellen in einem Bild zu finden.

Es werden 16 Pixel auf einem Kreis um den potentiellen Keypoint betrachtet. Sind davon mehr als n Pixel deutlich heller oder deutlich dunkler als der Kandidat, wird dieser als Keypoint identifiziert.

FAST ist zwar äußerst effizient, hat aber einige Schwächen gegenüber langsameren Feature Detektoren. Diese werden versucht durch Techniken des Machine Learning zu mindern. Als Deskriptor dienen beispielsweise BRIEF Deskriptoren.

3.2.4 BRIEF Deskriptor

Binary Robust Independent Elementary Features (BRIEF) [6] ist ein Binärer Deskriptor aus 128, 256 oder 512 Bits. Dabei entspricht jedes Bit einem Vergleich von zwei Pixeln miteinander. Je nachdem, welcher Pixel heller ist, erhält das Bit den Wert 0 oder 1. Es existieren verschiedene Varianten, die festlegen welche Pixel für ein bestimmtes Bit im Deskriptor verglichen werden. Am nützlichsten sind dabei Varianten, bei denen die Vergleichspaare zufällig gewählt wurden. Wichtig ist, dass die Vergleichspaare nur einmal zufällig gewählt werden und anschließend konsistent verwendet werden müssen. BRIEF Deskriptoren sind nicht rotations- bzw. skaleninvariant.

Der Vorteil von Binären Deskriptoren liegt darin, dass diese durch die Berechnung der Hemming Distanz wesentlich schneller verglichen werden können als beispielsweise SIFT-Features, bei denen 128 Gleitkommazahlen verglichen werden müssen.

3.2.5 Bags of Visual Words

Mit der Methode der Bags of Words, Bags of Binary Words oder auch Bags of Visual Words werden ähnliche Bilder zur globalen (Re-)Lokalisierung und Loop Closure gefunden. Die Technik besteht grundsätzlich darin, eine große Datenbank an Bildern aufzubauen, in der ähnliche Bilder anhand gemeinsamer visueller Wörter effizient gefunden werden.

Ein Vokabular aus visuellen Wörtern setzt sich aus häufig auftretenden Features eines Datensatzes zusammen. Diese können beispielsweise dadurch gefunden werden, dass alle mit FAST detektierten Features des Datensatzes nach ihren BRIEF Deskriptoren geclustert werden und jedem Cluster ein visuelles Wort zugeordnet wird, dem wiederum sehr ähnliche, dem Cluster zugehörige Features zugeordnet werden können. Dazu wird für jedes Bild eine Art Histogramm aus den Anzahlen der visuellen Wörter berechnet. Diese Histogramme werden verglichen, um die Ähnlichkeit zwischen Bildern effizient zu berechnen. Eine verbreitete Variante für die Berechnung eines Ähnlichkeitsmaßes ist es, die Histogrammeinträge zunächst nach der Häufigkeit eines Worts im Datensatz zu normalisieren, um signifikante Wörter stärker zu gewichten, und dann die Kosinus-Ähnlichkeit der Histogramme als Vektoren zu bestimmen.

Zu beachten ist, dass das Ähnlichkeitsmaß bei Bags of Visual Words ausschließlich von der Anzahl der visuellen Wörter in den Bildern abhängig ist. Rotationen oder die (relativen) Positionen der Wörter in den Bildern spielen hierbei keine Rolle.

Die Methode ist wesentlich schneller als vorherige, rein Feature-basierte Ansätze. Allerdings ist ein zusätzlicher Verifikationsschritt notwendig, der anhand von Features Korrespondenzen in ähnlichen Bildern findet und damit garantiert, dass es sich um eine Aufnahme der selben Umgebung handelt. [20]

3.3 Bundle Adjustment

Bundle Adjustment ist eine statistisch optimale Methode, um gleichzeitig die internen und externen Kameraparameter, sowie 3D Koordinaten von Punkten aus der Umgebung unter Berücksichtigung der Unsicherheiten zu schätzen. Im Gegensatz zu herkömmlichen Methoden der visuellen Odometrie werden hierfür nicht nur die beiden zuletzt aufgenommen Bilder verwendet, sondern beliebig viele Bilder, in denen beliebig viele 3D Punkte enthalten sind und als Features den Bildern zugeordnet werden.

Bundle Adjustment ist ein Spezialfall der sogenannten Block Adjustments, bei denen mehrere Einheiten an Punktwolken (zusammen ein Block) in ein globales Koordinatensystem fusioniert werden. Die Schwierigkeit besteht darin, dass die Anzahl dieser Einheiten und damit das zu lösende Gleichungssystem sehr groß werden kann. Das Problem wird gelöst, indem sich zunutze gemacht wird, dass die Matrizen im Gleichungssystem dünn besetzt sind. [19]

Es gibt verschiedenste Möglichkeiten zur Implementierung von Bundle Adjustment. Grundsätzlich wird ein Projektionsmodell aufgestellt, dessen Gleichungssystem durch die Minimierung einer Fehlerfunktion numerisch gelöst wird. Hierbei handelt es sich meist um einen Least-Squares-Ansatz. Für die numerische Minimierung der Fehlerfunktion sind initiale Schätzungen für die Unbekannten nötig. Diese werden entweder mit direkten Verfahren approximiert oder von alternativen Quellen, wie beispielsweise die Odometrie eines Fahrzeugs, bezogen. Ein Projektionsmodell sieht wie folgt aus:

$$x_{ij} \; + \; \widehat{v}_{x_{ij}} = \widehat{\lambda}_{ij} \; \widehat{P}_j(x_{ij}, p, q) \; \widehat{X}_i \; , \tag{3.26}$$

wobei x_{ij} die Bildkoordinaten des Punkts i im Bild j und $\widehat{v}_{x_{ij}}$ zugehörige Korrekturen in einem beliebigen Bildkoordinatensystem bezeichnen. Diese Position soll gleich den Koordinaten des Punkts $\widehat{X}_i$ im globalen Koordinatensystem sein, der mit der Matrix $\widehat{P}_j(x_{ij}, p, q)$ auf das Bildkoordinatensystem projiziert und mit dem Parameter $\widehat{\lambda}_{ij}$ skaliert wurde. Die Abweichung des in das Kamerakoordinatensystem projizierten 3D Punkts zu den wahren Bildkoordinaten heißt Reprojektionsfehler.

Die Unbekannten sind hierbei die Punkte $\widehat{X}_i$, die Skalenparameter, die Projektionsparameter p, die Verzeichnungssparameter q und die Orientierung der Kamera (enthalten in $\widehat{P}_j(x_{ij}, p, q)$). Die Skalierung ist notwendig, da im globalen Koordinatensystem mit homogenen Koordinaten gerechnet wird. Wird das Projektionsmodell mit Euklidischen Koordinaten aufgestellt, fallen

die Skalierungsparameter als Unbekannte weg. Um das Verfahren zu verbessern, können Passpunkte mitaufgenommen werden, deren 3D Position und Positionen in den Bildern bekannt ist. Hierdurch ist es zusätzlich möglich, absolute geometrische Informationen zu berechnen.

Für die Minimierung der Fehlerfunktion ist die Kenntnis der Unsicherheiten in den Bildkoordinaten von Nutzen, um die einzelnen Fehler entsprechend zu gewichten. Grobe Fehler bei der Zuordnung von 3D Punkten durch Features sollen möglichst vor der Anwendung von Bundle Adjustment eliminiert werden.

Weitere Details zur Implementierung und Theorie hinter Bundle Adjustment wurden von Triggs et al. [47] und Chen et al. [8] zusammengefasst.

3.4 Perspective-n-Point Problem und Lösungsansätze

Beim Perspective-n-Point (PnP) Problem wird unter Kenntnis von n 3D Punkten im Weltkoordinatensystem $\mathbf{X}_W$, der 2D Koordinaten der entsprechenden Projektionen im Bild $[u, v]$ und der Kameramatrix $\mathbf{K}$ die Pose einer kalibrierten Kamera geschätzt. Dabei wird folgender Zusammenhang zugrunde gelegt:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K}\,\mathbf{\Pi}\,{}^{c}\mathbf{T}_{\mathbf{W}} \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} \tag{3.27}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} \tag{3.28}$$

Bestimmt werden soll die Transformationsmatrix ${}^{c}\mathbf{T}_{\mathbf{W}}$, bestehend aus Rotation und Translation. Das Ergebnis wird unter anderem bei Augmented Reality oder zur Lokalisierung verwendet.

Zur Lösung des Problems sind bei Features mit Orientierung $n \geq 2$ und ansonsten $n \geq 3$ Punkte notwendig. Je nach Anzahl und Anordnung der Punkte im Raum existieren verschiedene Techniken. OpenCV enthält einige Lösungsmethoden, wobei standardmäßig ein iteratives Verfahren gewählt wird, bei dem der Reprojektionsfehler mit dem Levenberg-Marquardt-Algorithmus minimiert wird. [35, 40]

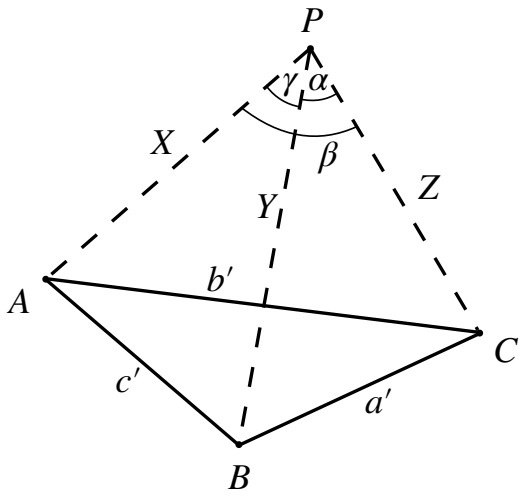


Abbildung 3.6: Das P3P Problem.

3.4.1 P3P

In der Minimalform kann das Perspective-n-Point Problem mit drei Punkten gelöst werden. In den meisten Fällen ist allerdings ein vierter Punkt notwendig, um Mehrdeutigkeiten aufzulösen. Zur Lösung des Problems verwendet die Bibliothek OpenCV die Methode von Gao et al. [22]. Hier wird das Problem wie folgt dargestellt (vgl. Abbildung 3.6): Sei P das Projektionszentrum und seien A, B, C die Kontrollpunkte, gilt durch den Kosinussatz angewandt auf die Dreiecke PBC, PAC und PAB das Gleichungssystem

$$Y^2 + Z^2 - 2YZ\cos(\alpha) - a'^2 = 0 \tag{3.29}$$
$$Z^2 + X^2 - 2XZ\cos(\beta) - b'^2 = 0 \tag{3.30}$$
$$X^2 + Y^2 - 2XY\cos(\gamma) - c'^2 = 0\,. \tag{3.31}$$

Die Werte α, β und γ werden aus den intrinsischen Parametern der Kamera berechnet. Das Gleichungssystem wird analytisch gelöst und die Distanzen X, Y und Z bestimmen die dreidimensionalen Koordinaten des Projektionszentrums. Bei drei Punkten existieren maximal vier unterschiedliche Lösungen, von denen durch eine vierte Beobachtung eine bestätigt wird.

3.4.2 EPnP

Meist sind deutlich mehr Kontrollpunkte vorhanden, deren Berücksichtigung das Problem des Rauschens vermindern. Allerdings ist die Berechnung der Lösung hierdurch komplexer. Efficient PnP (EPnP) [31] reduziert die Komplexität auf $O(n)$, indem die Kontrollpunkte im globalen Koordinatensystem $\mathbf{p}_i^w$ und im Kamerakoordinatensystem $\mathbf{p}_i^c$ als gewichtete Summe aus vier

virtuellen Kontrollpunkten $\mathbf{c}_j^w/\mathbf{c}_j^c$ ausgedrückt werden.

$$\mathbf{p}_i^w = \sum_{j=1}^{4} \alpha_{ij} \mathbf{c}_j^w \tag{3.32}$$

$$\mathbf{p}_i^c = \sum_{j=1}^{4} \alpha_{ij} \mathbf{c}_j^c \tag{3.33}$$

$$\sum_{j=1}^{4} \alpha_{ij} = 1 \tag{3.34}$$

Seien $\mathbf{u}_i$ die 2D-Projektionen der Punkte $\mathbf{p}_i$ ergibt sich mit der Kameramatrix $\mathbf{K}$:

$$\forall i, w_i \begin{bmatrix} \mathbf{u}_i \\ 1 \end{bmatrix} = \mathbf{K}\mathbf{p}_i^c = \sum_{j=1}^{4} \alpha_{ij} \mathbf{c}_j^c \tag{3.35}$$

$$\forall i, w_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_c \\ 0 & f_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \sum_{j=1}^{4} \alpha_{ij} \begin{bmatrix} x_j^c \\ y_j^c \\ z_j^c \end{bmatrix} , \tag{3.36}$$

wobei w_i skalare Projektionsparameter sind. Daraus lässt sich ein lineares Gleichungssystem ableiten. Für jeden Kontrollpunkt gilt

$$\sum_{j=1}^{4} \alpha_{ij} f_u x_j^c + \alpha_{ij}(u_c - u_i) z_j^c = 0 \tag{3.37}$$

$$\sum_{j=1}^{4} \alpha_{ij} f_v y_j^c + \alpha_{ij}(v_c - v_i) z_j^c = 0 \, . \tag{3.38}$$

Durch Konkatenation der Gleichungen (3.37) und (3.38) ergibt sich bei n Kontrollpunkten ein Gleichungssystem der Form

$$\mathbf{M}\mathbf{x} = \mathbf{0} \text{ , mit } \mathbf{x} = [\mathbf{c}_1^{cT}, \mathbf{c}_2^{cT}, \mathbf{c}_3^{cT}, \mathbf{c}_4^{cT}] \, , \tag{3.39}$$

wobei $\mathbf{M}$ eine $2n \times 12$ Matrix ist, deren Kern die Lösung des Gleichungssystems ist.

Die Methode kann für eine beliebige Anzahl an Kontrollpunkten in beliebiger Anordnung angewandt werden, ist aber nicht robust gegenüber Ausreißern. Um diesem Problem entgegen zu wirken, können Methoden zur Lösung von PnP Problemen mit RANSAC (vgl. Abschnitt 3.6) kombiniert werden.

3.4.3 IPPE

Infinitesimal Plane-Based Pose Estimation (IPPE) ist ein genaues und effizientes Verfahren, das mindestens vier Kontrollpunkte auf einer Ebene benötigt und ausnutzt, dass die Homographie (Transformation zweier Ebenen zueinander) an bestimmten Stellen genauer ermittelbar ist als an anderen Stellen. Die Methode ermittelt zunächst den Punkt, an dem die Transformation am besten geschätzt werden kann. Mit der verrauschten Schätzung der Transformation und deren Jakobi-Matrix wird die Pose aus der Lösung einer partiellen Differentialgleichung, die aus der Ableitung der Transformation der Kontrollpunktebene zur Kameraebene resultiert, ermittelt. Details können der Publikation von Collins und Bartoli [9] entnommen werden.

3.5 ICP Algorithmus

Die populärste Methode, um die relative Pose zweier beliebiger Punktwolken zueinander zu bestimmen (engl. Point Set Registration) ist der Iterative Closest Point (ICP) Algorithmus [4]: ein iteratives Verfahren, bei dem durch Rotation $\mathbf{R}$ und Translation $\mathbf{t}$ die Fehlerfunktion

$$E(\mathbf{R}, \mathbf{t}) = \sum_{(i,j)\in\mathcal{C}} \|\mathbf{m}_i - (\mathbf{R}\mathbf{d}_j + \mathbf{t})\|^2 \tag{3.40}$$

minimiert wird, wobei die Tupel $(i, j) \in \mathcal{C}$ Korrespondenzen der Punkte $\mathbf{m}_i \in M$ der *Model* Punktmenge mit Punkten $\mathbf{d}_j \in D$ der *Data* Punktmenge darstellen. Hierfür wird für jeden Punkt $\mathbf{d}_j$ in D der nächstgelegene Punkt in M gesucht, der in einem bestimmten Radius um $\mathbf{d}_j$ liegt.

Iterativ wird die Fehlerfunktion nach $\mathbf{R}$ und $\mathbf{t}$ minimiert, die Punktmenge D mittels der gefundenen Rotationsmatrix $\mathbf{R}$ und des Translationsvektors $\mathbf{t}$ transformiert und neue Tupel $(i, j) \in \mathcal{C}$ nach dem jeweils nächstgelegenen Punkt der transformierten Punktmenge gewählt. Der Algorithmus terminiert nach einer bestimmten Anzahl an Iterationen oder, wenn die Fehlerfunktion einen Schwellwert erreicht hat.

Der Kern des Verfahrens ist die Minimierung der Fehlerfunktion nach $\mathbf{R}$ und $\mathbf{t}$. Um die Rotation zu bestimmen, werden zunächst die Zentroide der Punktmengen bestimmt:

$$\mu_M = \frac{1}{|\mathcal{C}|} \sum_{(i,j)\in\mathcal{C}} \mathbf{m}_i \qquad \mu_D = \frac{1}{|\mathcal{C}|} \sum_{(i,j)\in\mathcal{C}} \mathbf{d}_j \tag{3.41}$$

Die Punktmengen werden anschließend zentriert.

$$M' = \{\mathbf{m}_i - \mu_M\} = \{\mathbf{m}'_i\} \qquad D' = \{\mathbf{d}_j - \mu_D\} = \{\mathbf{d}'_j\} \tag{3.42}$$

Im Folgenden bezeichnen $\mathbf{M}'$ und $\mathbf{D}'$ die Punktmengen M' und D' in Matrixform. Die Rotation $\mathbf{R}$ wird bestimmt, indem $\min_{\mathbf{R}} \|\mathbf{R}\mathbf{D}' - \mathbf{M}'\|_F$ minimiert wird. Die Frobenius-Norm einer Matrix Z berechnet sich aus $\|Z\|_F = \sum_{i,j} z_{i,j}^2$. Dieses Problem wird als Orthoganal Procrustes Problem

bezeichnet und wird gelöst, indem die Singulärwertzerlegung (SVD) des Produkts $C = \mathbf{M'D'}^T$ berechnet wird.

$$U\ S\ V^T = C \tag{3.43}$$

$$\mathbf{R} = U\, diag\left(1, 1, \det\left(UV^T\right)\right) V^T \tag{3.44}$$

Für eine detailliertere Beschreibung wird Söderkvists *Using SVD for some fitting problems* [45] empfohlen.

Die Translation wird mit $\mathbf{t} = \mu_M - \mathbf{R}\mu_D$ berechnet.

Neben dem erläuterten Verfahrens existieren verschiedene Varianten und Erweiterungen des ICP Algorithmus, bei denen unter anderem die Zuordnung der Korrespondenzen und die Behandlung von Ausreißern verbessert werden.

3.6 RANSAC

Random Sample Consensus (RANSAC) ist ein Algorithmus, der im Allgemeinen ein Modell aus einer Menge an Messwerten mit Ausreißern schätzt. Der Algorithmus wird unter anderem auch bei Point Set Registration angewandt und kommt gegenüber ICP oder Least Squares-basierten Verfahren deutlich besser mit größeren Anzahlen an Ausreißern zurecht. Der Algorithmus wurde 1981 von M. Fischer und R. Bolles [17] veröffentlicht und findet vor allem im Bereich Computer Vision Anwendung.

Der RANSAC Algorithmus verfolgt einen iterativen Ansatz, bei dem folgende Schritte wiederholt werden:

1. Zufällige Auswahl so vieler Datenpunkte aus der Menge der Messwerte, wie nötig sind, um das Modell zu berechnen
2. Berechnung des Modells (z.B. Homogene Transformationsmatrix bei Point Set Registration)
3. Bestimmung der Teilmenge der Messwerte (Consensus Set), die mit dem Modell vereinbar sind (z.B. durch die Distanz zu einer Modellkurve)

Nachdem die Schritte mehrfach wiederholt wurden, wird die größte der Teilmengen gewählt. Anhand dieser Teilmenge wird das Modell mit einem Ausgleichsverfahren berechnet. Bei der preemptiven Variante des Algorithmus wird die Iteration vorzeitig abgebrochen, falls ein Consensus Set eine Mindestgröße erreicht hat. Hierzu muss die Anzahl der Ausreißer im Groben bekannt sein.

Folgende Parameter müssen adäquat gewählt werden:

- **Maximaler Abstand eines Datenpunkts vom Modell:** Dieser Parameter ist entscheidend, um falsche Ergebnisse zu vermeiden. Ist der Abstand zu groß, werden zu viele Ausreißer nicht erkannt, sodass die Anzahl der Ausreißer nicht mehr aussagekräftig für die Güte des Consensus Sets ist. Ist der Wert zu klein, werden gültige Werte nicht in das Modell miteinbezogen, wodurch ebenfalls Fehler entstehen können.
- **Anzahl der Iterationen:** Dieser Wert ist weniger kritisch, da er beliebig hoch gewählt werden kann. Er wird so gewählt, dass mit einer hohen Wahrscheinlichkeit mindestens eine ausreißerfreie Teilmenge gefunden wird. Um einen passenden Wert zu berechnen muss der Ausreißeranteil bekannt sein.
- **Größe des Consensus Sets:** Bei der preemptiven Variante muss abgeschätzt werden, wie viele Datenpunkte in einer Teilmenge enthalten sein müssen, um ein plausibles Consensus Set darzustellen.

3.7 Bresenham Algorithmus

Der Bresenham Algorithmus [5] dient dazu, Geraden zwischen zwei Punkten in einem zweidimensionalen Raster zu zeichnen. Dabei ist die Besonderheit, dass der Algorithmus ganz ohne Multiplikationen und Divisionen auskommt. Infolgedessen treten keine Rundungsfehler auf und der Rechenaufwand ist gering.

Das Grundprinzip beim Bresenham Algorithmus ist, dass in einer Schleife jeweils ein Schritt in die schnellere Richtung (beispielsweise ist das bei einer Steigung von -1 bis 1 die x-Richtung) und bei Bedarf ein Schritt in die langsamere Richtung gemacht wird. Zur Entscheidung, ob ein Schritt in die langsamere Richtung nötig ist, wird ein Fehlerglied aufrechterhalten. Dieses resultiert aus der Geradengleichung zwischen den Punkten (x_{start}, y_{start}) und (x_{ende}, y_{ende})

$$y(x) = y_{start} + (x - x_{start}) \cdot \frac{y_{ende} - y_{start}}{x_{ende} - x_{start}} \quad , \tag{3.45}$$

die in

$$0 = (x_{ende} - x_{start}) \cdot (y - y_{start}) - (y_{ende} - y_{start}) \cdot (x - x_{start}) \tag{3.46}$$

umgestellt wird. Daraus ergibt sich aufgrund der Diskretisierung für das Fehlerglied E:

$$E = (x_{ende} - x_{start}) \cdot (y - y_{start}) - (y_{ende} - y_{start}) \cdot (x - x_{start}) \tag{3.47}$$

Das Fehlerglied wird sukzessive akkumuliert, indem bei einem Schritt in x-Richtung die Differenz $y_{ende} - y_{start}$ bzw. bei einem Schritt in y-Richtung $x_{ende} - x_{start}$ auf das Fehlerglied addiert wird. Sobald $E > 0$, ist ein Schritt in die langsamere Richtung notwendig.

Sobald der Endpunkt erreicht wird, wird die Schleife abgebrochen.

Algorithmus 1 fasst das Verfahren in kompakter Form zusammen.

Algorithmus 1: Bresenham Algorithmus

Eingabe: x_{start}, y_{start}, x_{end} und y_{end}

$dx \leftarrow \text{abs}(x_{end} - x_{start})$
$sx \leftarrow 1$ **if** $x_{end} > x_{start}$, **else** -1
$x \leftarrow x_{start}$
$dy \leftarrow -\text{abs}(y_{end} - y_{start})$
$y \leftarrow y_{start}$
$sy \leftarrow 1$ **if** $y_{end} > y_{start}$, **else** -1
$err \leftarrow dx + dy$
while $(x, y) \neq (x_{end}, y_{end})$ **do**
    setPixel(x)
    if $2 \cdot err > dy$ **then**
        $err \leftarrow err + dy$
        $x \leftarrow x + sx$
    end
    if $2 \cdot err < dx$ **then**
        $err \leftarrow err + dx$
        $y \leftarrow y + sy$
    end
end

3.8 Occupancy Grid Mapping

Beim Occupancy Grid Mapping wird auf Grundlage verrauschter Sensordaten eine Karte generiert. Dabei wird angenommen, dass die Pose des Roboters, der die Daten bereitstellt, bekannt ist. Die Karte wird als Feld gleich großer Zellen dargestellt, für die jeweils ein Zufallswert das Auftreten eines Hindernisses beschreibt.

Zur Lösung des Problems wird ein Zustands- und ein Messmodel benötigt. Der Zustand ist dabei die Karte der Umgebung, die aus Zellen besteht welche besetzt oder frei sind. Die Karte wird als zeitlich invariant angenommen. Das Messmodel $p(z_t|m, l_t)$ ist die Wahrscheinlichkeit, dass eine Beobachtung z_t bei einer Karte m und der Position l_t gemacht wird. Die einzelnen n Zellen werden als m_i bezeichnet. Das Ziel ist es daraus umgekehrt die Wahrscheinlichkeit $p(m|z_{1:t}, x_{1:t})$ für eine Karte m bei gegebenen Messungen zu bestimmen. Um die Komplexität der Lösung von 2^n auf $2n$ zu reduzieren, wird angenommen, dass die Belegung der Zellen unabhängig ist, sodass sich

$$p(m|z_{1:t}, x_{1:t}) = \prod_i p(m_i|z_{1:t}, x_{1:t}) \tag{3.48}$$

ergibt.

Sei X^i der Zustand der Zelle m_i, die entweder belegt (x) oder frei ($\bar{x}$) ist. Die Wahrscheinlichkeit, dass eine Zelle belegt ist, ist nach dem Satz von Bayes:

$$p(x|z_{1:t}) = \frac{p(z_t|x, z_{1:t-1})p(x|z_{1:t-1})}{p(z_t|z_{1:t-1})} \tag{3.49}$$

Mit der Markov-Annahme $p(z_t|x, z_{1:t-1}) = p(z_t|x)$ ergibt sich:

$$p(x|z_{1:t}) = \frac{p(z_t|x)p(x|z_{1:t-1})}{p(z_t|z_{1:t-1})} \tag{3.50}$$

Die Gültigkeit der Markov-Annahme beim Occupancy Grid Mapping ist begrenzt, da Messungen in der Praxis häufig von vergangenen Messungen abhängig sind. Nichtsdestotrotz wird die Vereinfachung gewählt, um mit einer weiteren Anwendung des Satz von Bayes auf $p(z_t|x)$ die Fortschreibungsregel (engl. *update rule*)

$$p(x|z_{1:t}) = \frac{p(x|z_t)p(z_t)}{p(x)} \frac{p(x|z_{1:t-1})}{p(z_t|z_{1:t-1})} \tag{3.51}$$

zu erhalten. Analog lässt sich die Fortschreibungsregel für $p(\bar{x}|z_{1:t})$ herleiten:

$$p(\bar{x}|z_{1:t}) = \frac{p(\bar{x}|z_t)p(z_t)}{p(\bar{x})} \frac{p(\bar{x}|z_{1:t-1})}{p(z_t|z_{1:t-1})} \tag{3.52}$$

Das Chancenverhältnis (engl. *odds ratio*) ergibt sich zu

$$\frac{p(x|z_{1:t})}{p(\bar{x}|z_{1:t})} = \frac{p(x|z_t)}{p(\bar{x}|z_t)} \frac{p(\bar{x})}{p(x)} \frac{p(x|z_{1:t-1})}{p(\bar{x}|z_{1:t-1})} \frac{p(z_t|z_{1:t-1})}{p(z_t|z_{1:t-1})} \tag{3.53}$$

$$= \frac{p(\bar{x})}{p(x)} \frac{p(x|z_t)}{p(\bar{x}|z_t)} \frac{p(x|z_{1:t-1})}{p(\bar{x}|z_{1:t-1})} \quad . \tag{3.54}$$

Häufig wird das Chancenverhältnis auch logarithmisch als $l_t(x)$ angegeben (engl. *log odds ratio*).

$$l_t(x) = \log \frac{p(x|z_{1:t})}{p(\bar{x}|z_{1:t})} = \log \frac{p(\bar{x})}{p(x)} + \log \frac{p(x|z_t)}{p(\bar{x}|z_t)} + l_{t-1}(x) \tag{3.55}$$

Für die Fortschreibungsregel muss demnach nur das inverse Sensormodell $p(x|z_t)$ aufgestellt und eine A-priori-Wahrscheinlichkeit $p(x)$ für die Belegung eines Felds festgelegt werden. Die zugehörigen Komplemente sind $p(\bar{x}|z_t)$ und $p(\bar{x})$. [2]

3.9 Kalmanfilter

Der Kalmanfilter [27] ist ein Beobachter für stochastische Größen, der iterativ aus verrauschten Messgrößen einen (potentiell nicht messbaren, aber beobachtbaren) Systemzustand schätzt. Er eignet sich hervorragend zur Sensordatenfusion und um aus fehlerbehafteten, verrauschten Messungen stabile Zustandsschätzungen zu erhalten.

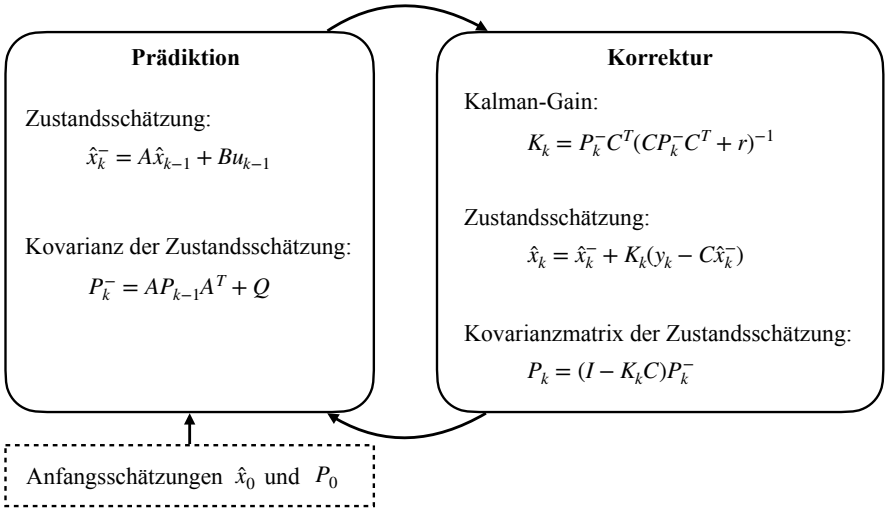


Abbildung 3.7: Überblick über die Berechnungen des Kalmanfilters, aufgeteilt in Prädiktion und Korrektur.

Für einen Kalmanfilter werden mehrere Annahmen getroffen. Zunächst wird von einem zeitdiskreten, stochastischen System der Form

$$x_{k+1} = Ax_k + Bu_k + w_k \tag{3.56}$$

$$y_k = Cx_k + v_k \tag{3.57}$$

ausgegangen, wobei x der Systemzustand, A die Systemmatrix, die den Zustandsübergang beschreibt, B die Eingabematrix und w das Prozessrauschen ist. Die messbare Ausgabe y berechnet sich über die Ausgabematrix C und das Messrauschen v. Der Index k bzw. $k+1$ spezifiziert den Zeitschritt. Wie bei allen Beobachtern, muss das Beobachtbarkeitskriterium von Kalman für das Paar A, C gelten. Zudem werden sowohl das Prozessrauschen als auch das Messrauschen als multivariates, weißes Rauschen modelliert. Das heißt beide Zufallsvariablen sind Mittelwertfrei. Für die Autokorrelationsmatrizen R_{ww} und R_{vv} gilt aufgrund der zeitlichen Unkorreliertheit des Rauschens:

$$R_{vv}(\kappa) = E[v_k\, v_{k+\kappa}^T] = \begin{cases} R & \text{für } \kappa = 0 \\ 0 & \text{für } \kappa \neq 0 \end{cases} \tag{3.58}$$

$$R_{ww}(\kappa) = E[w_k\, w_{k+\kappa}^T] = \begin{cases} Q & \text{für } \kappa = 0 \\ 0 & \text{für } \kappa \neq 0 \end{cases} \tag{3.59}$$

R und Q sind die Kovarianzmatrizen des Mess- und Prozessrauschens. Das Prozess- und Messrauschen sind zueinander zu allen Zeitpunkten unkorreliert. Das Messrauschen wird oft durch die empirische Varianz einer Messreihe angenähert. Das Prozessrauschen muss frei gewählt werden.

Der Kalmanfilter verfolgt grundsätzlich einen ähnlichen Ansatz wie der klassische zeitdiskrete Luenbergerbeobachter. Bei Eintreffen einer neuen Messung wird die Zustandsschätzung korrigiert:

$$\hat{x}_k = \hat{x}_k^- + K_k(y_K - C\hat{x}_k^-) \tag{3.60}$$

Hierbei ist $\hat{x}_k$ die a-posteriori Zustandsschätzung, das heißt die Zustandsschätzung nach der Korrektur, und $\hat{x}_k^-$ die a-priori Zustandsschätzung, die aus der Zustandsgleichung (3.58) berechnet wird.

Das Ziel des Kalmanfilters ist es den mittleren quadratischen a-posteriori Schätzfehler $E[(x_k - \hat{x}_k)^T(x_k - \hat{x}_k)]$ bezüglich des Kalman Gains K_k zu minimieren. Hierfür wird die Kovarianz P_k der Zustandsschätzungen mit einbezogen. Die sich ergebenden Terme zur iterativen Zustandsschätzung sind in Abbildung 3.7 dargestellt.

3.10 Adaptive Monte Carlo Localization

Adaptive Monte Carlo Localization (AMCL) [18] ist ein Lokalisierungsalgorithmus auf Basis von Partikelfiltern. Partikelfilter werden mit einer großen Anzahl an Partikeln, im Falle von AMCL mögliche Posen, die den Zustandsraum aufspannen, initialisiert. Bei jeder Iteration werden drei Schritte ausgeführt:

1. **Resampling:** Auf Grundlage der aktuellen multimodalen Wahrscheinlichkeitsverteilung für alle möglichen Posen werden zufällige Posen gewählt.
2. **Sampling:** Basierend auf den Messungen der Odometrie, wird aus den Posen aus dem vorherigen Schritt eine neue Wahrscheinlichkeitsverteilung für mögliche Posen bestimmt, aus der erneut zufällige Posen gewählt werden.
3. **Importance Sampling:** Die Posen aus dem letzten Schritt werden nach ihrer Wahrscheinlichkeit gewichtet, indem die aktuelle Messung (meist Laserscans) mit der erwarteten Messung bei der jeweiligen Pose verglichen wird.

Nach jeder Iteration entsteht demnach eine Menge an möglichen Posen, die nach ihrer Wahrscheinlichkeit gewichtet sind. Damit diese Menge schneller konvergiert und der Algorithmus weniger rechenaufwändig wird, begrenzen adaptive Partikelfilter die Anzahl der möglichen Posen, mit denen weiter gerechnet wird. Für detailreichere Erklärungen wird Thruns *Particle Filters in Robotics* [46] empfohlen.

3.11 Hand-Eye Kalibrierung

Die Hand-Eye Kalibrierung bezeichnet die Bestimmung der räumlichen Beziehung eines Sensors zu einem weiteren Sensor, einem Aktuator oder einem sonstigen Punkt auf einem Roboter. Sie ist immer dann von Bedeutung, wenn die Pose eines bestimmten Punkts aus der Pose eines Sensors errechnet werden soll. Im Falle dieser Arbeit wird die homogene Transformation zwischen der Sensoreinheit eines Referenzsystems und der Kamera, deren Pose vom SLAM-Algorithmus bestimmt wird, ermittelt, um deren jeweilige Messwerte zu vergleichen.

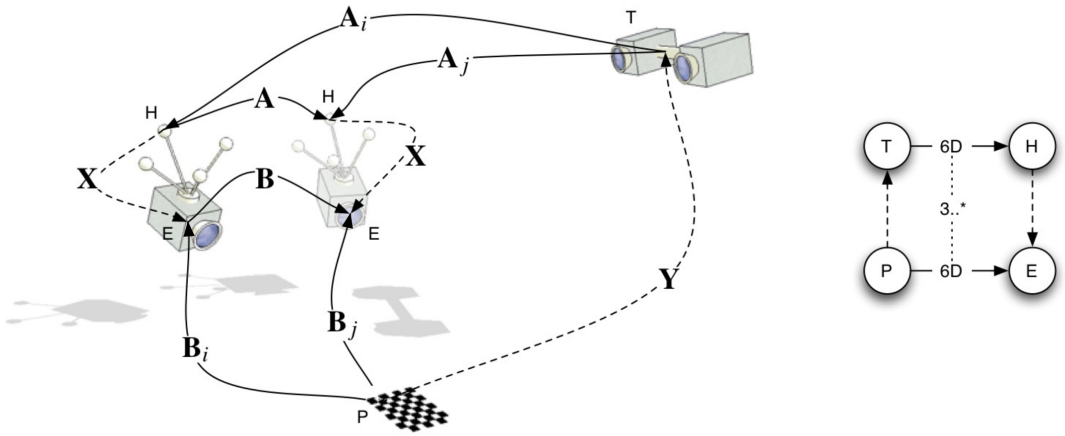


Abbildung 3.8: Veranschaulichung der Hand-Eye Kalibrierung. (basierend auf [15])

Hierzu bestehen verschiedene Lösungsmethoden, die – bis auf vereinzelte Ausnahmen – eine Gleichung der Form

$$AX = XB \tag{3.61}$$

lösen. Die homogenen Transformationsmatrizen A, B und X gehen aus Abbildung 3.8 hervor. Das Koordinatensystem des Tracking-System T ist in diesem Fall das Referenzkoordinatensystem. Die Position der Sensoreinheit des Referenzsystems auf dem Roboter H im Referenzkoordinatensystem wird durch die Transformation A_i bzw. A_j beschrieben. B_i und B_j enthalten die Pose der Kamera E im SLAM-Koordinatensystem, das durch das AprilTag Bundle P vorgegeben wird. Daraus berechnen sich die Matrizen A und B:

$$A = A_i^{-1} A_j \tag{3.62}$$

$$B = B_i^{-1} B_j \tag{3.63}$$

Somit resultieren aus den gemessenen Posen des SLAM-Systems und des Referenzsystems die Matrizen A und B, anhand derer die gesuchte Transformation X aus der Lösung der Gleichung 3.61 errechnet wird.

Um die gemessenen Posen zu vergleichen, muss zusätzlich Y berechnet werden.

$$Y = B_i \, X^{-1} \, A_i^{-1} = B_j \, X^{-1} \, A_j^{-1} \tag{3.64}$$

Um Gleichung (3.61) zu lösen, wird diese in zwei Gleichungen zerlegt.

$$R_A R_X = R_X R_B \tag{3.65}$$

hängt ausschließlich von den Rotationsmatrizen R_A, R_B und R_X der homogenen Transformationsmatrizen A, B und X ab und wird deshalb zuerst gelöst. Die zweite Gleichung beinhaltet zusätzlich die Translationsvektoren t_A, t_B und t_X:

$$(R_A - I) t_X = R_X t_B - t_A \tag{3.66}$$

Gleichung (3.65) kann durch rechtsseitige Multiplikation mit R_X^T als Ähnlichkeitstransformation umgeschrieben werden.

$$R_A = R_X R_B R_X^T \tag{3.67}$$

Daraus folgt, dass R_A und R_B die selben Eigenwerte besitzen. Einer dieser Eigenwerte ist bei Rotationsmatrizen allgemein 1. Sei n_B der zugehörige Eigenvektor von R_B, ergibt sich durch rechtsseitige Multiplikation

$$R_A R_X n_B = R_X R_B n_B \tag{3.68}$$

$$= R_X n_B \tag{3.69}$$

Daraus wird geschlossen, dass $R_X n_B$ der Eigenvektor n_A von R_A ist, der dem Eigenwert 1 zugeordnet ist. Somit kann an Stelle von Gleichung (3.61) folgende Gleichung gelöst werden:

$$n_A = R_X n_B \tag{3.70}$$

Der klassische Weg, die Gleichung zu lösen, wurde von Tsai und Lenz (1989) [48] vorgeschlagen. Dabei wird R_X durch seinen Einheitseigenvektor n_X und einen Winkel θ_X dargestellt. Dadurch lässt sich Gleichung (3.70) weiter vereinfachen.

Um das gesamte Gleichungssystem zu lösen sind insgesamt mindestens drei verschiedene Positionen nötig.

Kapitel 4

Implementierung

Auf Basis der erläuterten Grundlagen, wird in diesem Kapitel das vSLAM-System mit Hilfe des Frameworks ROS implementiert. Dafür werden zunächst geeignete vSLAM-Algorithmen ausgewählt und implementiert. Anschließend wird eine Methode entwickelt, durch die absolute Skaleninformationen aus den vSLAM-Posen gewonnen werden. Darauf baut ein Algorithmus auf, der aus der SLAM-Karte eine zweidimensionale Occupancy Grid Map erstellt. Schließlich wird das entwickelte vSLAM-System in das bestehende Navigationssystem des Zentrums für Telematik integriert.

4.1 Robot Operating System

Das Robot Operating System (ROS) ist ein Software Framework, das umfangreiche Bibliotheken für technische Funktionalitäten und nützliche Werkzeuge zur Visualisierung, Debugging, Beobachtung und Konfiguration von Programmen bereitstellt. Zudem verfügt ROS über ein Publisher/Subscriber System, über das die Prozesse miteinander kommunizieren. Im Folgenden werden die Kernelemente von ROS, die in dieser Arbeit eine Rolle spielen, beschrieben.

4.1.1 Nodes

Nodes sind Prozesse, die dynamisch gestartet und beendet werden können. Sie kommunizieren miteinander über Topics und werden über den ROS Parameter Server konfiguriert. Ein Softwaresystem besteht meist aus mehreren Nodes die ineinander greifen und somit eine verteilte Architektur implementieren. Hierdurch verringert sich die Komplexität bei der Entwicklung. Jedem Node ist eine ausführbare Datei zugeordnet, wobei der Programmcode in verschiedenen Sprachen verfasst sein kann. In dieser Arbeit wird vorrangig C++ verwendet.

4.1.2 Topics

Topics sind Informationskanäle, die über einen eindeutigen Namen identifiziert werden. Die Topics arbeiten nach einem anonymen Publisher/Subscriber Modell, bei dem mehrere beliebige Nodes eine Nachricht auf einem Topic veröffentlichen und empfangen können. Die Nachrichten sind streng typisiert, indem jedem Topic ein bestimmter Message Type zugeordnet ist.

4.1.3 Bags

Ein Bag speichert Nachrichten, die über einen bestimmten Zeitraum auf festgelegte Topics veröffentlicht wurden. Werkzeuge wie *rosbag* nehmen die Datei auf und rekonstruieren beim Abspielen der Datei die Nachrichtenfolge. So ist es möglich Sequenzen aufzunehmen und diese im Nachhinein zu visualisieren oder beispielsweise Messwerte erneut für Prozesse zu verwenden, ohne dass die benötigte Hardware eingesetzt wird. Gerade der letztgenannte Aspekt vereinfacht die Entwicklung mit Robotersystemen enorm.

4.1.4 Master

Der ROS Master ist der wichtigste Node, da er die Kommunikation und Zusammenarbeit zwischen den übrigen Nodes ermöglicht und koordiniert. Er verwaltet den Parameter Server, über den Parameter an einzelne Nodes übergeben werden. Befindet sich der festgelegte Master auf einem entfernten Gerät, können Nodes verschiedener Geräte miteinander kommunizieren.

4.1.5 roslaunch

Das Werkzeug *roslaunch* ermöglicht es, mehrere Nodes mit einem Befehl zu starten. Hierfür werden in XML-Dateien die zu startenden Nodes und deren Parameter festgelegt. Beim Start wird der Master automatisch gestartet, falls kein Master aktiv ist. Das Werkzeug ist besonders nützlich, um Systeme über SSH zu starten.

4.2 Visuelle Lokalisierung und Kartierung

In diesem Abschnitt werden zwei bestehende Algorithmen zur visuellen Lokalisierung implementiert, auf deren Basis das System zur absoluten Lokalisierung aufgebaut ist. Die vSLAM-Algorithmen erzeugen dabei eine Karte, die die Lokalisierung in einem zufällig initialisiertem – und damit nicht skalentreuem – Koordinatensystem ermöglicht.

Algorithmus	**LSD-SLAM**	**ORB-SLAM**	**LDSO**
Punktdichte	Semi Dense	Sparse	Sparse - Semi Dense
Methodik	Direkt und Dense	Indirekt	Direkt und Sparse
Globale Optimierung	Ja		
Loop Closure	Ja		

Tabelle 4.1: Vergleich der populärsten vSLAM Algorithmen

4.2.1 Auswahl der Algorithmen

Für visuelle Lokalisierung kommen verschiedenste Systeme in Frage. vSLAM-Algorithmen für RGB-D Kameras erzeugen genaue Punktwolken mit hoher Dichte. Allerdings sind RGB-D Kameras meist teurer als monokulare Kameras und weniger häufig auf bestehenden Roboterplattformen vorhanden. Je nach Funktionsweise ist ein zusätzliches Problem, dass die Kameras nur für Innenräume geeignet sind.

Systeme mit Stereokameras haben gegenüber monokularen Systemen den Vorteil, dass absolute Tiefeninformationen aus jedem Bildpaar gewonnen werden können. Hierfür ist allerdings eine hochgenaue und aufwändige Kalibrierung notwendig, die bei kleinsten mechanischen Änderungen, die durch Erschütterungen auftreten, erneuert werden muss. Zudem sind Stereokameras, wie auch RGB-D Kameras, meist teurer als monokulare Kameras.

Aus diesen Gründen wird sich für die Verwendung eines monokularen Systems entschieden. Diese Systeme können durch Informationen zusätzlicher Sensorik, wie Radencodern oder inertiale Messeinheiten, angereichert werden. In dieser Arbeit wird sich nichtsdestotrotz auf eine rein visuelle Lokalisierung und Kartierung beschränkt. Das hat zum Vorteil, dass bestehende Systeme unkomplizierter erweitert werden können.

Es existieren einige monokulare vSLAM-Algorithmen, die bereits eine hohe Genauigkeit erzielen. Deshalb wird die Neuentwicklung eines vSLAM-Algorithmus für diese Arbeit als nicht notwendig erachtet.

Die populärsten Algorithmen auf dem Stand der Technik sind ORB-SLAM [37] (bzw. dessen Nachfolger ORB-SLAM2 [38] und ORB-SLAM3 [7]), LSD-SLAM [13] und dessen Nachfolger LDSO [21]. Die genannten Algorithmen wurden von J. Wang und M. Shahbazi 2019 [49] anhand einer Drohne, deren Umgebung von einem hochgenauem Laserscanner erfasst wurde, evaluiert und verglichen. Die Hauptmerkmale sind in Tabelle 4.1 zusammengefasst. Direct Sparse Mapping (DSM) [53], das aktuellere Pendant der Universität Saragossa zu LDSO, wird nicht berücksichtigt, da es weder Loop Closing noch Relokalisierung implementiert.

Ein grundlegender Unterschied zwischen den Verfahren ist die Dichte der erzeugten Punktwolke. ORB-SLAM ermittelt als indirekte Methode lediglich die 3D Punkte der erkannten Features, weshalb die Punktwolke dünn besetzt ist. LSD-SLAM erzeugt dagegen 3D Punkte für alle Bildbereiche deren Intensitätsgradient ausreichend hoch ist. Dadurch wird eine wesentlich dichtere Punktwolke erstellt. Eine noch dichtere Punktwolke erzielt das Verwenden einer RGB-D Kamera

oder einer zusätzlichen Photogrammetrie Software. Diese benötigt allerdings viel Rechenleistung und Zeit, weshalb am Ansatz über den vSLAM-Algorithmus festgehalten wird.

Bei LDSO handelt es sich um Direct Sparse Odometry (DSO), bei der die Vorteile direkter Ansätze mit der Flexibilität von indirekten Methoden kombiniert werden. Die Dichte der Punktwolke ist entsprechend geringer als bei LSD-SLAM, kann allerdings künstlich durch die Anpassung des Schwellwerts für den Intensitätsgradienten erhöht werden. Diese Maßnahme geht allerdings mit einer Verschlechterung der Genauigkeit einher. Abbildung 4.1 zeigt sowohl die Dichte als auch die Genauigkeit der von den vSLAM-Algorithmen erzeugten Punktwolken. Ausreißer wurden zuvor aus den Punktwolken entfernt. Es ist klar zu erkennen, dass LSD-SLAM in diesem Beispiel die dichteste Punktwolke erzeugen konnte. ORB-SLAM erreicht die höchste Genauigkeit, wohingegen LDSO im Vergleich die geringste Genauigkeit bei einer ähnlich dichten Punktwolke wie LSD-SLAM aufweist.

LSD-SLAM ist nach LDSO und ORB-SLAM am stärksten von Rauschen betroffen. Zusammenfassend lässt sich dennoch sagen, dass nach Wang und Shahbazi [49] LSD-SLAM für Anwendungen in der Robotik am besten geeignet ist, da die Genauigkeit ausreichend und die Punktwolke am dichtesten ist.

Der Vergleich von Filipenko und Afanasyev [16] kommt zu einem ähnlichen Ergebnis und Krul et al. [29] sehen ORB-SLAM vor allem wegen der Genauigkeit eindeutig vorn.

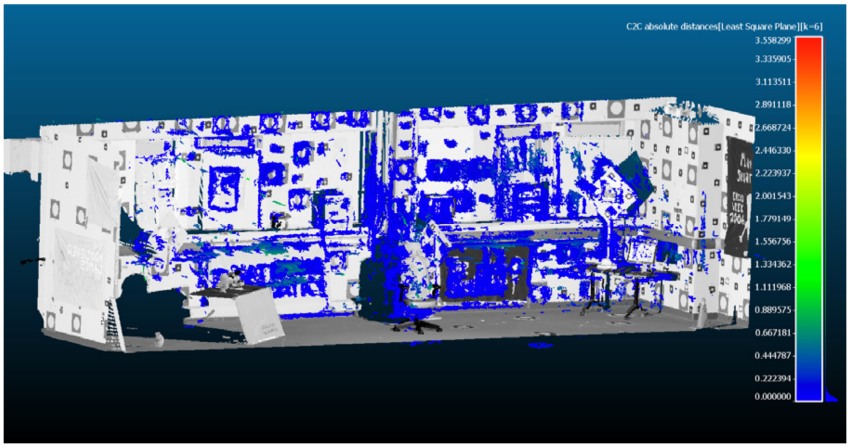
Im Rahmen dieser Arbeit spielen folgende Aspekte bei der Auswahl eines geeigneten Algorithmus eine Rolle:

- Genauigkeit der Lokalisierung
- Genauigkeit der erzeugten 3D-Punktwolke
- Dichte der 3D-Punktwolke
- Öffentliche Verfügbarkeit

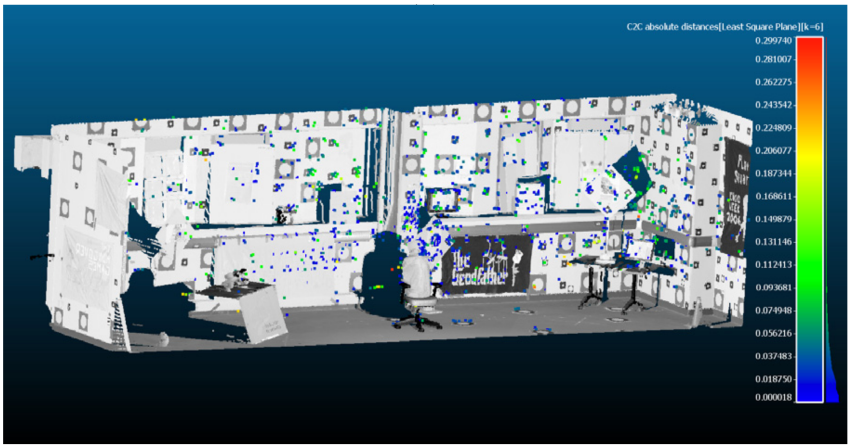
Die Genauigkeit der Lokalisierung ist bei allen Algorithmen hoch und für Anwendungen zur Freien Navigation in einer Industrieumgebung definitiv ausreichend. Bezüglich der Dichte und Genauigkeit der Punktwolke ist LSD-SLAM die beste Lösung für diese Arbeit, da eine möglichst dichte Punktwolke für Anwendungen in der Navigation essentiell ist. Gerade bei Objekten in der Umgebung ist es zur Kollisionsvermeidung erforderlich, ausreichend viele 3D-Punkte der Objekte zu erkennen. Dafür muss der Kompromiss mit einer schlechteren Genauigkeit der 3D-Punkte eingegangen werden.

Die drei genannten Algorithmen sind alle öffentlich verfügbar, wobei Anpassungen aufgrund von Änderungen der verwendeten Bibliotheken und Frameworks, wie ROS und OpenCV, nötig sind. Deshalb ist der Implementierungsaufwand aufgrund des Entwicklungszeitpunkts für LSD-SLAM am höchsten.

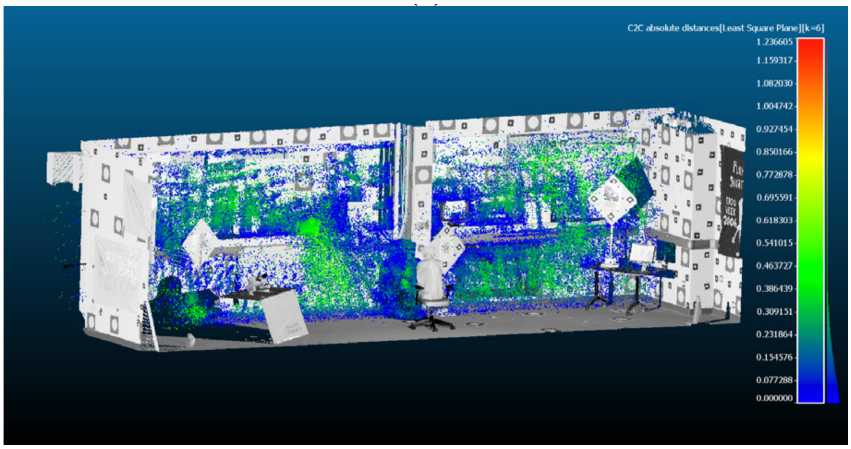
Insgesamt stand auf Grundlage der erläuterten Vorteile nichtsdestotrotz fest, dass im Rahmen dieser Arbeit zunächst LSD-SLAM implementiert wird. Da in der Praxis aber nicht die gewünschte Genauigkeit erzielt wurde, wird zusätzlich ORB-SLAM miteinbezogen.



(a) LSD-SLAM



(b) ORB-SLAM



(c) LDSO

Abbildung 4.1: Veranschaulichung der Cloud-to-Cloud (C2C) Distanzen zwischen der Laserscanner Punktwolke und den Punktwolken der vSLAM-Algorithmen. [49]

4.2.2 LSD-SLAM

Als Umgebung für die Entwicklung wurde das Betriebssystem Ubuntu 20.04 LTS gewählt. Da LSD-SLAM, wie auch ein Großteil der begleitenden Projekte des Zentrums für Telematik, das Framework ROS verwendet, sollen auch die Funktionalitäten, die im Rahmen dieser Arbeit entwickelt werden, mit ROS kompatibel sein.

Zunächst wird ROS hierfür in der Version *noetic* installiert. Die öffentlich verfügbare Version von LSD-SLAM [1] von 2014 wurde für Ubuntu 14.04 und ROS *indigo/fuerte* entwickelt und getestet. Das Projekt wurde in der Programmiersprache C++ entwickelt. Da sich mit den Folgeversionen das Buildsystem von *rosbuild* auf *catkin* geändert hat, werden die Konfigurationsdateien diesbezüglich angepasst bzw. erweitert.

Zudem haben sich seitdem die Schnittstellen zu verwendeten Frameworks geändert. Infolgedessen werden zunächst einige Änderungen im Code und den Konfigurationsdateien des Buildsystems CMake vorgenommen. Diese betreffen die Änderungen der Bibliotheken Qt und OpenCV, sowie des Frameworks G2o.

Für OpenCV ist eine herkömmliche Installation nicht ausreichend, da das Framework OpenFabMap, das LSD-SLAM für größere Loop-Closures verwendet, auf teils als kommerziell vorgesehene Erweiterungen von OpenCV zugreift. Deshalb wird OpenCV vom Quellcode mit einem Verweis auf die zusätzlichen Module gebuildet.

LSD-SLAM verfügt bereits über entsprechende ROS Nodes zur Ausführung des vSLAM-Algorithmus auf Bilder einer direkt zugeschalteten Kamera. Diese müssen zusammen mit den Kameraparametern auf einem entsprechenden ROS Topic veröffentlicht werden und werden von dort aus von LSD-SLAM abgegriffen und verarbeitet. Dafür wird der ROS Node *image_capture_node* implementiert, der mithilfe des Spinnaker SDK Bilder aufnimmt, diese als OpenCV `cv::Mat` umwandelt und bei Bedarf entzerrt. Die Bilder werden auf dem ROS Topic `image_raw` als `sensor_msgs/Image` und die Kameraparameter auf `camera_info` als `sensor_msgs/CameraInfo` veröffentlicht. LSD-SLAM gibt vor, dass Höhe und Breite der Bilder ein Vielfaches von 16 sein müssen. Der ROS Node *LSD_SLAM* übernimmt hierbei die Lokalisierung und Kartierung nach der Vorgehensweise aus Abschnitt 2.1.3. Die vom Algorithmus errechnete Schätzung der aktuellen Pose wird als `geometry_msgs/PoseStamped` auf dem Topic `/lsd_slam/pose` veröffentlicht und kann somit von jedem ROS Node verwendet werden. Durch Tastatureingaben können verschiedene Debug-Informationen visualisiert, die aktuelle Karte in Form der erstellten Keyframes mit Tiefeninformationen gesichert und Relokalisierung eingeleitet werden.

Der *viewer* Node visualisiert die vom vSLAM-Algorithmus erzeugte Punktwolke, die er aus den Keyframes und dem Keyframegraph erzeugt und darin den Keyframegraphen einzeichnet. Dabei werden auch die Posen der Keyframes visualisiert. Eine beispielhafte Punktwolke ist in Abbildung 4.3 zu sehen. Der Node arbeitet entweder live, indem er die auf dem Topic `/lsd_slam/keyframes` veröffentlichten Keyframes und den aktuellen Keyframegraphen von `/lsd_slam/graph` visualisiert oder im Nachhinein durch einen Verweis auf eine ROS Bag Datei, die die genannten Daten enthält. Des Weiteren verfügt der *viewer* Node über einige nützliche

[1] https://github.com/tum-vision/lsd_slam (Branch: `catkin`)

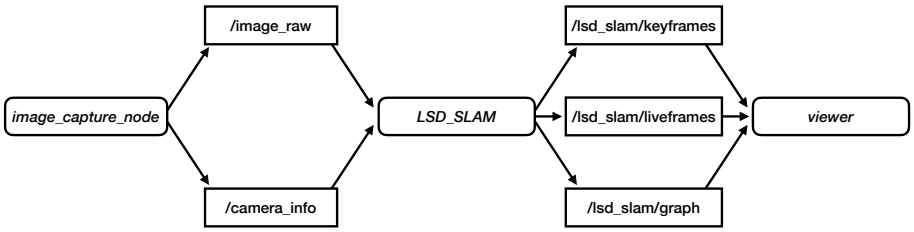


Abbildung 4.2: Überblick über ROS Nodes und Topics bei LSD-SLAM.

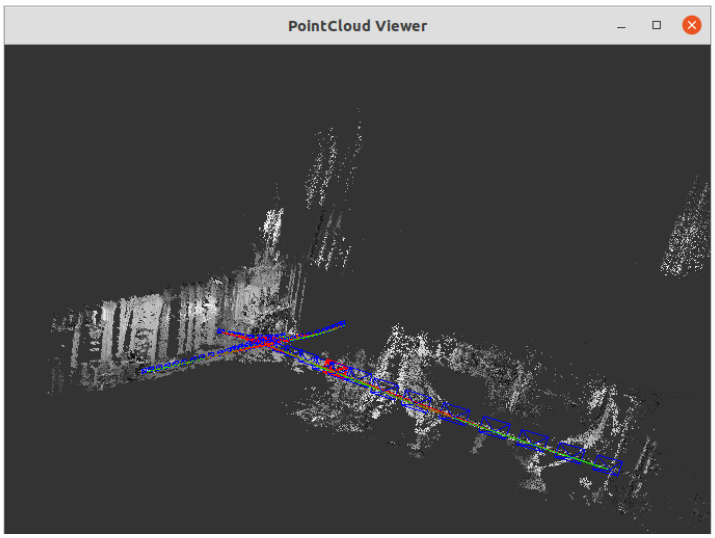


Abbildung 4.3: Bildschirmfoto des LSD-SLAM PointCloud Viewer Fensters

Funktionalitäten im Bezug auf die Weiterverarbeitung der Punktwolke. Solange die Punktwolke im RAM gehalten wird, können Schwellwerte bezüglich der Varianz der Punkte und ein Dichtheitsfaktor angegeben werden, nachdem die angezeigte Punktwolke gefiltert wird. Die gefilterte Punktwolke kann als binär kodierte *PLY*-Datei gesichert werden.

Eine Übersicht über alle ROS Nodes und Topics bei einer minimalistischen Anwendung von LSD-SLAM im *live*-Modus ist in Abbildung 4.2 dargestellt.

4.2.3 Anpassungen – LSD-SLAM

Um den Algorithmus in einer Testumgebung anzuwenden, werden einige Parameter angepasst. Diese Parameter werden je nach Umgebung und Kamera gewählt.

Ein wichtiger Schwellwert ist der **Instensitätsgradient**, der mindestens notwendig ist, um einen Pixel in die photometrische Fehlerfunktion aufzunehmen und den Pixel somit in das Tracking miteinzubeziehen. Je höher der Wert ist, desto besser können die Pixel bei der Verfeinerung der

Keyframes denen anderer Bilder zugeordnet werden. Die Tiefeninformationen der Keyframes werden also genauer. Allerdings verschlechtert sich das Tracking, je weniger Pixel einbezogen werden. Bei Aufnahmen mit besonders schwacher Textur kann das Tracking sogar fehlschlagen und die Relokalisierung wird eingeleitet. Geschieht das während der Kartierung, weist die Karte entsprechende Schwächen auf. Der Schwellwert für den minimalen Intensitätsgradienten muss demnach je nach Kameraauflösung (je höher die Auflösung, umso geringer sind die Intensitätsgradienten im Bild) und Textur der Umgebung angepasst werden, sodass ein Optimum an genauen Tiefeninformationen und ausreichend vielen Pixeln für das Tracking erzielt wird.

Die **Anzahl an Keyframes** ist ebenso entscheidend für die Güte des Tracking. Je näher ein Frame an einem Keyframe ist, desto genauer kann seine relative Pose geschätzt werden. Zudem werden mehr Loop Closures und Abhängigkeiten zwischen Keyframes gefunden, je mehr Keyframes vorhanden sind und je dichter diese aneinander liegen. Grundsätzlich verbessert sich das Tracking deshalb mit der Anzahl an Keyframes bis zu einem bestimmten Punkt. Da die Keyframes durch die folgenden Frames verfeinert werden, werden die Informationen eines Keyframes genauer und umfangreicher, je mehr Folgeframes das Keyframe verfeinern. Werden zu schnell neue Keyframes erstellt, leidet die Qualität der einzelnen Keyframes darunter und das Tracking (und damit letztendlich auch das Mapping) verschlechtert sich. Hinzu kommen Begrenzungen durch die verwendete Hardware. Gerade bei höheren Auflösungen ab 1280×960 erreicht die Karte, bestehend aus den Keyframes, schnell eine Größe von mehreren Giga Bytes. Da die gesamte Karte im Arbeitsspeicher gehalten wird, ist die Gesamtanzahl an Keyframes hierdurch beschränkt. Die Anzahl an Keyframes wird erhöht, indem der Schwellwert für die Erstellung eines neuen Keyframes (vgl. Abschnitt 2.1.3, Gleichung (2.7)) verringert wird oder die Gewichtungsmatrix W entsprechend erhöht wird, sodass der Schwellwert früher überschritten wird.

Für eine genaue Karte ist die Genauigkeit der Keyframeposen von essentieller Bedeutung, da die Lokalisierung jeweils relativ zum nächsten Keyframe erfolgt. Diese Genauigkeit profitiert von einem Keyframegraphen mit möglichst vielen guten Kanten, das heißt möglichst vielen erkannten **Korrespondenzen zwischen den Keyframes**, anhand derer die relativen Posen der Keyframes zueinander bestimmt werden. Die Genauigkeit der Keyframeposen wird dadurch mit Hilfe von Graph-Optimization-Algorithmen erhöht. Dieses Verfahren ist vor allem zur Bewältigung des Skalendrifts dringend notwendig. Hier spielen die sogenannten Large Loop Closures eine große Rolle, bei denen Korrespondenzen zum Schließen großer Schleifen gefunden werden. Um die Anzahl der Kanten zu erhöhen, kann die Zahl der zuletzt erstellten Keyframes, in denen nach Korrespondenzen gesucht wird, oder der die maximale Ungenauigkeit, die beim Tracking zweier Keyframes zueinander auftreten darf, erhöht werden. Hierdurch werden allerdings auch mehr fehlerhafte Kanten einbezogen, die die Genauigkeit verschlechtern.

Zusätzlich empfehlen die Entwickler, das Rauschen der Kamera (engl. *Pixel Noise*) der verwendeten Kamera entsprechend anzupassen. Dieses geht in die Varianz der Tiefeninformationen für jeden Pixel ein. In der Praxis konnte allerdings keine signifikante Verbesserung durch die Anpassung dieses Parameters erzielt werden.

Zur Verwendung von LSD-SLAM für kommerzielle Anwendungen muss der SURF Detektor in `FabMap.cpp` aus patentrechtlichen Gründen ersetzt werden. Für wissenschaftliche Zwecke ist das nicht notwendig.

4.2.4 ORB-SLAM3

Als Alternative zu LSD-SLAM wird ORB-SLAM3 auf Grundlage der Beta Version V0.4 [2] implementiert. Die Version setzt bereits die in Abschnitt 2.1.2 erläuterten Techniken ein und lässt sich ohne signifikante Anpassungen kompilieren. Das Sichern und Laden von erstellten Karten wird noch nicht unterstützt und im Rahmen dieser Arbeit implementiert.

Hierfür werden in der Klasse `System` jeweils eine Funktion zum Sichern und Laden der Karte implementiert. Die Funktion zum Sichern wird beim Abbruch des Programms aufgerufen, falls ein entsprechendes Flag beim Ausführen des Programms gesetzt wurde. Eine Karte wird geladen, falls zu Beginn der Ausführung eine Karte mit dem im Settings-File spezifizierten Namen existiert. Die geladene Karte wird anschließend in den Atlas mit aufgenommen, mit einer passenden ID versehen und als aktuelle Karte markiert, sodass das vSLAM-System sie verwendet.

Zudem wird für die geladenen Keyframes das BoW Vokabular auf das aktuell verwendete geändert und die BoW Representation neu berechnet. Die für die Klasse `KeyFrameDatabase` bestehende Funktion `SetORBVocabulary`, wird hierfür so angepasst werden, dass sich ausschließlich das Vokabular ändert.

Für das Laden und Sichern einer Karte wurden die Serialisierungsmethoden der Boost Bibliothek verwendet. Für alle Klassen aus ORB-SLAM, BoW und OpenCV, für die noch keine `serialize()` Funktion existiert, werden diese implementiert. Dafür ist auch ein Konstruktor ohne Parameter notwendig.

Wird eine geladene Karte verwendet, wechselt der Algorithmus automatisch in den Lokalisierungsmodus und stoppt damit das Mapping. Der Lokalisierungsmodus kann über die Benutzeroberfläche jederzeit wieder manuell verlassen bzw. betreten werden. Für das Tracking sind neue Zustände möglich, die zuvor nicht berücksichtigt wurden. Ist das Tracking nicht initialisiert und befindet sich im Lokalisierungsmodus, wird die Relokalisierung eingeleitet. Da die Relokalisierung im Lokalisierungsmodus auf dem aktuellen Stand nicht funktionsfähig ist, wurde der Code so angepasst, dass der Lokalisierungsmodus für die Relokalisierung verlassen wird. Das Problem liegt darin, dass mit Einführung des SLAM-Atlas eine globale Relokalisierung dadurch erfolgt, dass zunächst eine neue Karte angelegt wird und im Hintergrund versucht wird, diese mit der vorherigen Karte zu fusionieren. Eine globale Relokalisierung, bei der das Mapping deaktiviert ist, scheint in der aktuellen Version nicht vorgesehen zu sein, weshalb aus Gründen der Praktikabilität der beschriebene Workaround Anwendung findet.

Auch die Schnittstellen zu ROS werden erweitert. In der verwendeten Version ist lediglich ein Beispielcode vorhanden, der ORB-SLAM mit einem bestimmten Image Topic als Eingabeschnittstelle startet. Dieser wird um einen Publisher erweitert, der die von ORB-SLAM errechnete Pose als `geometry_msgs::PoseStamped` auf dem Topic `/orb_pose` veröffentlicht.

Zum Ausführen des Algorithmus wird eine Datei mit dem Bags of Words Vokabular und eine Settings Datei im *yaml* Format übergeben. Die mitgelieferte Datei `Vocabulary/ORBvoc.txt` eignet sich gut als Vokabular. Bei der Settings Datei dienen entsprechende Dateien, die für die

[2] https://github.com/UZ-SLAMLab/ORB_SLAM3

Anwendung auf populäre Datensätze (KITTI etc.) vorgesehen sind, als Orientierung. Neben den Algorithmus-spezifischen Parametern, die beispielsweise die Featuredetektion parametrisieren, werden die Kameraparameter und Viewer-Einstellungen angegeben. Für die Erweiterung zum Laden einer Karte wird der entsprechende Dateiname spezifiziert.

4.3 Skalenkalibrierung vom SLAM- ins Weltkoordinatensystem

Wie bereits erwähnt, ermitteln monokulare vSLAM-Algorithmen keine absoluten Skaleninformationen ihrer Karte. Da diese Informationen für die Navigation auf Grundlage eines vSLAM-Algorithmus notwendig sind, wird in diesem Abschnitt eine Methode entwickelt, die es ermöglicht die Posen im Koordinatensystem der SLAM-Karte in ein globales Koordinatensystem zu übertragen.

Der Ansatz hierbei ist, die Pose der Kamera im Weltkoordinatensystem über einen kurzen Zeitraum – neben der Lokalisierung durch den vSLAM-Algorithmus – durch ein zusätzliches Verfahren zu bestimmen. Aus den korrespondierenden Posen in Welt- und SLAM-Koordinatensystem wird die Transformation zwischen den Koordinatensystemen berechnet. Als alternatives Lokalisierungsverfahren wird die Fiducial Marker Localization gewählt, da hierdurch weiterhin nur die Kamera als Sensor notwendig ist und genaue Posenschätzungen durchgeführt werden können.

Zur Errechnung einer homogenen Transformation vom SLAM-Koordinatensystem in das Weltkoordinatensystem werden korrespondierende Posen in den jeweiligen Koordinatensystemen verwendet. Die Posen im Weltkoordinatensystem werden bestimmt, indem in Bildern ein AprilTag Bundle detektiert wird und mit den Eckpunkten und Zentren der Tags ein PnP Problem formuliert und gelöst wird (vgl. Abschnitt 3.4). Für die Genauigkeit der Transformation ist die Genauigkeit, Anzahl und räumliche Verteilung der Posen von Bedeutung. Die räumliche Verteilung ist durch die Kinematik des Roboters begrenzt. Es muss, darauf geachtet werden, dass die Posen möglichst gut verteilt sind. Die Genauigkeit der Posen wird gewährleistet, indem für die Lokalisierungsverfahren scharfe Bilder einer gut kalibrierten Kamera verwendet werden. Eine zusätzliche Verbesserung wird dadurch erzielt, dass mehrere Bilder aufgenommen werden ohne dass die Kamera bewegt wird. Aus den Posen, die aus den Bildern errechnet werden, werden Mittelwerte gebildet, wodurch Ausreißer gefiltert werden. Zudem ist es somit möglich die ermittelten Posen nach ihren Varianzen und Reprojektionsfehlern (der PnP Lösung) auszuwählen.

Um die Transformation vom SLAM-Koordinatensystem ins Weltkoordinatensystem zu berechnen, reichen die Positionen der Kamera aus, mit denen eine klassische Point Set Registration, beispielsweise mit dem ICP Algorithmus (vgl. Abschnitt 3.5), durchgeführt wird.

Korrespondenz-basierte Methoden gehen davon aus, dass jedem Punkt einer Punktmenge M ein Punkt in einer Punktmenge S zugeordnet werden kann. Im einfachsten Fall wird für die korrespondierenden Punkte $\mathbf{s}_i$ und $\mathbf{m}_i$ folgendes angenommen:

$$\mathbf{s}_i = l\ \mathbf{R}\ \mathbf{m}_i + \mathbf{t} + \varepsilon_i\ , \tag{4.1}$$

wobei die Ähnlichkeitstransformation durch eine Rotation $\mathbf{R}$, Translation $\mathbf{t}$ und einen Skalierungsfaktor l beschrieben wird und mit unbekanntem Rauschen ε_i versehen ist. Unter der

Annahme von weißem Rauschen mit einer Gaußschen Normalverteilung $\mathcal{N}(0, \sigma_i^2 I_3)$ lautet der Maximum Likelihood Schätzer:

$$l^*,\ \mathbf{R}^*,\ \mathbf{t}^* = \arg\min_{l,\,\mathbf{R},\,\mathbf{t}} \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \|\mathbf{s}_i - l\mathbf{R}\mathbf{m}_i - \mathbf{t}\|^2 \quad \text{mit } l > 0,\ \mathbf{R} \in \mathrm{SO}(3),\ \mathbf{t} \in \mathbb{R}^3 \tag{4.2}$$

Um daraus eine eindeutige Lösung zu erhalten (siehe Horn [26] und Arun et al. [1]), werden mindestens $N = 3$ nicht kollineare Punkte benötigt. Allerdings dürfen Ausreißer nicht berücksichtigt werden, um genaue Ergebnisse zu erzielen. Diese werden entweder im Vorhinein entfernt, oder beispielsweise indem die einzelnen Abweichungen der Punktepaare durch eine spezielle Kostenfunktion gewichtet werden.

4.3.1 Übersicht

Die Funktionalität zur Skalenrekonstruktion, die für eine absolute Lokalisierung notwendig ist, wird auf mehrere ROS Nodes verteilt implementiert. Ein Node, der Posen anhand von AprilTags errechnet und auf ein ROS Topic veröffentlicht, läuft parallel zum SLAM-Node. Ein dritter Node (*vSLAM_master*), verarbeitet die veröffentlichten Posen, indem er zunächst auf Grundlage von korrespondierenden Posen im SLAM- und Weltkoordinatensystem die Transformation vom SLAM- in das Weltkoordinatensystem errechnet und von da an die vom SLAM-Node veröffentlichte Pose in das Weltkoordinatensystem transformiert und wiederum auf einem ROS Topic veröffentlicht.

Die Nodes lassen sich durch eine *roslaunch* Datei gleichzeitig starten. In der Datei sind zusätzlich die passenden Parameter spezifiziert, sodass das System mit einem einfachen Kommando gestartet werden kann.

4.3.2 Lokalisierung mit AprilTags

Zur Berechnung der Kamerapose wurde ein unabhängiger ROS Node (*apriltag_ros_pnp_solver_node*) implementiert. Dieser greift, wie der vSLAM-Algorithmus, auf ein ROS Topic mit den aufgenommenen Bildern zu, die vom Node *image_capture_node* (vgl. Abschnitt 4.2.2) in verschiedenen Auflösungen veröffentlicht werden. Hierbei wurde darauf geachtet, dass die veröffentlichten Bilder verschiedener Auflösung mit dem jeweils selben Zeitstempel versehen sind. Der Zeitstempel der Bilder wird bei der Bestimmung der Pose auf Grundlage der jeweiligen Bilder übernommen. So ist es möglich, dass die Pose, die der vSLAM-Algorithmus ermittelt, der Pose relativ zu den AprilTags zugeordnet werden kann.

Der *apriltag_ros_pnp_solver_node* verwendet Bilder mit möglichst hoher Auflösung. Aus diesen werden mit Hilfe der AprilTag 3 Bibliothek[3] des APRIL Robotics Laboratory der University of Michigan die Bildkoordinaten der gefundenen Tags extrahiert und die jeweiligen Identifikationsnummern ermittelt. Da die Anordnung der Tags bekannt ist, wird aus der Identifikationsnummer die Pose der Tags im Weltkoordinatensystem ermittelt.

[3] https://github.com/AprilRobotics/apriltag

Mit der Funktion `solvePnP()` der Bibliothek OpenCV wird aus den Korrespondenzen aus zweidimensionalen Bildkoordinaten mit dreidimensionalen Weltkoordinaten und den intrinsischen Kameraparametern das zugehörige PnP Problem gelöst und damit die Transformation des Weltkoordinatensystems in das Kamerakoordinatensystem bestimmt. Um das iterative Verfahren als Lösungsalgorithmus zu wählen wird ein entsprechendes Flag gesetzt.

Der erhaltene Orientierungsvektor $\mathbf{r} = [r_x, r_y, r_z]^T$ wird im Anschluss mit der Rodrigues-Formel in eine Rotationsmatrix $\mathbf{R}$ umgerechnet:

$$\mathbf{R} = \cos(\|\mathbf{r}\|)\mathbf{I} + (1 - \cos(\|\mathbf{r}\|)) \left[\frac{\mathbf{r}}{\|\mathbf{r}\|}\right] \left[\frac{\mathbf{r}}{\|\mathbf{r}\|}\right]^T + \sin(\|\mathbf{r}\|) \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix} \tag{4.3}$$

Hiermit kann nun die inverse Transformation berechnet werden, die der Pose der Kamera im Weltkoordinatensystem entspricht:

$$\mathbf{R}' = \mathbf{R}^T \tag{4.4}$$

$$\mathbf{t}' = -\mathbf{R}'\mathbf{t} \tag{4.5}$$

Um die Pose als allgemein übliche `geometry_msgs/PoseStamped` zu veröffentlichen, wird die Rotationsmatrix $\mathbf{R}'$ in Quaternionen umgewandelt. Die Position entspricht den ersten drei Komponenten von $\mathbf{t}'$.

Werden nicht alle Tags des Bundles entdeckt, deutet das auf Störungen und damit auf eine hohe Ungenauigkeit der ermittelten Pose hin. Alternativ ist es möglich, dass das Bundle nicht vollständig im Bild enthalten ist. Auch wenn hier keine Störung vorliegt, wird nur eine geringere Genauigkeit erreicht, da sich die detektierten Tags im Randbereich des Bildes ($\Rightarrow$ starke Verzeichnung) befinden und der numerische Vorteil einer höheren Anzahl an Tags nicht ausgenutzt wird. Aus diesen Gründen wird die ermittelte Pose verworfen, falls weniger als fünf Tags von insgesamt 24 Tags detektiert werden, da aus anekdotischer Evidenz bekannt ist, dass für die Anwendung zur Skalenrekonstruktion die Qualität der Daten wichtiger ist als deren Quantität.

Auch wenn das quadratische Mittel (RMS) des Reprojektionsfehlers bei der Lösung des PnP Problems zu groß ist, wird die Pose nicht veröffentlicht. Hierfür wurde ein empirischer Schwellwert von 1 gewählt. Der erwartete Reprojektionsfehler einer guten Posenschätzung liegt dabei deutlich niedriger. Da der Reprojektionsfehler maßgeblich von der Größe der Tags im Bild und damit von Neigung und Entfernung abhängt, wird er lediglich dazu verwendet, Ausreißer, bei denen die genaue Posenschätzung vermutlich fehlschlug, zu erkennen.

4.3.3 Bestimmung der Transformation

Solange sich der *vSLAM_master* Node im Initialisierungszustand befindet, werden korrespondierende Positionen gesammelt. Dabei wird ein Puffer von 30 Positionspaaren aufrechterhalten. Liegt die Varianz der Positionen im Weltkoordinatensystem unter einem Schwellwert von

$10^{-7}\,\mathrm{m}^2$, was einer Standardabweichung von etwa 0,32 mm entspricht, wird ein Tupel gespeichert. So wird garantiert, dass sich die Kamera in den letzten 30 Bildern nicht bewegt hat und dass das Rauschen bei der Positionsbestimmung gering ist. Ein geringes Rauschen spricht in den meisten Fällen für eine hohe Genauigkeit. Das aufgenommene Tupel errechnet sich aus dem Mittelwert der Positionen, die vom Mittelwert der im Puffer gehaltenen Positionen weniger als 2σ abweichen. Hierdurch werden Ausreißer bei der Lösung der PnP Probleme verworfen.

Bei einer üblichen Bildrate von 10 Hz entsprechen 30 Positionspaare einer Aufnahmedauer von 3 s. Diese Zahl stellt sich als guter Kompromiss zwischen einer möglichst hohen Anzahl an Werten für die Mittelwertbildung und einer praktikablen Aufnahmedauer für den Initialisierungsprozess dar. Der Varianzschwellwert von $10^{-7}\,\mathrm{m}^2$ wurde empirisch festgelegt, indem die Varianz bei stehendem Roboter und guter Sicht auf das Tag Bundle aus verschiedenen Perspektiven, Abständen und in verschiedenen Regionen des Bilds verglichen wurde. Der Schwellwert ist entsprechend anzupassen, falls sich das verwendete Bundle in seiner Größe, Position oder Ausrichtung ändert. Für die Experimente im Rahmen dieser Arbeit wurde ein Pattern verwendet, das aus 24 April-Tags zu einer Seitenlänge von je 68 mm besteht und senkrecht auf Bodenhöhe an einer Wand stehen. Die Tags befinden sich hierbei auf foliertem Glas, um zu erzielen, dass alle Tags exakt auf einer Ebene und frei von Verzerrungen sind. Die senkrechte Ausrichtung ermöglicht eine unkompliziertere Reduktion der Positionsbestimmung von drei auf zwei Dimensionen. Das direkte Bedrucken des Bodens in einer Industriehalle mit den AprilTags hat den Vorteil, dass der Ursprung des Weltkoordinatensystems, der sich in dieser Implementierung im Zentrum des Tags mit der niedrigsten Identifikationsnummer befindet, auf dem Boden liegt und die z-Achse somit die Höhe im Raum angibt. Zudem ist das Bundle so rundum sichtbar, wohingegen ein senkrecht angebrachtes Bundle nur von einer Seite sichtbar ist. Allerdings ist die Kamera auf dem Roboter fest in horizontaler Ausrichtung angebracht, weshalb horizontale Tags so stets in einem unvorteilhaften Winkel aufgenommen werden. Aus diesen Gründen eignen sich senkrecht angebrachte Tags im Rahmen dieser Arbeit am besten.

Ist eine Mindestanzahl von acht bzw. 25 Positionstupeln aufgenommen, wird daraus die Transformation mit einem SVD-basiertem Verfahren bestimmt. Hierbei werden die beiden Punktwolken mit n Punkten als $3\times n$ Matrizen S (SLAM-Koordinatensystem) und T (Weltkoordinatensystem) behandelt. Um die Orientierung der Punktwolken zu bestimmen, werden beide Punktmengen zunächst zentriert. Die Korrelationsmatrix K der zentrierten Punktwolken berechnet sich aus $K = S_{centered}\, T'_{centered}$. Damit wird wiederum eine Singulärwertzerlegung $K = U\Sigma V'$ berechnet, aus der sich die Orientierung R der Punktwolken zueinander ergibt (vgl. Abschnitt 3.5):

$$R = UV' \,. \tag{4.6}$$

Die Skalierung s wird bestimmt, indem die Abstände der zentrierten Punkte zum Ursprung für die jeweilige Punktwolke aufsummiert werden und das Verhältnis der Summen berechnet wird. Dies ist nur möglich, da es sich nicht um beliebige Punkte (beispielsweise entlang der Trajektorie des Roboters) handelt, sondern um korrespondierende Punkte. Deshalb kann die Methode bei Point Set Registration im Allgemeinen nicht angewandt werden.

Wird die Punktwolke S mit R und s rotiert und skaliert, ergibt sich die Translation t aus dem

Vektor des Zentroids von S zum Zentroid von T. Die homogene Transformationsmatrix ist somit:

$$T_{abs}^{SLAM} = \begin{pmatrix} sR & t \\ 0 & 1 \end{pmatrix} \tag{4.7}$$

Wurde die Transformation berechnet, verlässt der *vSLAM_master* Node den Initialisierungszustand, transformiert die Posen des SLAM-Nodes und veröffentlicht die transformierten Posen mit gleichbleibendem Zeitstempel auf dem Topic `/abs_pose`. Die transformierten Posen können nun zur Navigation verwendet werden, da sie im durch das AprilTag Bundle vorgegebene Weltkoordinatensystem angegeben sind. Der *apriltag_ros_pnp_solver_node* wird beendet, sobald die Initialisierung abgeschlossen ist.

4.3.4 RANSAC-basierter Ausreißerfilter

Um die Qualität der Punktepaare zusätzlich zu verbessern, werden Ausreißer unter den Positionstupeln mit einem RANSAC-basiertem Verfahren detektiert und entfernt. Da herkömmliche RANSAC Algorithmen lediglich Rotation und Translation berücksichtigen, ist es notwendig den Skalenfaktor zunächst wie zuvor beschrieben zu bestimmen. Anschließend wird der RANSAC Algorithmus auf die gleich skalierten Punktwolken angewandt (siehe Abschnitt 3.6) und Tupel mit zu großer Abweichung bestimmt. Hierfür wird eine Funktion der *Point Cloud Library* (PCL) verwendet. Der maximale Abstand eines Datenpunkts zum Modell wird auf Basis der Standardabweichung des RMS Fehlers gewählt, um für unterschiedlichste Umgebungsparameter, die die Güte der Tupel beeinflussen, gültig zu sein. Im Anschluss wird mit den gefilterten Punktwolken das im vorherigen Abschnitt erläuterte Verfahren zur Schätzung der Transformation erneut ausgeführt. Insgesamt lässt sich das Verfahren durch Algorithmus 2 zusammenfassen.

Algorithmus 2: Berechnung der Transformation von SLAM- ins Weltkoordinatensystem

Eingabe: orb_points, pnp_points,
Ergebnis: Skalierte homogene Transformationsmatrix in das Weltkoordinatensystem

```
1 cloud_src ← orb_points                    // source point cloud: ORB-SLAM points
2 cloud_tgt ← pnp_points      // target point cloud: points based on AprilTags
3
4 transformation_matrix ← SVDbasedTransformationEstimation(cloud_src, cloud_tgt)
5 cloud_src_transformed ← transformation_matrix · cloud_src
  /* Filter cloud_src and cloud_tgt based on RANSAC performed on
     cloud_src_transformed and cloud_tgt                                    */
6 RANSACbasedOutlierDetector(cloud_src, cloud_src_transformed, cloud_tgt)
  /* Perform transformation estimation on filtered point clouds             */
7 transformation_matrix ← SVDbasedTransformationEstimation(cloud_src, cloud_tgt)
8 return transformation_matrix
```

4.4 Occupancy Grid Mapping

Um freie Navigation zu ermöglichen, ist eine geometrische Karte nötig, die Hindernisse und freie Flächen verzeichnet. Hierfür werden Occupancy Grid Maps erzeugt, die die Umgebung des Roboters in quadratische Zellen gleicher Größe einteilt. Für jede Zelle wird durch den Kartierungsprozess festgelegt, ob diese frei, belegt oder unbekannt ist, sodass letztendlich eine gesamtheitliche Karte der Umgebung entsteht, auf deren Grundlage ein Navigationsalgorithmus arbeitet. Hierfür ist es zusätzlich notwendig, dass die aktuelle Pose auf der Karte bekannt ist.

Der Kartierungsprozess ist, wie auch die Funktionalitäten zum Laden und Sichern einer Karte, in der Klasse `System` des ORB-SLAM3 Pakets implementiert. Die Prozedur wird aufgerufen, wenn das vSLAM-System beendet wird und falls ein entsprechender Parameter gesetzt wurde. Zunächst wird die Datei, die das Ergebnis einer Skalenkalibrierung enthält, über die Kommandozeile abgefragt und eingelesen.

Anschließend werden die Abmessungen der Karte berechnet. Die Höhe und Breite der Karte ergeben sich aus dem vierfachen der Standardabweichung (2σ Band) der x- bzw. y-Koordinaten aller in den Keyframes enthaltenen Featurepunkte. Die Lage des Ursprungs der Grid Map gegenüber der SLAM-Karte wird aus den Mittelwerten der x- bzw. y-Koordinaten aller Featurepunkte berechnet. Um die Größe der Zellen zu berücksichtigen müssen die jeweiligen Werte mit einem entsprechenden Faktor multipliziert werden.

Die Belegung der Zellen wird nach Algorithmus 3 bestimmt. Hierbei steht -1 für *frei*, 0 für *unbekannt* und $x > 0$ für die Anzahl an Beobachtungen von Punkten in einer Zelle. Die Zellen werden mit 0 initialisiert. Alle Featurepunkte werden auf die Grid Map projiziert und der Wert der Zelle wird um die Anzahl der Beobachtungen des Punkts inkrementiert, falls der Punkt oberhalb des Bodens und unterhalb der Decke liegt. Eine Mindestanzahl an Beobachtungen für die berücksichtigten Punkte sorgt dafür, dass Ausreißer früh gefiltert werden. Anschließend wird über die Keyframes iteriert und mittels einer modifizierten Version des Bresenham Algorithmus (Algorithmus 4) alle Zellen, die zwischen der Keyframezelle und den Zellen der beobachteten Punkte liegen, auf *frei* gesetzt. Falls dabei eine *belegte* Zelle getroffen wird, werden nur die Zellen zwischen der Keyframezelle und der *belegten* Zelle auf *frei* gesetzt. Damit wird die Kartierung mittels Laserscans nachgeahmt. Da die Featurepunkte auf verschiedenen Höhen aufgenommen werden, ist es möglich, dass Punkte in der Ebene hintereinander beobachtet werden. Hier wird der Raum zwischen den Punkten als *unbekannt* behandelt.

Aus diesem Grund bleiben Zellen, die einmal als *belegt* eingetragen wurden, im Zustand *belegt*. Eine Beobachtung, bei der eine Zelle nach dem Modell einer Messung mittels 2D Laserscanner *frei* ist, steht bei der Auswertung von dreidimensionalen Punktwolken nicht im Konflikt mit den vorigen Messungen. Somit ist eine Behandlung nach dem Bayes'schen Ansatz, wie in Abschnitt 3.8 hier unangebracht.

Algorithmus 3: Bestimmung der Occupancy Grid Map

Eingabe: Keyframes, MapPoints, minObservations, transformation, cell_size, size_x, size_y, offset_x, offset_y

```
1  int grid_map[size_y][size_x]
2  for i, j | i < size_y, j < size_x do
3  |  grid_map[i][j] ← 0                                   // set cell to unknown
4  end
5  foreach point ∈ MapPoints do
6  |  if point.numberOfObservations ≥ minObservations then
7  |  |  transformed_point ← transformation · point    // transform to world frame
8  |  |  if transformed_point.z ∈ [-0.35, 2.6] then
9  |  |  |  idx_x ← transformed_point.x / cell_size + offset_x
10 |  |  |  idx_y ← transformed_point.y / cell_size + offset_y
11 |  |  |  if (idx_x, idx_y) lies in grid map then
12 |  |  |  |  grid_map[idx_y][idx_x] ← grid_map[idx_y][idx_x] +
   |  |  |  |   point.numberOfObservations       // increment corresponding cell
13 |  |  |  end
14 |  |  end
15 |  end
16 end
17 foreach keyframe ∈ Keyframes do
18 |  kf_transformed ← transformation · keyframe.pos   // transform to world frame
19 |  kf_x ← kf_transformed.x / cell_size + offset_x
20 |  kf_y ← kf_transformed.y / cell_size + offset_y
21 |  foreach point ∈ keyframe.map_points do
22 |  |  transformed_point ← transformation · point    // transform to world frame
23 |  |  if transformed_point.z ∈ [-0.35, 2.6] then
24 |  |  |  idx_x ← transformed_point.x / cell_size + offset_x
25 |  |  |  idx_y ← transformed_point.y / cell_size + offset_y
   |  |  |  /* Set cells between keyframe cell and point to free          */
26 |  |  |  ModifiedBresenham(x_start ← kf_x, y_start ← kf_y, x_end ← idx_x, y_end ←
   |  |  |   idx_y )
27 |  |  end
28 |  end
29 end
```

Algorithmus 4: Modifizierter Bresenham Algorithmus

Eingabe: grid_map, x_{start}, y_{start}, x_{end} und y_{end}

```
1  dx ← abs(x_end − x_start)
2  sx ← 1 if x_end > x_start, else −1
3  x ← x_start
4
5  dy ← −abs(y_end − y_start)
6  y ← y_start
7  sy ← 1 if y_end > y_start, else −1
8
9  err ← dx + dy
10
11 while (x, y) ≠ (x_end, y_end) do
12     if 2 · err > dy then
13         err ← err + dy
14         x ← x + sx
15     end
16     if 2 · err < dx then
17         err ← err + dx
18         y ← y + sy
19     end
20     if grid_map[y][x] > 0 then
21         break                          // break if an occupied cell is reached
22     end
23     else
24         grid_map[y][x] ← -1                          // set cell to unknown
25     end
26 end
```

Um die erstellte Occupancy Grid Map weiter zu verfeinern werden individuelle morphologische Operationen auf den verschieden klassifizierten Zellen ausgeführt (siehe Abbildung 4.4). Diese verbessern die Occupancy Grid Map nach folgendem Schema:

1. **Für alle *belegten* Zellen:** Liegt die Summe aller benachbarten Zellen in einem 11×11 Feld um die betroffene Zelle unter -10, so wird diese Zelle als *frei* markiert. So werden fehlerhaft als *belegt* markierte Zellen, um die in einem großen Radius überwiegend *unbekannte* oder *freie* Zellen, sowie *belegte* Zellen mit wenigen Punkten liegen, korrigiert.

2. **Für alle *freien* Zellen:** Befinden sich im 3×3 Quadrat um die betroffene Zelle weniger als vier *freie* Zellen oder im 5×5 Quadrat um die betroffene Zelle weniger als zehn *freie* Zellen, wird diese Zelle als *unbekannt* markiert. Hiermit werden einzelne *frei* markierte Zellen und feine *freie* Strahlen in einer *unbekannten* bzw. *belegten* Umgebung gefiltert.

3. **Für alle *unbekannten* Zellen:** Sind mehr als vier der Zellen im umgebenden 3×3 Quadrat einer Zelle *frei*, wird die Zelle als *frei* markiert. So werden einzelne als *unbekannt* markierte Zellen in einer *freien* Umgebung gefiltert.

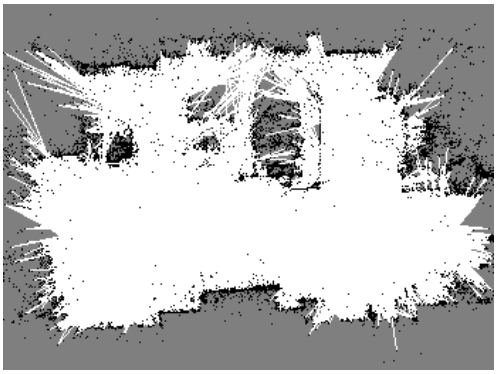
4. **Für alle *belegten* Zellen:** Liegen in einem 7×7 Feld um die betroffene Zelle ausschließlich als *unbekannt* markierte Zellen, wird diese Zelle ebenfalls als *unbekannt* klassifiziert. Diese Operation dient dazu, *belegte* Zellen, die in einem großen Radius von *unbekannten* Zellen umgeben sind, zu korrigieren.

5. Abschließend werden die Schritte 2. und 3. abwechselnd zehn mal wiederholt. In der Regel konvergiert die Karte hierbei bereits vor der zehnten Iteration.

Insgesamt ist zu bemerken, dass die Kriterien, um *belegte* Zellen zu eliminieren, deutlich schärfer sind als die Kriterien, um *freie* Felder als *unbekannt* zu korrigieren. Daraus folgt, dass Zellen im Zweifel bevorzugt als *belegt* oder *unbekannt* klassifiziert werden.

Die Abmessungen der Karte, sowie deren Auflösung und Ursprung, werden in einer *yaml*-Datei gesichert, die auf die eigentliche Occupancy Grid Map, die als separate *png*-Datei gespeichert wird, verweist. Somit ist die Occupancy Grid Map für ROS Map Server interpretierbar und kann unkompliziert auf ein entsprechendes ROS Topic veröffentlicht werden.

Die Karte wird nur bei Beendigung des Kartierungsprozesses erstellt, da ein sukzessiver Aufbau der Karte aufgrund der stetigen Anpassung der ORB-SLAM3 Karte durch Loop-Closures und Graphoptimierung erfordert, dass die Occupancy Grid Map häufig vollständig neu aufgebaut werden muss, um die Konsistenz zu wahren. Deshalb wird die Occupancy Grid Map erst erstellt, sobald die Umgebung vollständig und ausführlich kartiert wurde und die ORB-SLAM3 Karte somit gegen die höchstmögliche Genauigkeit konvergiert. Die Erweiterung des Kartierungsbereichs im Nachhinein ist nichtsdestotrotz möglich, indem der Kartierungsprozess mit Eingabe einer bestehenden Karte neu gestartet wird.



(a) Grid Map ohne morph. Operationen.

(b) Schritt 1: Filtern *belegter* Zellen.

(c) Schritt 2: Filtern *freier* Zellen.

(d) Schritt 3: Filtern *unbekannter* Zellen.

(e) Schritt 4: Erneutes Filtern *belegter* Zellen.

(f) Schritt 5: Glättung nicht-*belegter* Zellen.

Abbildung 4.4: Verbesserung der Occupancy Grid Map durch morphologische Operationen anhand einer beispielhaften Karte der Werkhalle des TGZ (*frei*: weiß, *unbekannt*: grau, *belegt*: schwarz).

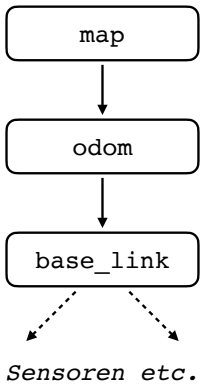


Abbildung 4.5: ROS Transformationsbaum des Navigationssystems.

4.5 Schnittstellen zum Navigationssystem des ZfT

Am Zentrum für Telematik wurden bereits diverse Funktionalitäten entwickelt, die Aufgaben zur Unterstützung von autonomer Navigation erfüllen. Diese reichen von einem System zur Lokalisierung mittels AMCL auf Basis von Laserscannern, das durch Odometrie verfeinert wird, bis hin zu Pfadplanungsalgorithmen, die den Roboter durch eine Occupancy Grid Map navigieren.

Um das entwickelte Verfahren zur Lokalisierung und Kartierung zusammen mit dem existierenden System zu verwenden, müssen einige zusätzliche Schnittstellen geschaffen werden. Für eine hohe Informationsrate ist es zudem erforderlich, die Posenbestimmung des vSLAM-Systems mit der Odometrie zu kombinieren. Das vSLAM-System bestimmt die Roboterpose mit 10 Hz, wohingegen Odometrie mit 50 Hz läuft. Allerdings ist bei Odometrie durch die Akkumulation der Fehler ein signifikanter Drift zu verzeichnen, der die Genauigkeit mit der Länge der gefahrenen Strecke sinken lässt. Deshalb wird die Odometrie durch die Berechnung des vSLAM-Systems korrigiert, sodass sich der Fehler nur über fünf Zeitschritte summiert, wodurch der resultierende akkumulierte Fehler stets gering bleibt.

Realisiert wird die Verfeinerung über den Transformationsbaum (*tf tree*), in dem verschiedene Koordinatensysteme in einer Baumstruktur miteinander verbunden sind. Für jede Kante wird eine Transformation spezifiziert. In diesem Fall besteht der Transformationsbaum vorrangig aus den Koordinatensystemen `map`, `odom` und `base_link` (siehe Abbildung 4.5). Die Basis bildet hierbei das Koordinatensystem der Karte (`map`). Darin liegt das Koordinatensystem der Odometrie (`odom`), in dem wiederum das Fahrzeugkoordinatensystem (`base_link`) des Roboters angegeben ist. Die einzelnen Koordinatensysteme der Sensoren, Aktuatoren und sonstigen wichtigen Punkte des Roboters liegen im Fahrzeugkoordinatensystem. Wird die Odometrie alleinstehend verwendet, ist die Transformation von `map` zu `odom` konstant.

Um die Posen der Odometrie zu korrigieren, wird die Transformation von `map` zu `odom` ständig korrigiert. Der ROS Node *tf_orbmap_to_odom* setzt bei jeder neuen Pose des vSLAM-Systems die Transformation von `map` zu `odom` so, dass die Odometriepose im Koordinatensystem der

Karte identisch zur Pose des vSLAM-Systems – angegeben im Koordinatensystem der Karte – ist. Zur Berechnung werden beim Erhalt einer neuen Pose des vSLAM-Systems diese und die zuletzt erhaltene Odometriepose als homogene Transformationen T_{odom}^{robot} und T_{map}^{robot} dargestellt. Mit der Korrektur T_{map}^{odom} gilt

$$T_{map}^{robot} = T_{map}^{odom} \; T_{odom}^{robot} \quad . \tag{4.8}$$

Somit berechnet sich die Korrektur aus

$$T_{map}^{odom} = T_{map}^{robot} \; T_{odom}^{robot^{-1}} \quad . \tag{4.9}$$

Um die Position der Kamera bezüglich der Roboter Basis zu berücksichtigen, wird die relative Position als intrinsische Transformation T_{cam}^{base} angehängt.

$$T_{map}^{odom} = T_{map}^{robot} \; T_{cam}^{base} \; T_{odom}^{robot^{-1}} \quad . \tag{4.10}$$

Der Node *tf orbmap_to_odom* ist der einzige Node, der die Transformation von `map` zu `odom` im Transformationsbaum überschreibt. AMCL sowie `static_transform_publisher` werden abgeschaltet.

Neben der genauen Pose wird für die Navigation eine Occupancy Grid Map benötigt. Diese wird, wie im vorherigen Kapitel erläutert, in einer *yaml*-Datei spezifiziert, die auf die eigentliche Occupancy Grid Map (als separate *png*-Datei) verweist. Ein ROS Map Server, der zusammen mit den übrigen Erweiterungen durch eine *roslaunch* Datei gestartet wird, interpretiert die Occupancy Grid Map und veröffentlicht sowohl die Karte selbst (*nav_msgs/OccupancyGrid*), als auch die dazugehörigen Metadaten (*nav_msgs/MapMetaData*) auf entsprechenden Topics.

4.6 Resultierendes Gesamtsystem zur autonomen Navigation

Insgesamt entsteht ein vollständiges System, das von hoch genauer Lokalisierung bis hin zur Pfadplanung und Kollisionsvermeidung alle Aufgaben erfüllt, die zur autonomen Navigation benötigt werden. Die zugrundeliegende topologische Karte (ORB-SLAM3) und die geometrische Occupancy Grid Map werden dabei automatisch erzeugt.

Der beispielhafte Ablauf bei Verwendung des Roboters Stäubli WFT Helmo des ZfT ist dabei wie folgt:

1. Einrichtung einer Secure Shell Verbindung zum Roboter zum Starten der Anwendungen auf dem PC des Roboters
2. Festlegung des ROS Masters auf dem verwendeten Laptop, wobei der Host „wft“ am Laptop unter */etc/hosts* eingetragen sein muss (sowie umgekehrt), und des eigenen ROS Hostnamen auf allen Shells des Laptops
3. Starten des ROS Master und der Funktionalitäten, die auf dem Rechner des Roboters laufen (`robot_wft_franz.launch` mit `useGroundstation:=true` und `useAMCL:=false`), in der Secure Shell

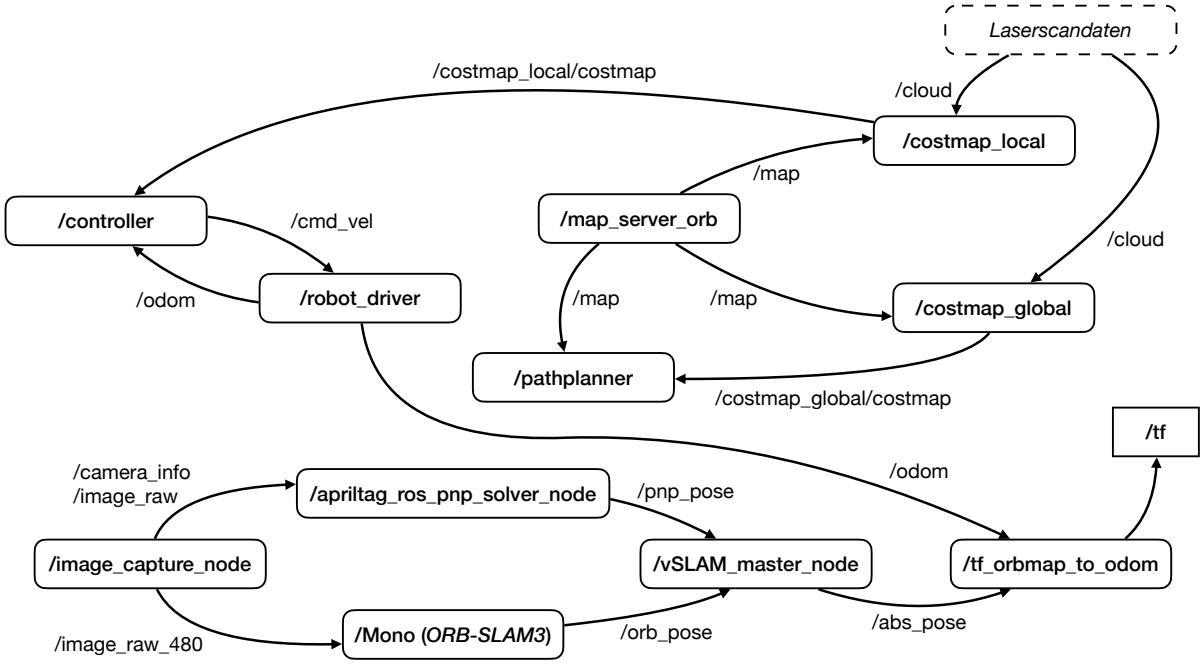


Abbildung 4.6: Übersicht über die wichtigsten ROS Nodes des Gesamtsystems zur autonomen Navigation und deren Zusammenspiel über ROS Topics.

4. Starten des Schnittstellen Nodes zur Kamera (`image_capture_node`)
5. Starten der übrigen Nodes auf dem Laptop (`vSLAM_master.launch` mit `save_map:=0` und `occ_map:=1`)

Mit dem Parameter `save_map` wird indiziert, ob bei Beendigung der Anwendung neue / aktualisierte Karten erstellt und gespeichert werden sollen. Der Parameter `occ_map` gibt an, ob bereits eine Occupancy Grid Map erstellt wurde. Optional kann zusätzlich das Visualisierungstool *rviz* gestartet werden. In diesem kann das Robotermodell in der Karte verfolgt und Ziele für die Pfadplanung graphisch vorgegeben werden. Alternativ genügt es, die Zielpose auf dem Topic */goal* zu veröffentlichen.

Grundsätzlich ist es ebenfalls möglich, alle Anwendungen auf dem PC des Roboters laufen zu lassen und alle Komponenten in einer *roslaunch* Datei zusammenzufassen. Da die Kommandierung und Visualisierung allerdings ohnehin einen verbundenen Rechner erfordert, ist es sinnvoll, dass dieser weitere Rechner auch einen Teil der Aufgaben übernimmt, um die Rechenlast zu verteilen. Dies erfordert ein stabiles Netzwerk, was durch ein herkömmliches WLAN gegeben ist.

Eine Übersicht über alle Nodes ist in Abbildung 4.6 zu sehen. Der */image_capture_node* nimmt die Bilder mit der Gigabit-Ethernet-Kamera auf und veröffentlicht diese in voller und heruntergesetzter Auflösung mit den zugehörigen Metadaten. Das hoch aufgelöste Bild wird vom */apriltag_ros_pnp_solver_node* verwendet, um die relative Position zum AprilTag Pattern zu

bestimmen. Parallel dazu bestimmt ORB-SLAM3 (*/Mono*) die Pose auf Basis der ORB-SLAM3 Karte. Der *vSLAM_master_node* verwendet beide Posen, um die ORB-SLAM3 Pose in das globale Koordinatensystem zu transformieren (siehe Abschnitt 4.3.3). Ist diese Transformation bereits bekannt, beendet der *vSLAM_master_node* den */apriltag_ros_pnp_solver_node*. Anhand der Pose im globalen Koordinatensystem wird die Odometrie über den Transformationsbaum korrigiert. Der Transformationsbaum wird von nahezu allen Nodes verwendet, weshalb die zugehörigen Kanten in Abbildung 4.6 zur besseren Übersichtlichkeit nur angedeutet wurden.

Die aus der ORB-SLAM3 Karte aufgebaute Occupancy Grid Map (Abschnitt 4.4) wird vom */map_server_orb* veröffentlicht. Zusammen mit den aktuellen, gefilterten Punktwolken der Laserscanner wird daraus eine lokale und globale Costmap bestimmt. Die globale Costmap ist Basis für den Pfadplanungsalgorithmus, wohingegen die lokale Costmap vom Controller zur Kollisionsvermeidung verwendet wird.

4.7 Ergänzender Kalmanfilter zur Fusion mit AMCL

Um das bisherige Lokalisierungssystem mit dem entwickelten vSLAM-System zu fusionieren, wird in diesem Abschnitt ein Kalmanfilter (vgl. 3.9) als ROS Node implementiert. Hierfür wird zunächst ein Zustandsraummodell aufgestellt und die nötigen Parameter festgelegt. Nach der Implementierung werden die Parameter, falls notwendig, angepasst.

Der Zustand des Roboters x wird durch die 2D Position und Orientierung beschrieben. Als Eingang werden die Steuerungseingaben mit der seit dem letzten Update vergangenen Zeit verrechnet, um absolute Positions- und Orientierungsänderungen $\Delta x, \Delta y$ und $\Delta\theta$ zu erhalten. Somit ergibt sich für das Zustandsraummodell:

$$x_{k|k-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}_{k-1} + \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{pmatrix}_{k-1} \tag{4.11}$$

Die Messung y_k berechnet sich aus der Ausgangsgleichung:

$$y_k = \begin{pmatrix} x_{vSLAM} \\ y_{vSLAM} \\ \theta_{vSLAM} \\ x_{AMCL} \\ y_{AMCL} \\ \theta_{AMCL} \end{pmatrix}_k = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} x_k \tag{4.12}$$

Der Einfachheit halber wird die Orientierung nicht wie bei den Lokalisierungsalgorithmen in Quaternionen angegeben, sondern auf die Fahrtebene projiziert und als Orientierungswinkel angegeben.

Problematisch bei der Auslegung des Kalmanfilters ist die ungleichmäßige Updaterate. Diese resultiert aus der Funktionsweise von AMCL, wobei neue Posenschätzungen bei einer gewissen Distanz zur vorherigen Pose berechnet werden. Zusätzlich spielt die Auslastung des Rechners eine Rolle. Um die Updaterate zu erhöhen, werden die Schwellwerte auf 0,05 m bzw. 0,05 rad verringert.

Das Messrauschen R ergibt sich aus der Ungenauigkeit des vSLAM-Systems und AMCL als Diagonalmatrix. Für das vSLAM-System dienen hierfür die Ergebnisse aus Kapitel 5.2.4. Da der Kalmanfilter mit der letzten Messung des vSLAM-Systems initialisiert wird, werden die entsprechenden Werte des Messrauschens auch als initiale Kovarianz der Zustandsschätzung P_0 verwendet. Nach praktischen Tests erwiesen sich folgende Parameter als passend:

$$R = \begin{pmatrix} 0,0016 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0,0016 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0,0004 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0,0025 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0,0025 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0,0004 \end{pmatrix}$$

$$Q = \begin{pmatrix} 0,25 & 0 & 0 \\ 0 & 0,25 & 0 \\ 0 & 0 & 0,001 \end{pmatrix} \quad P_0 = \begin{pmatrix} 0,0016 & 0 & 0 \\ 0 & 0,0016 & 0 \\ 0 & 0 & 0,0004 \end{pmatrix}$$

Das optimale Prozessrauschen Q wird experimentell ermittelt. Hierfür wird eine ROS Bag Datei aufgenommen, die die Messungen vom vSLAM-System und AMCL, sowie die Steuerungseingaben des Controllers wiedergibt. Q ist üblicherweise eine Diagonalmatrix und enthält in diesem Fall für x und y den selben Wert. Als erste Schätzung dient die maximale erwartete Änderung in Position und Orientierung. Die Werte werden schrittweise angepasst, bis die Zustandsschätzung des Kalmanfilters den Messungen optimal folgt und dabei nicht zu sensibel auf Änderungen reagiert. Der Wert für die Orientierung wird getrennt angepasst. Es werden feste Werte verwendet, da eine dynamische Berechnung für das vSLAM-System und AMCL mit hohem Aufwand verbunden sind.

Bei der Initialisierung des Kalmanfilters wird die erste Zustandsschätzung aus der Messung des vSLAM-Systems festgelegt. Sobald AMCL eine Pose maximal 0,05 s vor oder nach einer Pose des vSLAM-Systems bestimmt, wird die AMCL Pose in das Koordinatensystem des vSLAM-Systems transformiert und die Posen werden zusammen als Messung y_k zum Update der Zustandsschätzung verwendet. Die Transformation wird anhand des ersten Posenpaars bestimmt. Der Kalmanfilter verwendet das Publisher/Subscriber System von ROS, um die Messungen zu empfangen und die aktuelle Zustandsschätzung in Form einer Pose mit Zeitstempel zu veröffentlichen.

Bei Tests des Kalmanfilters mit den Steuerungseingaben des Controllers wurde deutlich, dass diese die Zustandsschätzung insgesamt eher verschlechtern als verbessern. Der Grund dafür sind

regelmäßig auftretende, sprunghafte und unverhältnismäßig große Steuerungseingaben, die die Zustandsschätzung kurzzeitig signifikant verschlechtern, da der Roboter diese Geschwindigkeitsvorgaben ohnehin nicht befolgen kann. Zudem steigt die Genauigkeit der Zustandsschätzung unter Einbezug der Eingaben nicht nennenswert. Deshalb werden die Steuerungseingaben im Folgenden nicht weiter vom Kalmanfilter berücksichtigt.

Kapitel 5

Experimente und Evaluierung

Das folgende Kapitel umfasst Versuche und Messreihen, anhand derer die implementierten Methoden aus Kapitel 4 ausgewertet werden.

5.1 Vergleich der vSLAM-Algorithmen

Die in Abschnitt 4.2 ausgewählten und eingerichteten vSLAM-Algorithmen werden in diesem Abschnitt hinsichtlich Ihrer Eignung zur Lokalisierung und Navigation eines mobilen Roboters in einer Industrieumgebung untersucht.

5.1.1 Hardware

Kamera und Objektiv

An die Kamera, die die Bilder für den vSLAM-Algorithmus aufnimmt, werden verschiedenste Anforderungen gestellt. Um den Rolling Shutter Effekt zu vermeiden, muss eine **Global Shutter** Kamera oder eine Kamera mit hoher Bildwiederholrate verwendet werden.

Bei der Verwendung einer Global Shutter Kamera ist eine hohe Bildwiederholrate nicht unbedingt notwendig. In dieser Arbeit soll die Lokalisierung und Kartierung stets live möglich sein. In diesem Fall ist die Laufzeit des vSLAM-Algorithmus der begrenzende Flaschenhals für die Anzahl an Bildern, die verarbeitet werden können. Aus diesem Grund ist je nach Rechenleistung eine **Bildwiederholrate** von 10-20 fps ausreichend.

Die Auflösung soll mindestens 480 p betragen. Eine höhere Auflösung ist gerade bei Feature-basierten vSLAM-Algorithmen von Vorteil. Eine Auflösung von deutlich über 1080 p darf nur bei einer entsprechend hohen Rechenleistung verwendet werden, da ansonsten die Bildverarbeitungsrate zu niedrig für zuverlässiges und genaues Tracking ist. Da die meisten vSLAM-Algorithmen mit Graustufenbildern arbeiten, ist keine Farbkamera nötig.

Zudem ist es notwendig, dass die Kamera über entsprechende Schnittstellen verfügt, die es ermöglichen, die Bilder mit möglichst geringer Zeitverzögerung an den vSLAM-Algorithmus weiterzugeben.

Das **Objektiv** ist für den Sichtbereich der Kamera maßgebend. Grundsätzlich ist ein möglichst großes Sichtfeld für vSLAM-Algorithmen von Vorteil. Mit besonders weitwinkligen Objektiven geht allerdings auch eine stärkere Verzeichnung einher, was eine genaue Kalibrierung erfordert. Das Objektiv soll feststellbar sein, um Neukalibrierungen zu entgehen.

Für die Experimente im Rahmen dieser Arbeit wird die Global Shutter Kamera FLIR Blackfly S GigE (Modell: BFS-PGE-16S2C-CS) mit einem 1,6 MP Sony IMX273 CMOS Sensor verwendet (siehe Abbildung 5.1 (b)). Die Kamera hat eine Auflösung von 1440×1080 und eine maximale Bildwiederholrate von 78 fps. Die Bilder werden über einen Gigabit-Ethernet-Port übertragen. Als Objektiv wird das Computar AG4Z2812FCS-MPIR (siehe Abbildung 5.1 (c)) mit einer variablen Brennweite von 2.8 mm-10 mm und einem diagonalen Bildwinkel von 154 °- 39 ° gewählt. Das Objektiv wird auf die kleinstmögliche Brennweite fixiert, um ein großes Sichtfeld aufzunehmen.

Rechner

Die Programme werden im Rahmen dieser Arbeit vorwiegend auf einem MacBook Pro (Anfang 2015) mit einem Intel Core i5-5257U Prozessor und einem 8 GB DDR3 Arbeitsspeicher ausgeführt. Für die Experimente dieser Arbeit ist der Rechner ausreichend.

Roboter

Die Ergebnisse dieser Arbeit sind hauptsächlich auf mobile Roboter in Innenräumen ausgerichtet. Das Ziel ist es, bestehende Robotersysteme unkompliziert um die Verfahren zur Lokalisierung, Kartierung und Navigation erweitern zu können.

Für die folgenden Tests und Experimente wird eine Prototypenversion des Stäubli WFT HelMo mit CS9 Steuerung verwendet, der vom Zentrum für Telematik zu Verfügung gestellt wird. Der Roboter besteht aus einer Antriebs- und einer Rotationseinheit. Hierdurch können die Experimente mit verschiedenen Ausrichtungen der Kamera, die auf der Oberseite der Rotationseinheit fixiert wird, durchgeführt werden. Unter anderem wegen seines hohen Gewichts von etwa 710 kg verfügt der Roboter über eine genaue Odometrie. Der Aufbau ist in Abbildung 5.1 (a) zu sehen. [44]

5.1.2 Kamera Kalibrierung

Für die Genauigkeit der vSLAM-Algorithmen sind genaue Kameraparameter erforderlich. Aus diesem Grund wird großen Wert auf die Kalibrierung gelegt, da diese eine entscheidende Grundlage für vSLAM ist.

(a) Stäubli WFT HelMo Roboter mit fixierter Kamera (FLIR Blackfly S) und Rechner.

(b) FLIR Blackfly S GigE mit Sony IMX273 CMOS Sensor. [1]

(c) Computar AG4Z2812FCS-MPIR. [2]

Abbildung 5.1: Verwendete Hardware.

Zur Kalibrierung dient das ROS Package `camera_calibration` und ein Schachbrettmuster mit 11×6 Feldern zu einer Seitenlänge von jeweils 6,5 cm. Ein Skript detektiert in Aufnahmen der Kamera die Ecken der Felder und bestimmt unter Kenntnis der dreidimensionalen Objektpunkte und der zweidimensionalen Bildpunkte die intrinsischen Parameter der Kamera. Die Nutzeroberfläche veranschaulicht hierbei, in welchem Winkel-, Skalen- und Bildbereich sich die verwendeten Bilder befinden. So wird garantiert, dass die Bilder für die Kalibrierung aus verschiedenen Perspektiven aufgenommen wurden. Im Anschluss wird der Reprojektionsfehler herangezogen, um die Güte der Kalibrierung abzuschätzen. Um eine genaue Kalibrierung zu erzielen, muss das Schachbrettmuster möglichst eben und exakt und die Bilder scharf sein.

LSD-SLAM bietet zum einen die Möglichkeit die Kameraparameter in einer Konfigurationsdatei zu hinterlegen, zum anderen können die Parameter über ein ROS Topic veröffentlicht werden. Für diese Arbeit wird der zweite Ansatz gewählt. Der Grund dafür ist, dass auf diese Weise ROS Bag Dateien aufgenommen werden können, die neben den Bildern auch die Kameraparameter

[2]https://www.flir.de/products/blackfly-s-gige/
[2]https://computar.com/product/1311/AG4Z2812FCS-MPIR

Parameter	Wert	Erläuterung
`minUseGrad`	3,3	Minimaler Intensitätsgradient eines Pixels für Tiefenschätzung
`cameraPixelNoise`	50	Geschätztes Pixelrauschen
`KFUsageWeight`	13	Gewichtung des *Pixelscore* für den Score eines Keyframes
`KFDistWeight`	9.75	Gewichtung der Distanz zu vorherigem Keyframe für den Score eines Keyframes
`maxLoopClosureCandidates`	15	Anzahl der vorherigen Keyframes, die als Loop Closure Kandidaten betrachtet werden
`loopClosureStrictness`	5,0	Erlaubte Ungenauigkeit der relativen Pose zwischen Keyframes bei Loop Closures

Tabelle 5.1: Angepasste Parameter für LSD-SLAM.

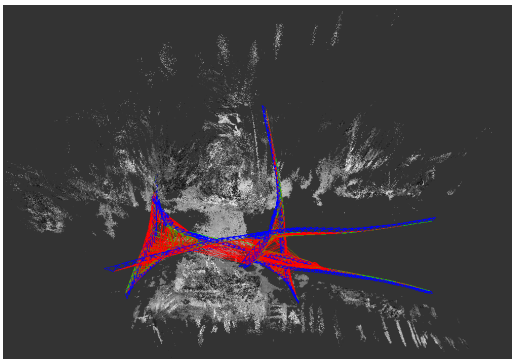
enthalten. So ist es möglich, den vSLAM-Algorithmus mit verschiedenen Kamerakonfigurationen mit einer ROS Bag Datei auszuführen, ohne dabei darauf achten zu müssen, dass jeweils auch die entsprechende Konfigurationsdatei eingebunden ist. In diesem Fall müssen die Bilder bereits entzerrt an den vSLAM-Algorithmus übergeben werden.

Bei **ORB-SLAM** müssen die Kameraparameter in einer Konfigurationsdatei eingetragen werden. Neben der Kameramatrix sind zwei Parameter bezüglich radialsymmetrischer Verzeichnung und zwei Parameter bezüglich tangentialer Verzeichnung anzugeben. Die Bilder werden vom vSLAM-Algorithmus entzerrt.

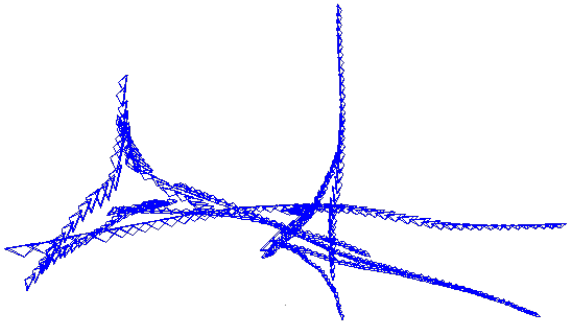
5.1.3 Evaluierung und Ergebnisse

Um die Eignung von LSD-SLAM und ORB-SLAM für diese Arbeit zu prüfen, werden Bildsequenzen in der Werkhalle des Technologie und Gründerzentrums (TGZ) in Würzburg aufgenommen, auf deren Grundlage die Algorithmen Karten erzeugen.

Hierfür wird die Kamera auf dem Stäubli WFT Roboter fixiert (siehe Abschnitt 5.1.1). Anschließend wird der Roboter per Fernbedienung durch die Werkhalle des TGZ gesteuert und dabei die Kamerabilder als Bag Datei aufgenommen, sodass die Fahrt für die vSLAM-Algorithmen im Nachhinein simuliert werden kann. Zuvor werden die Algorithmen live getestet, um deren Stärken und Schwächen zu identifizieren. Für den Vergleich werden mehrere Aufnahmen mit verschiedenen Auflösungen, Trajektorien und Kameraausrichtungen erstellt und die für LSD-SLAM am besten geeignete Konfiguration gewählt.



(a) Keyframegraph und Punktwolke



(b) Keyframes

Abbildung 5.2: LSD-SLAM: Karte der Werkhalle des TGZ.

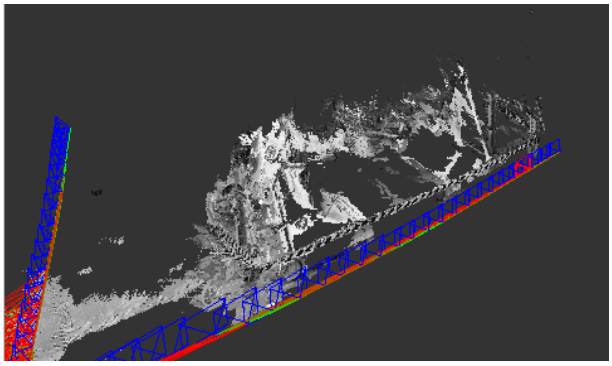
LSD-SLAM

Für LSD-SLAM ist neben der Anpassung der Parameter (siehe Tabelle 5.1) auch notwendig, dass die Trajektorie der Kamera bei der Aufnahme der Sequenz nach den Schwächen von LSD-SLAM gewählt wird. LSD-SLAM ist gerade bei Rotationen ohne Translation anfällig für starken Skalendrift, der die Karte teilweise unbrauchbar werden lässt. Auch Large Loop Closures helfen hierbei nur bedingt. Aus diesem Grund wird darauf geachtet, dass die Bewegungen vorwiegend translatorischer Natur sind und enge Kurven vermieden werden. Hierdurch werden die möglichen Trajektorien in der Werkhalle des TGZ stark eingeschränkt. Zudem wird deutlich, dass sich die Probleme bei LSD-SLAM signifikant verringern, wenn die Kamera am Roboter seitlich zur Fahrtrichtung ausgerichtet ist. Das hat allerdings auch den Nachteil, dass Gänge jeweils in zwei Richtungen abgefahren werden müssen, um beide Wände abzubilden. Da das Tracking bei LSD-SLAM ansonsten allerdings deutlich ungenauer ist, wird die Kamera für dieses Experiment dennoch seitlich ausgerichtet.

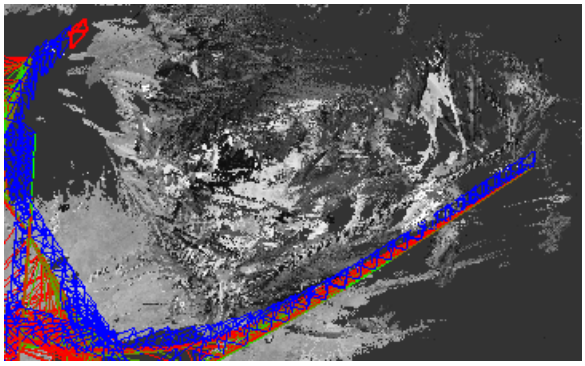
Als Auflösung wird für das Vergleichsexperiment 480 p gewählt, da bei einer höheren Auflösung der benötigte Speicherplatz pro Keyframe zu hoch ist, um einen Keyframegraphen mit möglichst vielen Keyframes zu erzeugen. Der Arbeitsspeicher wird bei LSD-SLAM deutlich stärker ausgelastet, als bei ORB-SLAM, was mit den dichten Tiefeninformationen der Keyframes bei LSD-SLAM zu begründen ist.

Die erstellte Karte, bestehend aus 354 Keyframes und 53630077 Punkten, ist in Abbildung 5.2 zu sehen. Grundsätzlich spiegeln sich viele Details der realen Umgebung wider. Die erstellte Punktwolke ist dicht und die Kameraposition wird relativ zu den nächsten Keyframes grob richtig lokalisiert. Die Genauigkeit reicht allerdings nicht aus, um eine genaue Lokalisierung und Kartierung zu ermöglichen. Zudem pflanzt sich der Fehler bei den relativen Posen der Keyframes zueinander fort, sodass bei der absoluten Lokalisierung selbst ohne die Verwendung eines Referenzsystems deutlich wird, dass die absoluten Positionen der Keyframes, abhängig von der Dichte des Keyframegraphen an der betroffenen Stelle und deren Abstand zum Ursprung, in der Größenordnung von bis zu über einen Meter falsch liegen. Ein Anhaltspunkt hierfür ist, dass

Abbildung 5.3: LSD-SLAM: Seitliche Ansicht des Keyframegraphs.



(a) Bewegungssimulationsanlage, beobachtet aus einer Perspektive



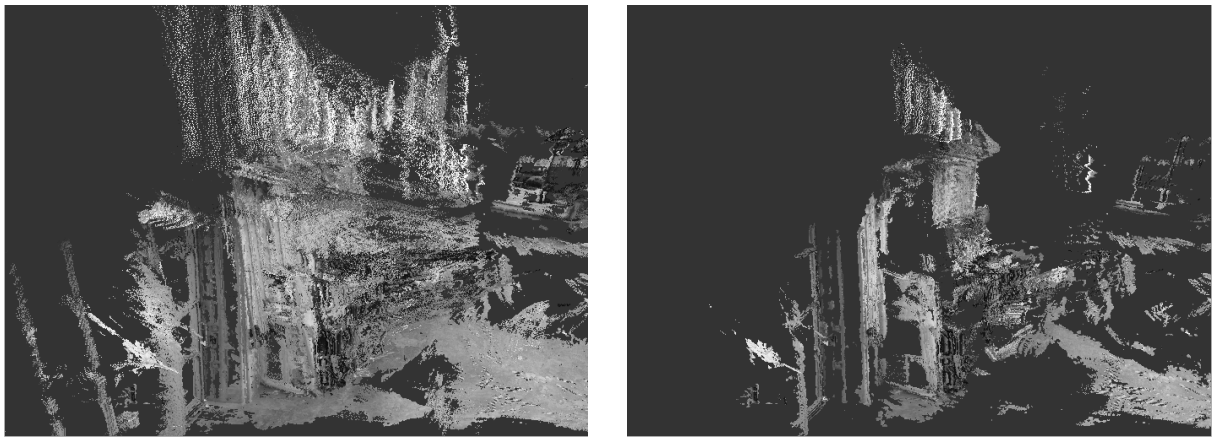
(b) Bewegungssimulationsanlage, beobachtet aus mehreren Perspektiven

Abbildung 5.4: Verschlechterung der Punktwolke durch ungenaue Keyframeposen bei LSD-SLAM.

die „Ausleger“, die in Abbildung 5.2 nach rechts hin zu sehen sind in Realität wesentlich näher aneinander liegen (vgl. Abbildung 5.6). Stattdessen driften sie immer weiter auseinander, was die eigentlich parallelen Wände schief zueinander wirken lässt. Ein zusätzlicher Anhaltspunkt für die mangelnde Genauigkeit ist in Abbildung 5.3 zu sehen. Da die Kamera fest am Roboter montiert ist und der Boden in der Werkhalle eben ist, muss auch die Trajektorie der Kamera, und damit die Position der Keyframes auf einer Ebene liegen. Wird der Keyframegraph im dreidimensionalen betrachtet, ist zu erkennen, dass bei Rotationen die Rotationsachse meist nicht exakt senkrecht ist, wodurch die Keyframes nicht auf einer Ebene liegen. Dies wird im Groben durch die Loop Closures korrigiert, wird aber insgesamt nur unzureichend kompensiert.

Die Ungenauigkeit der Keyframeposen wird zusätzlich deutlich, wenn Hindernisse aus verschiedenen Perspektiven aufgenommen werden. In Abbildung 5.4 ist zu sehen wie sich die Darstellung einer Bewegungssimulationsanlage in der Werkhalle des TGZ zunehmend verschlechtert, nachdem sie von einer anderen Blickrichtung aufgenommen wird.

LSD-SLAM bietet den Vorteil, dass den Punkten bei der Tiefenschätzung eine Varianz zugeordnet wird. Um die Genauigkeit der Punktwolke zu erhöhen, bietet es sich an, nach dieser Varianz zu filtern (siehe Abbildung 5.5). Allerdings ließ sich kein Schwellwert finden, der hierfür allgemein passend ist. Es existieren in der berechneten Punktwolke einerseits einige Ausreißer mit niedriger Varianz, andererseits haben genaue Punkte nicht immer eine niedrige Varianz, wodurch gerade in Bereichen mit schwacher Textur die ohnehin spärlichen Punkte herausgefiltert werden. Aus diesem Grund konnte der Vorteil im Vergleichsexperiment nur bedingt genutzt werden.



(a) Punktwolke bei maximaler relativer Varianz von $10^{-3,0}$

(b) Punktwolke bei maximaler relativer Varianz von $10^{-3,7}$

Abbildung 5.5: Einfluss des Filtern der Punktwolke nach Varianz bei LSD-SLAM am Beispiel einer Wand mit Transportboxen.

Verbessern neue Aufnahmen die bestehenden Keyframes über einen längeren Zeitraum hinweg, verringert sich die Varianz genauer Punkte weiter. Systematische Ausreißer (z.B. aufgrund von Reflexionen) werden hierdurch allerdings kaum gefiltert.

Zudem wird festgestellt, dass für hohe Performanz bei großen Karten und Punktwolken die Rechenleistung und der Arbeitsspeicher erhöht werden muss.

ORB-SLAM3

ORB-SLAM3 wird mit der selben Bildsequenz durchgeführt, die für LSD-SLAM verwendet wird. Da LSD-SLAM die Bilder nicht entzerrt, werden die Bilder bereits entzerrt aufgezeichnet. Aus diesem Grund werden die Verzeichnungsparameter für dieses Experiment auf 0 gesetzt.

Die erstellte Karte besteht aus 322 Keyframes und 14175 Punkten (siehe Abbildung 5.6). Im Vergleich zu LSD-SLAM unterscheidet sich die Anzahl an Punkten wie zu erwarten um mehrere Größenordnungen, wobei die Anzahl der Keyframes ähnlich groß ist. Die Auslastung des Arbeitsspeichers ist bei ORB-SLAM wesentlich geringer, was vor allem bei größeren Karten eine Rolle spielt.

An mehreren Merkmalen ist zu erkennen, dass die durch ORB-SLAM erstellte Karte genauer ist als das Pendant von LSD-SLAM. Die im vorherigen Abschnitt genannten „Ausleger“ liegen nun wesentlich näher aneinander und die gegenüberliegenden Wände sind hierdurch parallel. Beim genaueren Vergleich der Punktwolken wird deutlich, dass die Wände nicht nur in der Vogelperspektive, sondern auch im Querschnitt bei LSD-SLAM schief und bei ORB-SLAM parallel zueinander liegen (siehe Abbildung 5.7). Zudem befinden sich die Keyframes bei ORB-SLAM auf einer Ebene, wie in Abbildung 5.8 zu sehen ist.

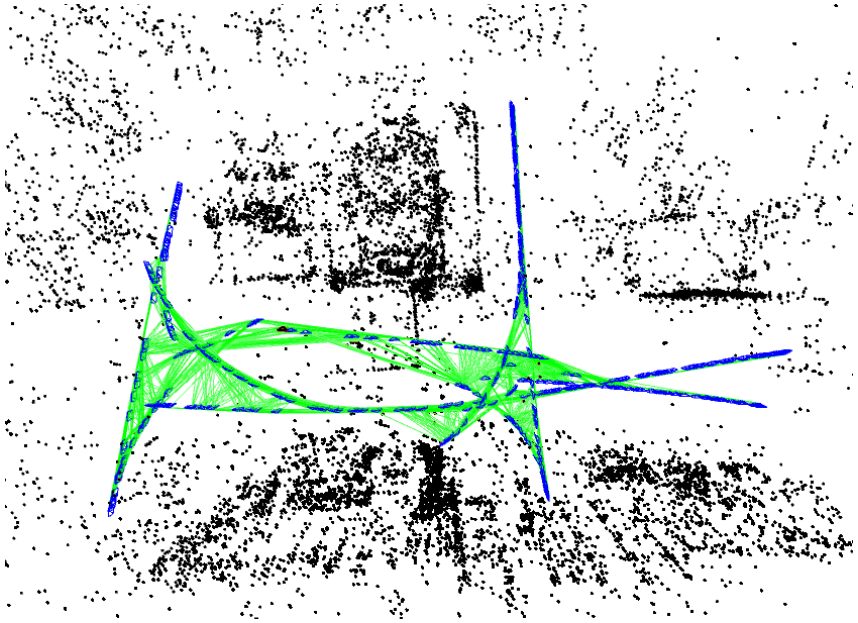
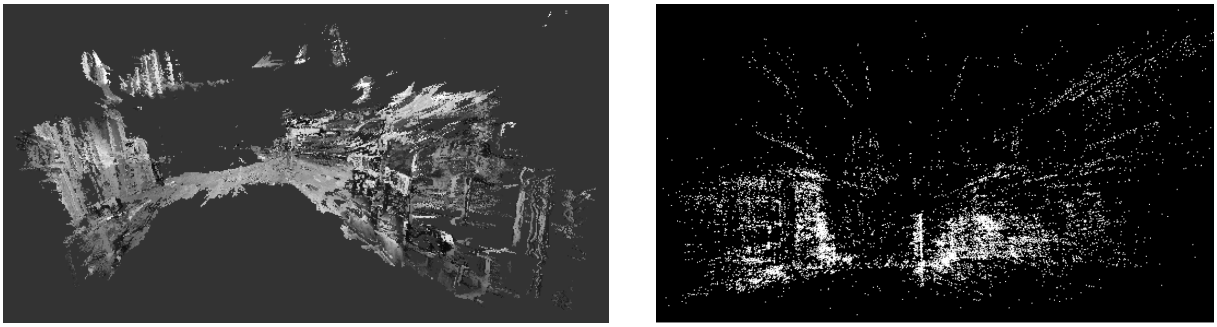


Abbildung 5.6: ORB-SLAM: Karte der Werkhalle des TGZ.



(a) **(b)**

Abbildung 5.7: Vergleich der Punktwolken von LSD-SLAM (a) und ORB-SLAM (b).

Abbildung 5.8: ORB-SLAM: Seitliche Ansicht des Keyframegraphs.

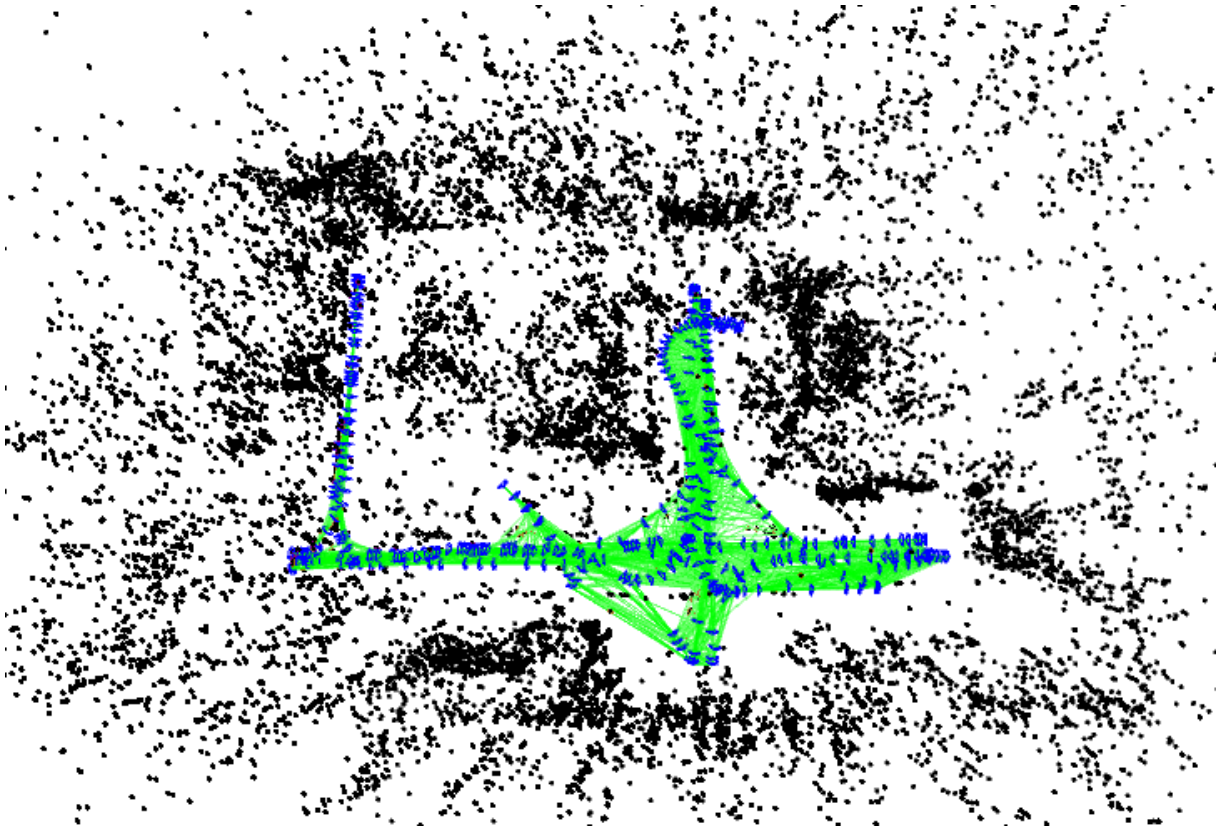


Abbildung 5.9: ORB-SLAM: Kartierung mit frontal ausgerichteter Kamera.

Fazit

Insgesamt lässt sich sagen, dass in diesem Vergleich ORB-SLAM ein deutlich besseres Ergebnis erzielt, obwohl die Robotertrajektorie und die Ausrichtung der Kamera so angepasst wurden, dass die Schwächen von LSD-SLAM umgangen werden. Dennoch hatte LSD-SLAM große Probleme mit dem Skalendrift und ungenau getrackten Rotationen, die aufgrund der Umgebung in der Werkhalle des TGZ nicht durch Large Loop Closures kompensiert werden konnten. Da die Bildsequenz aus Aufnahmen entlang einer Robotertrajektorie besteht, ergibt sich ein weniger dichter Keyframegraph als beispielsweise bei kreisenden Bewegungen mit konstanter Blickrichtung. Aus diesem Grund ist LSD-SLAM für die Anwendung auf einem mobilen Roboter schlecht geeignet, wohingegen es für die Anwendung in Bereichen wie der Augmented Reality vermutlich besser geeignet ist. Gerade bei besonders wendigen Robotern oder UAVs ist das Tracking im schlechtesten Fall unbrauchbar. Hierfür ist die Unterstützung durch eine IMU unabdingbar. ORB-SLAM zeigt sich dagegen deutlich robuster gegenüber Rotationen und Skalendrift. Zwar erzeugt LSD-SLAM eine wesentlich dichtere Punktwolke, was für Kollisionsvermeidung einen großen Vorteil darstellt, andererseits hatte die Punktwolke auch deutlich mehr schwer zu erkennende Ausreißer. Aus diesen Gründen stellt ORB-SLAM die passende Wahl für die folgenden Erweiterungen dar.

ORB-SLAM zeigt auch bei frontal ausgerichteter Kamera und Trajektorien mit engen Kurven eine gute Performanz. Ein Beispiel ist in Abbildung 5.9 dargestellt.

Da die deutlichen Nachteile von LSD-SLAM gegenüber ORB-SLAM bereits anhand der beschriebenen Merkmale deutlich werden, wird auf eine nähere Bewertung mithilfe genauer Referenzmessungen durch ein Indoor Positioning System und Laserscanner verzichtet.

Das Ergebnis steht nicht im Konflikt mit den im Abschnitt 4.2.1 genannten Vergleichen von Wang und Shahbazi [49] und Filipenko und Afanasyev [16]. Im Vergleich von Wang und Shahbazi [49] wurde lediglich die Punktwolke einer Wand rekonstruiert, wobei die genannten Schwächen von LSD-SLAM kaum Wirkung tragen. Filipenko und Afanasyev [16] fuhren mit einem Roboter in einer Schleife, weshalb eine Large Loop Closure die Ungenauigkeiten korrigieren konnte.

Der Vergleich im Rahmen dieser Arbeit beschränkt sich auf die Tauglichkeit des vSLAM-Algorithmus für die Anwendung zur Lokalisierung und Navigation eines mobilen Roboters in einer Industrieumgebung. Die Stärken und Schwächen beziehen sich auf diesen Anwendungsfall. Das Ergebnis der besseren Eignung von ORB-SLAM erhebt aus diesem Grund keinerlei Anspruch auf allgemeine Gültigkeit. Ein allgemeiner Vergleich zwischen LSD-SLAM und ORB-SLAM ist nur schwer möglich, da die individuellen Vor- und Nachteile je nach Anwendungsfall eine unterschiedlich große Rolle spielen.

5.2 Skalenrekonstruktion und absolute Lokalisierung

Im Folgenden wird das in Abschnitt 4.3 implementierte System zur absoluten Lokalisierung anhand einer Messreihe getestet und bewertet.

5.2.1 Anwendung des Systems zur Lokalisierung

Die in Kapitel 4.3 entwickelten Nodes (vgl. Abschnitt 4.3.1) stellen ein automatisiertes, absolutes Lokalisierungssystem dar. Nach dem Start des Systems wird zunächst das AprilTag Pattern zur Kalibrierung der Skalierung und des Weltkoordinatensystems angefahren. Die Kamera muss das Pattern aus verschiedenen Perspektiven aufnehmen, wobei sich die Kamera über mehrere Sekunden nicht bewegen darf. Die Messpunkte zur Kalibrierung werden automatisch aufgenommen und sobald genügend Punkte vorhanden sind, initialisiert sich das Lokalisierungssystem und veröffentlicht die errechnete Pose im skalentreuen Weltkoordinatensystem. Die Karte erweitert sich von selbst, solange sich das System nicht im Lokalisierungsmodus befindet.

Der Lokalisierungsmodus, bei dem die Kartierung abgeschaltet ist, muss mit Bedacht verwendet werden. Sobald die Kamera eine Position mit einer bestimmten Perspektive einnimmt, zu der kein passendes Keyframe zum Vergleich herangezogen werden kann, schlägt das Tracking fehl. Das passiert beispielsweise, wenn der Roboter sich in einem Gang an einer Stelle dreht, an der er sich während des Kartierungsprozesses nicht gedreht hat. Hinzu kommt, dass, falls bereits genügend Keyframes in einem Bereich existieren, ohnehin keine weiteren Keyframes aufgenommen werden.

5.2.2 Versuchsaufbau der Messreihe zur Auswertung

Um die Qualität des Lokalisierungssystems zu bewerten, werden die errechneten Posen mit den Messungen des iGPS der Firma Nikon Metrology (ehemals Metris) verglichen, das als optisches System eine Genauigkeit im niedrigen Millimeterbereich verspricht [36]. Das Messprinzip des Metris iGPS basiert auf mehreren festen Laser-Transmittern und einer Sensoreinheit, die auf dem Roboter fixiert ist. Ein Kalibrierprozess, bei dem zwei Sensoren mit bekanntem Abstand durch den Raum bewegt werden und im Optimalfall etwa 100 Datenpunkte aufgenommen werden, bestimmt die exakte Position der Transmitter zueinander. Aus den Winkeln der Laserstrahlen, die sich bei einer Messung im Sensor schneiden, berechnet sich die Position eines Sensors. Um die Pose des Roboters zu messen, werden vier zueinander kalibrierte Sensoren mit einer Position Calculation Engine (PCE) verbunden und auf dem Roboter fixiert. Das Tool Surveyor verbindet sich mit der PCE und stellt die Pose somit für Logging-Tools zur Verfügung. Details zur Funktionsweise und Genauigkeit des Metris iGPS sind in Depenthal [10] zu finden.

Als AprilTag Bundle für die Kalibrierung der Skalierung wird ein Pattern mit den Tags 300 bis 323 der Familie 36h11 verwendet. Die Tags sind bei einer Seitenlänge von 85 mm in sechs Reihen und vier Spalten angeordnet und auf Glas foliert, sodass Unebenheiten im Pattern vermieden werden und eine genauere Kalibrierung möglich ist (vgl. Abschnitt 4.3.3).

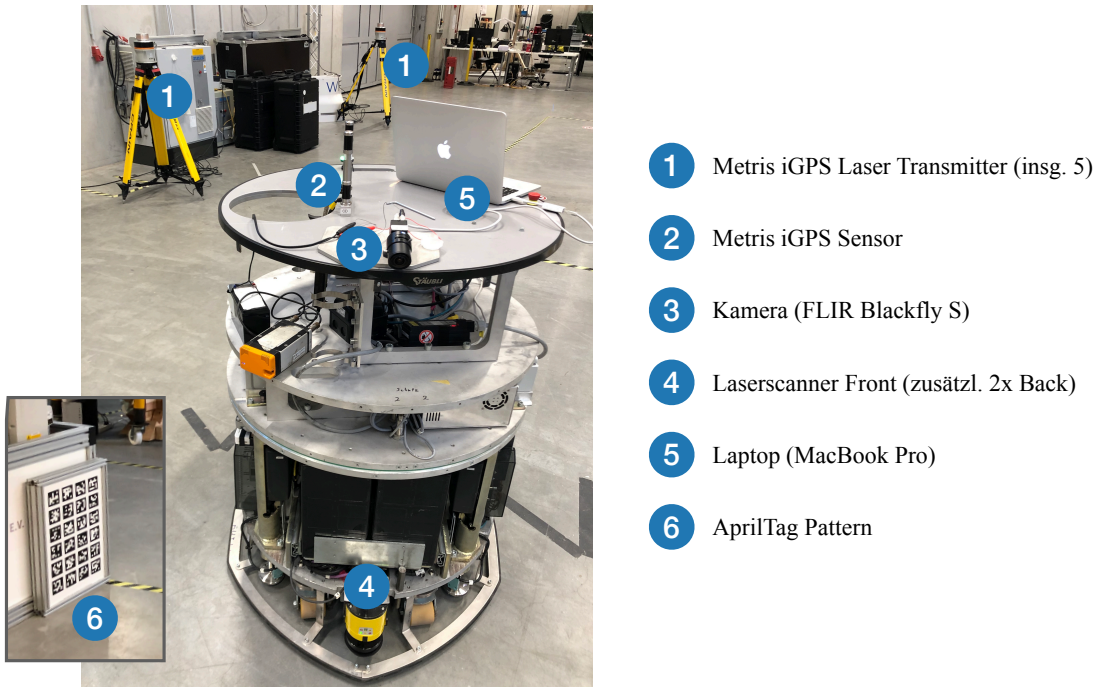


Abbildung 5.10: Aufbau zur Messreihe: Stäubli WFT HelMo (Roboter) mit FLIR Blackfly S (Kamera) und senkrechtem AprilTag Pattern in der Werkhalle des TGZ.

Um das Pattern weiter in den Mittelpunkt des Kamerabilds zu rücken, wird die Kamera an einer Halterung angebracht, die um 20 ° nach unten geneigt ist. So tritt der Effekt der Verzeichnung durch die Optik der Kamera weniger stark auf. Ein weiterer Vorteil ist, dass das Sichtfeld der Kamera, die auf dem Roboter angebracht ist, größere Teile des Bodens sowie bodennaher Hindernisse beinhaltet und Objekte, die deutlich über dem Roboter liegen und somit zur Hinderniserkennung ohnehin unbedeutend sind, weniger oft sichtbar sind. Neigt sich die Kamera allerdings zu stark, verschlechtert sich das vSLAM-System, da auf dem Boden meist nur wenige visuelle Features zum Tracking vorhanden sind.

Die Halterung wurde individuell für diesen Anwendungsfall entworfen und 3D gedruckt (siehe Abbildung 5.11). Die sonstige für die Messreihe verwendete Hardware entspricht der aus dem vorherigen Vergleichsexperiment (siehe Abschnitt 5.1.1). Der Aufbau besteht somit aus dem Stäubli WFT HelMo, an dem die Kamera FLIR Blackfly S befestigt ist, und einem AprilTag Pattern, das auf Glas foliert ist (siehe Abbildung 5.10). Die Software wird auf einem MacBook Pro (Anfang 2015) mit einem Intel Core i5-5257U Prozessor und einem 8 GB DDR3 Arbeitsspeicher ausgeführt.

Im Gegensatz zum vorherigen Vergleichsexperiment, wird die Kamera nicht anhand eines Schachbrettmusters, sondern anhand des beschriebenen AprilTag Patterns kalibriert, wodurch in diesem Fall aufgrund des verwendeten physikalischen Patterns, das besonders eben ist, deutlich mehr und genauere Punkte für die Kalibrierung verwendet werden.

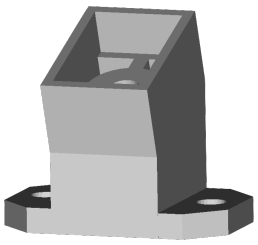


Abbildung 5.11: 3D-Modell der Kamerahalterung mit 20° Neigung.

Zur Evaluierung fährt der Roboter große Runden mit einer Länge von fast 10 m und einer Breite von etwa 3 m in der Werkhalle des TGZ ab. Dabei werden sowohl die vom Metris iGPS gemessenen Posen als auch die vom vSLAM-System errechneten Posen mit Zeitstempeln aufgezeichnet. Im Nachhinein werden die Zeitstempel synchronisiert, indem jeweils der Zeitpunkt bestimmt wird, zu dem der Roboter erstmals seine Position ändert. Da die iGPS Posen mit 40 Hz und die SLAM-Posen mit 10 Hz aufgezeichnet werden, erfolgt eine zusätzliche Verfeinerung. Der Betrag der ersten Positionsänderung, die vom vSLAM-System wahrgenommen wurde, wird mit den akkumulierten Beträgen der vier Positionsänderungen des iGPS nach der ersten Positionsänderung verglichen. Je nachdem wie viele Positionsschritte des iGPS summiert werden müssen, um den Positionsschritt des vSLAM-Systems zu erreichen, wird der zeitliche Versatz in 0,025 s Schritten angepasst. Anschließend wird jedem Datenpunkt des vSLAM-Systems ein passender Datenpunkt des iGPS zugeordnet, sodass das Datenpaar später verglichen werden kann.

Um eine aussagekräftige Datenmenge zu erhalten, wurden acht Karten aufgenommen. Für jede Karte wurde die Skalenkalibrierung fünf mal mit acht Punkten und fünf mal mit 25 Punkten durchgeführt und jeweils eine Runde gefahren. Insgesamt ergeben sich daraus 80 Runden bzw. Datensätze. Bei 14 der Datensätze traten unerwartete Fehler in den Zeitstempeln des iGPS auf, die erst zum Zeitpunkt der Auswertung festgestellt wurden und die Synchronisation der Daten behinderten. Aus diesem Grund werden in der Auswertung nur 66 Datensätze einbezogen, was nichtsdestotrotz die statistische Aussagekraft aufrechterhält. Generell gilt die Faustregel, dass bei Strichproben mit $n > 30$ approximativ eine standardisierte Normalverteilung angenommen werden kann.

5.2.3 Hand-Eye Kalibrierung

Für die Implementierung der Hand-Eye Kalibrierung (siehe Abschnitt 3.11) wird der Python Code der CRI (Control Robotics Intelligence) Group der Nanyang Technological University Singapur verwendet.[3] Darin sind Methoden zur Lösung der grundlegenden Gleichung (3.61) implementiert. Für die Auswertung wird der Tsai-Lenz Solver verwendet. In einem Python Skript werden aus den Paaren der gemessenen Posen die homogenen Transformationsmatrizen A_i/A_j und B_i/B_j und damit die Matrizen A und B bestimmt, die an den Solver der CRI Group übergeben werden. Da sich die Orientierung im SLAM-Koordinatensystem analog zur Orientierung

[3] https://github.com/crigroup/handeye

im Referenzkoordinatensystem ändert, wird die vom iGPS ermittelte Orientierung sowohl für A als auch B verwendet, da diese genauer als die vom vSLAM-System gemessene Orientierung ist.

Mit der erhaltenen Transformation X lässt sich Y, wie zuvor beschrieben, durch einfache Matrizenmultiplikation errechnen. Damit lassen sich die iGPS Posen P_{metris} in das SLAM-Koordinatensystem transformieren und um die Hand-Eye Kalibrierung korrigieren:

$$P_{SLAM} = Y \; P_{metris} \; X \tag{5.1}$$

Um eine hohe Genauigkeit trotz möglicher Fehler in den Messwerten zu erreichen, werden für die Kalibrierung wiederholt jeweils fünf zufällige Posenpaare ausgewählt und letztendlich das Ergebnis verwendet, das den kleinsten durchschnittlichen Fehler aufweist. Eine zusätzliche Korrektur erfolgt durch die Anwendung des ICP-Algorithmus auf die transformierten Punkte. Hierfür wird eine Implementierung der Bibliothek Open3D verwendet.

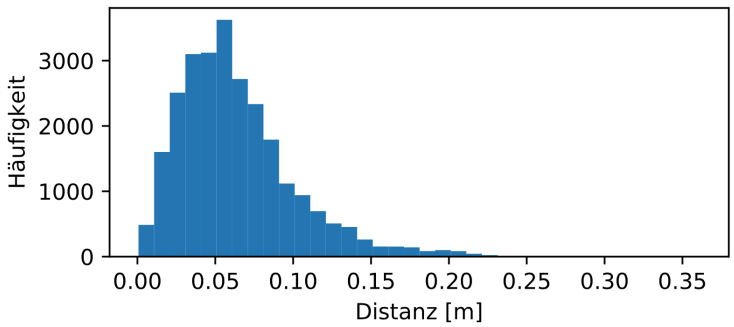
5.2.4 Analyse der Genauigkeit

Um die Genauigkeit zu analysieren werden die vom vSLAM-System ermittelten Posen einer Trajektorie mit den Posen des hochgenauen Referenzsystems Metris iGPS, die durch die Hand-Eye Transformation korrigiert wurden, verglichen.

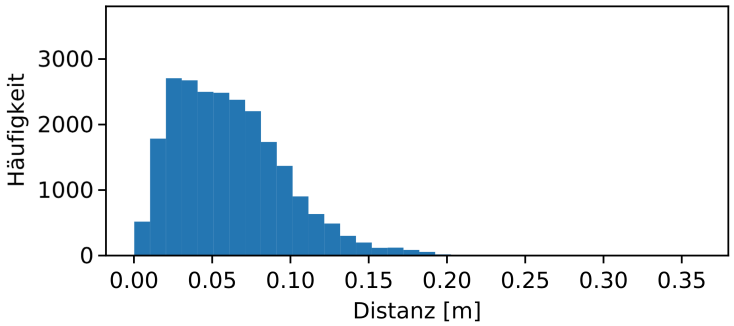
In Abbildung 5.12 wird das Gesamtergebnis als Histogramm der Positionsfehler über alle Durchläufe dargestellt. Dabei werden die Durchläufe nach der Anzahl der Punkte zur Skalenkalibrierung aufgeteilt. Bei der Kalibrierung mit acht Punkten wird eine durchschnittliche Distanz von 6,36 cm der SLAM-Punkte zum Referenzsystem erreicht. Das 95. Perzentil beträgt 13,74 cm. Einen ähnlichen durchschnittlichen Positionsfehler von 6,13 cm mit einem 95. Perzentil von 12,68 cm weist die Kalibrierung mit 25 Punkten auf. Die Kalibrierung mit 25 Punkten erzielt zwar ein besseres Ergebnis, allerdings ist der Mehraufwand bei der Kalibrierung gegen den Nutzen abzuwägen. Abbildung 5.12 zeigt, dass die Kalibrierung mit acht Punkten nicht signifikant schlechter ist.

Für eine deutliche Verbesserung der Kalibrierung muss das Kalibrierungsverfahren ausgereift werden anstatt die Anzahl der Kalibrierpunkte weiter zu erhöhen. Eine Möglichkeit ist es, das AprilTag Bundle nicht auf ein Pattern zu beschränken, sondern mehrere Patterns mit großem Abstand zueinander im Raum zu verteilen. Dadurch haben die Punkte, die für die Kalibrierung aufgenommen werden, größere Abstände, wodurch Ungenauigkeiten bei der Bestimmung der Transformation weniger ins Gewicht fallen. Das Problem ist hierbei, dass die genaue relative Pose der Patterns zueinander bekannt sein muss.

Ein Durchlauf mit hoher Genauigkeit ist in Abbildung 5.13 zu sehen. Hier beträgt der durchschnittliche Positionsfehler 3,86 cm, das 95. Perzentil 7,81 cm und der größte Positionsfehler 9,48 cm. Wie in der Abbildung zu erkennen ist, treten im Bereich $x = [6; 10]$ häufig Lücken auf. Diese resultieren aus Störungen des iGPS, das mit infrarotem Laser arbeitet. Da die skalenbedingten Fehler größer werden, je weiter ein Punkt vom Ursprung entfernt liegt, wird der



(a) Kalibrierung mit acht Punkten



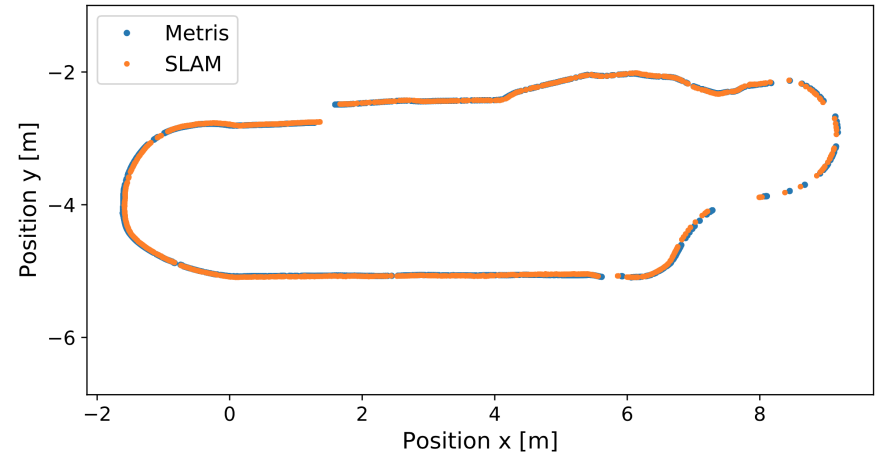
(b) Kalibrierung mit 25 Punkten

Abbildung 5.12: Histogramme der Positionsfehler über alle Runden, aufgeteilt nach der Anzahl der Punkte zur Kalibrierung.

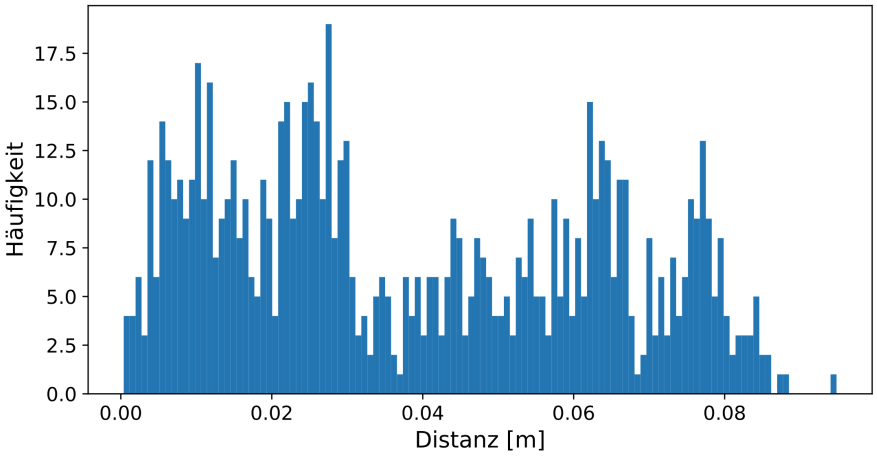
Mittelwert der Positionsfehler hierdurch verfälscht. Der Mittelwert bei gleich verteilten Punkten liegt demnach höher. Der Effekt tritt bei jedem Durchlauf auf, da das Metris iGPS in dem betroffenen Bereich generell Störungen aufwies.

Der verbleibende Positionsfehler resultiert aus verschiedenen Fehlerquellen. Aufgrund der Synchronisation der Zeitstempel können die Positionen vom iGPS und vSLAM-System einander nicht exakt zugeordnet werden. Durch das Messrauschen sind auch falsche Zuordnungen möglich. Ein zeitlicher Versatz von 0,025 s bewirkt bei einer Geschwindigkeit von 0,5 $^{m}/_{s}$ bereits einen Positionsfehler von 1,25 cm. Dazu kommt die Ungenauigkeit des vSLAM-Algorithmus, sowie der Kamerakalibrierung und zusätzliche Verzeichnung des Bilds. Änderungen der Lichtverhältnisse und sich bewegende Objekte führen zu Fehlern im Feature-basierten Tracking. Bei der Auswertung werden die Positionsfehler durch die Ungenauigkeiten bei der Hand-Eye Kalibrierung leicht verfälscht. Sprunghafte Ausreißer des vSLAM-Systems sind selten.

Am anfälligsten zeigt sich insgesamt jedoch die Skalenkalibrierung. Ein Durchlauf mit niedriger Genauigkeit ist in Abbildung 5.14 dargestellt (durchschnittlicher Positionsfehler: 8,53 cm, 95. Perzentil 13,96 cm). Hier ist zu sehen, dass die vom vSLAM-System aufgenommene Trajektorie von der iGPS Trajektorie umschlossen ist. Wird die vom vSLAM-System aufgenommene Trajektorie vergrößert, deckt sie sich annähernd mit der iGPS Trajektorie. Da die zugrundeliegende



(a) Trajektorie

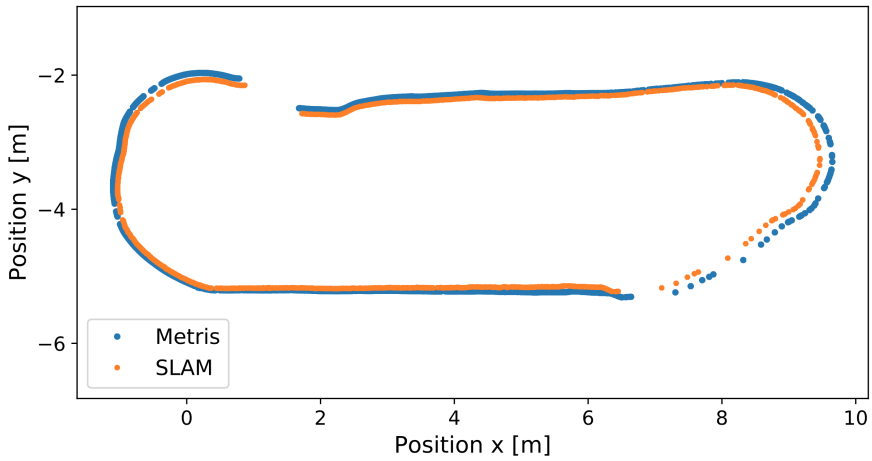


(b) Positionsfehler

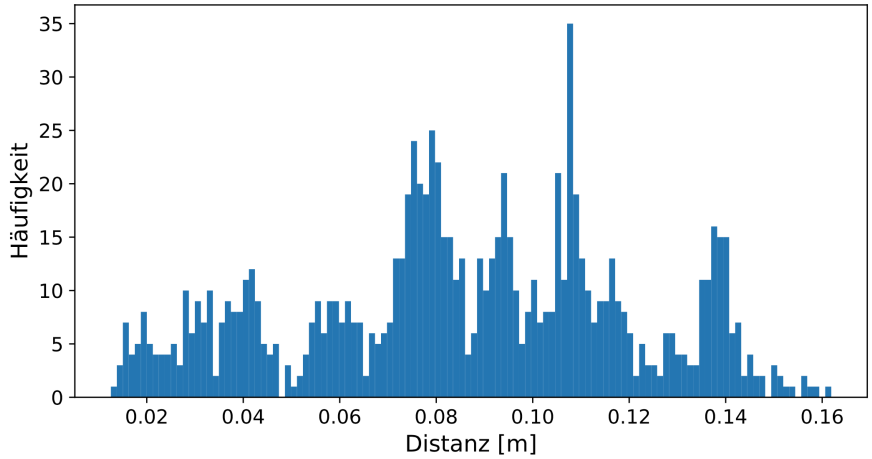
Abbildung 5.13: Trajektorie und Positionsfehler eines Durchlaufs mit hoher Genauigkeit.

ORB-SLAM3 Karte die selbe wie in Abbildung 5.13 ist, deutet das auf die Skalierung als hauptsächliche Fehlerquelle hin. Aufgrund der Tatsache, dass sich der RMS Reprojektionsfehler bei der Berechnung der Skalierung und Transformation bei beiden Durchläufen kaum unterscheidet, kann dieser nicht als Indiz für eine schlechte Kalibrierung verwendet werden. Als Ursache wird die Ungenauigkeit der Kalibrierpunkte in Zusammenhang mit der zu geringen räumlichen Verteilung gesehen.

Die Tabellen 5.2 und 5.3 geben einen Überblick über die Güte der einzelnen Karten. Sie enthalten die jeweiligen Durchschnittsfehler, Maximalfehler und 95. Perzentile der Position und Orientierung. Da die Werte der Karten bei acht und 25 Kalibrierpunkten nicht eindeutig korrelieren und auch bei den Werten der einzelnen Durchläufe jeder Karte (siehe Anhang A, Tabellen A.1 und A.2) kein klarer Zusammenhang zu erkennen ist, wird die Skalenkalibrierung als ausschlaggebend für die Güte gesehen. Die ORB-SLAM3 Karten ermöglichen alle eine genaue Lokalisierung.



(a) Trajektorie



(b) Positionsfehler

Abbildung 5.14: Trajektorie und Positionsfehler eines Durchlaufs mit niedriger Genauigkeit.

Der Kartierungsprozess erweist sich als robust.

Insgesamt wurde bei der Positionsbestimmung ein überaus zufriedenstellendes Ergebnis erzielt.

Ähnliche Ergebnisse liefert die Auswertung der Orientierungswinkel. In Abbildung 5.15 ist der Orientierungswinkel des Durchlaufs aus Abbildung 5.14 dargestellt. Der Orientierungswinkel weicht in diesem Durchlauf im Durchschnitt 3,64 ° von der Referenzmessung ab. Über alle Durchläufe mit acht Kalibrierpunkten ist die durchschnittliche Abweichung 3,23 ° und das 95. Perzentil 6,62 °. Bei 25 Kalibrierpunkten beträgt die durchschnittliche Abweichung 3,04 ° und das 95. Perzentil 6,30 °.

Durch eine höhere Anzahl an Kalibrierpunkten ist im Gegensatz zur Positionsbestimmung keine eindeutige Verbesserung zu verzeichnen. Der Grund dafür ist, dass der Orientierungswinkel

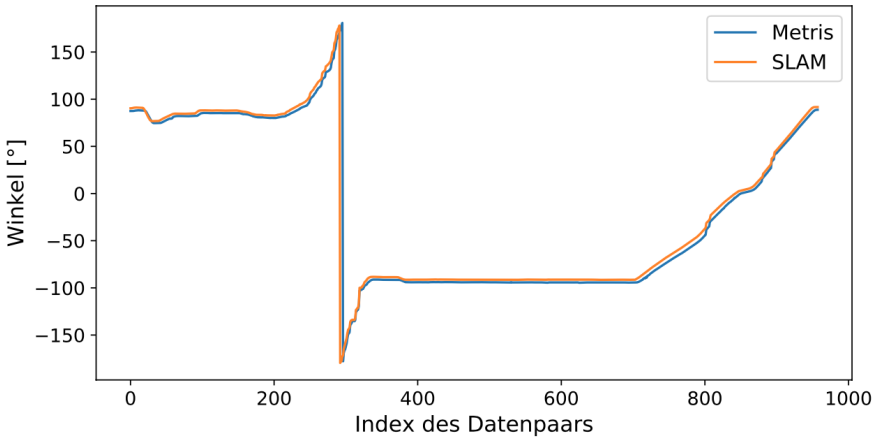
Durchlauf	ø Positionsfehler [m]	worst case Positionsfehler [m]	95. Perzentil [m]	ø Orientierungsfehler[°]	worst case Orientierungsfehler [°]	95. Perzentil [°]
Karte 1	0.07489	0.23233	0.13864	3.469	11.849	6.957
Karte 2	0.06810	0.25472	0.14014	2.476	12.115	6.278
Karte 3	0.07964	0.29926	0.15893	2.602	33.274	6.162
Karte 4	0.05180	0.30589	0.13873	2.744	9.512	6.009
Karte 5	0.06020	0.19871	0.13562	2.906	13.108	6.214
Karte 6	0.08030	0.36168	0.17286	3.794	15.259	7.704
Karte 7	0.04468	0.13937	0.08164	3.347	10.734	6.416
Karte 8	0.05044	0.19913	0.10870	4.651	35.617	12.921

Tabelle 5.2: Übersicht über die Durchläufe der Messreihe mit acht Punkten zur Skalenkalibrierung.

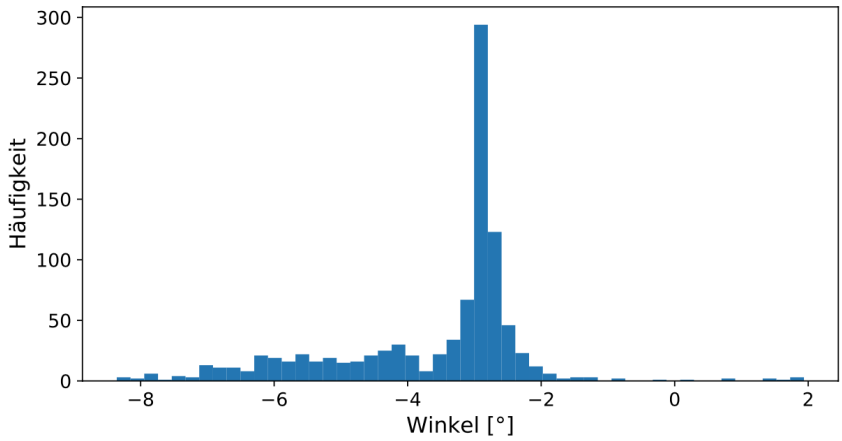
Durchlauf	ø Positionsfehler [m]	worst case Positionsfehler [m]	95. Perzentil [m]	ø Orientierungsfehler[°]	worst case Orientierungsfehler [°]	95. Perzentil [°]
Karte 1	0.06735	0.22463	0.13516	2.406	7.859	4.838
Karte 2	0.06792	0.15658	0.12786	3.832	11.332	7.531
Karte 3	0.05002	0.21793	0.10493	3.319	9.986	6.082
Karte 4	0.06824	0.16854	0.11334	3.485	12.966	6.834
Karte 5	0.05793	0.17393	0.10862	3.294	10.268	6.411
Karte 6	0.07053	0.20125	0.14236	2.577	8.407	4.515
Karte 7	0.04095	0.10529	0.07115	2.150	9.002	5.018
Karte 8	0.06983	0.23292	0.16209	2.585	22.491	12.019

Tabelle 5.3: Übersicht über die Durchläufe der Messreihe mit 25 Punkten zur Skalenkalibrierung

nicht von der Skalierung betroffen ist. Die Transformation vom SLAM- ins Weltkoordinatensystem des vSLAM-Systems wird bei der Hand-Eye Kalibrierung herausgerechnet, sodass diese ebenfalls keine Rolle spielt. Aus diesem Grund liegt die Fehlerquelle hier vor allem im vSLAM-Algorithmus und der Synchronisation, weshalb die Güte durch die Verbesserung der Skalenkalibrierung kaum beeinflusst wird. Aufgrund der Tatsache, dass der Orientierungswinkel des vSLAM-Systems meist etwas größer als der Orientierungswinkel des Metris iGPS ist, liegt die Vermutung nahe, dass die Hand-Eye Kalibrierung einen kleinen systematischen Fehler hervorruft, der aus einer unzureichenden Projektion in die Fahrtebene resultiert.



(a) Verlauf des Orientierungswinkels



(b) Orientierungsfehler

Abbildung 5.15: Verlauf und Abweichung des Orientierungswinkels eines Durchlaufs mit durchschnittlicher Genauigkeit.

5.3 Occupancy Grid Map zur freien Navigation

Das in Abschnitt 4.4 entwickelte Verfahren zur Erstellung einer Occupancy Grid Map ermöglicht es, neben der hochgenauen Pose des Roboters auch eine Karte mit Hindernissen zur Verfügung zu stellen. Damit ist die Basis für autonome und freie Navigation gegeben. In diesem Kapitel werden zunächst das Kartierungsverfahren und die Eigenschaften der erstellten Occupancy Grid Maps qualitativ bewertet. Im Anschluss wird durch eine Versuchsreihe die Eignung zur autonomen Navigation demonstriert und das Ergebnis quantitativ festgehalten.

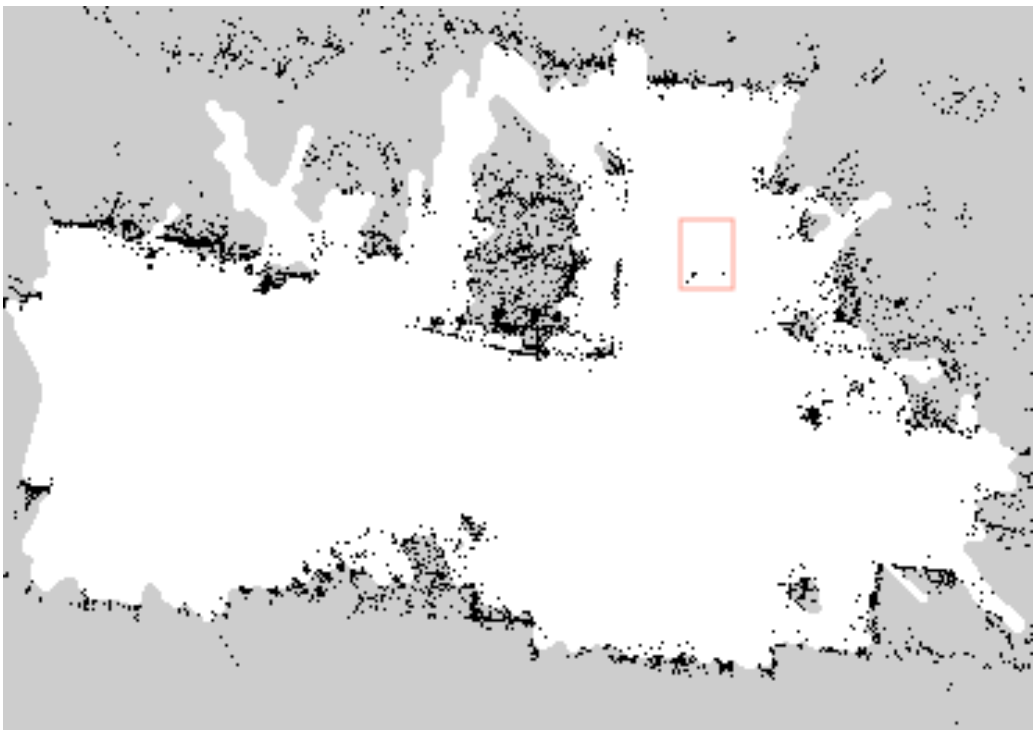
5.3.1 Stärken und Schwächen des Verfahrens zum Occupany Grid Mapping

Eine beispielhafte Occupancy Grid Map der Werkhalle des TGZ ist in Abbildung 5.16 (a) dargestellt. Die Auflösung beträgt 6,25 cm, was einer Unterteilung eines Quadratmeters in $16 \times 16 = 256$ gleichgroße Zellen entspricht. Diese Auflösung ist für die reine Anwendung zur Pfadplanung und Kollisionsvermeidung gut geeignet, da eine höhere Auflösung mehr Rechenaufwand und größere Karten-Dateien zu Folge hat, die gerade bei großflächigen Umgebungen für Mikrocontroller schwer zu verarbeiten sind. Zudem genügt die Auflösung für die Anwendung zur Pfadplanung und Kollisionsvermeidung. Wird die Karte für AMCL in Verbindung mit Laserscannern verwendet, ist die Auflösung ausschlaggebend für die Genauigkeit der Lokalisierung.

Die Karte für AMCL wird in der Regel vom *gmapping*[4] Algorithmus, einem laserbasierten SLAM-Algorithmus, erstellt. Im Vergleich zu der Karte, die der *gmapping* Algorithmus in der selben Umgebung erstellt (siehe Abbildung 5.16 (b)), werden Wände weniger klar wahrgenommen. Zur Pfadplanung reicht es jedoch aus, wenn die Hindernisse im Groben erkennbar sind, da die Karten ohnehin durch eine Costmap, die zeigt wo sich der Referenzpunkt des Roboters bewegen darf, verfeinert wird. Hierdurch reichen wenige Punkte aus um ein großes Hindernis in der Karte zu verzeichnen. Einen flachen Rollwagen erkennen beiden Systemen beispielsweise nur schlecht. Das vSLAM-System erkennt nur wenige Punkte und *gmapping* erkennt die vier Rollen, jedoch nicht den restlichen Aufbau (vgl. Abbildung 5.16, rot markiert). Im rechten unteren Bereich der Karten befinden sich ein kleiner Antennenmast und eine größere Box auf Paletten. Die Laserscanner, die niedrig am Roboter angebracht sind, erfassen die gesamte Palette, wohingegen das vSLAM-System lediglich den Mast und die Box selbst erfassen. Hier ist das Ergebnis des *gmapping* Algorithmus besser zur Kollisionsvermeidung geeignet. Anders verhält es sich beim zentralen Hindernis (links des Rollwagens), das zur rechten und unteren Seite von einem Absperrband umgeben ist. Die Laserscanner erkennen hier nur die Pfosten, an denen das Band angebracht ist, nicht jedoch – wie das vSLAM-System – die Absperrbänder selbst.

Große freie Flächen werden vom Algorithmus zuverlässig erkannt. Flache Hindernisse, wie auf dem Boden liegende Kabel oder leichte Absätze werden allerdings nicht berücksichtigt, da diese kaum von Linien und Markierungen am Boden unterscheidbar sind. Im Allgemeinen werden texturreiche Objekte gut erkannt. Bei eintönigen und kontrastarmen Hindernissen sind dagegen kaum ORB-Features zu erkennen. Davon sind vor allem kahle Wände und Tore betroffen. In

[4] http://wiki.ros.org/gmapping



(a) Karte der Werkhalle des ZfT, aufgenommen mit dem entwickelten vSLAM-System. Die Position des Rollwagens wurde nachträglich rot markiert.

(b) Karte der Werkhalle des ZfT, aufgenommen mit dem *gmapping* Algorithmus.

Abbildung 5.16: Vergleich der Kartierung des vSLAM-Systems mit *gmapping*.

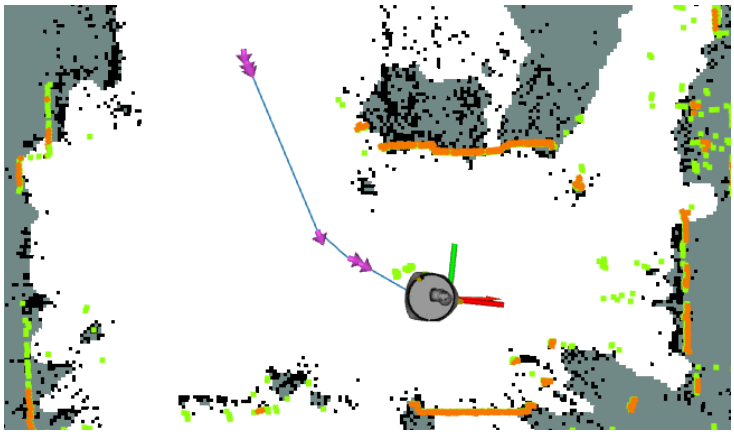


Abbildung 5.17: Beispielhafter Pfad von Start- zu Zielpose in zuvor aufgenommener Occupancy Grid Map. In orange/grün sind die Messungen der Laserscanner zu sehen.

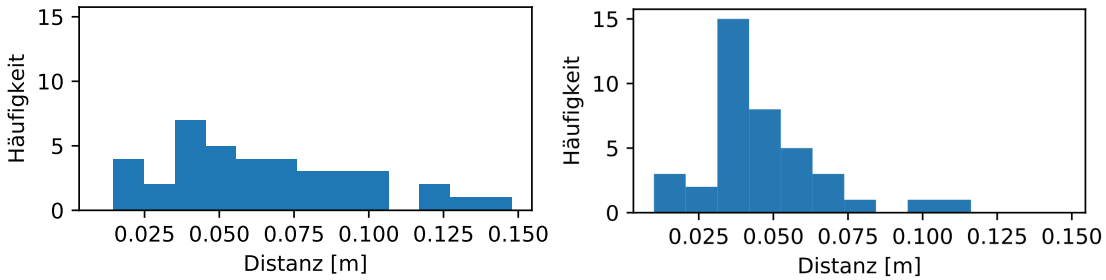
Abbildung 5.16 ist allerdings zu sehen, dass diese zwar schlechter aber dennoch ausreichend gut erkannt werden.

Die Genauigkeit der Karte ist abhängig von der Genauigkeit der Lokalisierung und damit vor allem von der Qualität der Skalenrekonstruktion. Da sich der skalenbedingte Fehler in der Karte allerdings mit dem skalenbedingten Lokalisierungsfehler ausgleicht, ist dieser Fehler für die Pfadplanung und Kollisionsvermeidung nicht von Bedeutung. Die Abweichung muss lediglich bei der Vorgabe von Zielposition und ähnlichem beachtet werden. Dafür bietet es sich an, die Positionen in der Karte zu verzeichnen, statt diese separat zu vermessen. Eine Karte, die auf Basis von Laserscans aufgenommen wird, kann eine deutlich höhere Genauigkeit erreichen und enthält mehr Details als die Occupancy Grid Map, die der entwickelten Algorithmus erzeugt. Da Roboter jedoch ohnehin mit Sensoren zur Kollisionsvermeidung (Infrarot, Sonar, etc.) ausgestattet sein müssen, um auf bewegliche Objekte und Veränderung der Umgebung zu reagieren, ist die genannte Beeinträchtigung akzeptabel. Aus diesem Grund kann eine Laserscan-basierte Karte durch das entwickelte Verfahren prinzipiell ersetzt werden, wodurch teure Laserscanner nicht mehr erforderlich sind. Insgesamt wurde somit ein praktikables und damit akzeptables Ergebnis erzielt.

5.3.2 Versuchsaufbau und Messreihe zur Navigation

Um die Eignung zur autonomen Navigation zu prüfen, wird dem Roboter wiederholt ein festes Ziel vorgegeben, das dieser von einem festen Startpunkt aus anfährt. Hierbei wird überprüft, mit welcher Genauigkeit das Ziel angefahren wird und ob beim Versuch anderweitige Komplikationen auftreten.

Der Versuchsaufbau entspricht dem des vorherigen Abschnitts (siehe Abschnitt 5.2.2). Der Roboter lokalisiert sich mit dem vSLAM-System, wobei die vSLAM-Pose zusätzlich mit den hoch genauen Daten des Metris iGPS verglichen werden. Um auch beim vSLAM-System eine hohe



(a) Distanzen der Metris iGPS Endpositionen. **(b)** Distanzen der vSLAM-Endpositionen.

Abbildung 5.18: Histogramme der Distanzen der Metris iGPS und vSLAM-Endposen zur Zielposition über alle Runden.

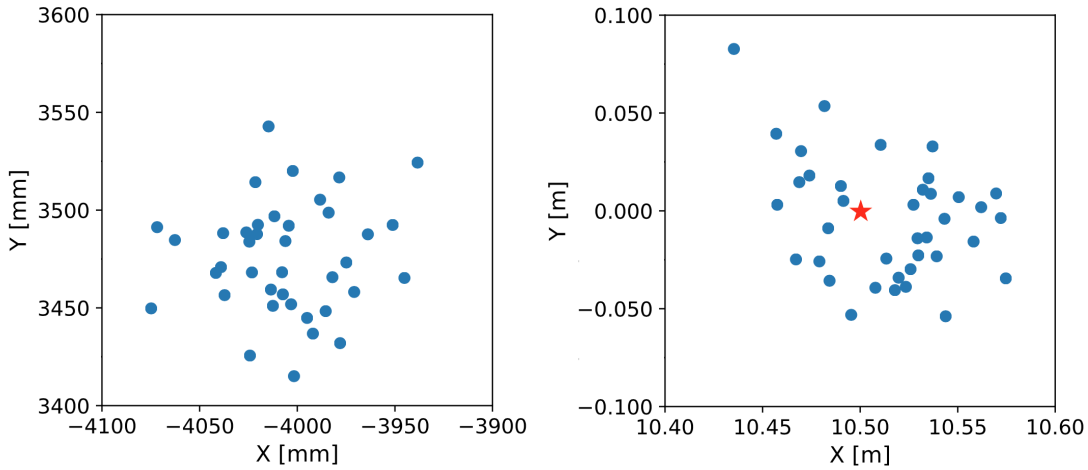
Genauigkeit zu garantieren, wird eine bestehende ORB-SLAM3-Karte verwendet, bei der die Genauigkeit der Skalentransformation bereits durch Messungen mit Metris iGPS bestätigt ist. Auf Basis dieser Karte wird durch den in Abschnitt 4.4 beschriebenen Algorithmus eine Occupancy Grid Map generiert, in der der Roboter in der Versuchsreihe navigiert.

Zu Beginn eines Durchlaufs wird der Roboter zur Startpose ($x = 6\,\mathrm{m}, y = -3.5\,\mathrm{m}$) gefahren. Von hier aus wird die Zielpose ($x = 10.5\,\mathrm{m}, y = 0\,\mathrm{m}$) vorgegeben. Der Pfadplanungsalgorithmus bestimmt nun einen Pfad in der zugrundeliegenden Occupancy Grid Map, die durch eine Costmap für die Pfadplanung aufbereitet wird. Ein beispielhafter Pfad ist in Abbildung 5.17 dargestellt, wobei zu sehen ist, dass um ein Hindernis herum navigiert werden muss. Auf Grundlage dieses Pfades sendet der Controller Kommandos an den Roboter, durch die die Pfadsegmente abgefahren werden. Die vom vSLAM-System ermittelte Pose wird dafür verwendet, die Roboterpose zu korrigieren, falls diese vom geplanten Pfad abweicht. Die Genauigkeit und Stabilität des vSLAM-Systems ist demnach fundamental für die Navigation zur Zielpose.

Insgesamt wurden 39 valide Durchläufe ausgewertet.

5.3.3 Evaluierung

Für eine quantitative Evaluierung des Systems zur autonomen Navigation werden die Endposen mit Metris iGPS gemessen, durch die Hand-Eye Kalibrierung korrigiert und den Endposen des vSLAM-Systems, sowie den vorgegebenen Zielposen gegenübergestellt. Aufgrund der hohen Genauigkeit von Metris iGPS kann die Referenzmessung annähernd als Ground Truth angenommen werden. Das Ergebnis ist in Abbildung 5.18 dargestellt. Die vom vSLAM-System gemessenen Endpositionen liegen durchschnittlich 4,669 cm von der Zielposition entfernt. Dabei liegen 90% der Endpositionen näher als 7,065 cm an der Zielposition. Da die vSLAM-Posen für die Navigation verwendet werden, liegen diese näher an der Zielposition als die von Metris iGPS gemessenen Endpositionen, die im Schnitt 6,437 cm von der Zielposition abweichen. Das 90%-Perzentil beträgt hier 10,85 cm. Die Abweichung kommt durch den Lokalisierungsfehler von durchschnittlich 4,347 cm zustande.



(a) Metris iGPS. **(b)** vSLAM-System.

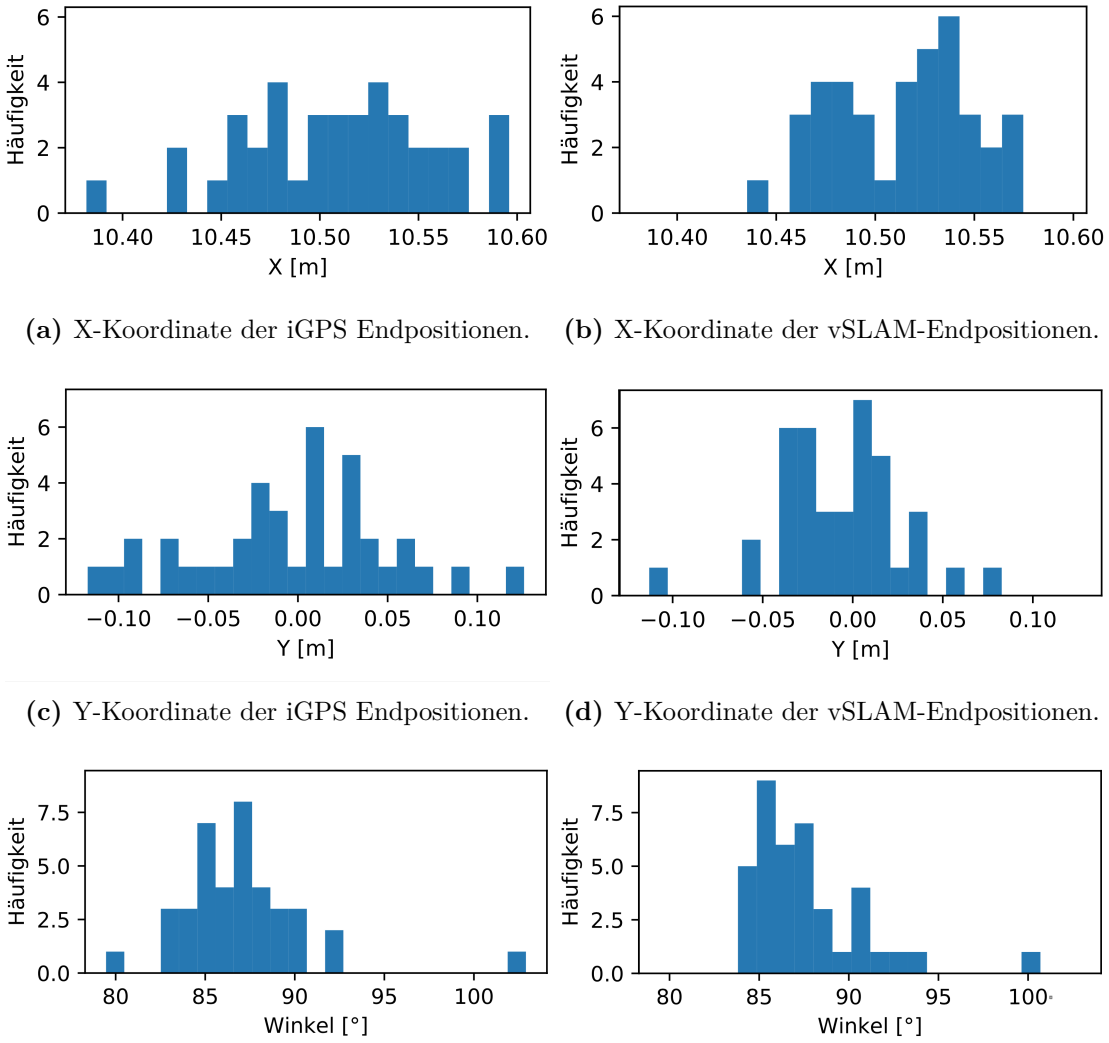
Abbildung 5.19: Wiederholungsgenauigkeit der Endpositionen, gemessen mit Metris iGPS und dem vSLAM-System (Zielposition mit Stern markiert).

Neben der Abweichung zum Zielpunkt ist auch die Wiederholungsgenauigkeit von Interesse. Bei Betrachtung der unverarbeiteten Messungen der Endposen durch Metris iGPS beträgt die Standardabweichung in x-Richtung 3,114 cm, in x-Richtung 2,802 cm und bei der Orientierung 2,293 °. Die Messungen der Endpunkte mit Metris iGPS sind den vSLAM-Endpunkten in Abbildung 5.19 graphisch gegenübergestellt.

Um systematische Fehler zu erkennen, sind die Endposen aller Durchläufe in Abbildung 5.20 in Histogrammdarstellung, aufgeteilt nach dem jeweiligen Lokalisierungssystem, zu sehen. Die X-Koordinaten liegen mit durchschnittlich 10,509 m (Metris iGPS) bzw. 10,513 m (SLAM-System) etwas über dem Ziel bei 10,5 m. Dagegen liegen die Y-Koordinaten mit durchschnittlich -0,00316 m (Metris iGPS) bzw. -0,00692 m (SLAM-System) nah am vorgegebenen Ziel von 0 m. Das liegt daran, dass der Roboter an der Zielpose in y-Richtung ausgerichtet ist und Abweichungen zur Zielposition durch vorwärts/rückwärts Fahren ausgleichen kann. Abweichungen in x-Richtung, also senkrecht zur Fahrtrichtung können nur durch umfangreichere Pfade ausgeglichen werden, was erneute Abweichungen mit sich führt. Das spiegelt sich auch in der durchschnittlichen Distanz der vSLAM-Position zur Zielposition wider, die in x-Richtung 3,295 cm und in y-Richtung lediglich 2,659 cm beträgt. Eine Übersicht über die Kennzahlen aller Durchläufe ist in Anhang B zu finden.

Die Orientierung der Endposen liegt im Schnitt bei 87,023 ° (Metris iGPS) bzw. 87,648 ° (SLAM-System) und damit um einige Grad unter der vorgegebenen Orientierung von 90 °. Für dieses Verhalten wurde keine Erklärung gefunden, da der Lokalisierungsfehler bei nur 0,823 ° liegt. Die erreichte Orientierung erfüllt nichtsdestotrotz die Anforderungen für Navigation in Industrieumgebungen, da für Anwendungen wie Docking, die eine deutlich höhere Genauigkeit benötigen, separate Technologien verwendet werden.

Alles in allem bestätigt die Messreihe, dass das entwickelte vSLAM-System als Basis zur au-



(a) X-Koordinate der iGPS Endpositionen.

(b) X-Koordinate der vSLAM-Endpositionen.

(c) Y-Koordinate der iGPS Endpositionen.

(d) Y-Koordinate der vSLAM-Endpositionen.

(e) Orientierung der iGPS Endposen.

(f) Orientierung der vSLAM-Endposen.

Abbildung 5.20: Histogramme der Endposen aller Durchläufe, gemessen durch Metris iGPS und das vSLAM-System.

tonomen Navigation seinen Zweck mehr als erfüllt. Die erreichte durchschnittliche Distanz von 6,437 cm zum Zielpunkt ist unter Berücksichtigung des Lokalisierungsfehlers von durchschnittlich 4,347 cm und der Ungenauigkeiten bei der Pfadverfolgung, die durch die Streuung der vSLAM-Endpositionen deutlich wurden (vgl. Abbildung 5.19 und Abbildung 5.20 (b) und (d)), durchaus zufriedenstellend.

5.4 Vergleich mit bisherigen Lokalisierungsmethoden

Um die bisherigen Ergebnisse zusätzlich im Vergleich zu den bestehenden Lokalisierungsverfahren einzuordnen, wird in diesem Kapitel eine Messreihe durchgeführt, die das entwickelte Lokalisierungssystem auf Basis des vSLAM-Algorithmus ORB-SLAM3 der Adaptive Monte Carlo Localization (AMCL) und der Odometrie gegenüberstellt, indem deren Lokalisierungsgenauigkeit auf prägnanten Trajektorien verglichen wird. Zusätzlich wurde ein Kalmanfilter, basierend auf den Messungen des vSLAM-Systems und AMCL (siehe Abschnitt 4.7), evaluiert.

5.4.1 Bisherige Lokalisierungsmethoden und Kalmanfilter

Von der aktuellen ROS Navigationssoftware des Zentrum für Telematik für den Stäubli WFT HelMo wird bisher nur die Lokalisierung mittels AMCL und Odometrie unterstützt. AMCL (vgl. Abschnitt 3.10) arbeitet dabei mit den drei integrierten Laserscannern, die die Umgebung zusammen um 360° abtasten. Bei einer festgesetzten Distanz zum vorherigen Messpunkt, veröffentlicht AMCL eine neue Posenschätzung. Da die Updaterate aus diesem Grund ungleichmäßig und zu niedrig zur Navigation ist, wird die Messung durch die Odometrie verfeinert, indem die Posenänderung seit dem letzten Update von AMCL, gemessen durch die Odometrie, auf die letzte Posenschätzung addiert wird. Da der Stäubli WFT HelMo über eine sehr genaue Odometrie verfügt, ist die Genauigkeit des Lokalisierungssystems hierdurch kaum beeinträchtigt.

Ein großer Nachteil von AMCL ist jedoch, dass die globale Relokalisierung in einer gegeben Karte längere Trajektorien und damit einige Zeit erfordert. Dies wird häufig dadurch umgangen, dass eine grobe Posenschätzung – beispielsweise über das Visualisierungstool rviz – an AMCL übergeben wird. Hier ist ORB-SLAM3 deutlich im Vorteil, da ein Bild ausreicht, um die Position in der gesamten Karte neu zu bestimmen. Dagegen ist AMCL durch die genauen Laserscans weniger anfällig für Skalenfehler als das vSLAM-System. Um die Stärken des vSLAM-Systems mit denen von AMCL zu verbinden, wird ein Kalmanfilter, der beide Messungen verarbeitet, in das Vergleichsexperiment miteinbezogen.

5.4.2 Beschreibung des Vergleichsexperiments

Um die Lokalisierungsverfahren zu vergleichen, wird zunächst eine aktuelle Karte der Umgebung für das Vergleichsexperiment aufgenommen. Für AMCL kommt hierfür *gmapping* zum Einsatz. Das vSLAM-System verwendet eine von den vorherigen Experimenten bestehende Karte. Anschließend werden mit dem Stäubli WFT HelMo Trajektorien in Form eines Quadrats, eines Kreises und einer Acht abgefahren, wobei die Lokalisierungsverfahren AMCL, Odometrie und das entwickelte vSLAM-System parallel laufen und die Pose des Roboters mit Metris iGPS gemessen wird. Der Versuchsaufbau gleicht somit dem Aufbau aus Abschnitt 5.2.2.

Zusätzlich läuft der Kalmanfilter, der AMCL und das vSLAM-System kombiniert und bei jedem Aktualisierungsschritt die Messungen aller Lokalisierungsverfahren aufzeichnet. Parallel wird die

Trajektorie in höchster zeitlicher Auflösung vom Metris iGPS und vSLAM-System aufgezeichnet. Diese Aufzeichnungen dienen zur Synchronisierung der Metris iGPS Messpunkte zu den ROS Zeitstempeln. Die synchronisierten Metris iGPS Posen werden, wie auch die Messungen des vSLAM-Systems, durch eine Hand-Eye Kalibrierung auf den Mittelpunkt des Roboters korrigiert. Zur Hand-Eye Kalibrierung von Metris iGPS wurde die Trajektorie des vSLAM-Systems verwendet, da diese zeitlich deutlich höher aufgelöst ist und zuvor angenommen wurde, dass diese genauer ist als die Trajektorie von AMCL und Odometrie.

Hierdurch lässt sich jeder Posenmessung ein Vergleichswert des hoch genauen Metris iGPS zuordnen, womit aus der Distanz zur Metris iGPS Pose die Genauigkeit einer Messmethode abgeschätzt wird.

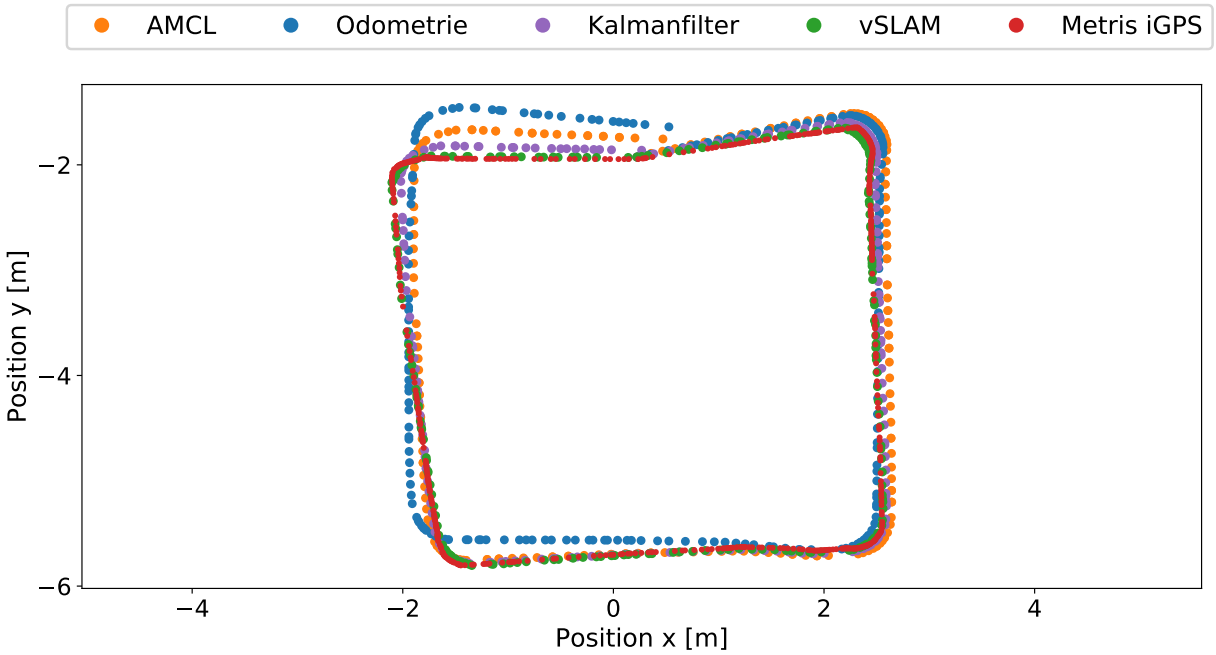
5.4.3 Ergebnisse des experimentellen Vergleichs

Insgesamt wurden vier gültige Durchläufe mit kreisförmigen Trajektorien, fünf Durchläufe mit Trajektorien in Form einer Acht und sechs Durchläufe mit annähernd quadratischen Trajektorien durchgeführt. Die einzelnen Ergebnisse sind in Anhang C aufgeführt. In diesem Abschnitt wird von jeder Form ein Beispiel betrachtet und das Gesamtergebnis evaluiert.

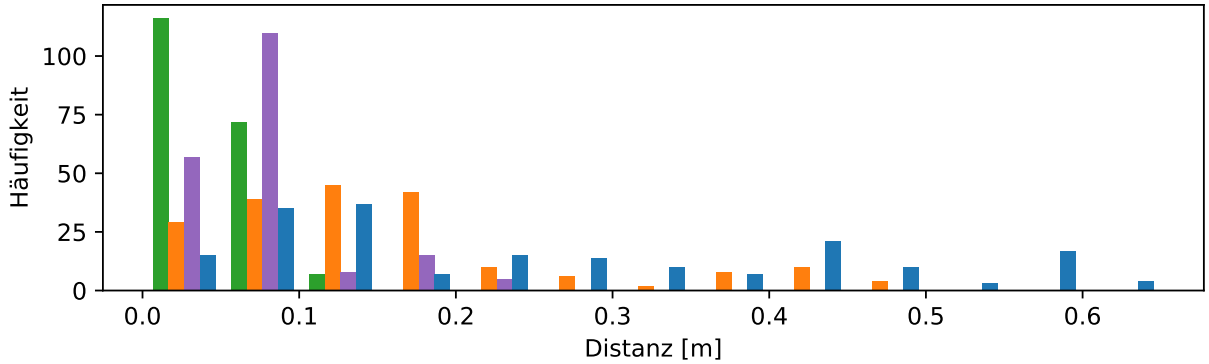
Im Vergleich der Positions- und Orientierungsfehler einer quadratischen Trajektorie (siehe Abbildung 5.21) bestätigt sich zunächst die Vermutung, dass sich die Genauigkeit der Odometrie über den Streckenverlauf zunehmend verschlechtert. Für AMCL wird die Schätzung der Odometrie zugrunde gelegt und die Pose durch Sampling korrigiert. Aus diesem Grund wird AMCL durch den zunehmenden Fehler der Odometrie beeinflusst. Praktisch wird das deutlich, indem die AMCL Pose gerade bei Rotationen durch die Odometrie verfälscht wird, da die Odometrie hier aufgrund von leicht durchdrehenden Rädern – deutlich als quietschen wahrzunehmen – am fehleranfälligsten ist. In der abgebildeten Trajektorie betrifft dieser Effekt AMCL vor allem in den letzten beiden Kurven (gefahren wurde im Uhrzeigersinn, oben beginnend).

Bei den quadratischen Trajektorien zeigt sich beim vSLAM-System die höchste Genauigkeit. Signifikant schlechter liegt AMCL, gefolgt von Odometrie. Aufgrund der geringeren Genauigkeit von AMCL liegt die Genauigkeit der Posenschätzungen des Kalmanfilter leicht unter der des vSLAM-Systems. Hinzu kommt, dass AMCL in den äußerst engen Kurven teilweise einen derartigen Orientierungsfehler aufweist, dass sich der Positionsfehler in der Folge so stark erhöht, dass eine globale Relokalisierung eingeleitet wird. Schlägt diese fehl, springt die Position deutlich und die Posen sind unbrauchbar. Hier erweist sich Odometrie und ORB-SLAM3 als wesentlich robuster, wobei für ORB-SLAM3 eine hohe Rechenleistung (aufgrund der hohen Bildrate) aufzuwenden oder die Kurve langsam abzufahren ist, um die schnelle Perspektivenänderung adäquat zu tracken.

Bei den Trajektorien in Form einer Acht (siehe Abbildung 5.22) werden die Ungenauigkeiten in Kurvenfahrten besonders deutlich. Auch ORB-SLAM3 erreicht dabei nicht die gewohnte Genauigkeit. Gut zu sehen ist in Abbildung 5.22 (a), dass die Ungenauigkeit der Odometrie zum Ende hin immer weiter zunimmt, wohingegen AMCL durch die Laserscans weniger stark driftet. Auch hier erreicht der Kalmanfilter keine höhere Genauigkeit als das vSLAM-System.



(a) Positionsverlauf der Lokalisierungsmethoden über eine quadratische Trajektorie.



(b) Histogramm über die Positionsfehler aller Lokalisierungsmethoden.

Abbildung 5.21: Vergleich der Lokalisierungsmethoden anhand einer quadratischen Trajektorie.

Da die gefahrene Acht die größte Spannweite hat, wird hier das Problem der Skalenkalibrierung des vSLAM-Systems als hauptsächliche Ungenauigkeitsquelle deutlich. Je weiter sich der Roboter vom Ursprung entfernt, desto größer wird der Lokalisierungsfehler (siehe Abschnitt 5.2.4). Außerdem verschlechtert sich die Odometrie zunehmend durch die vielen Kurven, wovon auch AMCL betroffen ist. Somit beschreibt die Form einer Acht eine Trajektorie, bei der die Schwächen aller Lokalisierungsmethoden aufgezeigt werden.

Bei den kreisförmigen Trajektorien (vgl. Abbildung 5.23) zeigen sich ähnliche Ergebnisse. Da

Lokalisierungsmethode	Maximaler Positionsfehler [m]	Maximaler Orientierungsfehler [°]
vSLAM	0.8853	167.82
AMCL	0.7048	20.41
Odometrie	0.9685	28.34
Kalmanfilter	0.4966	129.79

Tabelle 5.4: Maxima der Positions- und Orientierungsfehler.

Lokalisierungsmethode	95. Perzentil Positionsfehler [m]	95. Perzentil Orientierungsfehler [°]
vSLAM	0.2037	4.27
AMCL	0.5239	8.60
Odometrie	0.7216	11.39
Kalmanfilter	0.2277	5.26

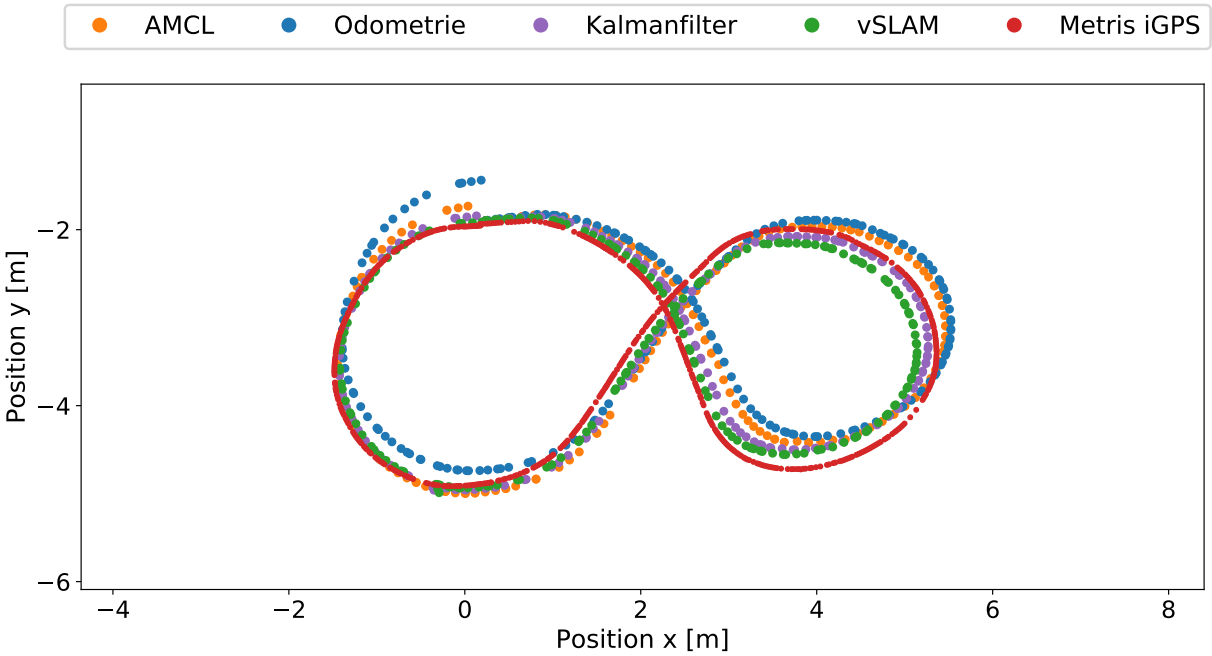
Tabelle 5.5: 95. Perzentile der Positions- und Orientierungsfehler.

der Kreis mit etwa 1-2 m einen geringen Radius hat, fallen die Abweichungen zu den Messungen des Metris iGPS hier geringer aus. Nichtsdestotrotz ist erneut deutlich zu erkennen, dass das vSLAM-System die höchste Genauigkeit aufweist.

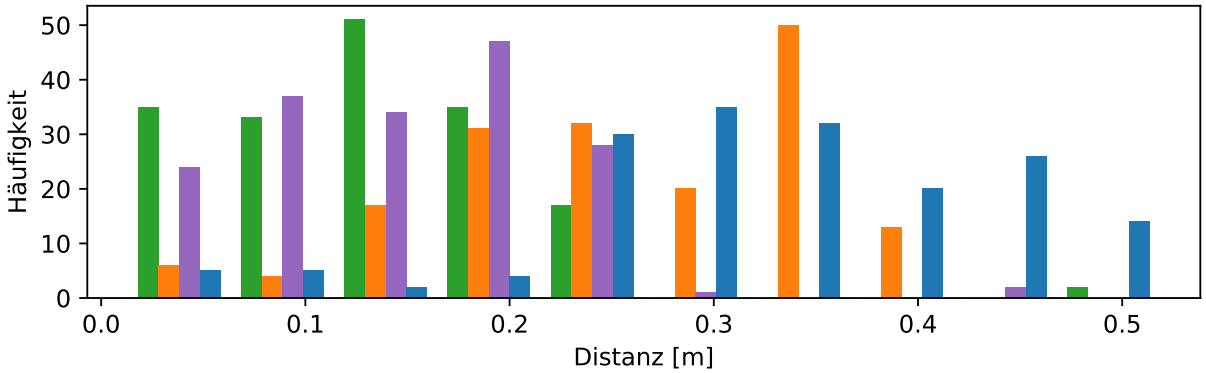
Das Gesamtergebnis ist in Abbildung 5.24 zu sehen. Am genausten ist sowohl bei der Position als auch bei der Orientierung das entwickelte vSLAM-System. Darauf folgt der Kalmanfilter, der von der Genauigkeit des vSLAM-Systems profitiert. AMCL ist zwar deutlich genauer als Odometrie, erreicht aber nicht die Genauigkeit des vSLAM-Systems. Dazu ist anzumerken, dass die Schwäche des vSLAM-Systems bezüglich des Skalendrifts besonders bei Trajektorien mit hohen Spannweiten (> 10 m) sichtbar wird. AMCL ist dafür nicht anfällig, weshalb ein Kalmanfilter bei höheren Spannweiten mehr bewirkt.

In Abbildung 5.24 werden ausschließlich die Messwerte innerhalb des 95%-Bands dargestellt, da teils deutliche Ausreißer wesentlich breitere Histogramme erfordern, in denen die Kernaussage der Abbildung schlechter deutlich wird. Die Maxima der Fehler sind in Tabelle 5.4 und die 95. Perzentile in Tabelle 5.5 aufgeführt. Bei den Maxima schneidet das vSLAM-System aufgrund eines starken Ausreißers schlecht ab. Insgesamt spiegelt dies jedoch nicht das Gesamtbild über die einzelnen Durchläufe wider, bei denen die maximalen Positions- und Orientierungsfehler bei den meisten Durchläufen geringer sind als bei den Vergleichstechnologien (siehe Anhang C).

Wichtig ist anzumerken, dass der aufgeführte Vergleich nicht für die verglichenen Technologien im Allgemeinen sprechen kann, sondern lediglich eine praktische Gegenüberstellung mit dem System des Zentrums für Telematik darstellt. Die Tendenz und das Ergebnis, dass das entwickelte vSLAM-System ausgehend von seiner Genauigkeit mit dem bestehenden Lokalisierungssystem mehr als mithalten kann, ist jedoch deutlich zu erkennen.

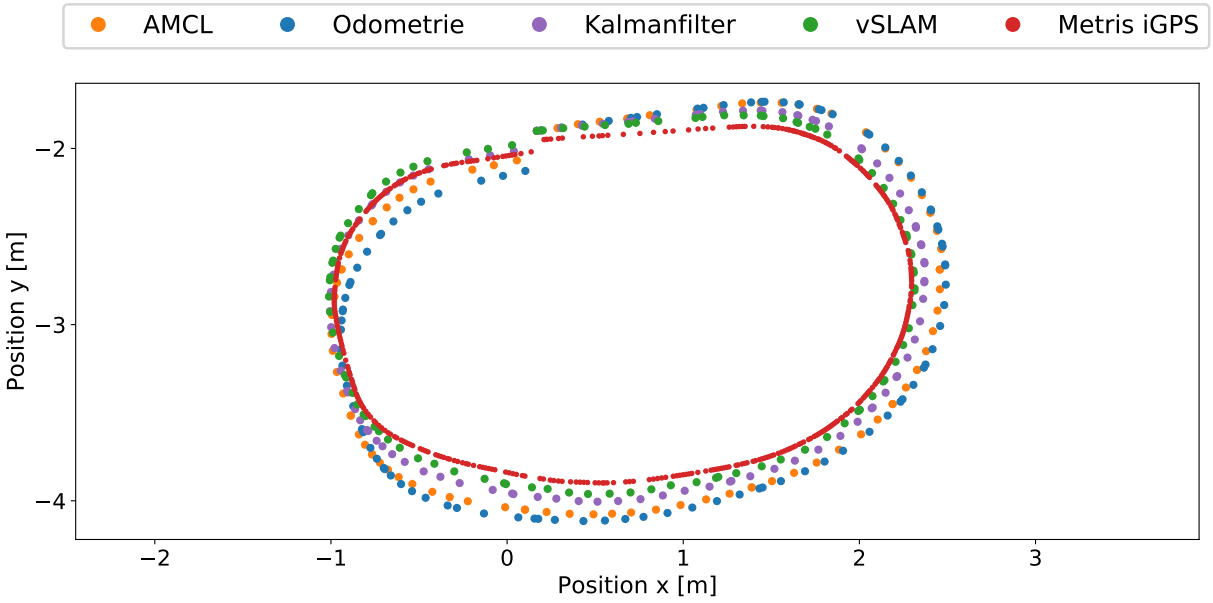


(a) Positionsverlauf der Lokalisierungsmethoden über eine Trajektorie in Form einer Acht.

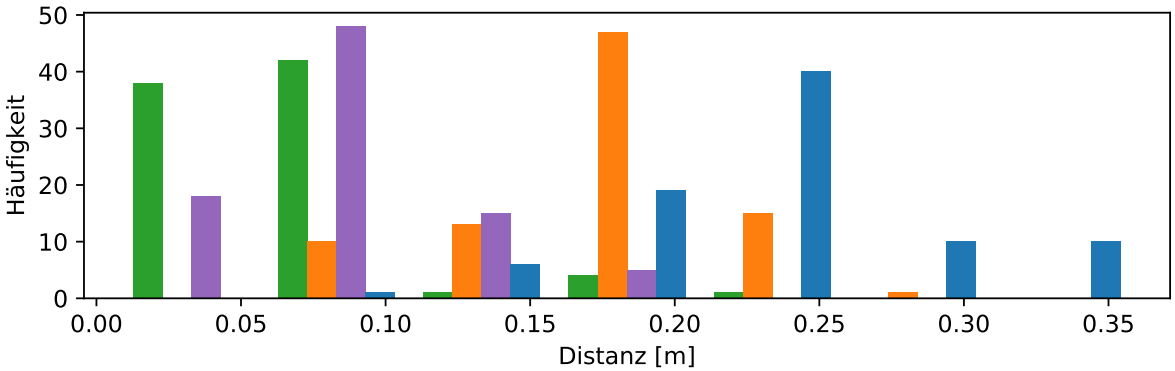


(b) Histogramm über die Positionsfehler aller Lokalisierungsmethoden.

Abbildung 5.22: Vergleich der Lokalisierungsmethoden anhand einer Trajektorie in Form einer Acht.

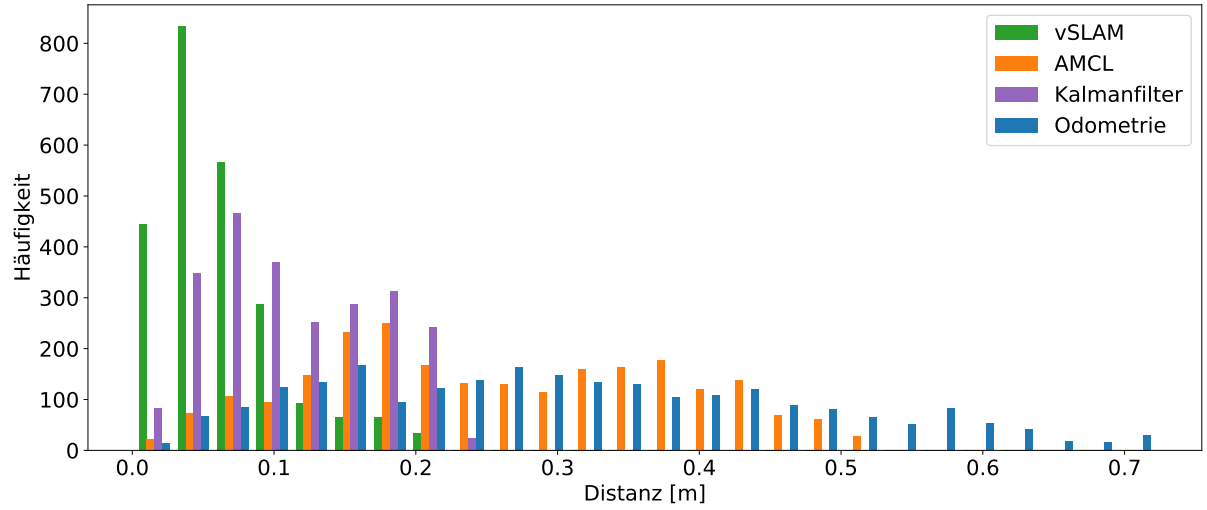


(a) Positionsverlauf der Lokalisierungsmethoden über eine kreisförmigen Trajektorie.

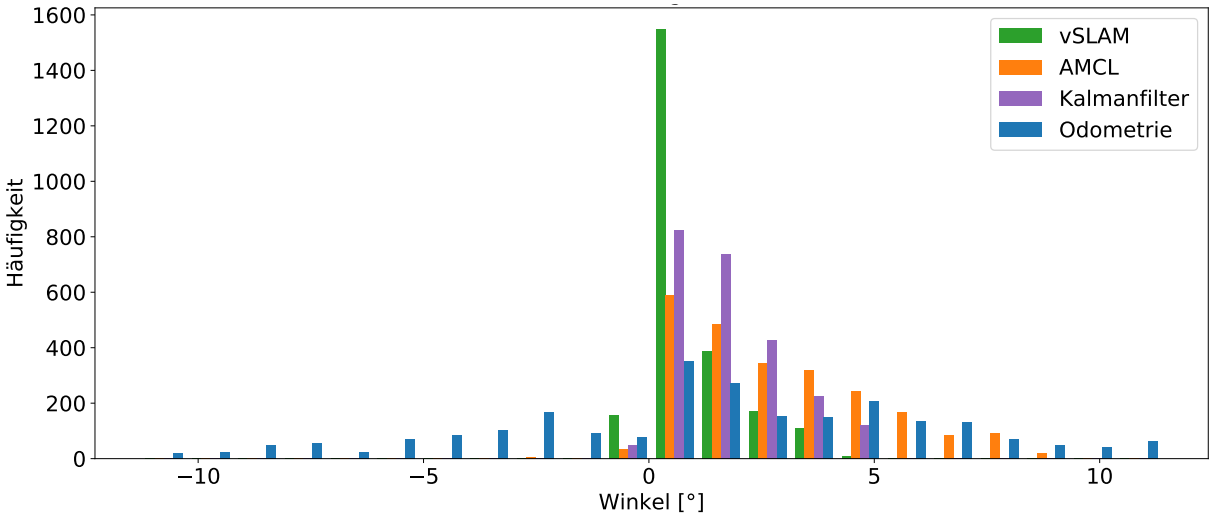


(b) Histogramm über die Positionsfehler aller Lokalisierungsmethoden.

Abbildung 5.23: Vergleich der Lokalisierungsmethoden anhand einer kreisförmigen Trajektorie.



(a) Histogramm über die Positionsfehler aller Durchläufe.



(b) Histogramm über die Orientierungsfehler aller Durchläufe.

Abbildung 5.24: Vergleich der Lokalisierungsmethoden anhand der Messungen aller Durchläufe im jeweiligen 95%-Band.

Kapitel 6

Zusammenfassende Diskussion und Ausblick

Resümee

Grundlage für einen autonomen mobilen Roboter ist ein zuverlässiges und genaues Lokalisierungssystem. Diese Arbeit vergleicht zunächst vSLAM-Algorithmen, die als Grundlage für ein umfangreiches Lokalisierungssystem dienen können. Anhand praktischer Experimente und vergangener Ergebnisse der Forschung wurde ORB-SLAM3 als optimal für diese Arbeit geeignet erachtet.

Da monokulare vSLAM-Algorithmen keine absolute Skaleninformationen gewinnen, wurde auf ORB-SLAM3 aufbauend ein Verfahren entwickelt, mit dem die Skalierung des vSLAM-Algorithmus kalibriert wird. Dies geschieht, indem ein AprilTag Pattern von mehreren Perspektiven betrachtet wird und die gemessenen Posen anhand des AprilTag Patterns mit den Posen des vSLAM-Algorithmus abgeglichen werden. Über die Genauigkeit der absoluten Skalierung gab eine Messreihe Aufschluss. Bei einer Kalibrierung mit acht Punkten wurde ein durchschnittlicher Fehler von 6,36 cm bei einem 95. Perzentil von 13,74 cm erreicht. Einen geringeren durchschnittlichen Fehler von 6,13 cm bei einem 95. Perzentil von 12,68 cm ergab die Kalibrierung mit 25 Punkten. Die Skalenkalibrierung wurde neben Fehlern bei der Auswertung und Ungenauigkeiten des vSLAM-Algorithmus als Hauptfehlerquelle identifiziert. Eine deutliche Verbesserung ist durch die Erhöhung der Kalibrierpunkte allerdings nicht zu erzielen.

Um das entwickelte System zur Pfadplanung einzusetzen, wurde ein Algorithmus entwickelt, der aus der topologischen Karte des ORB-SLAM3 Algorithmus, bestehend aus Keyframes, eine zweidimensionale Occupancy Grid Map erstellt. Die erstellte Karte erwies sich als brauchbar für Navigationsanwendungen. Zwar werden flache Hindernisse, wie auf dem Boden liegende Kabel oder Leisten, nicht erkannt, jedoch ergaben sich andere Vorteile gegenüber Laser-basierten Karten, die meist nur Hindernisse auf einer bestimmten Höhe verzeichnen. Insgesamt konnte zwar nicht die Qualität einer Laser-basierten Kartierung erreicht werden, nichtsdestotrotz erfüllt die Occupancy Grid Map ihren Zweck in einem ausreichenden Maß. In einem Experiment wurde das

Lokalisierungssystem zusammen mit der erstellten Occupancy Grid Map getestet, indem wiederholt von einem Startpunkt zu einem bestimmten Zielpunkt um ein Hindernis herum navigiert wurde. Hier lagen 90% der Endposen näher als 7,065 cm an der Zielposition.

Abschließend wurde das neu entwickelte Lokalisierungssystem mit den bisherigen Lokalisierungssystemen verglichen. Hierbei wurde deutlich, dass die Genauigkeit des vSLAM-Systems in Bereichen mit kleiner Spannweite deutlich über der Genauigkeit von AMCL und Odometrie liegt. In Bereichen, die sich über mehr als 10 m erstrecken, wird jedoch der Skalierungsfehler immer deutlicher, sodass AMCL im Vergleich genauer wird. In diesem Bereich ist ein Kalmanfilter sinnvoll. Um den Skalierungsfehler zu minimieren, ist ein aufwändigeres Kalibrierverfahren notwendig. Ein weiteres Problem ist die Anfälligkeit des vSLAM-Systems gegenüber sich ändernden Lichtverhältnissen durch Helligkeitsunterschiede und Schattierungen. In Industrieumgebungen mit fester Beleuchtung spielen diese jedoch eine untergeordnete Rolle. Das vSLAM-System zeigte sich insgesamt im Gegensatz zu AMCL als robuster, weist eine höhere Datenrate auf und benötigt keine initiale Posenschätzung, sondern relokalisiert sich stets fehlerfrei.

Das entstehende Gesamtsystem ist in der Lage eine zuverlässige und genaue Posenschätzung bei einer hohen Datenrate zu liefern. Zudem wird eine zweidimensionale Occupancy Grid Map erstellt, wodurch Lasersysteme an einen mobilen Roboter zur autonomen Navigation vollständig durch ein monokulares visuelles System ersetzt werden können. Lediglich zu Vermeidung von Kollisionen mit dynamischen Objekten ist zusätzliche Sensorik nötig. Hierfür kommen preiswerte Infrarot- oder Ultraschallabstandssensoren in Frage.

Ausblick

Das Schlüsselelement, um aus einem monokularen vSLAM-System skalentreue Posenschätzungen zu erhalten, ist das Verfahren zur Skalenkalibrierung. Ein völlig anderer Weg, um Skaleninformationen in einen vSLAM-Algorithmus aufzunehmen, ist es die Odometrie direkt in die Berechnungen des vSLAM-Algorithmus eingehen zu lassen, indem beim Tracking zum vorherigen Bild die absolute Posenänderung (gemessen durch die Odometrie) miteinbezogen wird. Ähnliche Verfahren existieren bereits mit Messungen von Inertial Measurement Units (IMUs). Dies ist in der Entwicklung und Umsetzung deutlich umfangreicher, da die Messsysteme tiefer ineinander greifen. Allerdings wird mit dieser Anpassung von vornherein eine skalentreue Lokalisierung ohne vorherige Kalibrierung unterstützt.

Möglich ist auch eine Skalenkalibrierung, die nicht nur mit visuellen Messungen arbeitet, sondern bei der Kalibrierung Odometrie oder alternative inertiale Lokalisierungsverfahren einbezieht. Da die Kalibrierpunkte somit über eine größere Spannweite aufgenommen werden können, verringert sich der Skalierungsfehler deutlich. Eine Kalibrierung mit höherer Spannweite ist jedoch auch mit dem Verfahren, das in dieser Arbeit vorgestellt wurde, möglich, indem mehrere AprilTag Patterns zur Lokalisierung verwendet werden. Bei einem Pattern liegt das Problem darin, dass bei der Lokalisierung in großem Abstand zum Pattern der Fehler zunimmt und somit nur eine begrenzte Spannweite der Kalibrierpunkte möglich ist. Werden mehrere Patterns in möglichst großem Abstand zueinander im Raum verteilt und an jedem Pattern Kalibrierpunkte aufgenommen, erhöht sich die Spannweite der Kalibrierpunkte ohne dass deren Genauigkeit sinkt.

Infolgedessen wird die bestimmte Skalierung deutlich genauer, da die Ungenauigkeit der Kalibrierpunkte weniger stark eingeht. Das Problem ist allerdings, dass die exakte relative Position der Patterns zueinander bekannt sein muss. Dies erfordert eine aufwändige und hochgenaue Messung und fest angebrachte Patterns. Eine andere Möglichkeit ist es, deutlich größere AprilTag Patterns zu verwenden, die eine genaue Lokalisierungen über eine größere Entfernung ermöglichen.

Ein bestehendes Problem ist die zuverlässige Erkennung von dynamischen Objekten durch das visuelle System. Im Feature-basierten SLAM-Verfahren sind die auf den dynamischen Objekten erkannten Features nur schwer von Ausreißern zu unterscheiden und können kein Gesamtbild des Objekts liefern. Monokulare Structure from Motion Verfahren können dieses Problem ebenfalls nur schwierig lösen, wenn sich die beobachteten Objekte bewegen. Einfacher zu lösen ist das Problem bei der Verwendung von Stereo- und Tiefenkameras, da diese Systeme Tiefeninformationen aus einzelnen Aufnahmen gewinnen. Zu untersuchen wäre, ob direkte visuelle Tracking Algorithmen, wie in LSD-SLAM, dynamische Objekte in ausreichender Genauigkeit wahrnehmen können.

Ein weiterer interessanter Punkt ist die Erstellung einer Elevation Grid Map, bei der in jeder Zelle Höhe und Steigung angegeben ist. Derartige Karten sind besonders für den Außenbereich und zur Abbildung komplexer Strukturen erforderlich. Industrieumgebungen besitzen dagegen meist eine planare Fahrtebene. Jedoch können auch hier mehrere Fahrtebenen und entsprechende Rampen existieren, die in einer Elevation Grid Map abgebildet werden können.

Das entwickelte Lokalisierungssystem kann auch für die Bewegungsschätzung des AMCL Algorithmus verwendet werden. Bisher bezieht sich AMCL hier auf die Odometrie, wodurch die Fehler der Odometrie Drifts in AMCL hervorrufen. Da das vSLAM-System eine deutlich höhere Genauigkeit als Odometrie aufweist, wird AMCL hierdurch verbessert. Ein solcher Ansatz ist wesentlich sinnvoller als die Kombination der beiden Messverfahren durch einen Kalmanfilter.

Ein anderes Anwendungsgebiet ist die 3D Lokalisierung von Unmanned Aerial Vehicles (UAV). Hier ist zu überprüfen, ob der vSLAM-Algorithmus robust genug gegenüber schnellen Drehungen, mit denen zügige Perspektivenänderungen einhergehen, und Vibrationen, die Unschärfe in den Bildern verursachen, ist. Zudem ist ein alternatives System zur geometrischen Kartierung erforderlich, das Hindernisse nicht nur in der Fahrtebene sondern als 3D-Objekte erkennt.

In erster Linie ist das entwickelte System jedoch für autonome mobile Roboter in Industrieumgebungen prädestiniert. Da in dieser Arbeit festgestellt wurde, dass die Lokalisierung besonders in Bereichen mit einer Spannweite von weniger als 10 m eine hohe Genauigkeit aufweist, sind Kombinationen mit ungenaueren Lokalisierungssystemen vorteilhaft, bei denen der Roboter seine Pose größtenteils durch diese bestimmt und nur in besonderen Bereichen, in denen eine genauere Lokalisierung notwendig ist, auf das vSLAM-System zurückgreift. Derartige Bereiche sind beispielsweise Umgebungen einer Maschine oder Lagerslots, die der Roboter genau anfahren muss, oder Bereiche, wie enge Durchfahrten, in denen aufgrund der Umgebung die Navigation erschwert ist.

Anhänge

Anhang A

Messreihe zur Auswertung der absoluten Lokalisierung

Die folgenden Tabellen A.1 und A.2 geben einen Gesamtüberblick über die einzelnen Durchläufe der Messreihe aus Kapitel 5.2. Die Tabellen enthalten den durchschnittlichen und maximalen Fehler, sowie das 95. Perzentil der Position und Orientierung.

Es ist zu erkennen, dass die Genauigkeit hauptsächlich von der Skalenrekonstruktion abhängt und bei den verschiedenen Karten jeweils ähnlich ist.

Durchlauf	ø Positionsfehler [m]	worst case Positionsfehler [m]	95. Perzentil [m]	ø Orientierungsfehler[°]	worst case Orientierungsfehler [°]	95. Perzentil [°]
Karte 1 Runde 1	0.06661	0.10595	0.09698	2.692	11.849	7.861
Karte 1 Runde 2	0.07365	0.22612	0.15545	3.262	8.738	6.607
Karte 1 Runde 3	0.08525	0.16188	0.13962	3.640	8.357	6.672
Karte 1 Runde 4	0.06179	0.18209	0.09269	3.797	8.690	6.089
Karte 1 Runde 5	0.08714	0.23233	0.20569	3.952	7.536	7.072
Karte 2 Runde 1	0.08002	0.25472	0.20238	3.522	12.115	8.311
Karte 2 Runde 2	0.06776	0.20956	0.11326	1.647	6.930	3.387
Karte 2 Runde 3	0.07934	0.19975	0.08603	3.180	3.373	3.262
Karte 2 Runde 4	0.03892	0.07770	0.06958	2.092	2.515	2.466
Karte 2 Runde 5	0.07446	0.12980	0.11613	1.937	7.184	4.614
Karte 3 Runde 1	0.06647	0.29926	0.12620	3.957	16.447	7.818
Karte 3 Runde 2	0.07541	0.25889	0.19830	2.128	33.274	4.651
Karte 3 Runde 3	0.08271	0.19237	0.17275	1.743	7.515	4.307
Karte 3 Runde 4	0.08010	0.17104	0.12938	2.964	9.572	5.586
Karte 3 Runde 5	0.09350	0.24419	0.14254	2.217	9.119	5.407
Karte 4 Runde 2	0.03351	0.08867	0.06455	1.007	4.067	3.261
Karte 4 Runde 3	0.06714	0.17687	0.12930	3.456	6.635	5.752
Karte 4 Runde 4	0.07705	0.30589	0.18944	2.501	8.910	6.118
Karte 4 Runde 5	0.02948	0.08301	0.05824	4.011	9.512	6.720
Karte 5 Runde 1	0.04489	0.10258	0.07814	1.996	7.778	5.330
Karte 5 Runde 2	0.05852	0.15021	0.11615	2.735	9.225	5.785
Karte 5 Runde 3	0.04109	0.09440	0.08051	3.958	8.306	6.183
Karte 5 Runde 4	0.07855	0.19871	0.14007	2.842	13.108	7.736
Karte 5 Runde 5	0.07797	0.19342	0.14628	3.000	8.620	6.237
Karte 6 Runde 1	0.07246	0.24877	0.15452	2.566	8.201	5.256
Karte 6 Runde 2	0.10246	0.36168	0.17252	4.916	15.259	11.543
Karte 6 Runde 5	0.06598	0.23996	0.18664	3.900	9.301	6.009
Karte 7 Runde 1	0.02359	0.07664	0.04462	3.246	7.311	6.219
Karte 7 Runde 2	0.05967	0.12284	0.08960	2.247	8.486	5.147
Karte 7 Runde 3	0.04403	0.13937	0.10586	3.580	9.808	6.474
Karte 7 Runde 4	0.05062	0.13249	0.08204	3.575	10.734	6.777
Karte 7 Runde 5	0.04547	0.08190	0.06168	4.086	7.776	6.681
Karte 8 Runde 1	0.04112	0.09127	0.08119	4.104	10.251	8.686
Karte 8 Runde 3	0.07748	0.19913	0.16046	6.406	35.617	23.707
Karte 8 Runde 4	0.05059	0.17128	0.09685	5.849	21.002	13.560
Karte 8 Runde 5	0.03257	0.11167	0.05702	2.246	6.960	4.538

Tabelle A.1: Übersicht über die Durchläufe der Messreihe mit acht Punkten zur Skalenkalibrierung.

Durchlauf	ø Positionsfehler [m]	worst case Positionsfehler [m]	95. Perzentil [m]	ø Orientierungsfehler[°]	worst case Orientierungsfehler [°]	95. Perzentil [°]
Karte 1 Runde 1	0.05498	0.12359	0.09952	3.528	6.074	5.441
Karte 1 Runde 2	0.03863	0.09482	0.07806	3.013	7.020	4.612
Karte 1 Runde 3	0.08393	0.22463	0.17845	1.916	7.744	4.574
Karte 1 Runde 4	0.08120	0.18922	0.13601	1.307	7.553	3.867
Karte 1 Runde 5	0.07803	0.13141	0.12205	2.267	7.859	5.275
Karte 2 Runde 2	0.04004	0.09915	0.06719	3.919	8.589	7.386
Karte 2 Runde 3	0.07675	0.14349	0.13203	3.893	10.151	6.658
Karte 2 Runde 4	0.09620	0.14660	0.13791	3.181	10.185	7.007
Karte 2 Runde 5	0.05869	0.15658	0.10428	4.336	11.332	8.024
Karte 3 Runde 1	0.05087	0.14860	0.12550	3.667	9.152	5.675
Karte 3 Runde 2	0.04911	0.19587	0.08874	3.805	9.865	7.172
Karte 3 Runde 3	0.06987	0.21793	0.10974	2.057	6.810	5.738
Karte 3 Runde 4	0.04306	0.16877	0.08743	3.842	9.986	6.554
Karte 3 Runde 5	0.03719	0.16737	0.10255	3.222	9.163	5.674
Karte 4 Runde 1	0.05918	0.10591	0.08035	3.461	6.942	6.184
Karte 4 Runde 2	0.08222	0.14447	0.13269	4.242	12.966	9.294
Karte 4 Runde 3	0.08635	0.14853	0.12172	3.281	9.591	5.504
Karte 4 Runde 4	0.05410	0.16854	0.11041	3.725	11.421	7.448
Karte 4 Runde 5	0.05935	0.15043	0.10586	2.718	8.724	6.098
Karte 5 Runde 1	0.07368	0.17393	0.13074	3.254	9.380	5.416
Karte 5 Runde 4	0.05561	0.13181	0.09169	3.605	10.268	7.171
Karte 5 Runde 5	0.04449	0.09199	0.08415	3.021	7.500	6.465
Karte 6 Runde 3	0.09567	0.20125	0.16657	2.218	8.407	5.023
Karte 6 Runde 5	0.04540	0.16367	0.08791	2.935	4.873	4.385
Karte 7 Runde 3	0.03488	0.06415	0.05197	1.086	7.066	3.478
Karte 7 Runde 4	0.03671	0.09266	0.06722	2.327	7.663	5.107
Karte 7 Runde 5	0.05125	0.10529	0.07535	3.038	9.002	5.868
Karte 8 Runde 1	0.06311	0.23292	0.18047	4.710	16.776	14.135
Karte 8 Runde 2	0.10524	0.19937	0.16849	1.849	22.491	7.448
Karte 8 Runde 4	0.04113	0.07659	0.07139	1.196	5.622	3.151

Tabelle A.2: Übersicht über die Durchläufe der Messreihe mit 25 Punkten zur Skalenkalibrierung.

Anhang B

Messreihe zur Auswertung der freien Navigation

Eine Übersicht über die Messreihe zur Navigation aus Kapitel 5.3 ist in Tabelle B.1 dargestellt. Die Tabelle enthält die x- und y-Koordinate, sowie den Orientierungswinkel θ der Endpunkte jedes Durchlaufs, gemessen durch das SLAM-System und Metris iGPS.

Durchlauf	X Metris [m]	Y Metris [m]	θ Metris [°]	X SLAM [m]	Y SLAM [m]	θ SLAM [°]
Runde 1	10.55558	-0.01185	84.777	10.53920	-0.023213	84.988
Runde 2	10.50219	0.03647	92.195	10.50780	-0.039286	93.546
Runde 3	10.52346	0.03462	87.153	10.53710	0.032928	86.737
Runde 4	10.51663	-0.03407	85.158	10.52730	0.003115	85.566
Runde 5	10.46875	0.06243	89.721	10.47400	0.018037	90.184
Runde 6	10.58703	0.00669	86.729	10.57210	-0.003652	86.880
Runde 7	10.49714	-0.11718	83.213	10.51780	-0.040512	85.227
Runde 9	10.54717	-0.02413	84.853	10.54320	-0.00403	85.715
Runde 10	10.56127	0.03348	85.285	10.55810	-0.015681	86.028
Runde 11	10.47905	0.04755	86.529	10.53620	0.008726	86.423
Runde 13	10.54423	-0.01239	86.698	10.51340	-0.024366	87.753
Runde 14	10.59595	-0.06738	79.464	10.51970	-0.034147	84.698
Runde 15	10.47957	0.01329	86.796	10.48360	-0.008885	87.466
Runde 16	10.46610	-0.02911	89.135	10.46970	0.030518	90.208
Runde 17	10.45902	-0.02145	88.395	10.48430	-0.035709	88.913
Runde 19	10.45564	-0.07113	90.338	10.49140	0.005095	90.546
Runde 20	10.51905	-0.09198	82.687	10.52340	-0.038812	83.862
Runde 21	10.50343	-0.09689	89.694	10.46710	-0.024791	91.472
Runde 22	10.38167	0.08835	87.778	10.43530	0.082722	87.687
Runde 23	10.56968	0.03713	86.559	10.47910	-0.025813	87.138
Runde 24	10.57443	0.01081	89.493	10.56200	0.00187	90.409
Runde 25	10.50878	0.01431	85.142	10.53500	0.016675	85.464
Runde 26	10.59015	0.00786	88.171	10.57470	-0.034494	88.565
Runde 27	10.53786	-0.05523	84.397	10.52940	-0.014013	85.565
Runde 28	10.52457	0.02910	87.522	10.53410	-0.013588	87.699
Runde 29	10.53589	0.02497	84.032	10.55050	0.00702	84.508
Runde 31	10.50914	-0.05831	83.872	10.52580	-0.029803	84.769
Runde 32	10.48244	-0.02188	86.068	10.49540	-0.053138	86.410
Runde 33	10.54618	-0.03675	83.365	10.56970	0.008882	83.816
Runde 35	10.51392	0.00453	84.630	10.52980	-0.022744	85.440
Runde 36	10.42726	0.03148	88.811	10.45750	0.003117	89.368
Runde 38	10.52612	0.00220	86.879	10.53210	0.010822	87.164
Runde 39	10.53051	-0.01903	85.463	10.54380	-0.053856	85.633
Runde 40	10.52788	0.12605	92.028	10.48170	0.053553	93.305
Runde 41	10.42498	0.06772	85.955	10.45700	0.039397	86.744
Runde 42	10.46272	-0.09955	102.903	10.47280	-0.11301	100.689
Runde 43	10.48091	0.01495	87.903	10.49010	0.012681	88.042
Runde 44	10.44987	-0.01132	86.881	10.46880	0.01467	85.854
Runde 45	10.49235	0.06232	87.212	10.51050	0.033726	87.776

Tabelle B.1: Übersicht über die Durchläufe der Messreihe mit den korrigierten Metris iGPS Posen gegenüber den SLAM-Posen.

Anhang C

Messreihe zum Vergleich verschiedener Lokalisierungsmethoden

Im Folgenden sind die Experimente zum Vergleich der Lokalisierungsverfahren AMCL und Odometrie zum entwickelten visual SLAM-System, einem Kalmanfilter und Metris iGPS dargestellt. Zu jedem Durchlauf werden der Positionsverlauf, das Fehlerhistogramm gegenüber Metris iGPS und die Verläufe der Koordinaten und des Orientierungswinkel abgebildet. Die Tabellen C.1 - C.4 führen die Genauigkeiten der einzelnen Lokalisierungsverfahren gegenüber Metris iGPS über alle Durchläufe auf.

Durchlauf	ø Positions-fehler [m]	Maximaler Positions-fehler [m]	95. Perzentil Positions-fehler [m]	ø Orientierungs-fehler [°]	Maximaler Orientierungs-fehler [°]	95. Perzentil Orientierungs-fehler [°]
Quadrat 1	0.0456	0.1138	0.0936	0.5135	5.0164	2.2854
Quadrat 2	0.0442	0.1331	0.0880	0.7826	3.8638	2.4784
Quadrat 3	0.0432	0.1682	0.0846	0.7759	5.0518	2.8675
Quadrat 4	0.0566	0.1251	0.1057	1.2769	6.0761	5.0782
Quadrat 5	0.0549	0.1314	0.0949	1.0744	5.4184	4.4684
Quadrat 6	0.0501	0.1171	0.1012	2.2075	168.8320	5.0700
Kreis 1	0.0648	0.1176	0.1110	1.0855	1.7143	1.5428
Kreis 2	0.0337	0.1361	0.0581	0.5439	2.5900	1.2102
Kreis 3	0.0393	0.0759	0.0686	1.0355	2.4003	1.9168
Kreis 4	0.0641	0.2125	0.1718	1.9667	5.7054	4.6571
Acht 1	0.1343	0.4852	0.2280	0.7416	4.3904	1.5098
Acht 2	0.1041	0.2595	0.2383	1.5881	4.8481	4.4615
Acht 3	0.1083	0.2262	0.2159	1.5767	4.1676	3.7473
Acht 4	0.1120	0.8854	0.2307	2.9829	176.8208	5.6137
Acht 5	0.1075	0.2371	0.2247	0.4002	5.1277	1.2666

Tabelle C.1: Auswertung des visual SLAM-Systems über alle Durchläufe.

Durchlauf	ø Positions-fehler [m]	Maximaler Positions-fehler [m]	95. Perzentil Positions-fehler [m]	ø Orientierungs-fehler [°]	Maximaler Orientierungs-fehler [°]	95. Perzentil Orientierungs-fehler [°]
Quadrat 1	0.1579	0.4998	0.4158	6.1927	20.4089	12.3756
Quadrat 2	0.2488	0.4524	0.4273	3.8235	12.6571	8.2528
Quadrat 3	0.3094	0.5442	0.4609	3.5078	14.8678	9.8541
Quadrat 4	0.2163	0.4043	0.3799	3.2320	14.2152	8.1603
Quadrat 5	0.3993	0.7048	0.5846	3.0834	11.9097	7.3420
Quadrat 6	0.1660	0.3115	0.2653	2.9800	16.8754	9.0674
Kreis 1	0.3301	0.5571	0.5464	2.5415	5.8845	5.7345
Kreis 2	0.3236	0.4817	0.4631	2.4751	9.1178	6.7649
Kreis 3	0.3477	0.6492	0.6167	1.8698	4.7826	4.3489
Kreis 4	0.1765	0.2593	0.2514	3.0663	9.5862	8.3118
Acht 1	0.2601	0.3868	0.3690	1.9758	5.6136	4.8772
Acht 2	0.2771	0.4939	0.4649	1.7866	7.8307	5.0314
Acht 3	0.2848	0.6649	0.5399	3.4065	10.2391	8.7664
Acht 4	0.3452	0.6755	0.5332	4.0028	9.0242	8.1128
Acht 5	0.2506	0.4073	0.3896	2.4631	8.2139	6.0919

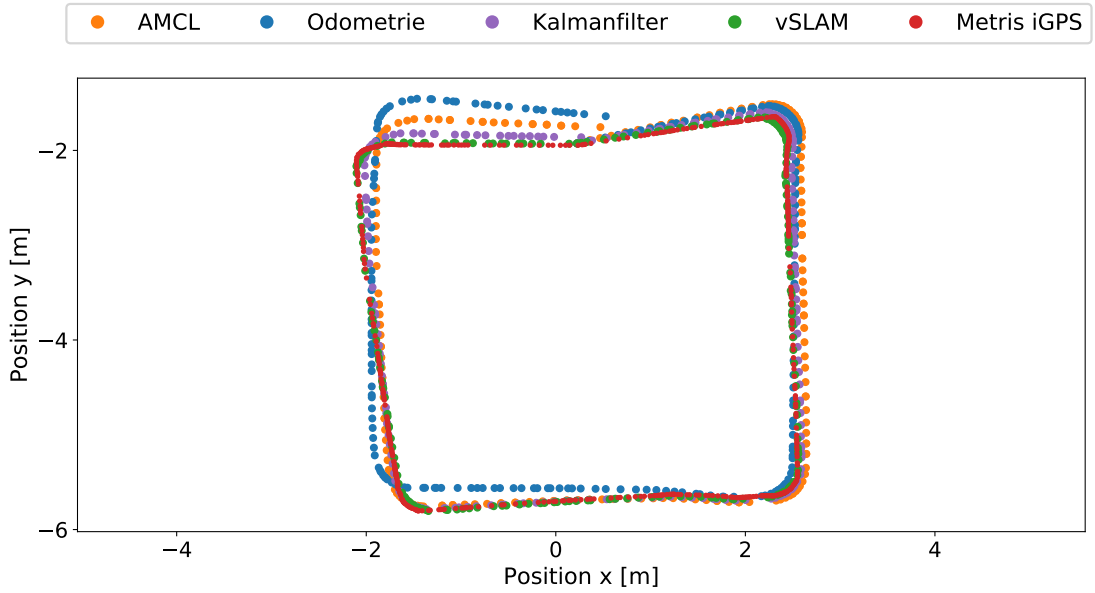
Tabelle C.2: Auswertung von AMCL über alle Durchläufe.

Durchlauf	ø Positions-fehler [m]	Maximaler Positions-fehler [m]	95. Perzentil Positions-fehler [m]	ø Orientierungs-fehler [°]	Maximaler Orientierungs-fehler [°]	95. Perzentil Orientierungs-fehler [°]
Quadrat 1	0.2566	0.6504	0.5869	8.4441	28.3443	17.0788
Quadrat 2	0.3027	0.6204	0.4920	4.6850	14.4570	10.6102
Quadrat 3	0.3059	0.6526	0.5876	4.7890	16.9820	13.6085
Quadrat 4	0.1567	0.3099	0.2987	4.6657	20.6016	12.3068
Quadrat 5	0.4588	0.9685	0.7833	4.5850	18.7917	12.3572
Quadrat 6	0.1456	0.4806	0.3735	5.2580	22.6083	14.5584
Kreis 1	0.4820	0.8203	0.8086	5.1254	11.0812	10.7489
Kreis 2	0.5068	0.8130	0.7887	5.6449	15.1065	12.0630
Kreis 3	0.4727	0.9144	0.8861	3.7323	9.0054	8.3718
Kreis 4	0.2292	0.3593	0.3430	2.7976	8.1597	7.3290
Acht 1	0.3252	0.5185	0.4782	2.3554	5.5023	4.9315
Acht 2	0.2765	0.4529	0.4307	3.0329	9.6846	7.6480
Acht 3	0.3520	0.6091	0.5230	3.2795	9.8224	7.5860
Acht 4	0.4714	0.9553	0.8328	5.0014	11.6007	10.5451
Acht 5	0.3521	0.5626	0.5276	2.4454	6.4819	5.2249

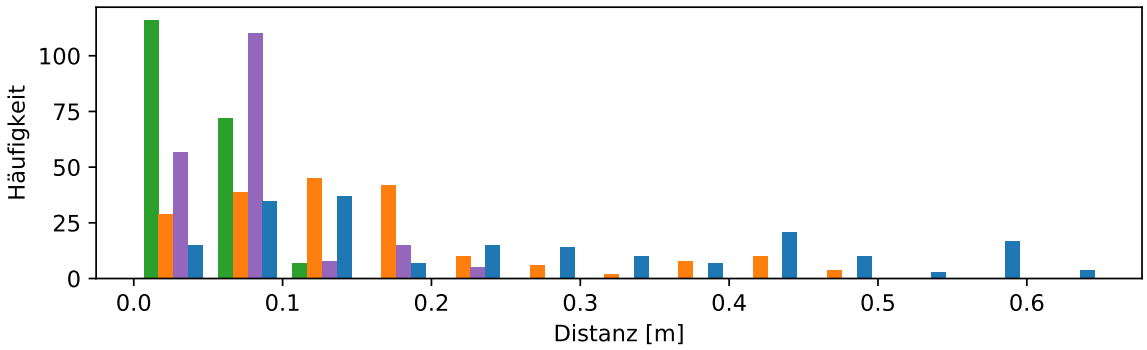
Tabelle C.3: Auswertung der Odometrie über alle Durchläufe.

Durchlauf	ø Positions-fehler [m]	Maximaler Positions-fehler [m]	95. Perzentil Positions-fehler [m]	ø Orientierungs-fehler [°]	Maximaler Orientierungs-fehler [°]	95. Perzentil Orientierungs-fehler [°]
Quadrat 1	0.0724	0.2145	0.1931	3.2720	56.2084	6.2116
Quadrat 2	0.1029	0.2001	0.1703	2.2034	56.4967	4.6216
Quadrat 3	0.1262	0.2333	0.2079	1.8548	13.9910	5.8004
Quadrat 4	0.0891	0.2018	0.1577	2.1888	56.1900	6.6231
Quadrat 5	0.1608	0.3332	0.2679	2.0287	54.5243	5.1687
Quadrat 6	0.0681	0.1556	0.1241	3.0573	72.4510	7.7170
Kreis 1	0.1446	0.2550	0.2495	2.7388	50.6471	3.9024
Kreis 2	0.1316	0.2165	0.2059	2.1231	51.4321	4.4927
Kreis 3	0.1286	0.2681	0.2468	1.8704	51.2442	3.1765
Kreis 4	0.0858	0.1999	0.1693	2.5823	53.1781	5.7007
Acht 1	0.1498	0.4234	0.2410	2.3391	51.3083	3.7471
Acht 2	0.1416	0.2530	0.2408	2.0427	55.0167	4.2894
Acht 3	0.1426	0.3067	0.2424	2.6002	55.5754	4.6650
Acht 4	0.1558	0.4966	0.2255	3.4071	129.7929	4.9014
Acht 5	0.1322	0.2634	0.2203	2.2762	50.7799	4.1538

Tabelle C.4: Auswertung des Kalmanfilters über alle Durchläufe.

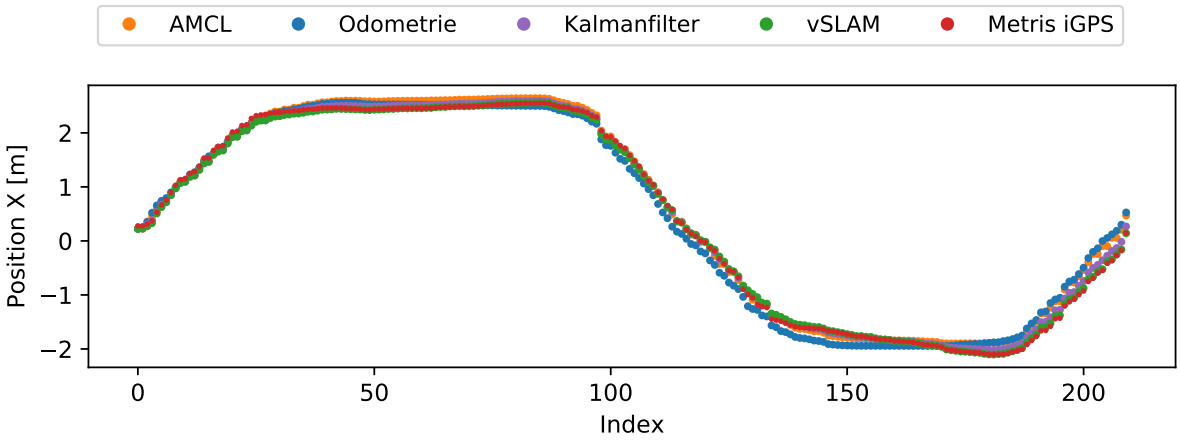


(a) Positionsverlauf der Lokalisierungsmethoden über eine quadratische Trajektorie.

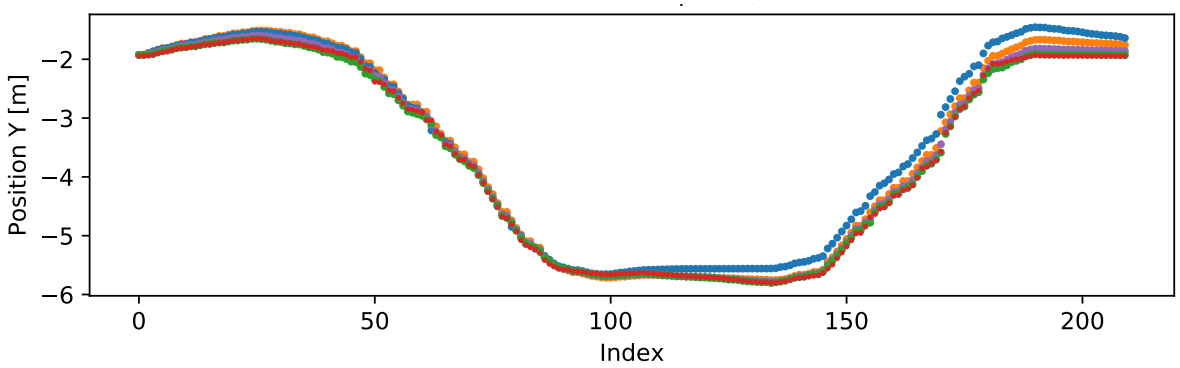


(b) Histogramm über die Positionsfehler aller Lokalisierungsmethoden.

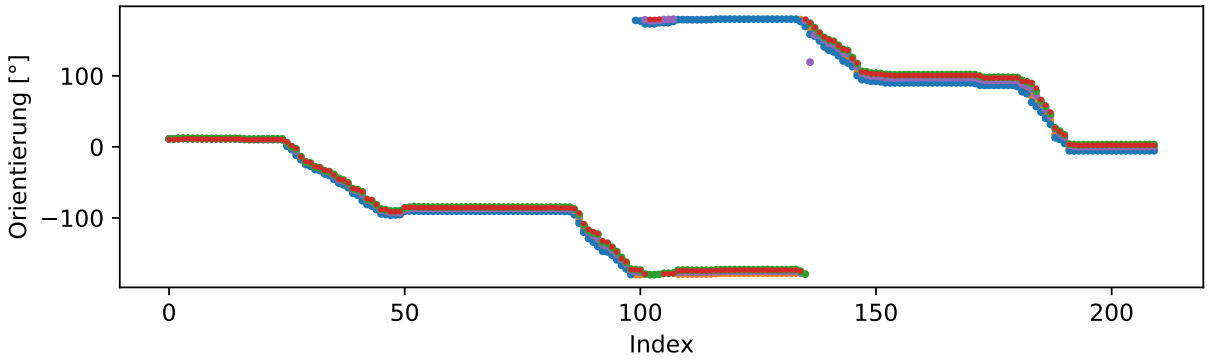
Abbildung C.1: Durchlauf 1 mit quadratischer Trajektorie: Fehlerhistogramm und Positionsverlauf.



(a) Verlauf der X-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

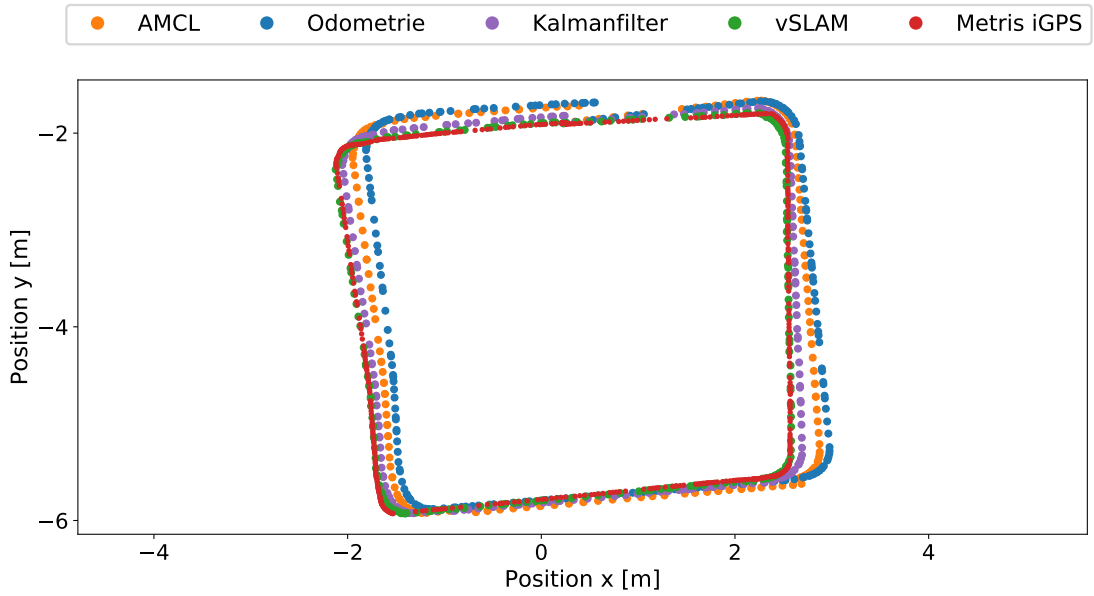


(b) Verlauf der Y-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

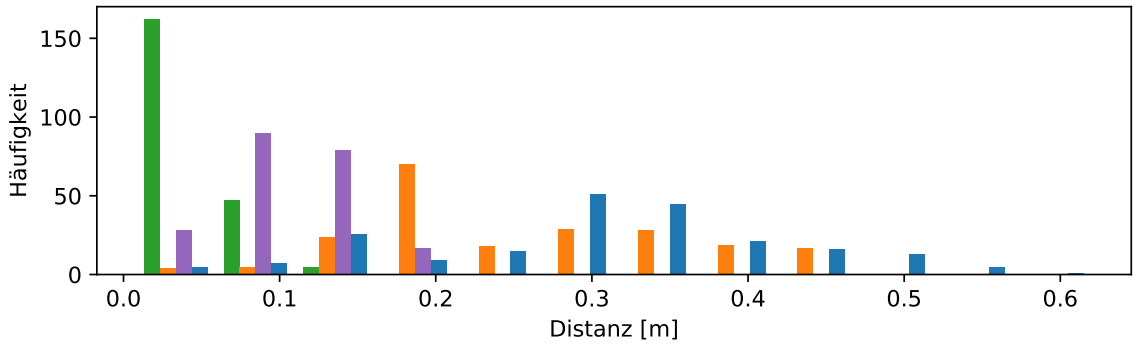


(c) Verlauf der Orientierung der Posenmessungen aller Lokalisierungsmethoden.

Abbildung C.2: Durchlauf 1 mit quadratischer Trajektorie: Verlauf der Koordinaten und des Orientierungswinkels.

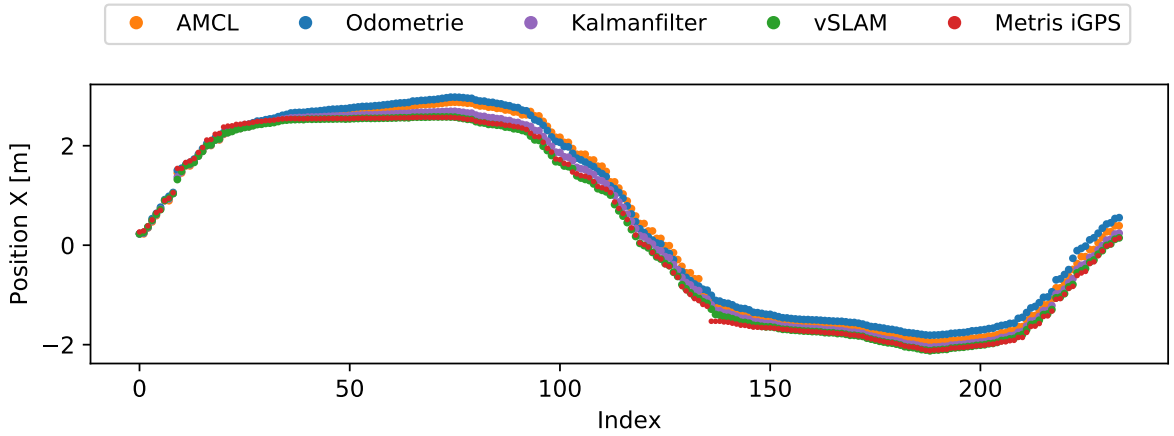


(a) Positionsverlauf der Lokalisierungsmethoden über eine quadratische Trajektorie.

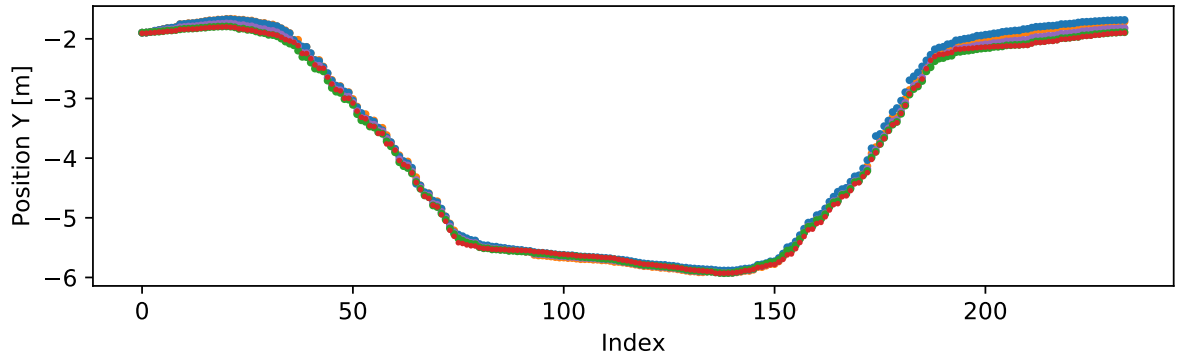


(b) Histogramm über die Positionsfehler aller Lokalisierungsmethoden.

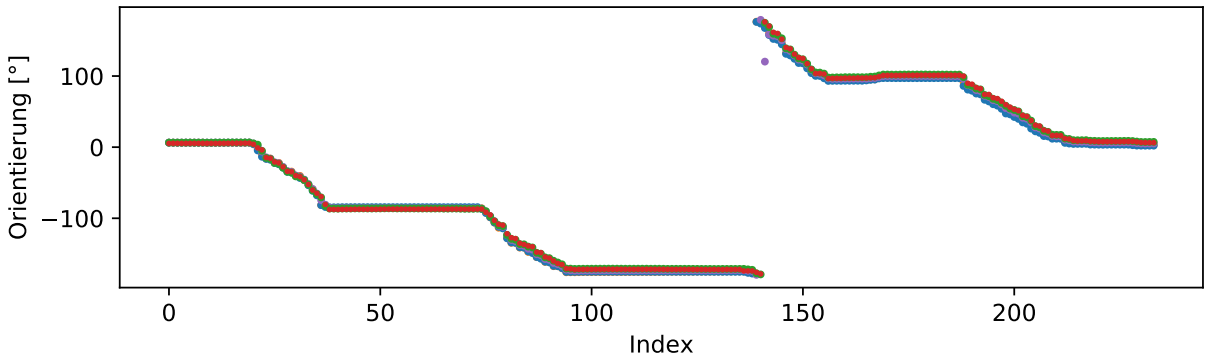
Abbildung C.3: Durchlauf 2 mit quadratischer Trajektorie: Fehlerhistogramm und Positionsverlauf.



(a) Verlauf der X-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

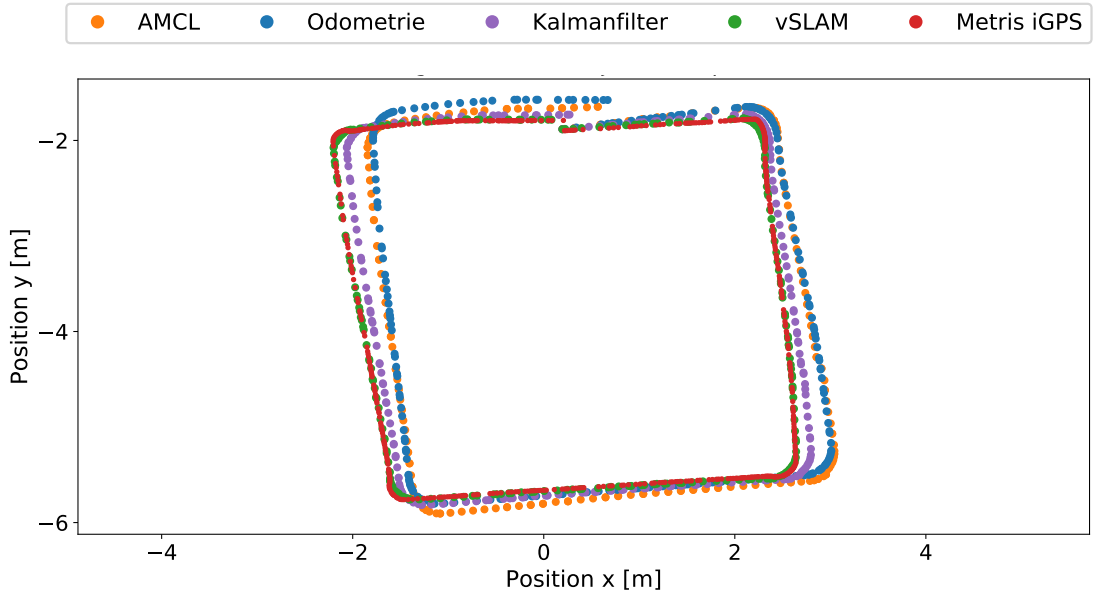


(b) Verlauf der Y-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

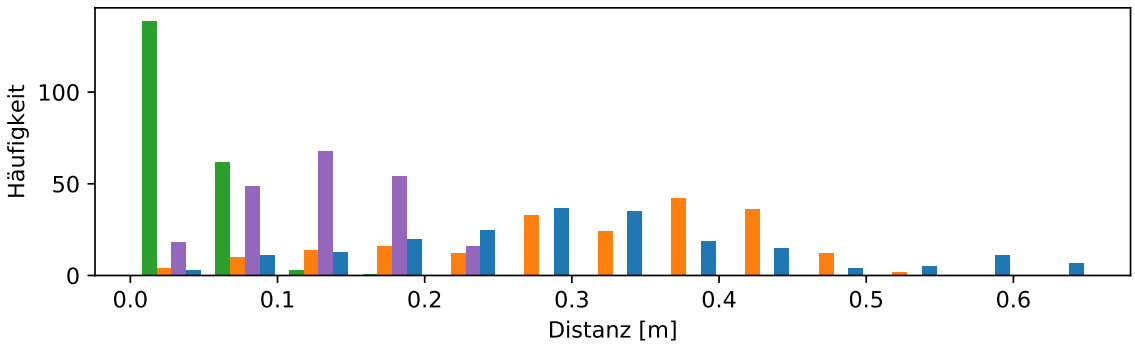


(c) Verlauf der Orientierung der Posenmessungen aller Lokalisierungsmethoden.

Abbildung C.4: Durchlauf 2 mit quadratischer Trajektorie: Verlauf der Koordinaten und des Orientierungswinkels.

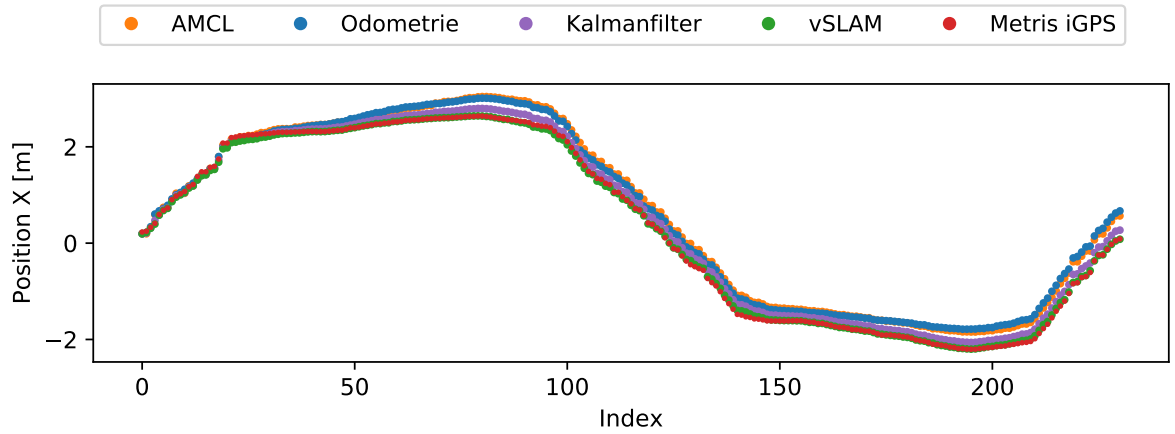


(a) Positionsverlauf der Lokalisierungsmethoden über eine quadratische Trajektorie.

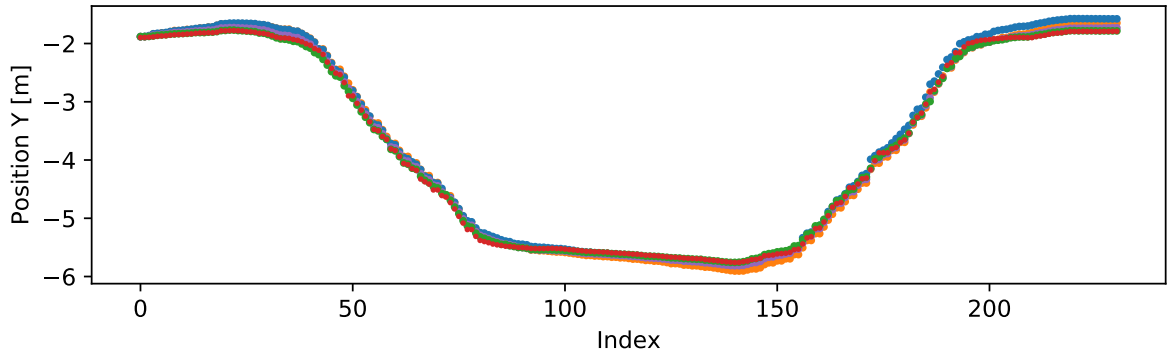


(b) Histogramm über die Positionsfehler aller Lokalisierungsmethoden.

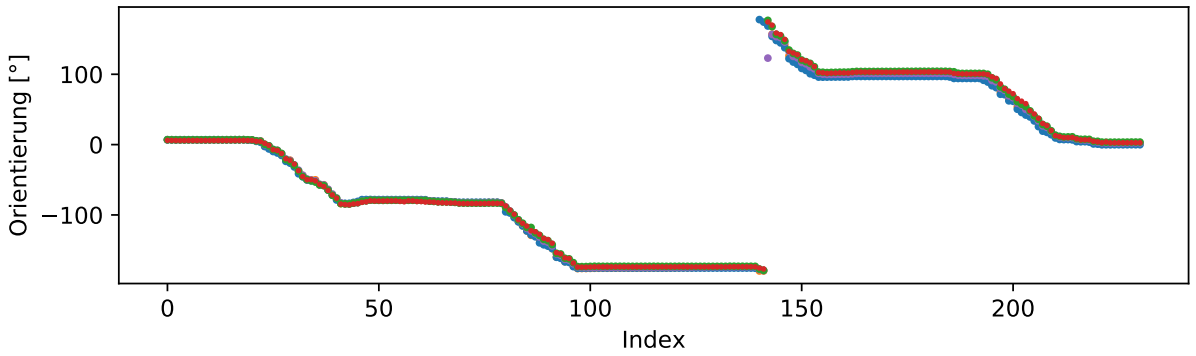
Abbildung C.5: Durchlauf 3 mit quadratischer Trajektorie: Fehlerhistogramm und Positionsverlauf.



(a) Verlauf der X-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

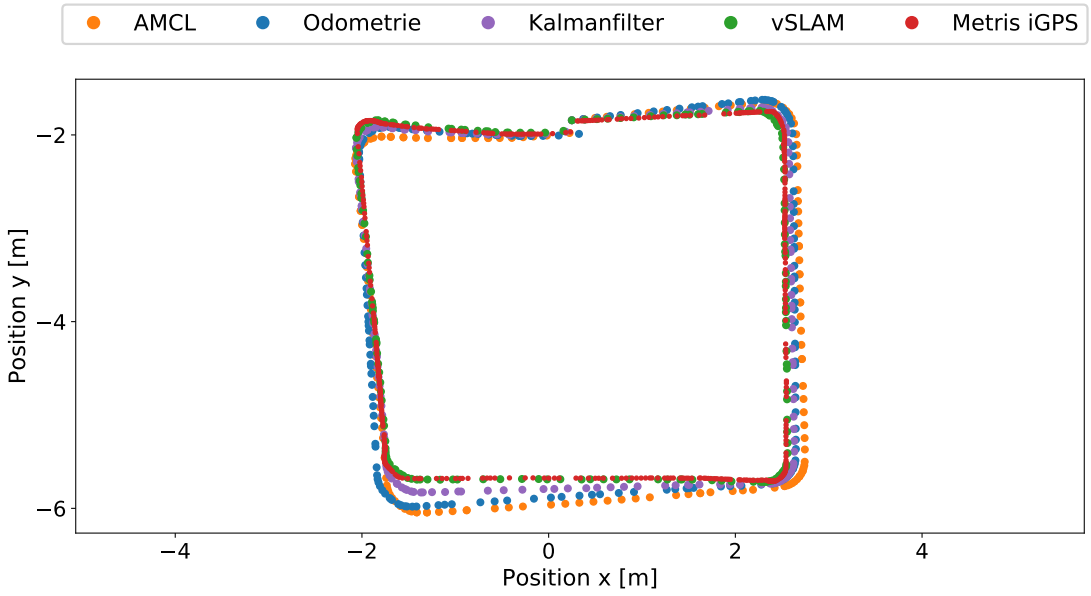


(b) Verlauf der Y-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

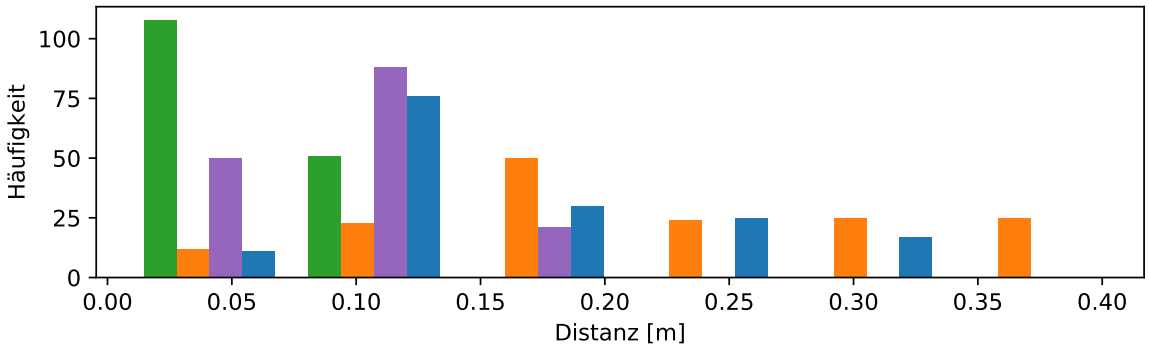


(c) Verlauf der Orientierung der Posenmessungen aller Lokalisierungsmethoden.

Abbildung C.6: Durchlauf 3 mit quadratischer Trajektorie: Verlauf der Koordinaten und des Orientierungswinkels.

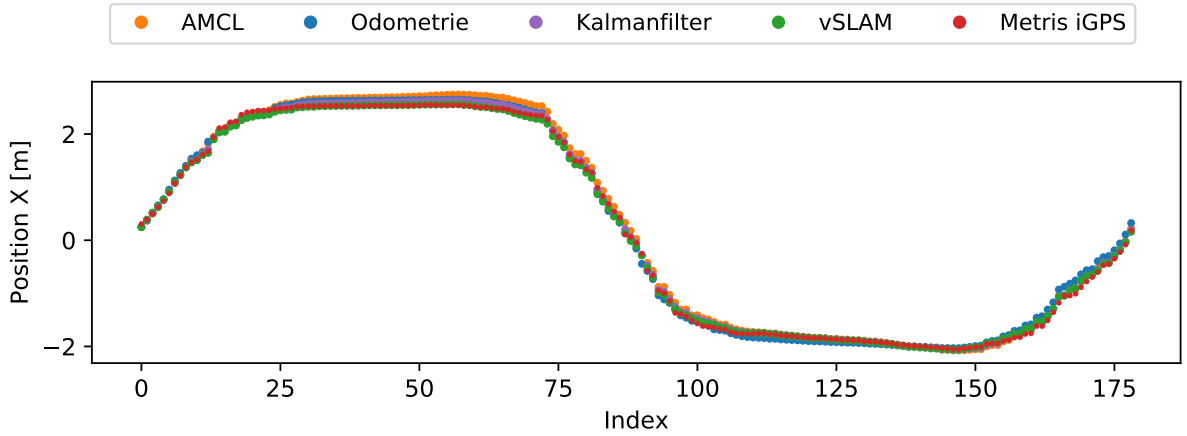


(a) Positionsverlauf der Lokalisierungsmethoden über eine quadratische Trajektorie.

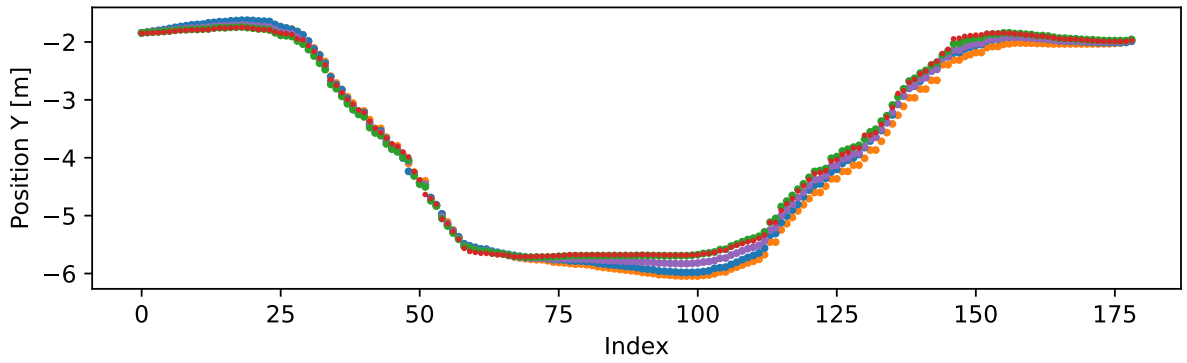


(b) Histogramm über die Positionsfehler aller Lokalisierungsmethoden.

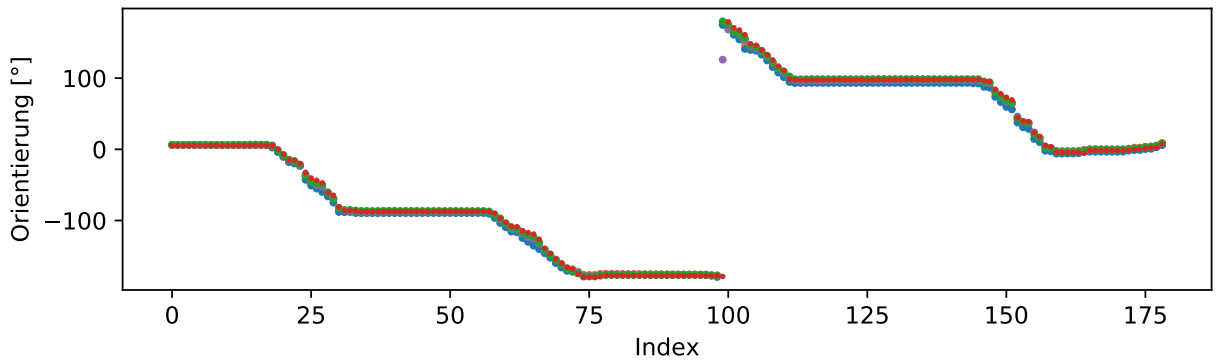
Abbildung C.7: Durchlauf 4 mit quadratischer Trajektorie: Fehlerhistogramm und Positionsverlauf.



(a) Verlauf der X-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

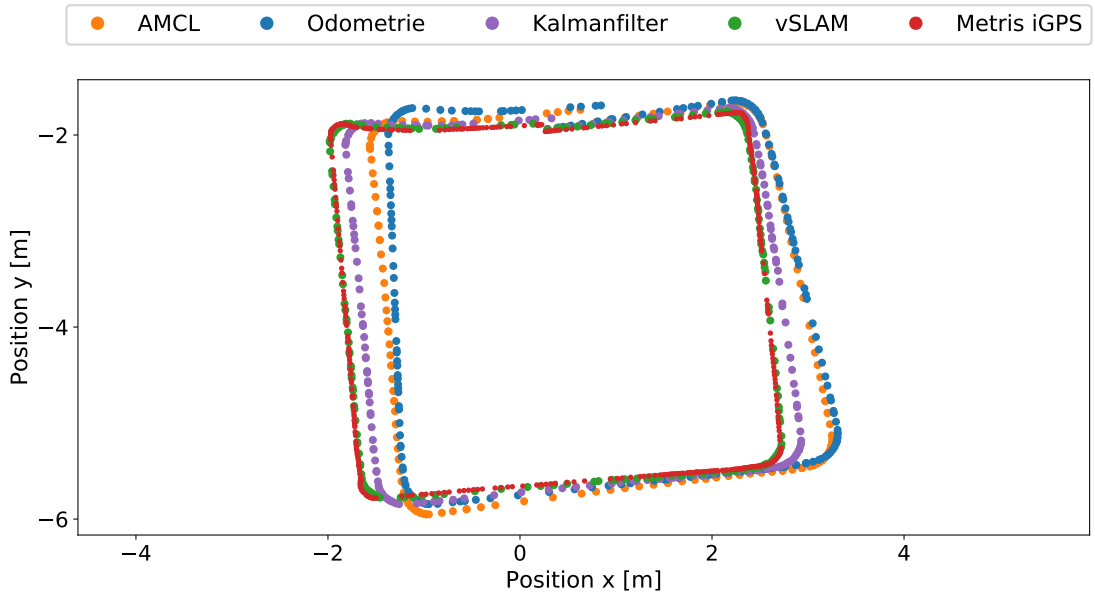


(b) Verlauf der Y-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

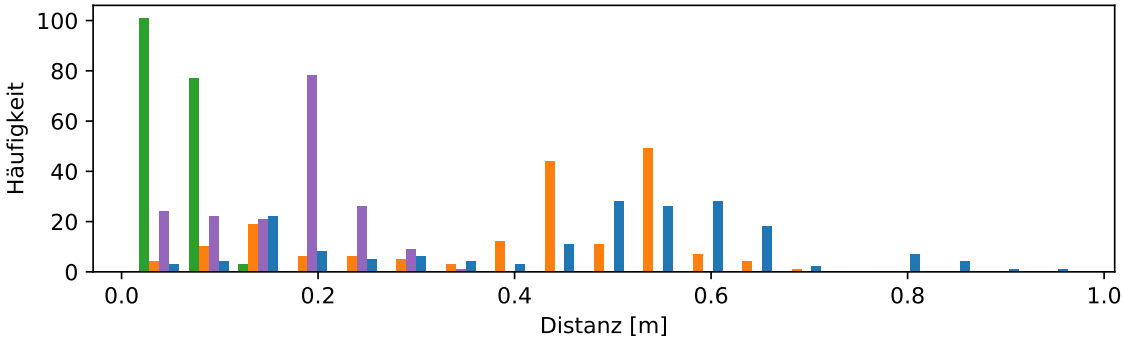


(c) Verlauf der Orientierung der Posenmessungen aller Lokalisierungsmethoden.

Abbildung C.8: Durchlauf 4 mit quadratischer Trajektorie: Verlauf der Koordinaten und des Orientierungswinkels.

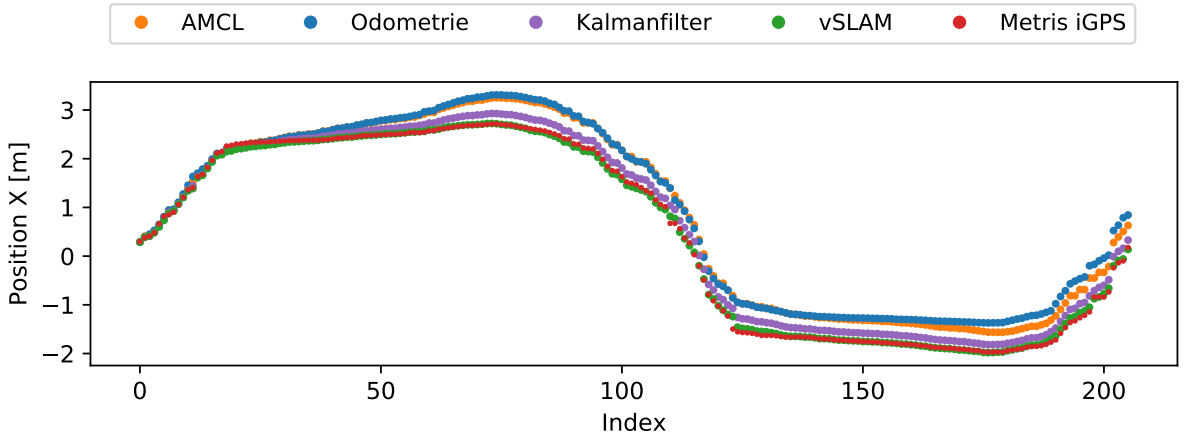


(a) Positionsverlauf der Lokalisierungsmethoden über eine quadratische Trajektorie.

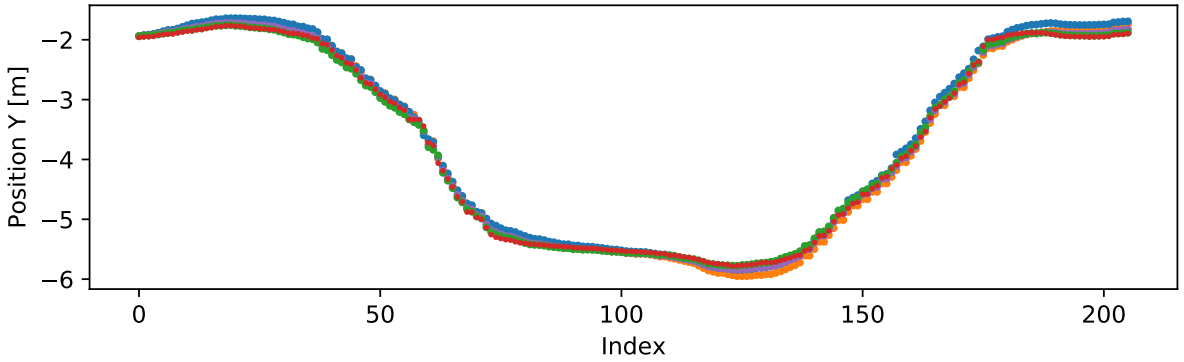


(b) Histogramm über die Positionsfehler aller Lokalisierungsmethoden.

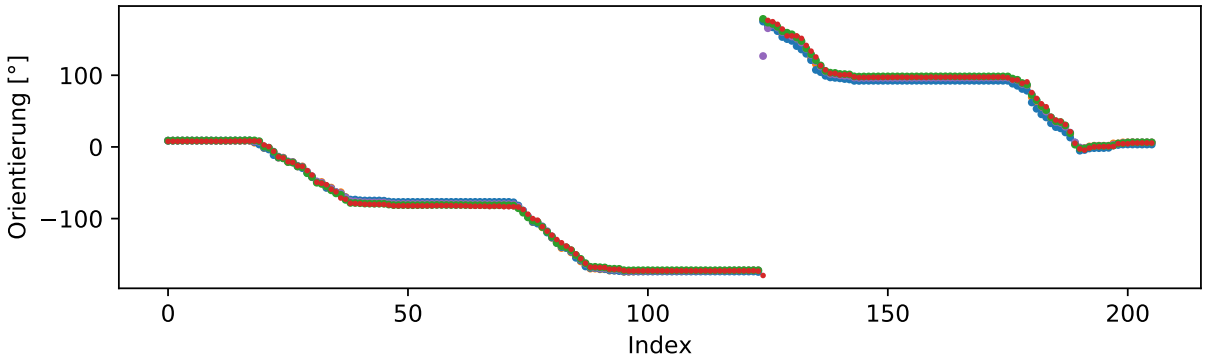
Abbildung C.9: Durchlauf 5 mit quadratischer Trajektorie: Fehlerhistogramm und Positionsverlauf.



(a) Verlauf der X-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

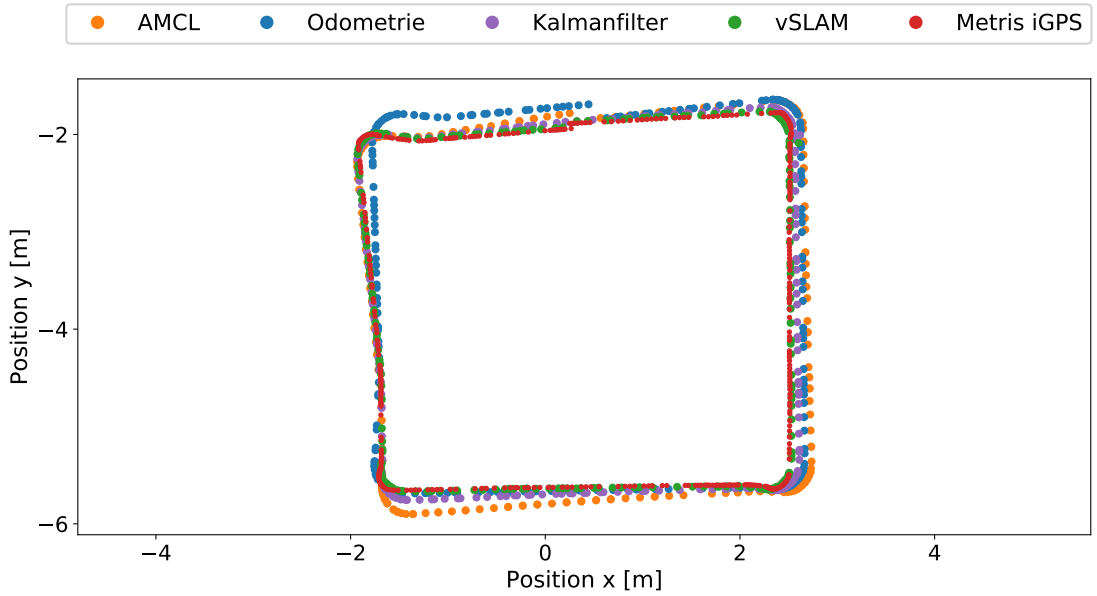


(b) Verlauf der Y-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

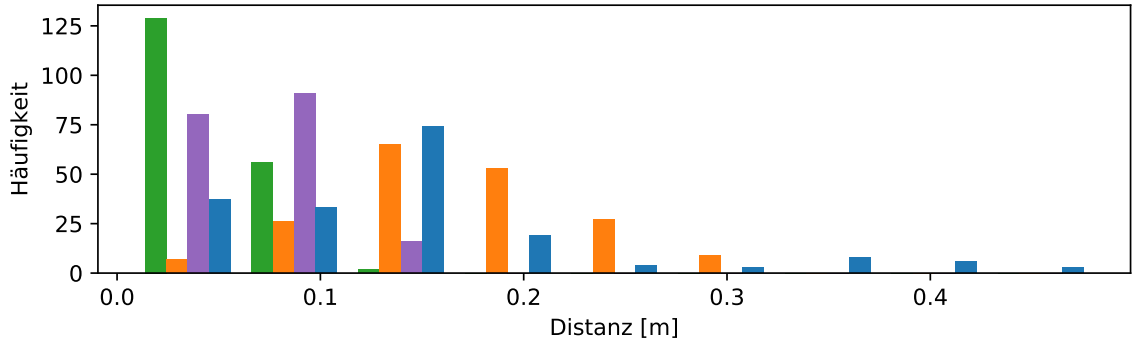


(c) Verlauf der Orientierung der Posenmessungen aller Lokalisierungsmethoden.

Abbildung C.10: Durchlauf 5 mit quadratischer Trajektorie: Verlauf der Koordinaten und des Orientierungswinkels.

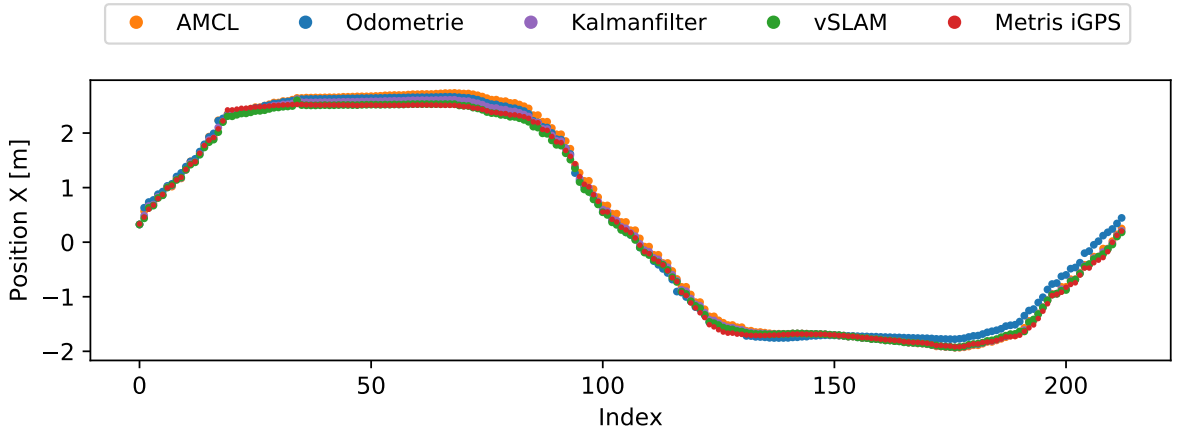


(a) Positionsverlauf der Lokalisierungsmethoden über eine quadratische Trajektorie.

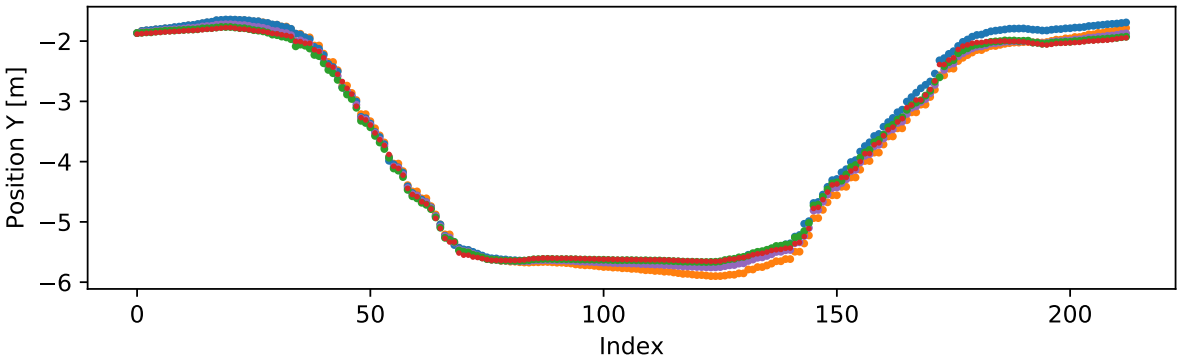


(b) Histogramm über die Positionsfehler aller Lokalisierungsmethoden.

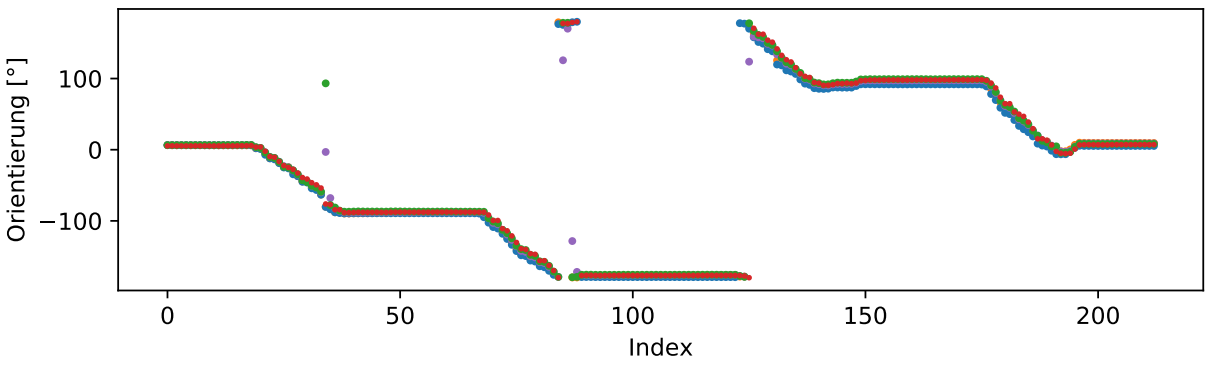
Abbildung C.11: Durchlauf 6 mit quadratischer Trajektorie: Fehlerhistogramm und Positionsverlauf.



(a) Verlauf der X-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

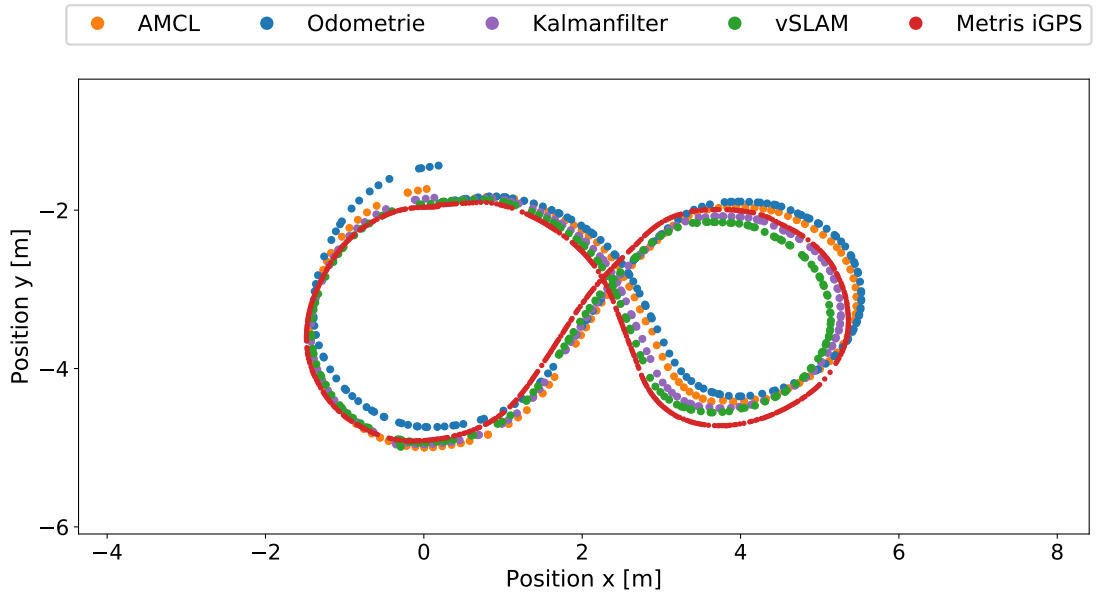


(b) Verlauf der Y-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

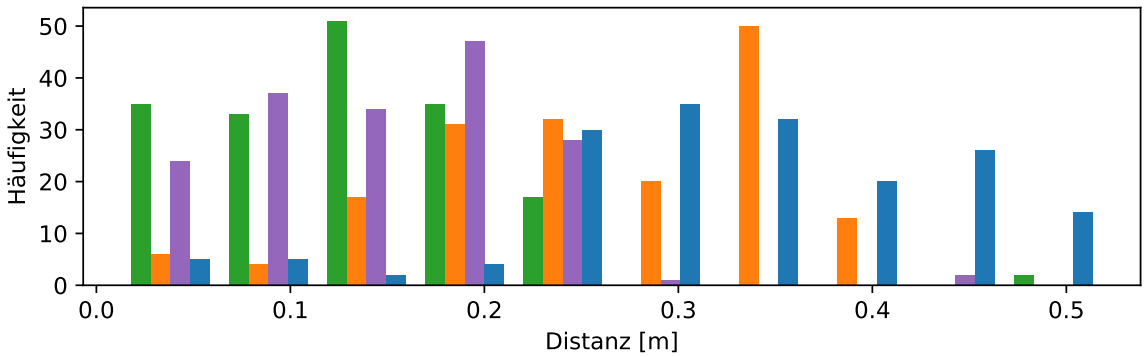


(c) Verlauf der Orientierung der Posenmessungen aller Lokalisierungsmethoden.

Abbildung C.12: Durchlauf 6 mit quadratischer Trajektorie: Verlauf der Koordinaten und des Orientierungswinkels.

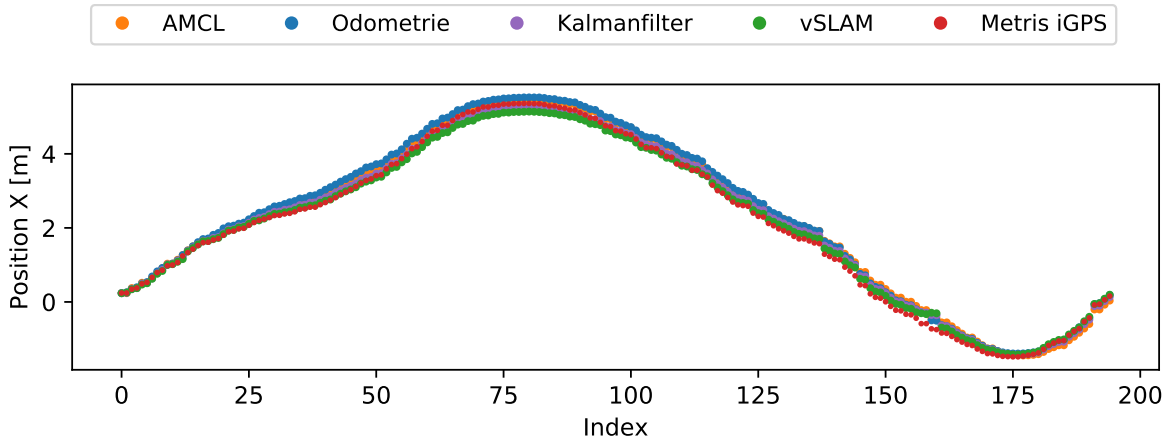


(a) Positionsverlauf der Lokalisierungsmethoden über eine achtförmige Trajektorie.

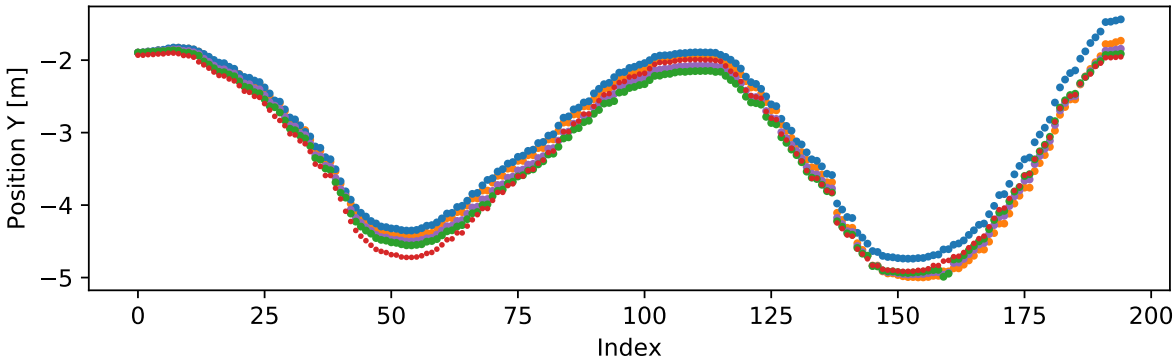


(b) Histogramm über die Positionsfehler aller Lokalisierungsmethoden.

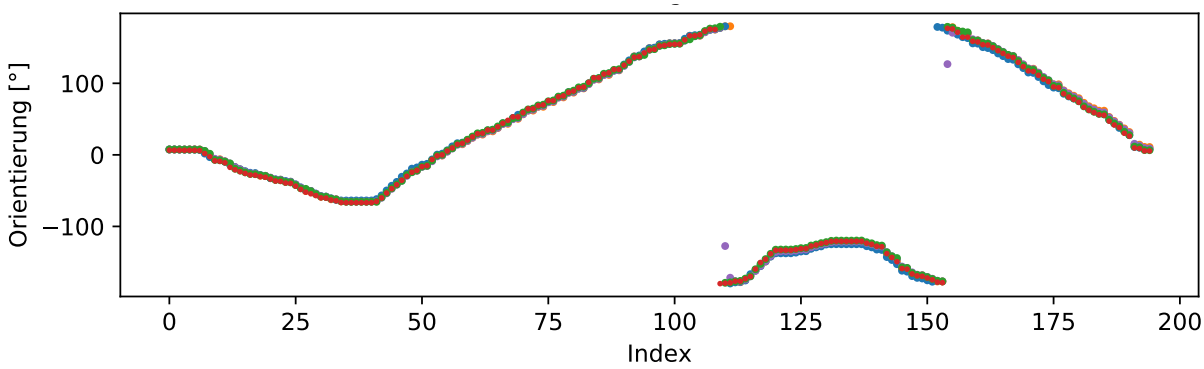
Abbildung C.13: Durchlauf 1 mit Trajektorie in Form einer Acht: Fehlerhistogramm und Positionsverlauf.



(a) Verlauf der X-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

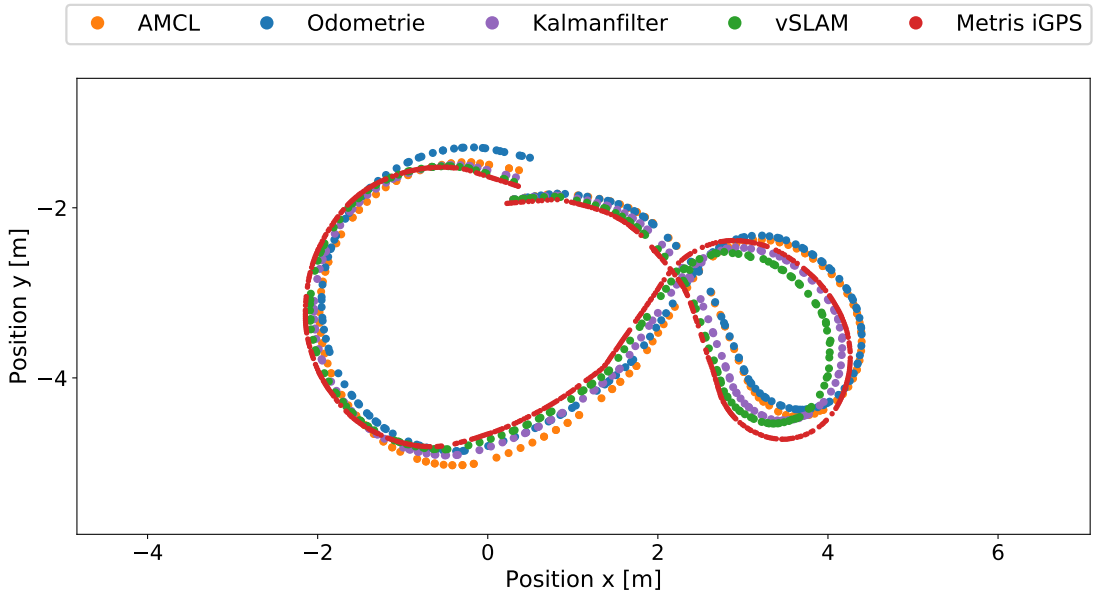


(b) Verlauf der Y-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

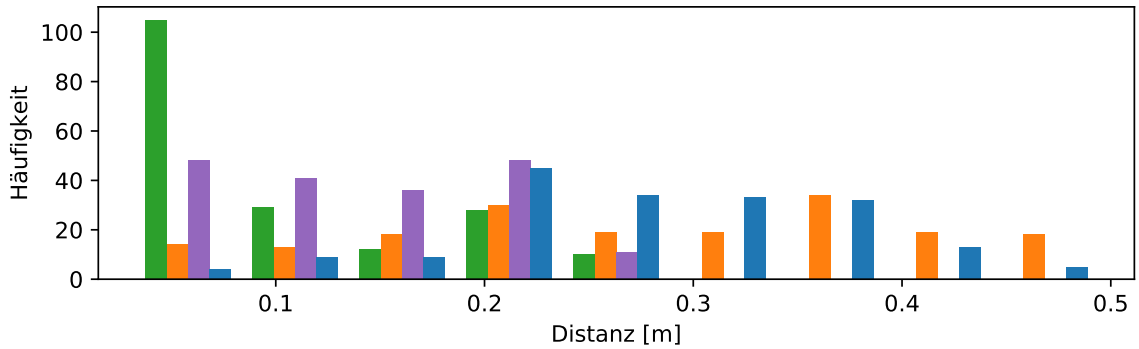


(c) Verlauf der Orientierung der Posenmessungen aller Lokalisierungsmethoden.

Abbildung C.14: Durchlauf 1 mit Trajektorie in Form einer Acht: Verlauf der Koordinaten und des Orientierungswinkels.

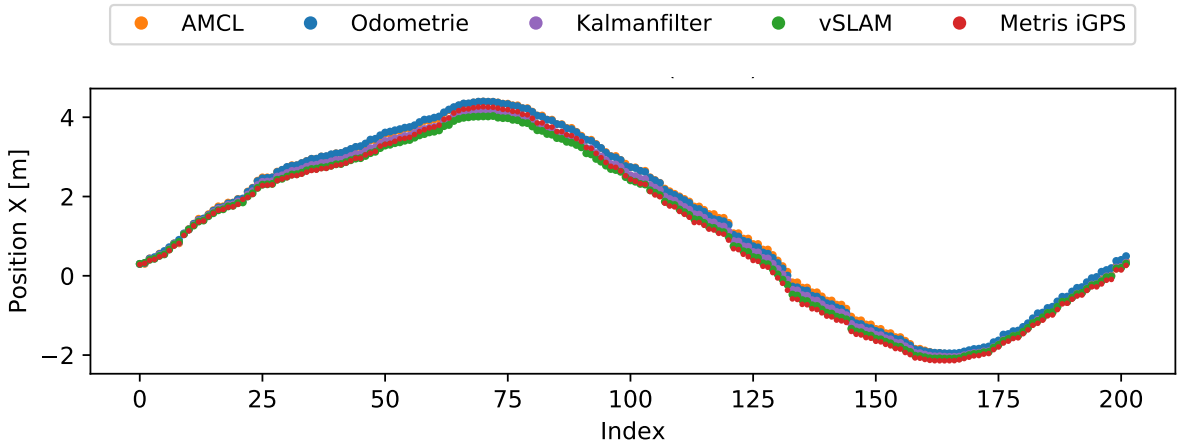


(a) Positionsverlauf der Lokalisierungsmethoden über eine achtförmige Trajektorie.

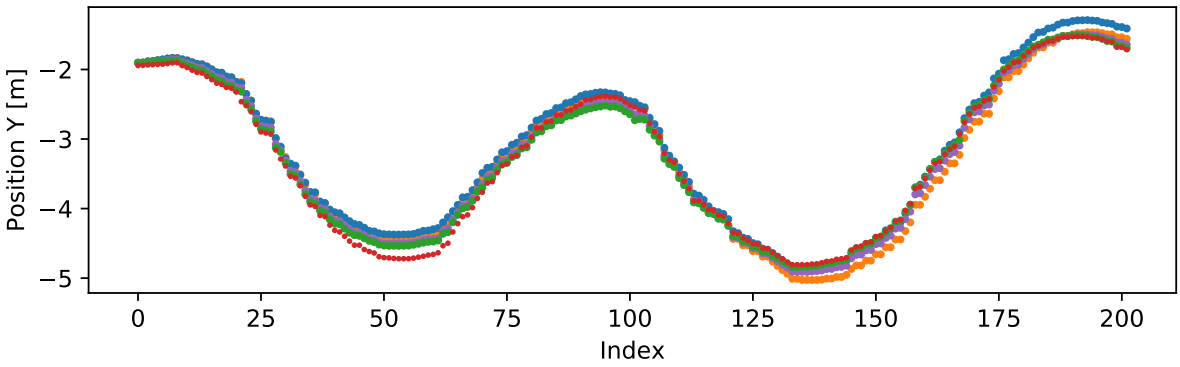


(b) Histogramm über die Positionsfehler aller Lokalisierungsmethoden.

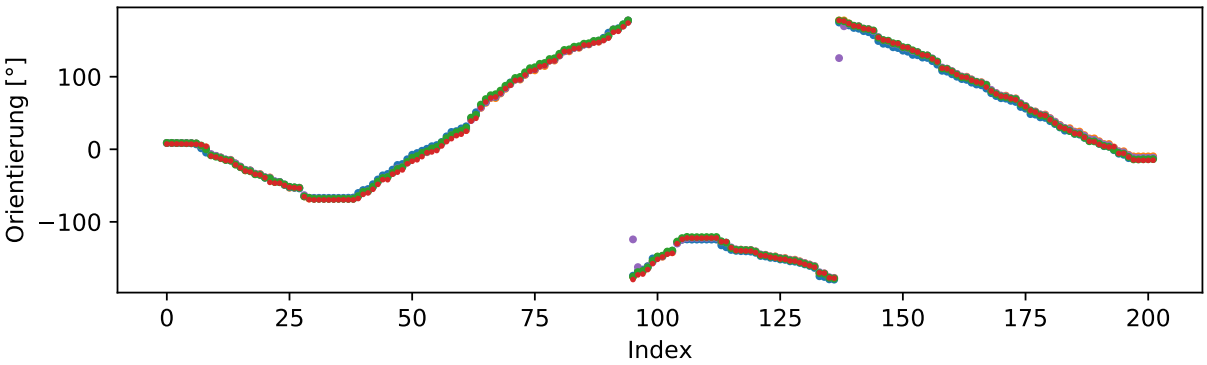
Abbildung C.15: Durchlauf 2 mit Trajektorie in Form einer Acht: Fehlerhistogramm und Positionsverlauf.



(a) Verlauf der X-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

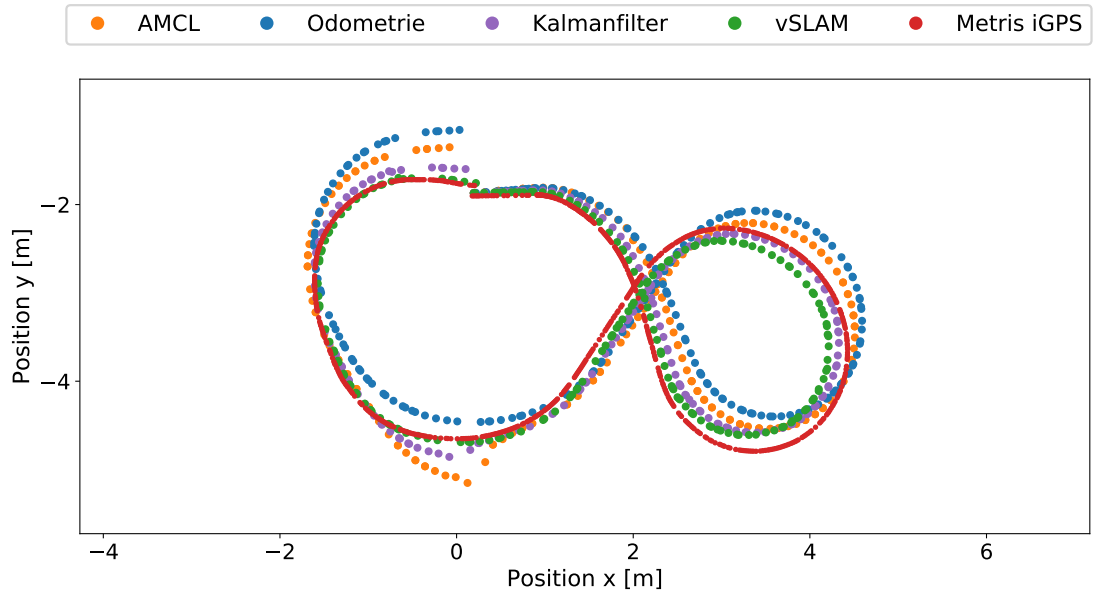


(b) Verlauf der Y-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

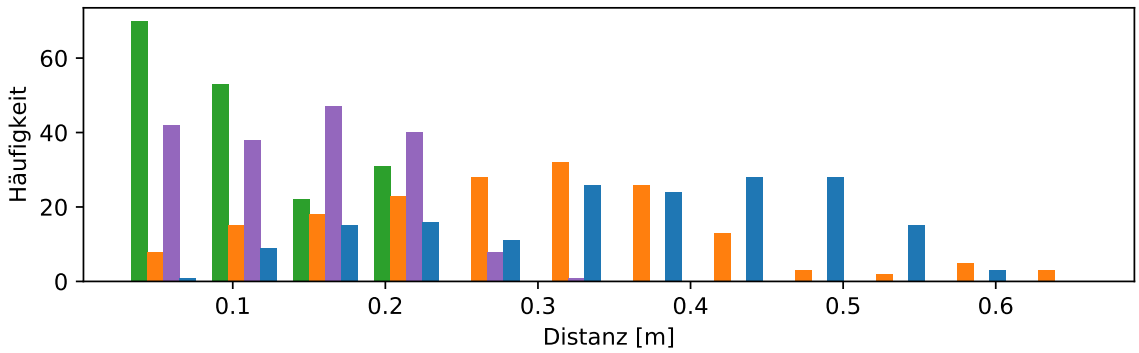


(c) Verlauf der Orientierung der Posenmessungen aller Lokalisierungsmethoden.

Abbildung C.16: Durchlauf 2 mit Trajektorie in Form einer Acht: Verlauf der Koordinaten und des Orientierungswinkels.

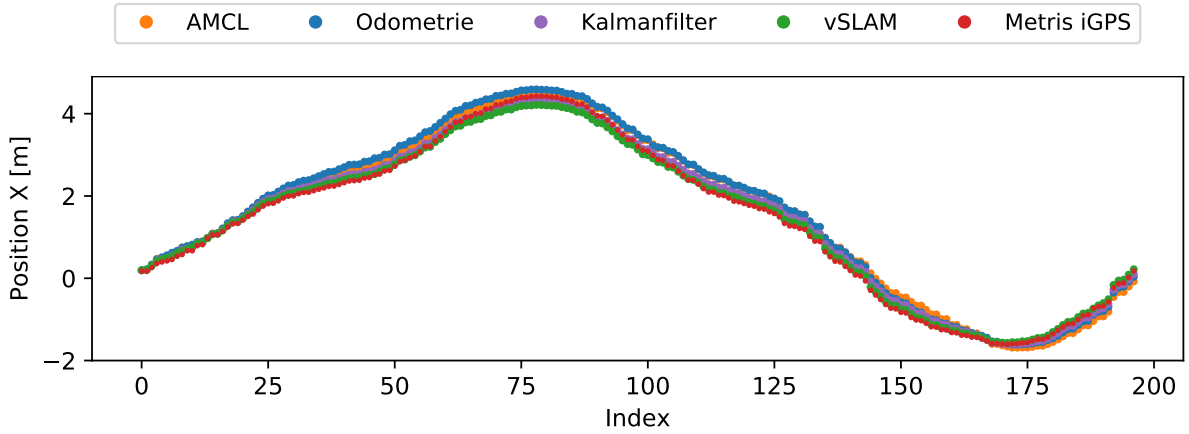


(a) Positionsverlauf der Lokalisierungsmethoden über eine achtförmige Trajektorie.

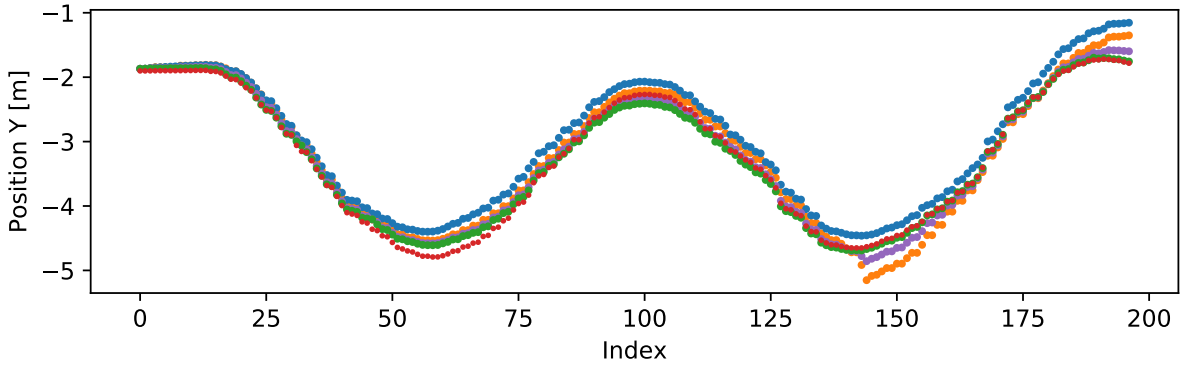


(b) Histogramm über die Positionsfehler aller Lokalisierungsmethoden.

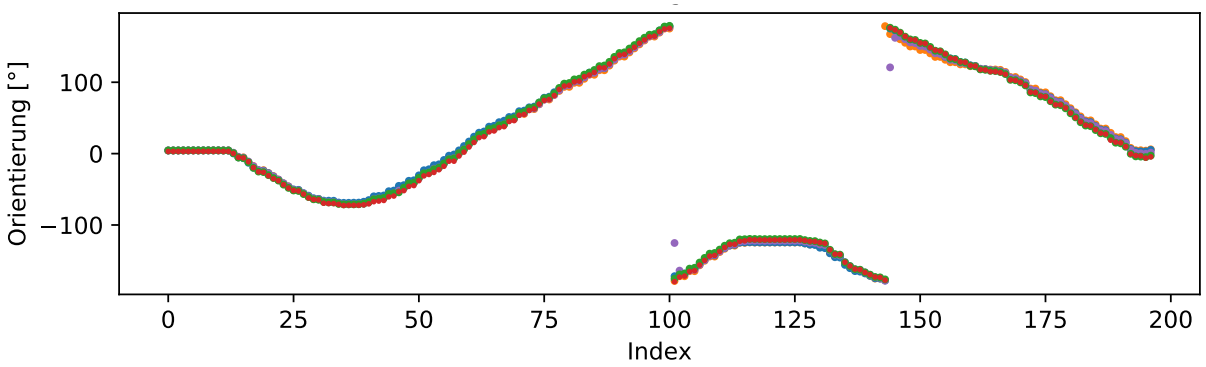
Abbildung C.17: Durchlauf 3 mit Trajektorie in Form einer Acht: Fehlerhistogramm und Positionsverlauf.



(a) Verlauf der X-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

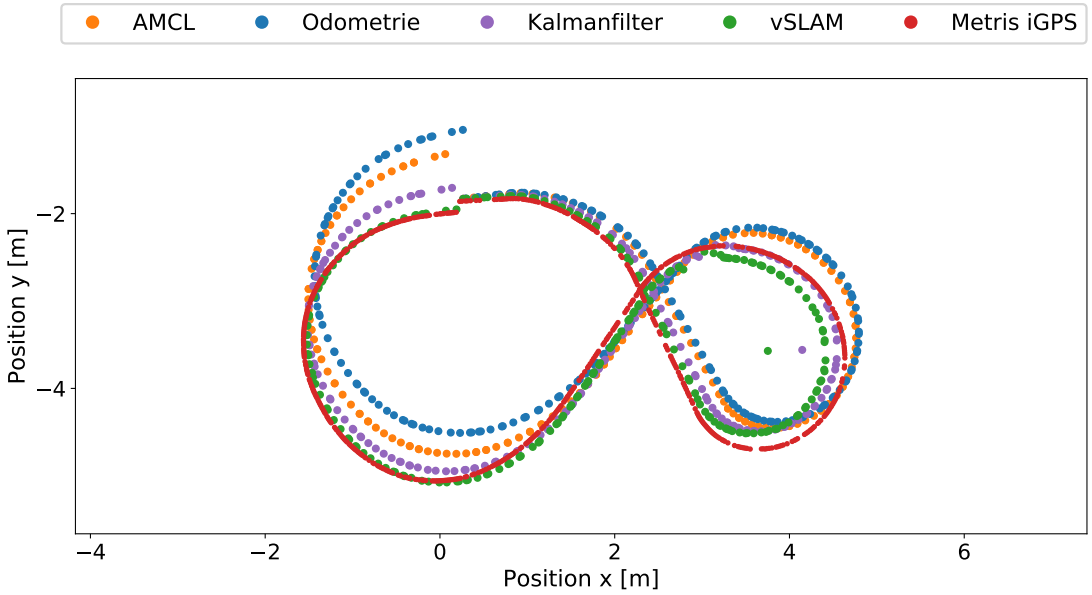


(b) Verlauf der Y-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

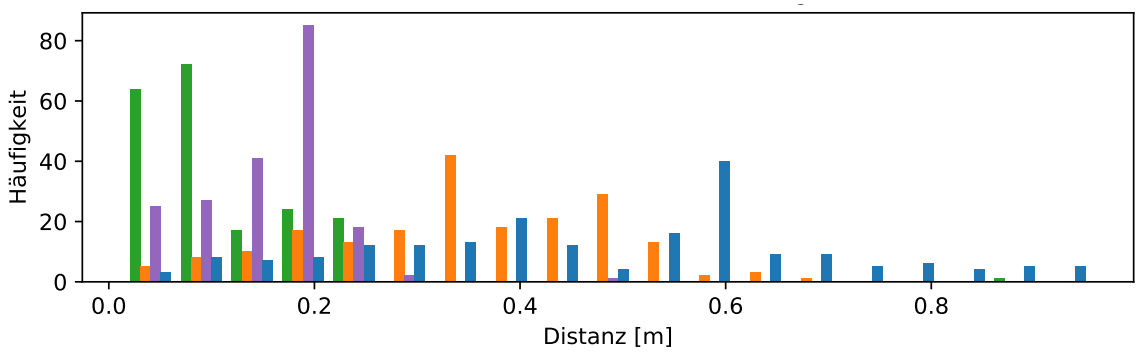


(c) Verlauf der Orientierung der Posenmessungen aller Lokalisierungsmethoden.

Abbildung C.18: Durchlauf 3 mit achtf. Trajektorie: Koordinaten und Orientierungswinkel.

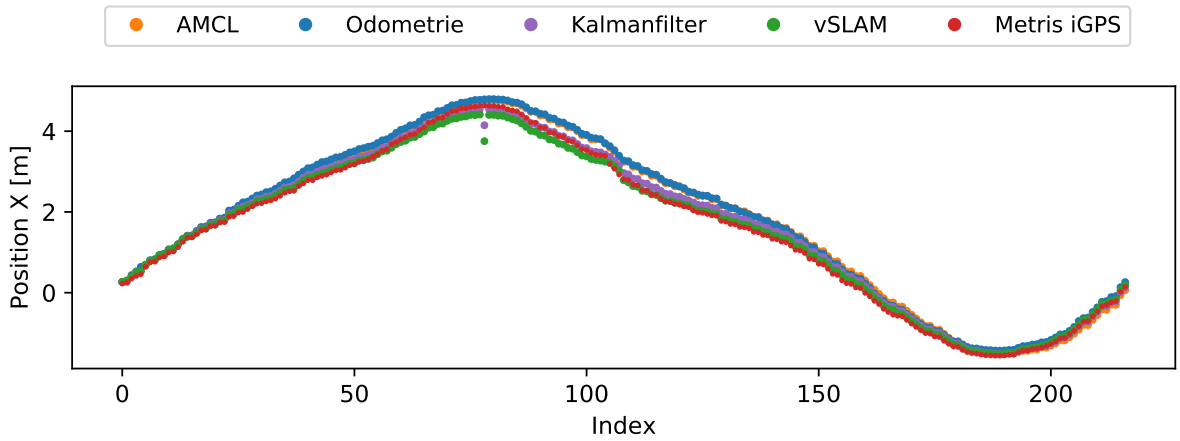


(a) Positionsverlauf der Lokalisierungsmethoden über eine achtförmige Trajektorie.

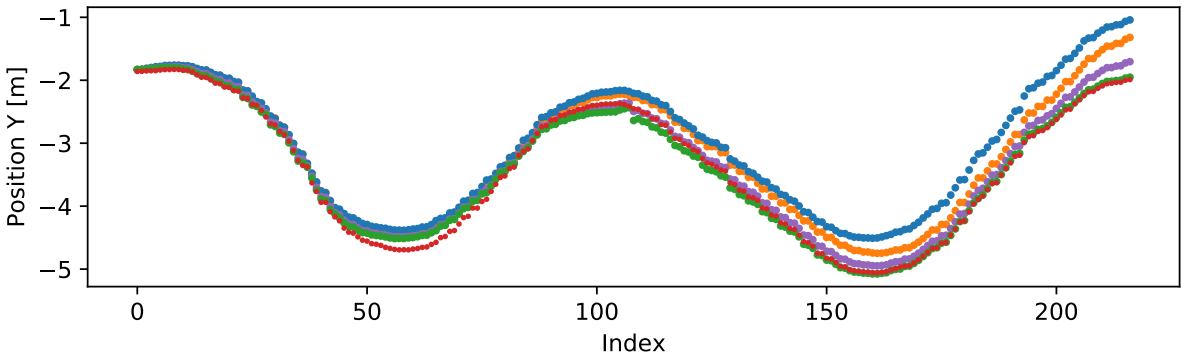


(b) Histogramm über die Positionsfehler aller Lokalisierungsmethoden.

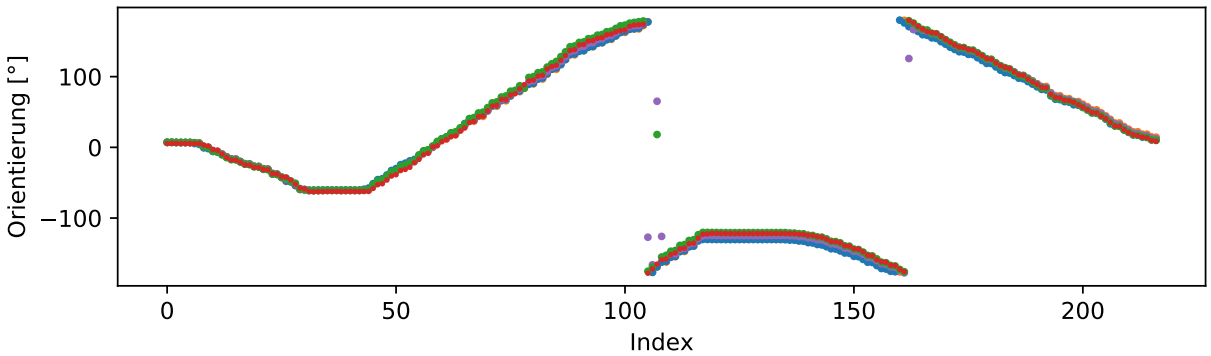
Abbildung C.19: Durchlauf 4 mit Trajektorie in Form einer Acht: Fehlerhistogramm und Positionsverlauf.



(a) Verlauf der X-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

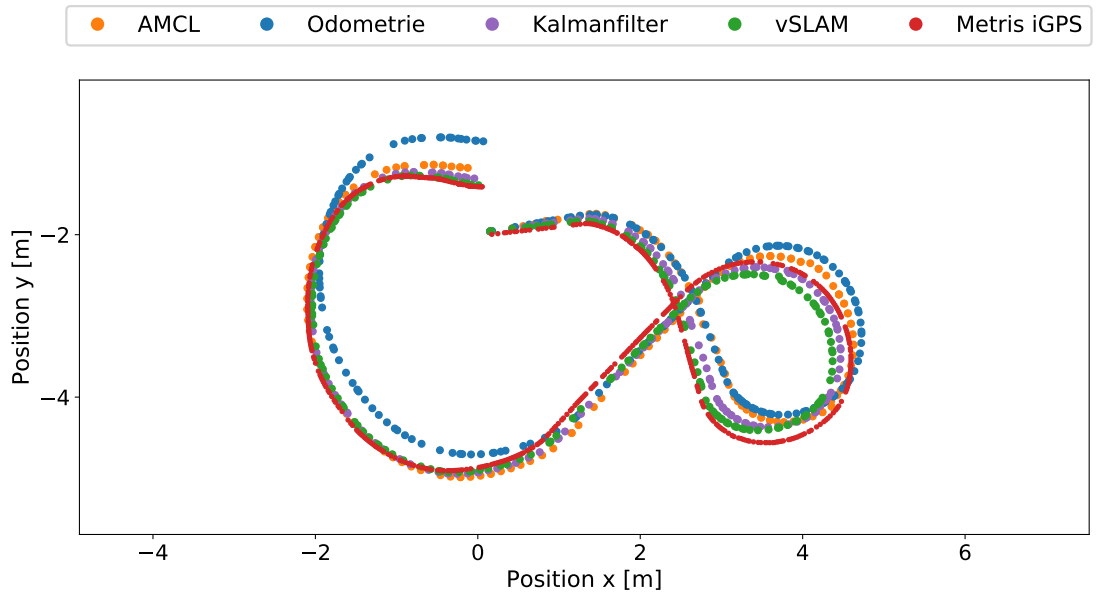


(b) Verlauf der Y-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

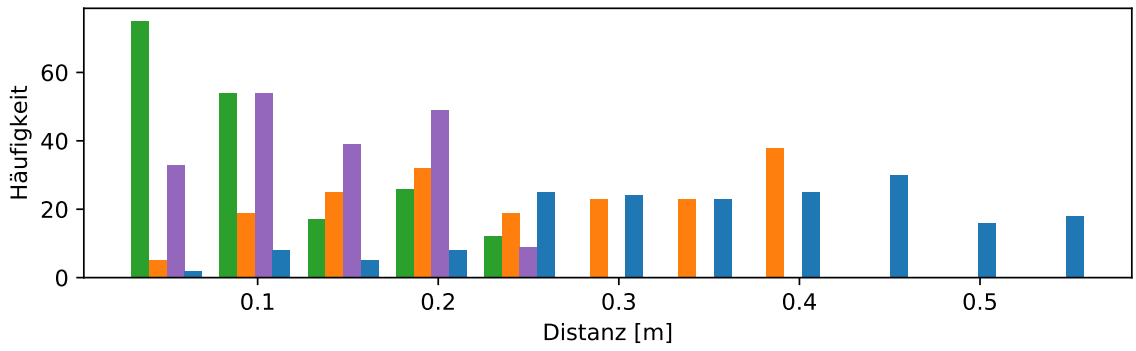


(c) Verlauf der Orientierung der Posenmessungen aller Lokalisierungsmethoden.

Abbildung C.20: Durchlauf 4 mit Trajektorie in Form einer Acht: Verlauf der Koordinaten und des Orientierungswinkels.

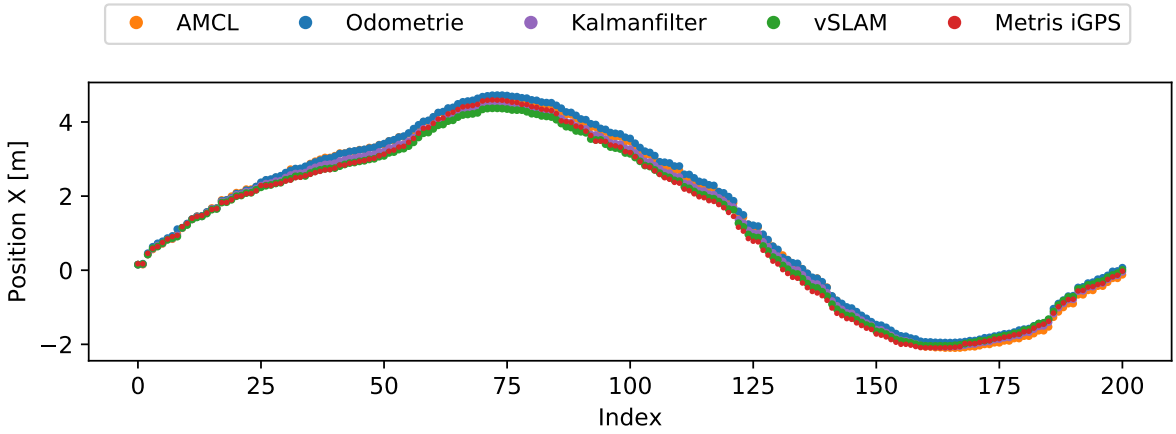


(a) Positionsverlauf der Lokalisierungsmethoden über eine achtförmige Trajektorie.

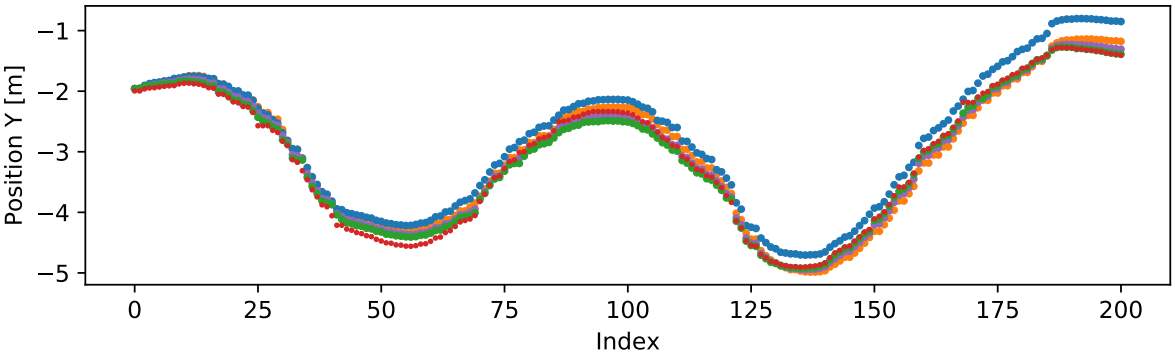


(b) Histogramm über die Positionsfehler aller Lokalisierungsmethoden.

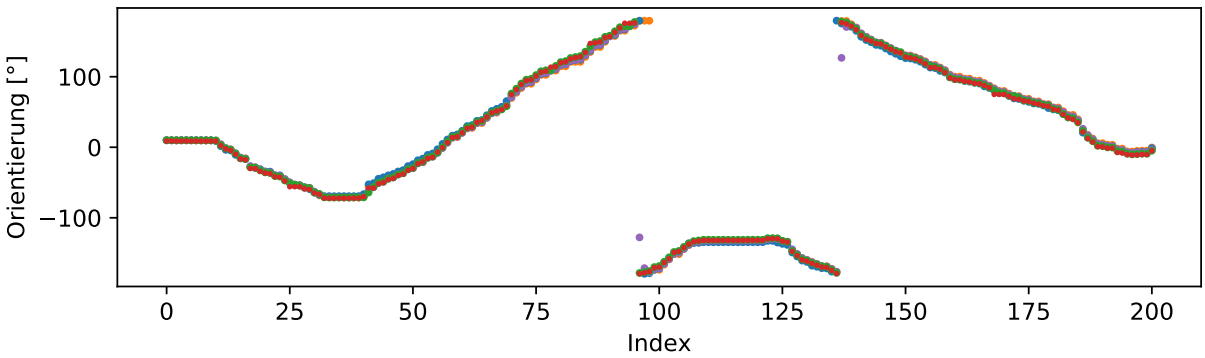
Abbildung C.21: Durchlauf 5 mit Trajektorie in Form einer Acht: Fehlerhistogramm und Positionsverlauf.



(a) Verlauf der X-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

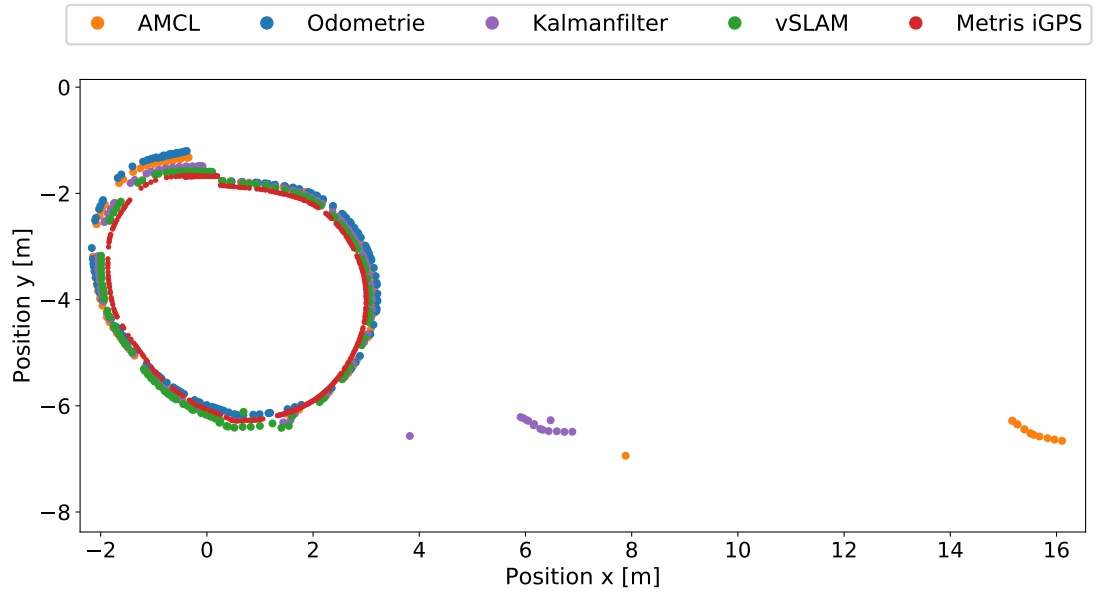


(b) Verlauf der Y-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

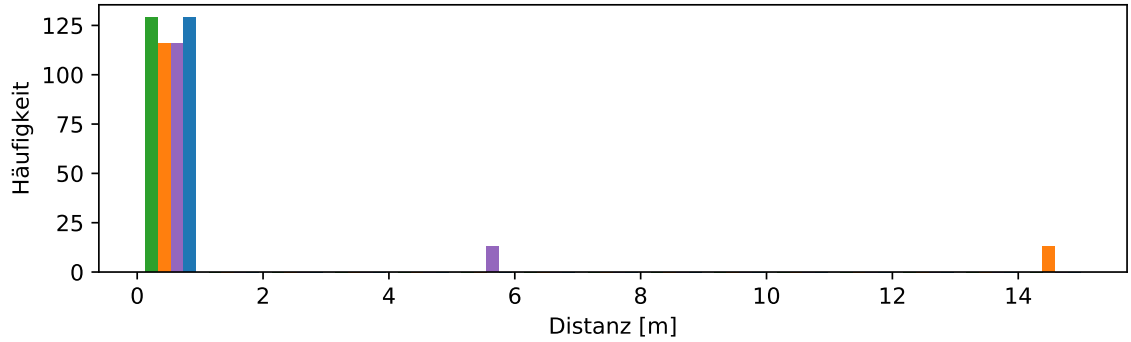


(c) Verlauf der Orientierung der Posenmessungen aller Lokalisierungsmethoden.

Abbildung C.22: Durchlauf 5 mit Trajektorie in Form einer Acht: Verlauf der Koordinaten und des Orientierungswinkels.

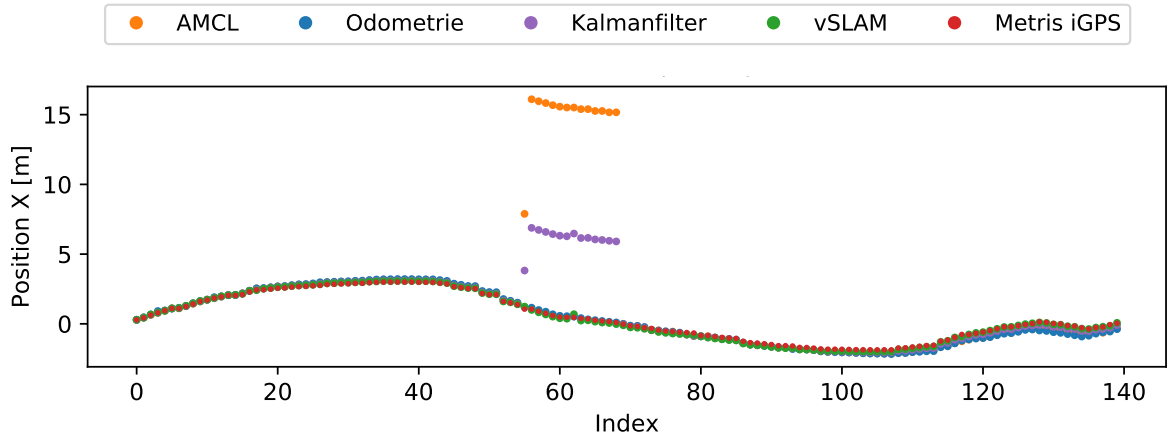


(a) Positionsverlauf der Lokalisierungsmethoden über eine kreisförmige Trajektorie.

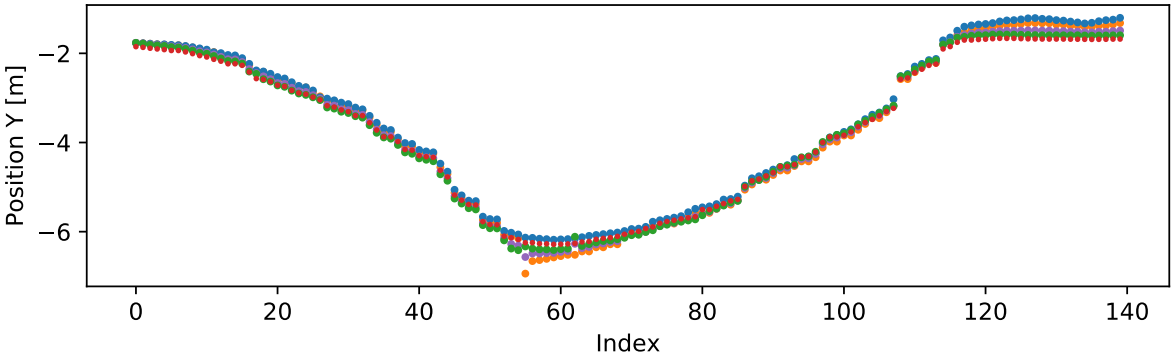


(b) Histogramm über die Positionsfehler aller Lokalisierungsmethoden.

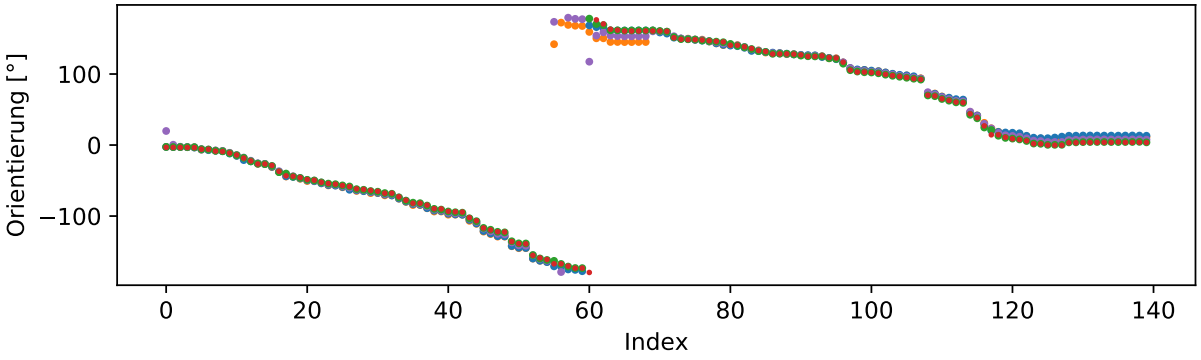
Abbildung C.23: Durchlauf 1 mit kreisförmiger Trajektorie: Fehlerhistogramm und Positionsverlauf (aufgrund des Ausreißers von AMCL nicht in die Gesamtauswertung miteinbezogen).



(a) Verlauf der X-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

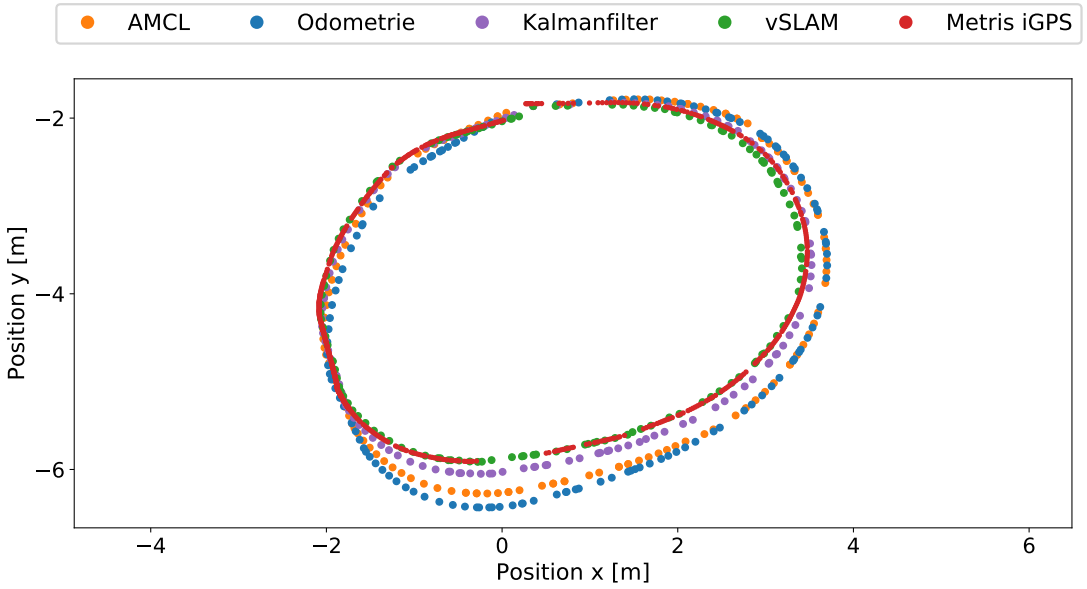


(b) Verlauf der Y-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

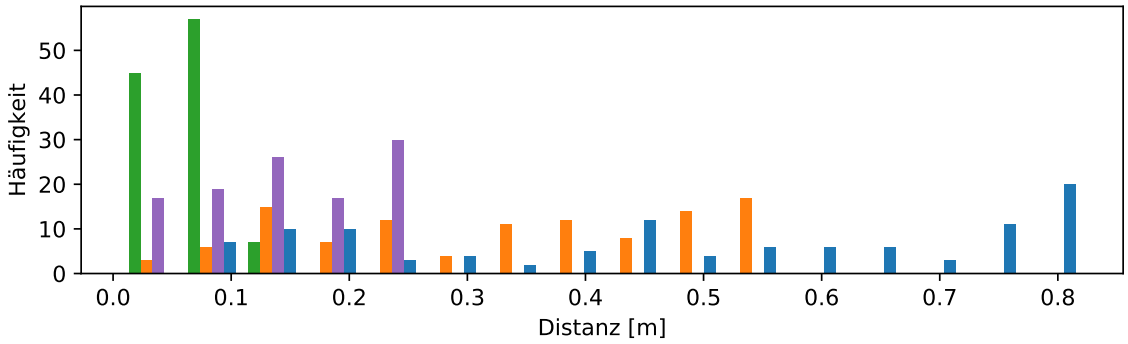


(c) Verlauf der Orientierung der Posenmessungen aller Lokalisierungsmethoden.

Abbildung C.24: Durchlauf 1 mit kreisförmiger Trajektorie: Verlauf der Koordinaten und des Orientierungswinkels (aufgrund des Ausreißers von AMCL nicht in die Gesamtauswertung miteinbezogen).

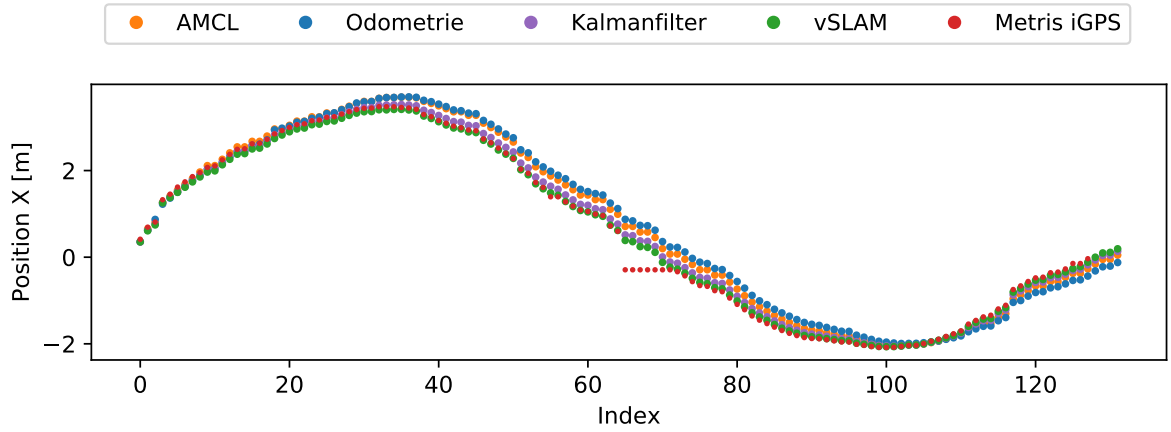


(a) Positionsverlauf der Lokalisierungsmethoden über eine kreisförmige Trajektorie.

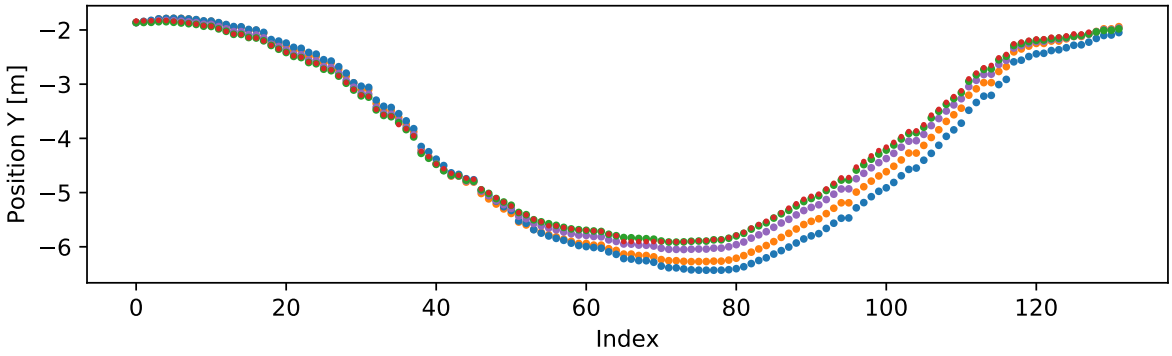


(b) Histogramm über die Positionsfehler aller Lokalisierungsmethoden.

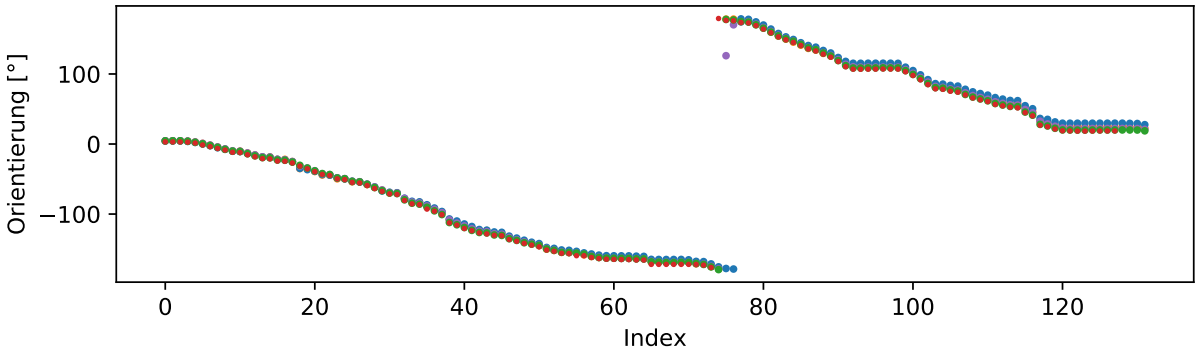
Abbildung C.25: Durchlauf 2 mit kreisförmiger Trajektorie: Fehlerhistogramm und Positionsverlauf.



(a) Verlauf der X-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

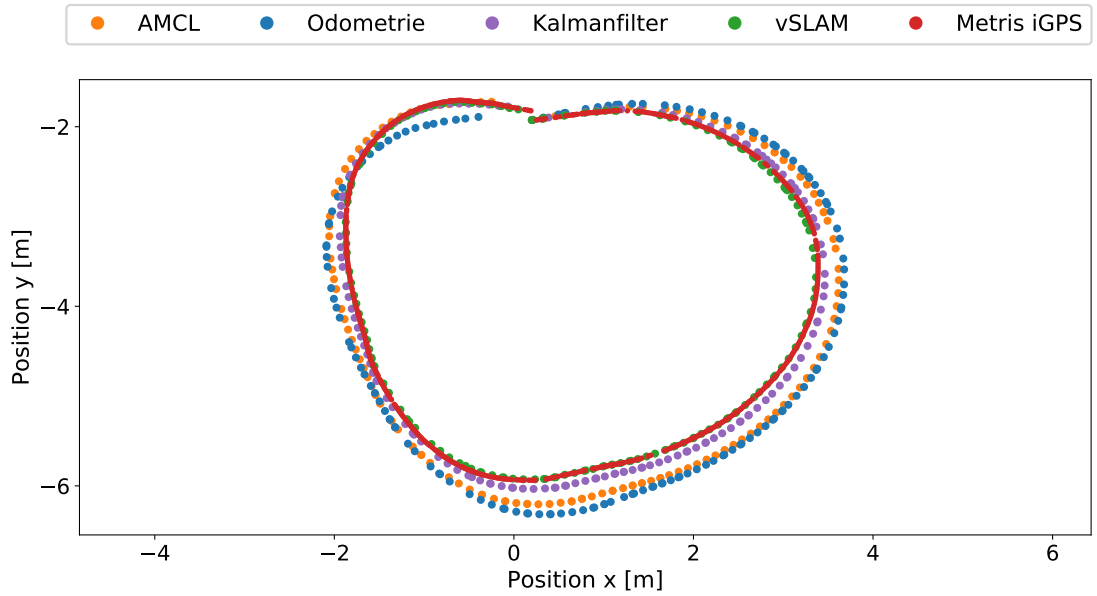


(b) Verlauf der Y-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

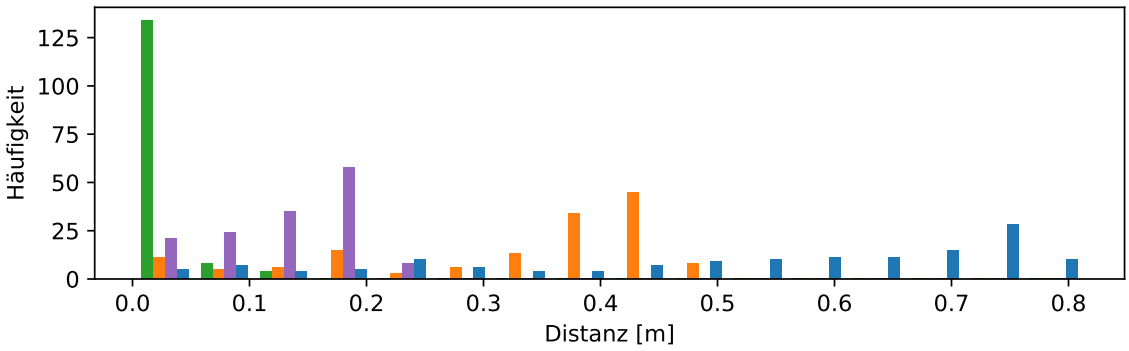


(c) Verlauf der Orientierung der Posenmessungen aller Lokalisierungsmethoden.

Abbildung C.26: Durchlauf 2 mit kreisförmiger Trajektorie: Verlauf der Koordinaten und des Orientierungswinkels.

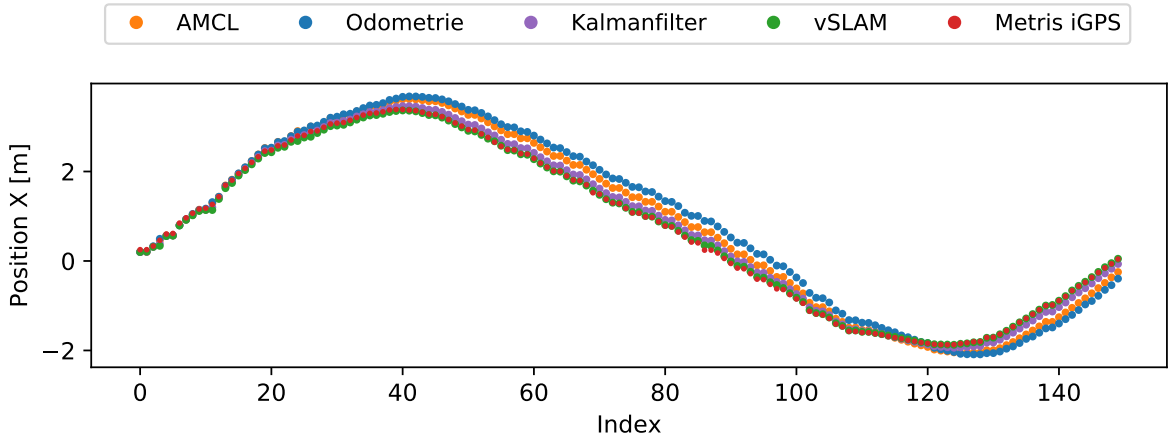


(a) Positionsverlauf der Lokalisierungsmethoden über eine kreisförmige Trajektorie.

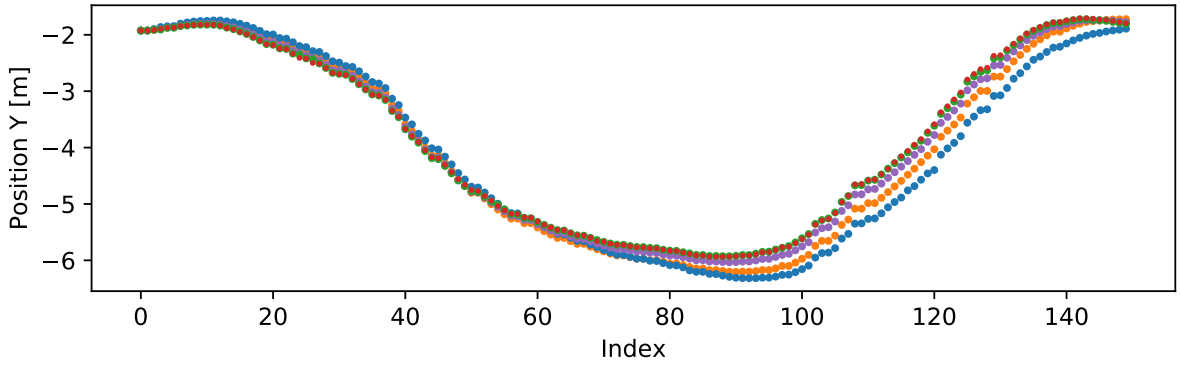


(b) Histogramm über die Positionsfehler aller Lokalisierungsmethoden.

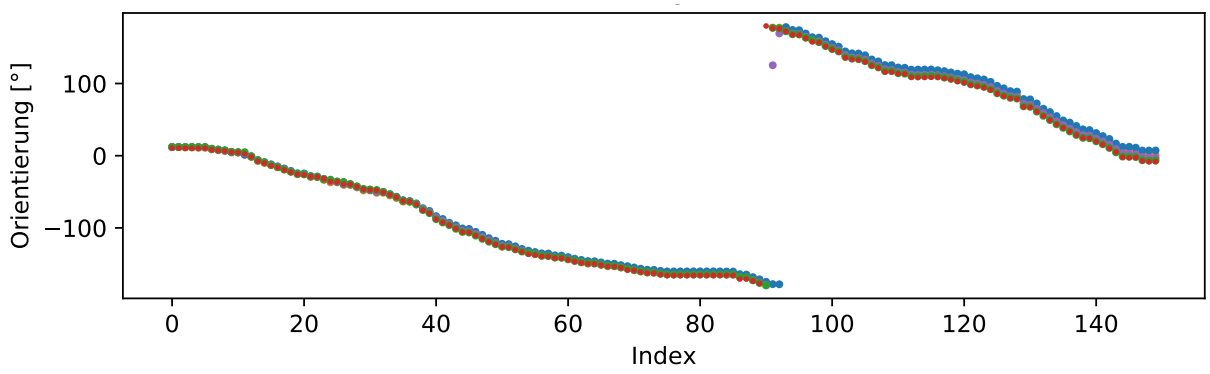
Abbildung C.27: Durchlauf 3 mit kreisförmiger Trajektorie: Fehlerhistogramm und Positionsverlauf.



(a) Verlauf der X-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

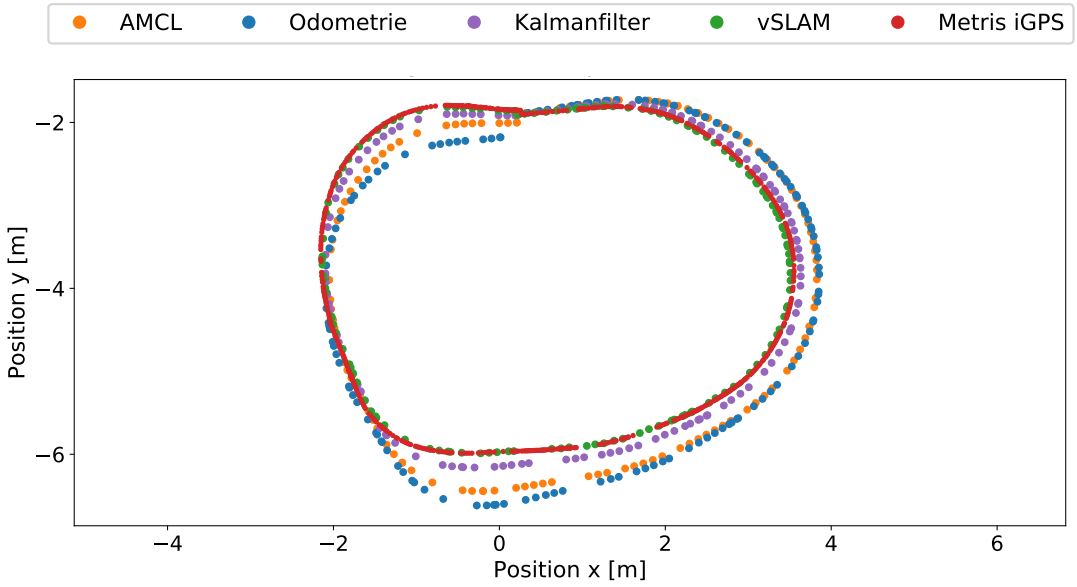


(b) Verlauf der Y-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

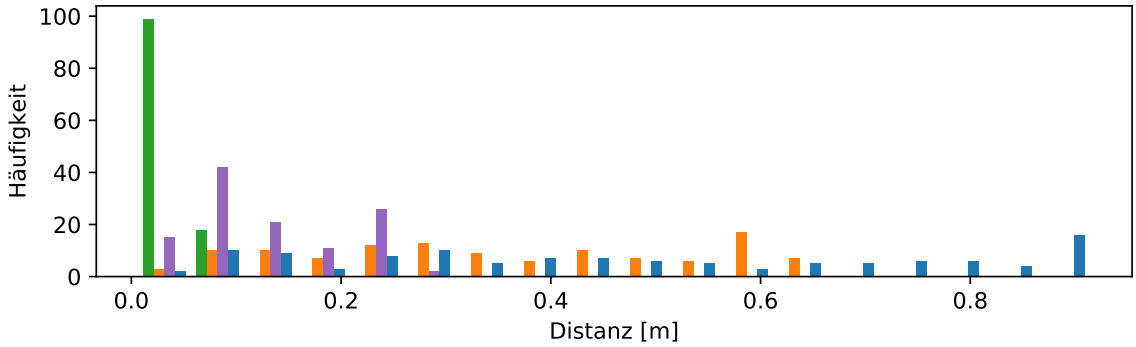


(c) Verlauf der Orientierung der Posenmessungen aller Lokalisierungsmethoden.

Abbildung C.28: Durchlauf 3 mit kreisförmiger Trajektorie: Verlauf der Koordinaten und des Orientierungswinkels.

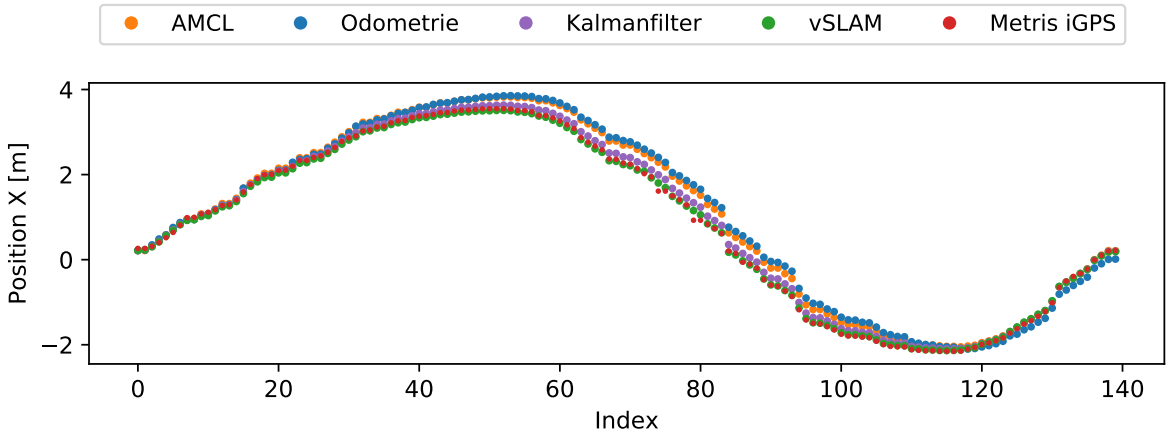


(a) Positionsverlauf der Lokalisierungsmethoden über eine kreisförmige Trajektorie.

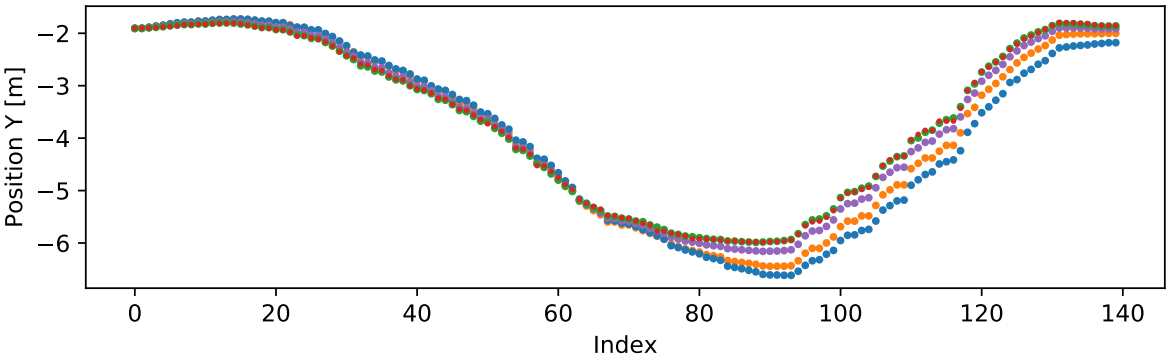


(b) Histogramm über die Positionsfehler aller Lokalisierungsmethoden.

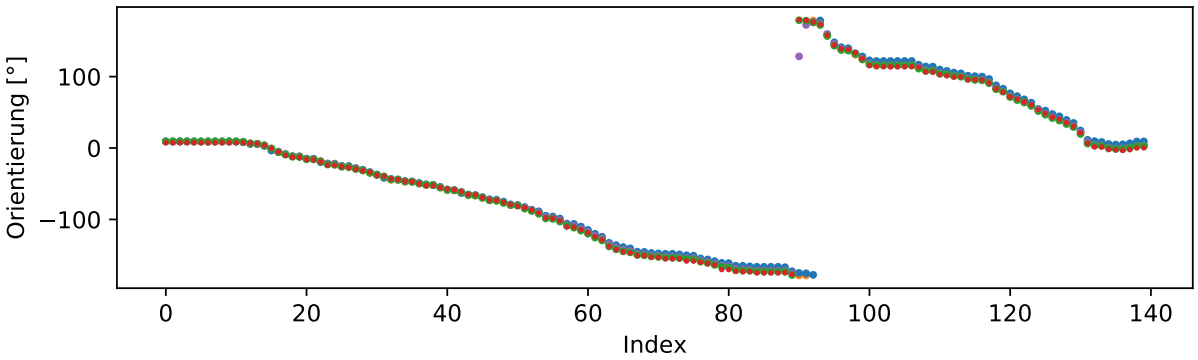
Abbildung C.29: Durchlauf 4 mit kreisförmiger Trajektorie: Fehlerhistogramm und Positionsverlauf.



(a) Verlauf der X-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

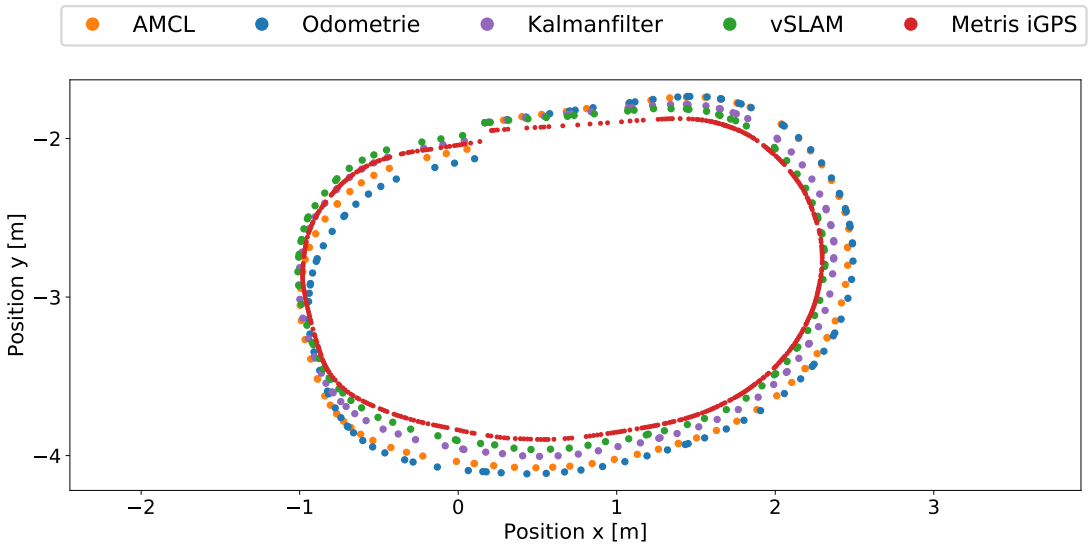


(b) Verlauf der Y-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.

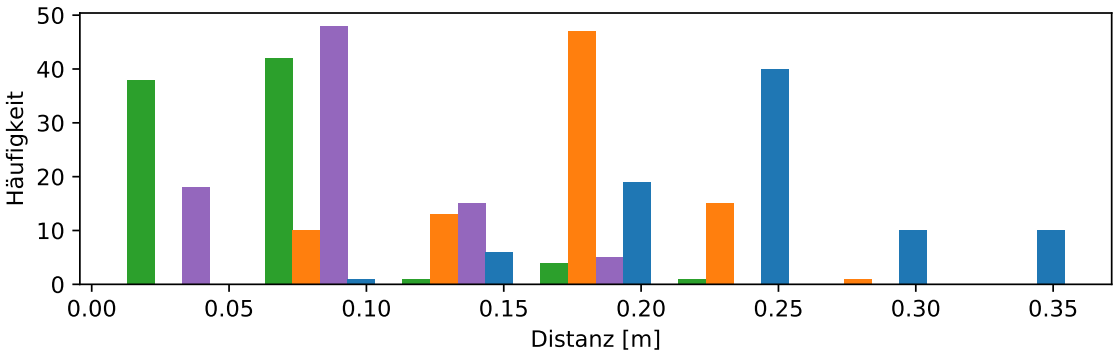


(c) Verlauf der Orientierung der Posenmessungen aller Lokalisierungsmethoden.

Abbildung C.30: Durchlauf 4 mit kreisförmiger Trajektorie: Verlauf der Koordinaten und des Orientierungswinkels.

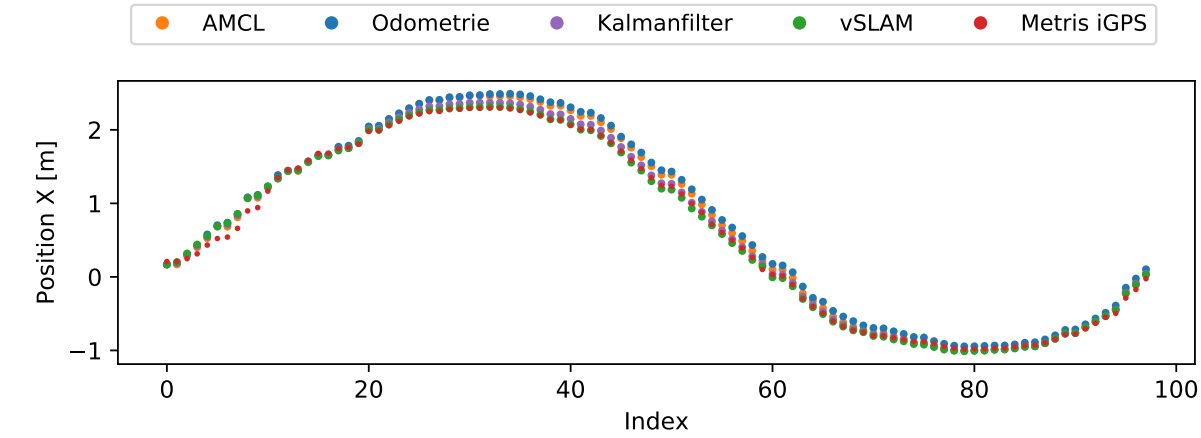


(a) Positionsverlauf der Lokalisierungsmethoden über eine kreisförmige Trajektorie.

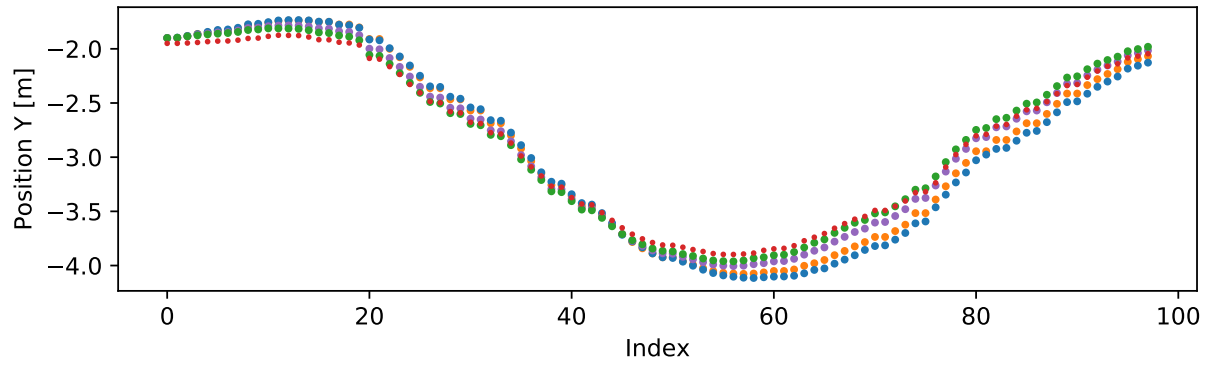


(b) Histogramm über die Positionsfehler aller Lokalisierungsmethoden.

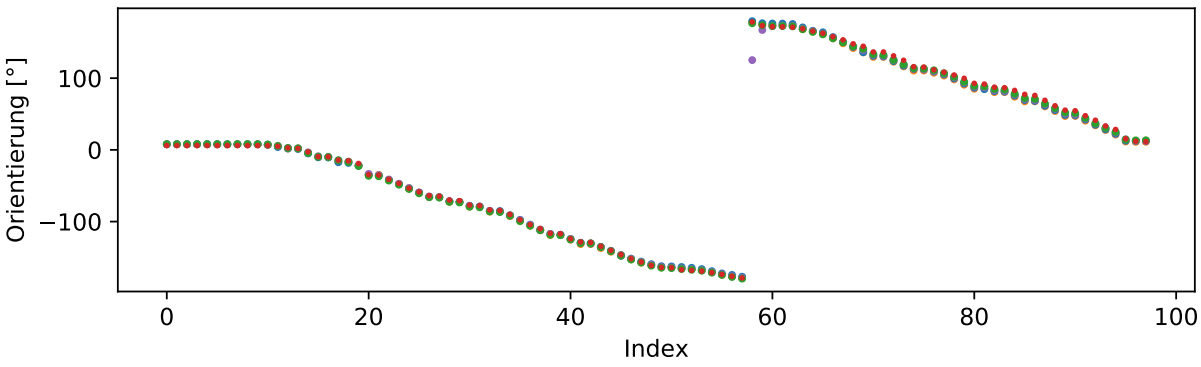
Abbildung C.31: Durchlauf 5 mit kreisförmiger Trajektorie: Fehlerhistogramm und Positionsverlauf.



(a) Verlauf der X-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.



(b) Verlauf der Y-Koordinaten der Positionsmessungen aller Lokalisierungsmethoden.



(c) Verlauf der Orientierung der Posenmessungen aller Lokalisierungsmethoden.

Abbildung C.32: Durchlauf 5 mit kreisförmiger Trajektorie: Verlauf der Koordinaten und des Orientierungswinkels.

Anhang D

Inhalt des beigefügten Datenträgers

Folgenden Dateien befinden sich auf dem beigefügten Datenträger:

- die vorliegende Arbeit als PDF-Dokument (*Masterarbeit_Darius_Deubert.pdf*)
- *Referenzen.zip*: Enthält die Veröffentlichungen, auf die in der Arbeit verwiesen wird.
- Ordner *lsd-slam*: Enthält den Quellcode der angepassten Version von LSD-SLAM sowie Auswertungsdaten.
- Ordner *extended-orb-slam*: Enthält den Quellcode der angepassten Version von ORB-SLAM3, den Quellcode der Erweiterungen zur Skalenrekonstruktion, zum Occupancy Grid Mapping und des Kalmanfilters sowie die Skripte zur Auswertung der Experimente und die entsprechenden Datensätze.

Weitere Erläuterungen zu den beigefügten Inhalten befinden sich in zusätzlichen Text-Dateien (*README.md*) in den jeweiligen Ordnern. Hierin wird auch beschrieben, wie das entwickelte System auszuführen ist.

Abbildungsverzeichnis

2.1 Überblick über die Threads und Funktionsweise von ORB-SLAM 7
2.2 Überblick über die Komponenten und Funktionsweise von LSD-SLAM 10
2.3 Apriltag (ID: 0) der Familie 36h11 13

3.1 Überblick über die Transformationsschritte eines Kameramodells 16
3.2 Geometrische Zusammenhänge bei der Abbildung durch eine Linse 17
3.3 Kamerakoordinatensystem mit Bildkoordinatensystem 17
3.4 Sensorkoordinatensystem und Bildkoordinatensystem 18
3.5 Überblick über verschiedene Kameramodelle 20
3.6 Das P3P Problem 28
3.7 Überblick über die Berechnungen des Kalmanfilters 35
3.8 Veranschaulichung der Hand-Eye Kalibrierung 37

4.1 Distanzen zwischen Laserscanner und Punktwolken der vSLAM-Algorithmen . . 43
4.2 Überblick über ROS Nodes und Topics bei LSD-SLAM. 45
4.3 Bildschirmfoto des LSD-SLAM PointCloud Viewer Fensters 45
4.4 Verbesserung der Occupancy Grid Map durch morphologische Operationen . . . 57
4.5 ROS Transformationsbaum des Navigationssystems. 58
4.6 Übersicht über ROS Nodes des Gesamtsystems zur autonomen Navigation 60

5.1 Verwendete Hardware 67
5.2 LSD-SLAM: Karte der Werkhalle des TGZ 69
5.3 LSD-SLAM: Seitliche Ansicht des Keyframegraphs 70
5.4 Verschlechterung der Punktwolke durch ungenaue Keyframeposen (LSD-SLAM) . 70
5.5 Einfluss des Filtern der Punktwolke nach Varianz (LSD-SLAM) 71
5.6 ORB-SLAM: Karte der Werkhalle des TGZ 72
5.7 Vergleich der Punktwolken von LSD-SLAM und ORB-SLAM 72
5.8 ORB-SLAM: Seitliche Ansicht des Keyframegraphs 72
5.9 ORB-SLAM: Kartierung mit frontal ausgerichteter Kamera 73
5.10 Aufbau zur Messreihe 76
5.11 3D-Modell der Kamerahalterung mit 20° Neigung 77
5.12 Histogramme der Positionsfehler über alle Runden 79
5.13 Trajektorie und Positionsfehler eines Durchlaufs mit hoher Genauigkeit 80

5.14 Trajektorie und Positionsfehler eines Durchlaufs mit niedriger Genauigkeit 81
5.15 Verlauf und Abweichung des Orientierungswinkels eines Durchlaufs 83
5.16 Vergleich der Kartierung des vSLAM-Systems mit *gmapping* 85
5.17 Beispielhafter Pfad von Start- zu Zielpose 86
5.18 Histogramme der Distanzen der Endposen zur Zielposition 87
5.19 Wiederholungsgenauigkeit der Endpositionens 88
5.20 Histogramme der Endposen aller Durchläufe 89
5.21 Vergleich der Lokalisierungsmethoden anhand einer quadratischen Trajektorie . . 92
5.22 Vergleich der Lokalisierungsmethoden anhand einer achtförmigen Trajektorie . . 94
5.23 Vergleich der Lokalisierungsmethoden anhand einer kreisförmigen Trajektorie . . 95
5.24 Vergleich der Lokalisierungsmethoden anhand der Messungen aller Durchläufe . . 96

C.1 Durchlauf 1 mit quad. Trajektorie: Fehlerhistogramm und Positionsverlauf 112
C.2 Durchlauf 1 mit quad. Trajektorie: Koordinaten und Orientierungswinkel 113
C.3 Durchlauf 2 mit quad. Trajektorie: Fehlerhistogramm und Positionsverlauf 114
C.4 Durchlauf 2 mit quad. Trajektorie: Koordinaten und Orientierungswinkel 115
C.5 Durchlauf 3 mit quad. Trajektorie: Fehlerhistogramm und Positionsverlauf 116
C.6 Durchlauf 3 mit quad. Trajektorie: Koordinaten und Orientierungswinkel 117
C.7 Durchlauf 4 mit quad. Trajektorie: Fehlerhistogramm und Positionsverlauf 118
C.8 Durchlauf 4 mit quad. Trajektorie: Koordinaten und Orientierungswinkel 119
C.9 Durchlauf 5 mit quad. Trajektorie: Fehlerhistogramm und Positionsverlauf 120
C.10 Durchlauf 5 mit quad. Trajektorie: Koordinaten und Orientierungswinkel 121
C.11 Durchlauf 6 mit quad. Trajektorie: Fehlerhistogramm und Positionsverlauf 122
C.12 Durchlauf 6 mit quad. Trajektorie: Koordinaten und Orientierungswinkel 123
C.13 Durchlauf 1 mit achtf. Trajektorie: Fehlerhistogramm und Positionsverlauf 124
C.14 Durchlauf 1 mit achtf. Trajektorie: Koordinaten und Orientierungswinkel 125
C.15 Durchlauf 2 mit achtf. Trajektorie: Fehlerhistogramm und Positionsverlauf 126
C.16 Durchlauf 2 mit achtf. Trajektorie: Koordinaten und Orientierungswinkel 127
C.17 Durchlauf 3 mit achtf. Trajektorie: Fehlerhistogramm und Positionsverlauf 128
C.18 Durchlauf 3 mit achtf. Trajektorie: Koordinaten und Orientierungswinkel. 129
C.19 Durchlauf 4 mit achtf. Trajektorie: Fehlerhistogramm und Positionsverlauf 130
C.20 Durchlauf 4 mit achtf. Trajektorie: Koordinaten und Orientierungswinkels 131
C.21 Durchlauf 5 mit achtf. Trajektorie: Fehlerhistogramm und Positionsverlauf 132
C.22 Durchlauf 5 mit achtf. Trajektorie: Koordinaten und Orientierungswinkel 133

C.23 Durchlauf 1 mit kreisf. Trajektorie: Fehlerhistogramm und Positionsverlauf . . . 134
C.24 Durchlauf 1 mit kreisf. Trajektorie: Koordinaten und Orientierungswinkel 135
C.25 Durchlauf 2 mit kreisf. Trajektorie: Fehlerhistogramm und Positionsverlauf . . . 136
C.26 Durchlauf 2 mit kreisf. Trajektorie: Koordinaten und Orientierungswinkel 137
C.27 Durchlauf 3 mit kreisf. Trajektorie: Fehlerhistogramm und Positionsverlauf . . . 138
C.28 Durchlauf 3 mit kreisf. Trajektorie: Koordinaten und Orientierungswinkel 139
C.29 Durchlauf 4 mit kreisf. Trajektorie: Fehlerhistogramm und Positionsverlauf . . . 140
C.30 Durchlauf 4 mit kreisf. Trajektorie: Koordinaten und Orientierungswinkel 141
C.31 Durchlauf 5 mit kreisf. Trajektorie: Fehlerhistogramm und Positionsverlauf . . . 142
C.32 Durchlauf 5 mit kreisf. Trajektorie: Koordinaten und Orientierungswinkel 143

Tabellenverzeichnis

4.1 Vergleich der populärsten vSLAM Algorithmen 41

5.1 Angepasste Parameter für LSD-SLAM 68
5.2 Übersicht über die Durchläufe mit acht Punkten zur Skalenkalibrierung 82
5.3 Übersicht über die Durchläufe mit 25 Punkten zur Skalenkalibrierung 82
5.4 Maxima der Positions- und Orientierungsfehler 93
5.5 95. Perzentile der Positions- und Orientierungsfehler 93

A.1 Übersicht über die Durchläufe der Messreihe mit acht Punkten 104
A.2 Übersicht über die Durchläufe der Messreihe mit 25 Punkten 105

B.1 Übersicht über die Durchläufe der Messreihe zur Navigation 108

C.1 Auswertung des visual SLAM-Systems über alle Durchläufe 110
C.2 Auswertung von AMCL über alle Durchläufe 110
C.3 Auswertung der Odometrie über alle Durchläufe 111
C.4 Auswertung des Kalmanfilters über alle Durchläufe 111

Literaturverzeichnis

[1] Arun, K. S. ; Huang, T. S. ; Blostein, S. D.: Least-Squares Fitting of Two 3-D Point Sets. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-9 (1987), Nr. 5, S. 698–700

[2] Bagnell, A. : *Occupancy Maps.* 2014

[3] Bay, H. ; Tuytelaars, T. ; Van Gool, L. : SURF: Speeded up robust features, 2006, S. 404–417

[4] Besl, P. ; McKay, H. : A method for registration of 3-D shapes. IEEE Trans Pattern Anal Mach Intell. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on Robotics* 14 (1992), März, S. 239–256

[5] Bresenham, J. E.: Algorithm for Computer Control of a Digital Plotter. In: *IBM Syst. J.* 4 (1965), März, Nr. 1, S. 25–30

[6] Calonder, M. ; Lepetit, V. ; Strecha, C. ; Fua, P. : BRIEF: Binary Robust Independent Elementary Features, 2010, S. 778–792

[7] Campos, C. ; Elvira, R. ; Rodriguez, J. J. G. ; M. Montiel, J. M. ; D. Tardos, J. : ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial, and Multimap SLAM. In: *IEEE Transactions on Robotics* (2021), S. 1–17

[8] Chen, Y. ; Chen, Y. ; Wang, G. : *Bundle Adjustment Revisited.* 2019

[9] Collins, T. ; Bartoli, A. : Infinitesimal Plane-Based Pose Estimation. In: *International Journal of Computer Vision* 109 (2014), September

[10] Depenthal, C. : *iGPS used as Kinematic Measuring System.* 2010

[11] Elvira, R. ; Tardós, J. D. ; Montiel, J. M. M.: *ORBSLAM-Atlas: a robust and accurate multi-map system.* 2019

[12] Engel, J. ; Sturm, J. ; Cremers, D. : Semi-dense Visual Odometry for a Monocular Camera. In: *2013 IEEE International Conference on Computer Vision*, 2013, S. 1449–1456

[13] Engel, J. ; Schöps, T. ; Cremers, D. : LSD-SLAM: Large-Scale Direct Monocular SLAM. In: *Computer Vision – ECCV 2014.* Cham : Springer International Publishing, 2014, S. 834–849

[14] Ergen, E. ; Akinci, B. ; East, B. ; Kirby, J. : Tracking components and maintenance history within a facility utilizing radio frequency identification technology. In: *ASCE J. Comput. Civil Eng.* 21 (2006), Januar, S. 11–20

[15] Feuerstein, M. : *Hand-Eye Calibration: Robot-Camera-Calibration.* 2009. – `http://campar.in.tum.de/Chair/HandEyeCalibration`; abgerufen am 13. September 2021.

[16] Filipenko, M. ; Afanasyev, I. : Comparison of Various SLAM Systems for Mobile Robot in an Indoor Environment. In: *2018 International Conference on Intelligent Systems (IS)*, 2018, S. 400–407

[17] Fischler, M. ; Bolles, R. : Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In: *Commun. ACM* 24 (1981), S. 381–395

[18] Fox, D. ; Burgard, W. ; Dellaert, F. ; Thrun, S. : Monte Carlo Localization: Efficient Position Estimation for Mobile Robots, 1999, S. 343–349

[19] Förstner, W. ; Wrobel, B. P.: *Photogrammetric Computer Vision.* Braunschweig / Wiesbaden : Springer, Cham, 2016. – ISBN 978–3–319–11549–8

[20] Gálvez-López, D. ; Tardós, J. D.: Bags of Binary Words for Fast Place Recognition in Image Sequences. In: *IEEE Transactions on Robotics* 28 (2012), S. 1188–1197

[21] Gao, X. ; Wang, R. ; Demmel, N. ; Cremers, D. : LDSO: Direct Sparse Odometry with Loop Closure. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, S. 2198–2204

[22] Gao, X.-S. ; Hou, X.-R. ; Tang, J. ; Cheng, H.-F. : Complete Solution Classification for the Perspective-Three-Point Problem. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 25 (2003), September, S. 930– 943

[23] Glover, A. ; Maddern, W. ; Warren, M. ; Reid, S. ; Milford, M. ; Wyeth, G. : OpenFABMAP: An open source toolbox for appearance-based loop closure detection. In: Oh, P. (Hrsg.) ; Tadokoro, S. (Hrsg.) ; Roumeliotis, S. (Hrsg.) ; Kyriakopoulos, K. (Hrsg.): *Proceedings of the 2012 IEEE International Conference on Robotics and Automation.* United States : IEEE, 2012, S. 4730–4735

[24] Grisetti, G. ; Stachniss, C. ; Burgard, W. : Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling. In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005, S. 2432–2437

[25] Grisetti, G. ; Stachniss, C. ; Burgard, W. : Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters. In: *IEEE Transactions on Robotics* 23 (2007), Nr. 1, S. 34–46

[26] Horn, B. K. P.: Closed-form solution of absolute orientation using unit quaternions. In: *J. Opt. Soc. Am. A* 4 (1987), April, Nr. 4, S. 629–642

[27] Kalman, R. E.: A New Approach to Linear Filtering and Prediction Problems. In: *Journal of Basic Engineering* 82 (1960), 03, Nr. 1, S. 35–45

[28] Kohlbrecher, S. ; Stryk, O. von ; Meyer, J. ; Klingauf, U. : A flexible and scalable SLAM system with full 3D motion estimation. In: *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, 2011, S. 155–160

[29] Krul, S. ; Pantos, C. ; Frangulea, M. ; Valente, J. : Visual SLAM for Indoor Livestock and Farming Using a Small Drone with a Monocular Camera: A Feasibility Study. In: *Drones* 5 (2021), Nr. 2

[30] Kümmerle, R. ; Grisetti, G. ; Strasdat, H. ; Konolige, K. ; Burgard, W. : G2o: A general framework for graph optimization. In: *2011 IEEE International Conference on Robotics and Automation* (2011), S. 3607–3613

[31] Lepetit, V. ; Moreno-Noguer, F. ; Fua, P. : EPnP: An Accurate O(n) Solution to the PnP Problem. In: *International Journal of Computer Vision* 81 (2008), S. 155–166

[32] Lowe, D. : Distinctive Image Features from Scale-Invariant Keypoints. In: *International Journal of Computer Vision* 60 (2004), November, S. 91–110

[33] Lowe, D. : Object recognition from local scale-invariant features. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision* Bd. 2, 1999, S. 1150–1157 vol.2

[34] Lu, W. ; Huang, G. Q. ; Li, H. : Scenarios for applying RFID technology in construction project management. In: *Automation in Construction* 20 (2011), Nr. 2, S. 101–106. – Building Information Modeling and Changing Construction Practices

[35] Lu, X. : A Review of Solutions for Perspective-n-Point Problem in Camera Pose Estimation. In: *Journal of Physics: Conference Series* 1087 (2018), September, S. 052009

[36] Metris: *Metris - iGPS*. 2007. – `http://www.metris3d.hu/igps_de_1107.pdf`; abgerufen am 16. September 2021.

[37] Mur-Artal, R. ; Montiel, J. M. M. ; Tardós, J. D.: ORB-SLAM: a Versatile and Accurate Monocular SLAM System. In: *CoRR*

[38] Mur-Artal, R. ; Tardos, J. D.: ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. In: *IEEE Transactions on Robotics* 33 (2017), Oktober, Nr. 5, S. 1255–1262

[39] Olson, E. : AprilTag: A robust and flexible visual fiducial system. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011, S. 3400–3407

[40] OpenCV: *Camera Calibration and 3D Reconstruction*. Website, . – `https://docs.opencv.org/master/d9/d0c/group__calib3d.html`; abgerufen am 06. August 2021.

[41] Rosten, E. ; Drummond, T. : Machine Learning for High-Speed Corner Detection. In: *Proceedings of the 9th European Conference on Computer Vision - Volume Part I*. Berlin, Heidelberg : Springer-Verlag, 2006 (ECCV'06). – ISBN 3540338322, S. 430–443

[42] Rublee, E. ; Rabaud, V. ; Konolige, K. ; Bradski, G. : ORB: An efficient alternative to SIFT or SURF. In: *2011 International Conference on Computer Vision*, 2011, S. 2564–2571

[43] Song, J. ; Haas, C. T. ; Caldas, C. H.: A proximity-based method for locating RFID tagged objects. In: *Adv. Eng. Informatics* 21 (2007), S. 367–376

[44] Stäubli: *Mobiles Robotersystem HelMo*. März 2018

[45] Söderkvist, I. : *Using SVD for some fitting problems*. 2009

[46] Thrun, S. : Particle Filters in Robotics. In: *Proceedings of the 17th Annual Conference on Uncertainty in AI (UAI)*, 2002

[47] Triggs, B. ; Mclauchlan, P. ; Hartley, R. ; Fitzgibbon, A. : Bundle adjustment - A modern synthesis. In: *ICCV '99 Proceedings of the International Workshop on Vision Algorithms: Theory and Practice* (2000), Januar, S. 198–372

[48] Tsai, R. ; Lenz, R. : A new technique for fully autonomous and efficient 3D robotics hand/eye calibration. In: *IEEE Transactions on Robotics and Automation* 5 (1989), Nr. 3, S. 345–358

[49] Wang, J. ; Shahbazi, M. : Mapping Quality Evalutation of Monoocular SLAM Solutinos for Micro Aerial Vehicles. In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-2/W17 (2019), November, S. 413–420

[50] Wang, J. ; Olson, E. : AprilTag 2: Efficient and robust fiducial detection. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016

[51] Xu, L. ; Feng, C. ; Kamat, V. ; Menassa, C. : An Occupancy Grid Mapping enhanced visual SLAM for real-time locating applications in indoor GPS-denied environments. In: *Automation in Construction* 104 (2019), August, S. 230–245

[52] Zhang, Z. : A flexible new technique for camera calibration. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (2000), Nr. 11, S. 1330–1334

[53] Zubizarreta, J. ; Aguinaga, I. ; Montiel, J. M. M.: Direct Sparse Mapping. In: *IEEE Transactions on Robotics* 36 (2020), August, Nr. 4, S. 1363–1370

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Darius Deubert
Würzburg, Dezember 2021