HAW HAMBURG

Bachelorarbeit

David Berschauer

Vergleich und Evaluation LiDAR-basierter 3D-SLAM-Verfahren zur kombinierten Nutzung für Navigation und Pflanzenanalyse in Baumobstkulturen

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

David Berschauer

Vergleich und Evaluation LiDAR-basierter 3D-SLAM-Verfahren zur kombinierten Nutzung für Navigation und Pflanzenanalyse in Baumobstkulturen

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Informatik Technischer Systeme*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Tim Tiedemann
Zweitgutachter: Prof. Dr. Peer Stelldinger

Eingereicht am: 12. Juni 2023

David Berschauer

Thema der Arbeit

Vergleich und Evaluation LiDAR-basierter 3D-SLAM-Verfahren zur kombinierten Nutzung für Navigation und Pflanzenanalyse in Baumobstkulturen

Stichworte

SLAM, LiDAR, Robotik, Kartierung, autonome Navigation, Obstanbau, Obstplantage

Kurzzusammenfassung

Der Einsatz von autonom navigierenden Robotern im Anbau von Baumobstkulturen wird zunehmend nachgefragt. Gleichzeitig steigt der Bedarf an baumbezogenen Informationen wie dem Wachstum, um eine präzise und bedarfsgerechte Bewirtschaftung der Obstplantage zu ermöglichen. Für beide Zwecke wurden bereits unabhängig voneinander erfolgreich LiDAR-Sensoren eingesetzt. In dieser Arbeit wird die Eignung von LiDAR-basierten 3D-SLAM-Verfahren für die Nutzung in beiden Anwendungsfällen untersucht. Dafür werden die frei verfügbaren und ROS-fähigen SLAM-Verfahren LIO-SAM, RTAB-Map und HDL Graph SLAM mit acht für diese Arbeit aufgezeichneten Datensätzen evaluiert und die Ergebnisse verglichen. Die Datensätze wurden in einer typischen Ertragsanlage in der Obstanbauregion Altes-Land erstellt und zeigen Apfelbäume im Winter. Anhand von Evaluationskriterien, die für die Zwecke der Navigation und Pflanzenanalyse ausgewählt wurden, wird gezeigt, dass RTAB-Map unter diesen Bedingungen keine weiterverarbeitbaren Karten erzeugt. Mit LIO-SAM und HDL Graph SLAM entstehen mit ausgewählten Konfigurationen und horizontaler Ausrichtung des LiDAR-Sensors Karten, die die Struktur der Obstanlage widerspiegeln und für die Zwecke der Navigation eingeschränkt nutzbar sind. Für die Analyse von Baummerkmalen wie dem Wachstum sind diese Karten ungeeignet. Es wird gezeigt, dass vor allem die Abweichung der SLAM-basierten Lokalisierung von der Realität ein Problem für diese Zwecke darstellt.

David Berschauer

Title of Thesis

Comparison and evaluation of LiDAR-based 3D-SLAM methods for combined use for navigation and plant analysis in tree fruit crops

Keywords

SLAM, LiDAR, robotics, mapping, autonomous navigation, fruit growing, orchard

Abstract

The use of autonomously navigating robots in the cultivation of tree fruit crops is increasingly in demand. At the same time, the need for tree-related information such as growth is increasing in order to enable precise and needs-based management of the orchard. Independently of each other, LiDAR sensors have already been used successfully for both purposes. In this work, the suitability of LiDAR-based 3D-SLAM methods for use in both applications is examined. For this purpose, the freely available and ROS-capable SLAM methods LIO-SAM, RTAB-Map and HDL Graph SLAM are evaluated with eight datasets recorded for this work and the results are compared. The datasets were created in a typical yield system in the fruit-growing region Altes-Land and show apple trees in winter. Based on evaluation criteria selected for the purposes of navigation and plant analysis, it is shown that RTAB-Map does not produce any further processable maps under these conditions. With LIO-SAM and HDL Graph SLAM, maps, which reflect the structure of the orchard and can be used to a limited extent for navigation purposes, are created with selected configurations and horizontal alignment of the LiDAR sensor. These maps are unsuitable for analyzing tree characteristics such as growth. It is shown that especially the deviation of the SLAM-based localization from reality is a problem for these purposes.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

DOF Freiheitsgrade (englisch: *degree-of-freedom*).

EKF Extended Kalman Filter.

GNSS Globales Navigationssatellitensystem (englisch: *Global Navigation Satellite Systems*).

HAW Hochschule für Angewandte Wissenschaften.

HDL High Definition LiDAR.

HTTP Hypertext Transfer Protocol.

ICP Iterativ nähester Punkt (englisch: *iterative closest point*).

IMU Inertiale Messeinheit (englisch: *Inertial Measurement Unit*).

LiDAR Light Detection and Ranging.

MEMS Mikro-Elektro-Mechanische Systeme (englisch: *Micro-Electro-Mechanical-System*).

NTRIP Networked Transport of RTCM via Internet Protocol.

ROS Robot Operating System.

RTK-GNSS Realtime Kinematic GNSS.

SLAM Simultane Lokalisierung und Kartenerstellung (englisch: *Simultaneous Localization and Mapping*).

ToF Laufzeit (englisch: *Time-of-Flight*).

UTM Universale Transversale Mercator Projektion (englisch: *Universal Transverse Mercator*).

WGS84 World Geodetic System 1984.

1 Einleitung

Im Zuge der Digitalisierung und Automatisierung werden auch in der Landwirtschaft zunehmend Roboter entwickelt und eingesetzt. In Baumobstkulturen wie Äpfeln ist dabei die Erkennung der Bäume selbst als Hindernis und entsprechende Berücksichtigung in der Pfadplanung eine große Herausforderung. Ein Ansatz ist dabei, mit LiDAR-Sensoren dreidimensionale Tiefeninformationen zu erfassen, wobei hoch aufgelöste Daten über die Bäume gewonnen werden. Diese können neben den Informationen über den Baum als Hindernis wertvolle Informationen über die Eigenschaften der Nutzpflanzen enthalten. In dieser Arbeit werden 3D-SLAM-Verfahren verglichen und evaluiert, die aus den Rohdaten der Sensoren dreidimensionale Karten und Positionsschätzungen erstellen. Es wird untersucht, inwieweit diese sowohl für die Navigation, als auch für die Analyse der Bäume genutzt werden können. Zu diesem Zweck werden die Verfahren auf gezielt erstellten Datensätzen von Apfelbäumen angewandt und die Qualität der Karten in Bezug auf Navigation und Baumanalyse eingeschätzt.

1.1 Stand der Technik

In der Landwirtschaft hat sich die RTK-GNSS-basierte Spurführung für Traktoren durchgesetzt, um Ackerflächen, auf ihrem Umriss basierend geplant, Spur an Spur zu bearbeiten. In einigen kommerziellen Anwendungen wie dem AgBot von AgXeed [1] wird damit das vollautomatische Bewirtschaften eines Feldes versprochen. Zur Gewährleistung eines sicheren Betriebs, werden Umfeldsensoren wie LiDAR-Scanner eingesetzt, mit denen Hindernisse erkannt werden. Diese Technik wird zunehmend auch auf den Obst- und Weinbau übertragen und ermöglicht es, zuvor festgelegte Spuren zwischen den Baumreihen abzufahren. Beispiele hierfür sind die für den Obstbau angepassten AgBot Modelle [1] und ab Werk verfügbare Spurführung für die Wein- und Obstbauschlepper von Fendt [4]. In beiden Fällen werden die Fahrspuren im voraus geplant oder durch einmaliges Abfahren der Fläche angelernt.

Die Analyse von einzelnen Bäumen in Baumobstkulturen erfolgt in der Regel manuell durch geschultes Personal und ist sehr aufwendig. Im ESTEBURG Obstbauzentrum Jork werden beispielsweise für Versuchszwecke Triebzuwachs, Kronenvolumen, Baumhöhe und Stammstärke gemessen. Die Daten werden vor allem für die Bewertung von Sorteneigenschaften und Auswirkungen von verschiedenen Anbau- und Erziehungsmaßnahmen verwendet. Im Erwerbsobstbau werden solche Messungen in der Regel aufgrund der Wirtschaftlichkeit nicht durchgeführt. Unter anderem in Bezug auf Wachstumsregulierung, z.B. durch Wurzelschnitt, wird bei der ESTEBURG allerdings ein Potential der baumbezogenen Wachstumsanalyse auch auf Ewerbsbetrieben gesehen. Weltweit gibt es Bestrebungen, diese Baumparameter automatisiert mit Sensorik und Algorithmik zu bestimmen. Mit LiDAR-Sensoren wurde beispielsweise 2010 das Volumen von Bäumen bestimmt [31]. Im Jahr 2012 wurde bereits aufgezeigt, welche Bedeutung die Erfassung der individuellen Baumeigenschaften haben könnte [36]. Dabei wird auch skizziert, wie LiDAR-Scanner an mobilen Robotern die Bäume automatisiert analysieren könnten. An der Washington State University wurde 2019 die 3D-LiDAR-basierte Kronenvermessung in Baumobstkulturen evaluiert [18] und es konnte gezeigt werden, dass die Baumhöhe und das Kronenvolumen mit dieser Technik gut bestimmt werden können. In Deutschland wurde 2019 ein ähnliches Verfahren basierend auf einem 2D-LiDAR vorgestellt [33]. Im Jahr 2021 konnte gezeigt werden, dass Obstbäume auch mit visuellen SLAM-Verfahren [19] kartiert werden können. Der Fokus lag dabei auf dem Pflücken der Früchte, was eine sehr detaillierte Karte der Bäume erfordert. Viele dieser Forschungsansätze wurden 2021 von Ruth et al. zusammengetragen [25].

1.2 Motivation

Die in Abschnitt 1.1 vorgestellten Navigationsansätze erfordern einen zuvor festgelegten Pfad und Umfeldsensorik, um auf Hindernisse und äußere Störeinflüsse reagieren zu können. Durch die aufgrund der Flächenstruktur häufig nicht perfekt rechtwinklig und parallel gepflanzten Baumreihen ist es meistens notwendig, die Fahrspur anzulernen, anstatt diese aus theoretischen Reihenabständen, Vorgewenden etc. zu berechnen. Dieser Prozess ist sehr aufwendig und muss bei jeder Änderung der Obstanlage wiederholt werden. Um auch in unbekanntem Gelände navigieren zu können, haben sich in der Robotik SLAM-Verfahren (siehe Abschnitt 2.1) etabliert, deren Karten und Positionsinformationen von Navigationsalgorithmen (siehe Abschnitt 2.3) verwendet werden. Als Umfeldsensorik kommen häufig LiDAR-Scanner zum Einsatz, die im Verhältnis zum

Preis des Fahrzeuges sehr teuer sind. Es wurde bereits gezeigt, dass diese ebenfalls dazu geeignet sind, die Eigenschaften von Bäumen zu bestimmen.

Es liegt daher nahe, die teuren LiDAR-Scanner an autonomen Fahrzeugen in Baumobstkulturen sowohl für die Navigation, als auch für die Analyse der Einzelbäume zu nutzen. Ferner ist es denkbar, dass mit 3D-SLAM Verfahren Karten erstellt werden können, die sowohl die Navigation mit bekannter Algorithmik ermöglicht, als auch die Struktur jedes Baumes als Teilpunktwolke beinhalten. Bei bekannter Position eines spezifischen Baumes in der SLAM-Karte kann diese Punktwolke extrahiert und die Analyse des einzelnen Baumes auf dieser Basis erfolgen. Würde ein Roboter bei wiederkehrenden Tätigkeiten regelmäßig aktualisierte 3D-Karten der Obstanlagen erstellen, könnten die einzelnen Bäume auch im Erwerbsobstbau gezielt im Jahresverlauf analysiert und individuell behandelt werden. Durch ihre über etwa 20 jährige Standzeit unveränderliche Position, ließen sich zudem andere Informationen wie Pflanzenschutzbehandlungen auf die Einzelbäume anhand ihrer Koordinaten beziehen und in Zusammenhang mit den 3D-Informationen bringen.

Ein weiterer potentieller Vorteil von SLAM-Verfahren besteht darin, Messungen von verschiedenen Positionen um die einzelnen Bäume mit beliebigem Zeitabstand zu fusionieren.

1.3 Zielsetzung

Aufgrund der Funktionsweise von SLAM-Verfahren und den Erkenntnissen des Einsatzes von SLAM-Verfahren in der Robotik, ist anzunehmen, dass bei den üblichen Implementierungen Probleme mit Scan-Matching durch die wiederkehrenden Strukturen der sehr ähnlichen Bäume entstehen. Es ist daher zu Zeigen, ob und mit welchen Einschränkungen SLAM-Verfahren für diese Zwecke eingesetzt werden können. In dieser Arbeit werden dafür übliche 3D-SLAM-Verfahren auf gezielt erstellten Datensätzen von Apfelbäumen angewandt und ihre Ergebnisse verglichen. Die Auswahl ist auf 3D-Verfahren beschränkt, da für die Baumanalyse dreidimensionale Daten notwendig sind. Ziel ist es, aufzuzeigen, wo die Probleme beim Einsatz von SLAM-Verfahren zur Erstellung dreidimensionaler Karten von Obstbäumen liegen, und ob die Qualität für die Weiterverwendung für die Navigation von Robotern und die Analyse der Bäume ausreicht. Außerdem werden die Auswirkungen von verschiedener Ausrichtung des LiDAR-Sensors und verschiedene Fahrgeschwindigkeiten bei der Datenaufnahme verglichen. Aufgrund der geographischen Nähe

der HAW Hamburg zum Obstanbaugebiet Altes-Land und der dort stark überwiegenden Apfelproduktion, fokussiert sich diese Arbeit auf den Kulturapfel (*Malus domestica*) als Baumobstkultur.

1.4 Aufbau der Arbeit

Diese Arbeit teilt sich in sieben Kapitel auf. Das erste Kapitel zeigt, warum der Einsatz von 3D-SLAM-Verfahren in Baumobstkulturen sinnvoll sein könnte und welches Ziel mit dieser Arbeit verfolgt wird. Im zweiten Kapitel werden die notwendigen Grundlagen für die Einordnung und das Verständnis der folgenden Kapitel erläutert. Im dritten Kapitel werden die in dieser Arbeit verglichenen und evaluierten SLAM-Verfahren vorgestellt und ihre Auswahl erklärt. Die Einrichtung der SLAM-Verfahren und Implementierung von Tools wird im vierten Kapitel dokumentiert. In Kapitel fünf wird die Evaluation und der Vergleich der drei 3D-SLAM-Verfahren beschrieben. Dazu wird zunächst auf die Erstellung der Datensätze eingegangen, mit denen die Evaluation durchgeführt wird. Nach Erläuterung der Evaluationskriterien werden die Ergebnisse vorgestellt. Die Diskussion der Ergebnisse und eine Bewertung der Eignung der drei SLAM-Verfahren für die Einsatzzwecke Navigation und Pflanzenanalyse erfolgt in Kapitel sechs. Abschließend wird in Kapitel sieben ein Fazit gezogen und die Ergebnisse dieser Arbeit in Bezug zu zukünftiger Robotik im Obstanbau und Datenverarbeitung im Hinblick auf Präzisionslandwirtschaft (auch bekannt unter dem englischen Begriff *Precision Farming*) gesetzt. Außerdem werden mögliche Lösungsansätze für die in den vorangegangenen Kapiteln aufgedeckten Probleme und Herausforderungen aufgezeigt.

2 Grundlagen

In diesem Kapitel werden die Grundlagen der in Kapitel 3 vorgestellten SLAM-Verfahren und der für die Datenaufnahme in Abschnitt 5.1 verwendeten Sensoren erklärt. Desweiteren werden die in der Robotik etablierten Navigationstechniken in Bezug auf SLAM und mögliche Übertragung auf den Obstanbau erklärt. Damit wird die Grundlage für die Evaluation der SLAM-Verfahren für den Einsatz in Navigationsanwendungen gelegt. Zum besseren Verständnis der Datenaufnahme und der Einordnung der in Kapitel 4 dargestellten Implementierung wird außerdem das Robot Operating System in seinen Grundzügen vorgestellt.

2.1 Simultaneous Localization and Mapping

Simultaneous Localization and Mapping (SLAM) beschäftigt sich nach Siciliano [35] mit dem Problem, dass ein Roboter sich durch eine unbekannte Umgebung mit unbestimmten Bewegungen bewegt. Der Startpunkt ist dabei global bekannt oder wird als Ursprung angesehen. Während seiner Bewegung soll der Roboter seine Umgebung kartieren und seine Position relativ zu dieser Karte bestimmen.

Für die formale Beschreibung von SLAM bezeichnen wir die Zeit mit t und die Position des Roboters x_t. Eine Sequenz von Positionen wird als Pfad bezeichnet und wird mit

$$X_T = \{x_0, x_1, x_2, ..., x_T\}$$

beschrieben. Die Anfangsposition x_0 ist dabei bekannt, die darauf folgenden Positionen können nicht sensorisch erfasst werden. Durch Odometrie kann die Änderung zwischen zwei aufeinander folgenden Positionen erfasst werden. Wir geben mit u_t die Odometrie Information für die Bewegung zwischen $t-1$ und t an. Die relative Bewegung des Roboters

wird somit mit der Sequenz

$$U_T = \{u_1, u_2, u_3, ..., u_T\}$$

beschrieben. Bei einer perfekten Odometrie ließe sich aus U_T auch X_T bestimmen, doch in der echten Welt weicht die Odometrie mit der Zeit immer weiter von der Realität ab. Die wahre Karte der Umgebung des Roboters, die dieser mit seinen Sensoren erfassen kann, wird m bezeichnet. Es wird angenommen, dass diese Karte nicht von der Zeit abhängig ist, sich also nicht ändert. Der Roboter erfasst durch seine Sensoren immer einen Teil z_t von m in seiner unmittelbaren Umgebung. Die Sequenz

$$Z_T = \{z_1, z_2, z_3, ..., z_T\}$$

ist die Abfolge dieser Messungen. Der Zusammenhang zwischen den Variablen ist in Abbildung 2.1 dargestellt. Das SLAM-Problem beschreibt dabei, basierend auf den Odometriedaten U_T und den Messungen der Umwelt Z_T, die Karte m und die Positionen X_T zu ermitteln.

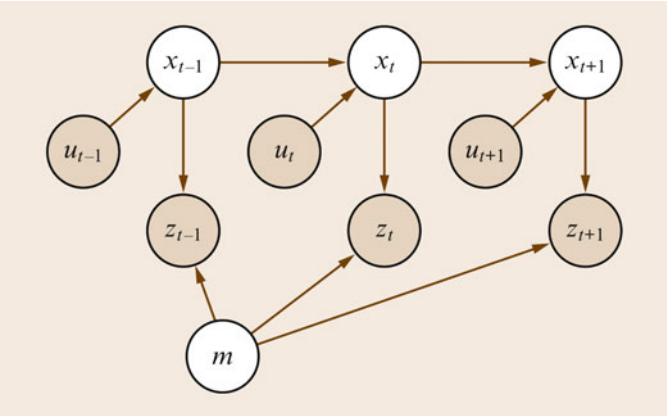


Abbildung 2.1: Graphisches Modell des SLAM-Problems aus [35]. Pfeile geben kausale Zusammenhänge an. Gefärbte Knoten werden vom Roboter erfasst, die weißen Knoten sind unbekannt und sollen bestimmt werden.

Dabei wird zwischen zwei Arten des SLAM-Problems unterschieden. Beim sogenannten vollständigen, oder auch offline, SLAM-Problem geht es darum, den gesamten Pfad und die Karte zu bestimmen:

$$p(X_T, m|Z_T, U_T)$$

Dabei sind Z_T und U_T direkt vom Roboter erfassbar und X_T und m sollen bestimmt werden. Beim online SLAM-Problem geht es darum, nur die aktuellste Position anstatt des gesamten Pfades zu bestimmen:

$$p(x_t, m|Z_T, U_T)$$

Online SLAM-Verfahren arbeiten meist inkrementell.

Zur Lösung des SLAM-Problems benötigt der Roboter ein Modell, welches die Beziehung zwischen der Odometrie u_t und den Positionen x_{t-1} und x_t beschreibt. Außerdem wird ein Modell benötigt, das die Beziehung zwischen den Messungen z_t, der Umgebung m und der Position x_t beschreibt. Dafür gibt es drei verschiedene Ansätze:

Extended Kalman Filter (EKF): Einer der ersten SLAM-Ansätze ist die Verwendung eines EKF. Die Position des Roboters und Merkmale der Umgebung werden dabei als Vektor mit einer zugehörigen Kovarianzmatrix gespeichert. In dieser Matrix wird die Unsicherheit der Schätzungen und der Zusammenhang zu den Positionsschätzungen repräsentiert. Der EKF aktualisiert diese Matrix laufend. Werden neue Merkmale in der Umgebung entdeckt, wird ein neuer Zustand zum Zustandsvektor hinzugefügt und die Kovarianzmatrix wächst quadratisch. Auch wenn diese Technik erfolgreich auf verschiedene SLAM-Probleme angewandt wurde, ist das quadratische Wachsen der Kovarianzmatrix eine starke Limitierung, für die verschiedenste Lösungsansätze gesucht wurden.

Graphbasierte Optimierung: Hierbei wird das SLAM-Problem als Graph repräsentiert. Damit wird üblicherweise die Lösung des vollständigen SLAM-Problems angestrebt. Landmarken und die Positionen des Roboters werden als Knoten und die Beziehung zwischen ihnen als Kanten in einem Graphen modelliert. Graphbasierte SLAM-Verfahren können im Vergleich zu EKF-SLAM wesentlich besser in der Kartengröße skalieren, weil ihr Aktualisierungsaufwand konstant ist. Der Speicheraufwand wächst linear.

Partikelfilter: Zur Lösung des SLAM-Problems können als dritte Variante Partikelfilter eingesetzt werden. Dabei wird eine Menge von Partikeln verwendet, die je gewissermaßen einen Vorschlag zur Repräsentation der Wirklichkeit beinhalten. Jedes Partikel hält also einen wahrscheinlichen Pfad des Roboters und eine Karte mit Merkmalen der Umgebung, die bei jeder neuen Messung überprüft und aktualisiert wird. Da Partikelfilter exponentiell mit der Dimension des Zustandsraumes wachsen, ist Skalierbarkeit bei diesem SLAM-Ansatz ein Problem.

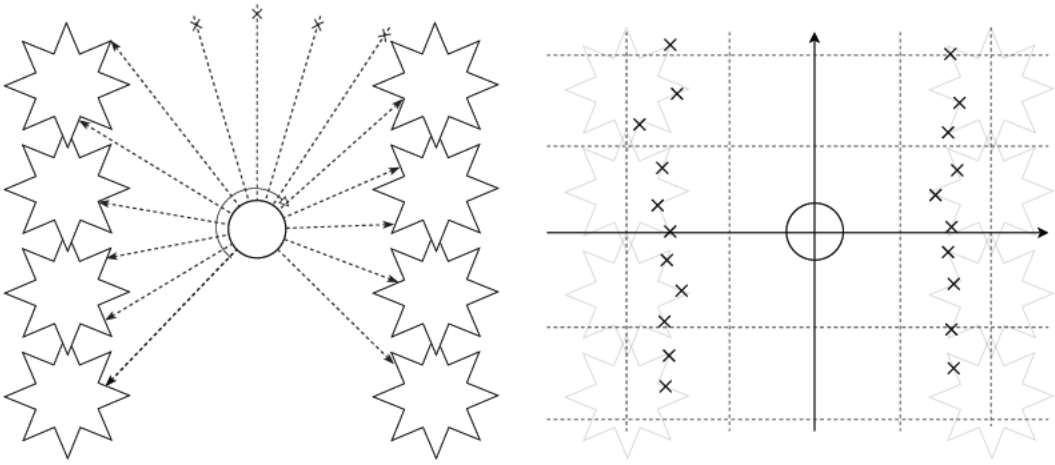
2.2 Verwendete Sensoren

In der Sensorik wird grundsätzlich zwischen exterozeptiven Sensoren und propriozeptiven Sensoren unterschieden. Propriozeptive Sensoren messen dabei den inneren Zustand eines Systems, zum Beispiel eines Roboters. Exterozeptive Sensoren hingegen sammeln Informationen aus der Umgebung [16]. Für die in Abschnitt 5.1 beschriebene Erstellung der Datensätze für diese Arbeit wurden Sensoren verwendet, um die Eigenschaften der Bäume, also der Umgebung, zu erfassen. Daher handelt es sich bei den verwendeten Sensoren überwiegend um exterozeptive Sensoren. Als einziger propriozeptiver Sensor wurde ein Gyroskop als Teil der IMU verwendet, um die Daten der übrigen Sensoren in Bezug zum internen Zustand des Sensoraufbaus setzen zu können.

2.2.1 LiDAR

Light Detection and Ranging (LiDAR) Sensoren sind in der Forstwirtschaft und dem Bauwesen sehr weit verbreitete Sensoren [16]. Auch in der Landwirtschaft werden sie zunehmend eingesetzt. LiDAR-Sensoren messen eine Reihe von Entfernungen zu Objekten in der Umgebung, in dem sie Licht aussenden und wieder empfangen. Anhand des Zeitunterschieds zwischen dem Senden und Empfangen und der bekannten Geschwindigkeit des Lichtes lässt sich die Entfernung zum Reflexionsobjekt berechnen. LiDAR-Sensoren sind daher ein typisches Beispiel für exterozeptive Sensoren. Für die Messung des Zeitunterschiedes gibt es üblicherweise zwei verschiedene Techniken: time-of-flight und phase-shift. Eine dritte Technik basiert auf Triangulation, wobei Sender und Empfänger in einem festen Abstand platziert sind und über den Einfallswinkel der Reflexion der Abstand zum Objekt bestimmt werden kann. Diese Technik wird in der Regel für Abstände von wenigen Metern eingesetzt, weshalb sie weniger in landwirtschaftlichen Anwendungen zu finden ist [16]. Laufzeit (englisch: *Time-of-Flight*) (ToF) LiDAR-Sensoren senden einzelne Lichtimpulse aus und empfangen diese wieder. Durch die eindeutige zeitliche Zuordnung der Impulse kann für jeden Impuls über die Laufzeit der Abstand zum reflektierenden Objekt berechnet werden. Bei der phase-shift Technik (Phasenverschiebung) wird ein kontinuierliches moduliertes Signal ausgesendet [17]. Über die Modulation kann der Zeitunterschied zwischen Senden und Empfangen bestimmt werden. Unabhängig von der Technik zur Zeitmessung werden so zunächst nur einzelne Abstände gemessen. Für das Erfassen der gesamten Umgebung, wird das Licht hintereinander in verschiedene Richtungen ausgesendet. Dieses Vorgehen wird als Scannen bezeichnet, weshalb LiDAR-Sensoren auch häufig

als LiDAR-Scanner bezeichnet werden. Die Summe der Einzelmessungen wird als Scan bezeichnet. Eine weit verbreitete Technik, das Licht gezielt in verschiedene Richtungen zu senden, ist die Verwendung rotierender Spiegel. Das Licht wird dabei von einer im Sensor fixierten Lichtquelle über den Spiegel ausgesendet, am Objekt reflektiert, und über einen zweiten Spiegel wieder auf einen fixierten Empfänger gelenkt. Durch das kontinuierliche Rotieren des Spiegels wird das Licht immer in eine andere Richtung gelenkt und so auch von einem anderen Objekt in anderer Entfernung reflektiert. Dieses Verfahren in 2D ist



(a) Messung einzelner Entfernungen während einer Rotation im Uhrzeigersinn

(b) Ergebnis des Scans. Bäume als Referenz ausgegraut.

Abbildung 2.2: Skizzierte Draufsicht eines Scans eines rotierenden 2D-LiDAR-Sensors zwischen Baumreihen. LiDAR-Sensor im Zentrum

in Abbildung 2.2a dargestellt. Im oberen Bereich ist zu erkennen, wie einige Lichtimpulse nicht reflektiert werden. In diesem Fall kann keine Entfernung gemessen werden. Als Resultat des Scans ergibt sich eine Sammlung von Entfernungen zwischen LiDAR-Sensor und Objekten in der Umgebung und den zugehörigen Winkeln des rotierenden Spiegels zum Zeitpunkt der Messung. Daraus lassen sich Punkte in einem dem LiDAR-Sensor zugeordneten Koordinatensystem berechnen (siehe Abbildung 2.2b). Diese Sammlung von Punkten wird als Punktwolke bezeichnet.

Zur Abdeckung einer dritten Raumdimension werden verschiedene Techniken verwendet. Ähnlich wie beim Scannen in 2D durch einen rotierenden Spiegel, kann dieser oder ein weiterer Spiegel um eine zweite unabhängige Achse rotieren. Die Aktorik dafür sitzt wahlweise innerhalb des Sensorgehäuses oder außerhalb. Insbesondere bei externer Sensorik kann hier üblicherweise nur eine deutlich geringere Drehzahl als um die erste Achse er-

reicht werden, meistens ist sogar nur ein hin und her schwenken aufgrund der Verkabelung möglich. Alternativ werden mehrere Lichtquellen und Empfänger in verschiedenen Ebenen verbaut. Daraus ergeben sich mehrere gleichzeitige Messungen in der ersten Achse und eine feste Winkelauflösung in der zweiten Achse.

2.2.2 IMU

Eine Inertial Measurment Unit (IMU) ist ein Gerät, dass Messgeräte wie ein Gyroskop und Beschleunigungssensoren verwendet, um relative Positionen zu schätzen [35]. Im Folgenden werden daher zunächst die einzelnen Bestandteile von IMUs erklärt, gefolgt von einer Erklärung der Kombination als Gesamtsystem.

Beschleunigungssensoren messen ihre eigene Beschleunigung. Dies schließt die Erdbeschleunigung mit ein, weshalb sie häufig eingesetzt werden, um die Lage relativ zum Schwerefeld der Erde, also relativ zum Erdboden, zu messen. Damit messen sie den Zustand in Relation zur Umwelt und werden als exterozeptive Sensoren eingestuft. Meistens werden drei unabhängige lineare Beschleunigungssensoren für drei orthogonale Raumachsen zu einer Einheit kombiniert [35]. Heutzutage sind die meisten Beschleunigungssensoren als Mikro-Elektro-Mechanische Systeme (englisch: *Micro-Electro-Mechanical-System*) (MEMS) ausgeführt. Dabei wird die Trägheit von sehr kleinen Massen genutzt und die für deren Beschleunigung nötigen Kräfte gemessen.

Gyroskope sind Drehraten-Sensoren. Sie messen als propriozeptive Sensoren die eigene Winkelgeschwindigkeit um eine Achse. Gyroskope sind ebenfalls meistens als MEMS mit drei orthogonalen Achsen ausgeführt. Gyroskope weisen typischerweise einen starken Drift auf, da geringe Drehraten sehr schwierig zu messen sind.

Mit Magnetometern wird die Stärke von Magnetfeldern gemessen. Sie finden häufig in der Messung des Erdmagnetfeldes Anwendung. So lässt sich die Lage des Sensors relativ zum magnetischen Nordpol bestimmen. Auch Magnetometer sind häufig mit 3 orthogonalen Achsen ausgeführt, wobei zur Messung der Feldstärke meistens der Hall-Effekt genutzt wird. Wie auch bei einem Kompass wird die Messung des Erdmagnetfeldes stark von Störeinflüssen wie großen Metallkörpern in der Nähe beeinflusst.

Eine Kombination von Beschleunigungssensor, Gyroskop und Magnetometer mit je drei orthogonalen Achsen besitzt neun Freiheitsgrade (englisch: *degree-of-freedom*) (DOF) und wird als 9-DOF IMU bezeichnet. Über die Integration der Beschleunigung und der

Drehraten ist eine Positionsschätzung in der Position (x, y, z) und Ausrichtung (roll, pitch, yaw) möglich [35]. Über das Magnetometer lassen sich einige Integrationsfehler herausrechnen und die Ausrichtung relativ zum Erdmagnetfeld schätzen.

Im Rahmen dieser Arbeit wurde eine IMU verwendet, um die Bewegungen des Sensoraufbaus zu erfassen und so über die Lage des LiDAR-Sensors zu den Zeitpunkten der einzelnen Entfernungsmessungen die gemessenen Punkte besser im Raum zu verorten. Dieses Vorgehen ist in Anwendungsfällen wie diesem grundsätzlich empfohlen [31].

2.2.3 RTK-GNSS

GNSS sind globale satellitengestützte Navigationssysteme, die in verschiedensten Umgebungen Anwendung finden. GNSS steht dabei als Sammelbegriff für einzelne entsprechende Systeme. Aktuell sind dies GPS (USA), GLONASS (Russland), Galileo (Europa) und Beidou/Compass (China) [5]. Bei diesen Systemen erfolgt die Postionsbestimmung über Laufzeitmessungen zu Satelliten [35]. Vor allem durch Störungen in der Atmosphäre entstehen dabei Fehler [20], die zu einer Positionierungsgenauigkeit von einigen Metern führen können.

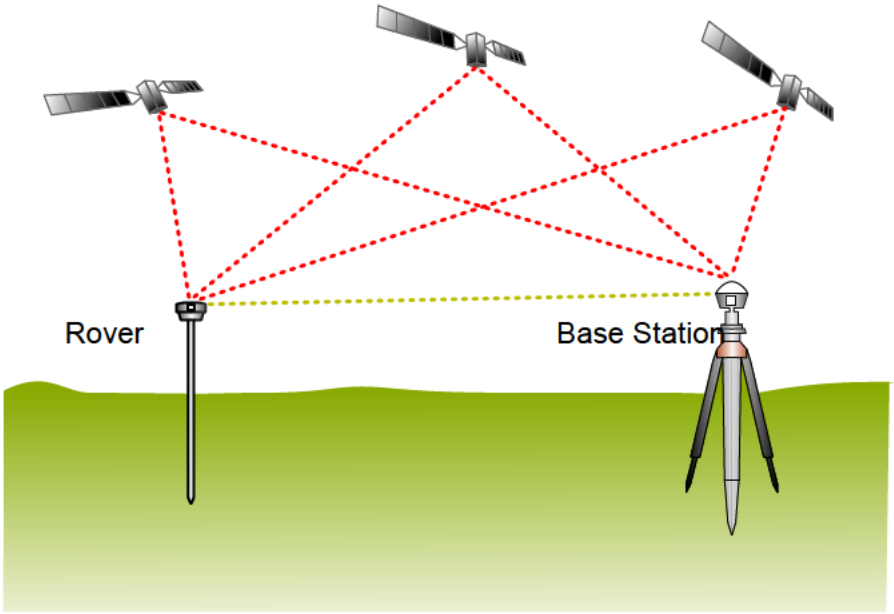


Abbildung 2.3: Beispiel für RTK-GNSS mit Rover und Basestation [11]

Zur Steigerung der Genauigkeit wurden Realtime Kinematic Global Navigation Satellite Systems (RTK-GNSS) [32] als Erweiterung von GNSS entwickelt. Diese Technik beruht auf differentiellem GNSS, wobei zwei unabhängige Empfänger die Signale der gleichen Satelliten empfangen (siehe Abbildung 2.3). Die Position des einen Empfängers, der sogenannten Base Station, wird nicht verändert und in vielen Fällen über externe Vermessungsysteme eindeutig in einem Globalen Koordinatensystem wie dem UTM-System verortet. Dadurch können die bei der Messung entstehenden Fehler je Satellit bestimmt werden und Korrekturdaten berechnet werden, die die Fehler ausgleichen würden. Die Korrekturdaten werden dann in Echtzeit an den zweiten Empfänger, den Rover, geschickt und dort mit den Laufzeitmessungen verrechnet. So wird auch bei günstiger Hardware eine Positionierungsgenauigkeit von wenigen Zentimetern erreicht [23]. Für die Übertragung der Korrekturdaten gibt es einige Standards, die auf drahtlose Übertragung setzen. Sehr weit verbreitet ist der Einsatz von sogenannten NTRIP-Servern, die die Korrekturdaten über HTTP bereitstellen. Im Rahmen dieser Arbeit wurden Korrekturdaten des SAPOS-HEPS-Dienstes vom Land Niedersachsen [12] verwendet. Dabei handelt es sich um einen Netzwerk-RTK-Dienst, das heißt es werden die Korrekturdaten von vielen verschiedenen Base Stations verarbeitet und die für die aktuelle Rover-Position besten Daten gesendet.

Das Ausgabedatenformat von RTK-GNSS Empfängern ist mit dem von GNSS-Empfängern identisch, daher wird im folgenden und bei der Einbindung von GNSS-Sensoren in SLAM-Verfahren nicht zwischen RTK-GNSS und GNSS unterschieden.

2.3 Navigation

Die Navigation ist einer der Kernaufgaben von mobilen Robotern. Sie ist häufig Teil einer höheren Verhaltenssteuerung, um die Ziele des Roboters zu erreichen [35]. Für die Navigation ist es notwendig, die Umgebung zu kennen, sich darin zu lokalisieren und ein Vorgehen basierend darauf zu planen. Dabei gibt es eine große Spanne zwischen der globalen Sicht und der Erreichung eines Zielpunktes und der aktuellen, lokalen Umgebung des Roboters, in dem sich beispielsweise ein unbekanntes Hindernis befinden kann. Ein häufiger Ansatz, die Komplexität der gesamten Navigation aufzuteilen, ist ein gestuftes Verfahren mit einer globalen und einer lokalen Planung [26]. Dieses Vorgehen ist auch als hybride Navigation bekannt und in Abbildung 2.4 dargestellt. Die gesamte Umwelt ist dabei in Form einer Karte repräsentiert, in der eine Zielposition von außen festgelegt

wird. Der Globale Planer berechnet einen Weg zwischen der aktuellen Position des Roboters und der Zielposition. Je nach Kartenart kann hierfür beispielsweise ein Dijkstra

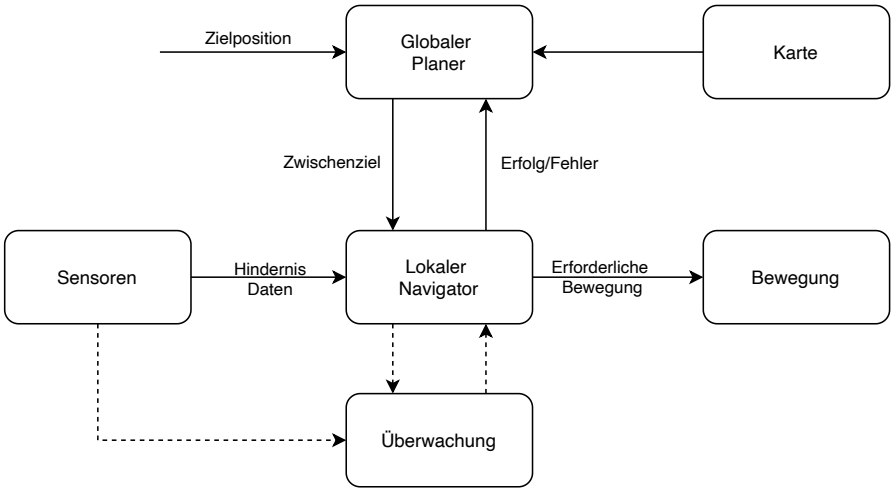


Abbildung 2.4: Bestandteile der hybriden Navigation nach [26]

Algorithmus verwendet werden. Aus der globalen Planung ergeben sich Zwischenziele, die an den lokalen Navigator oder Planer weitergegeben werden. Dieser versucht diese Ziele durch die Ansteuerung des Roboters mit konkreten Bewegungen zu erreichen. Dabei werden stetig die Daten der Umfeldsensorik verarbeitet, um Hindernisse oder ähnliches zu erkennen und diese zum Beispiel zu umfahren. Hierfür kann in einer lokalen Karte beispielsweise ebenfalls mit dem Dijkstra Algorithmus ein Weg gefunden werden. Muss von der global geplanten Route abgewichen werden oder ist ein Weg nicht erreichbar, wird der globale Planer darüber informiert und es kann zum Beispiel eine alternative Route um ein Hindernis herum geplant werden.

Die Navigation ist stark von einer Lokalisierung und der Repräsentation der Umwelt in Form einer Karte abhängig. SLAM-Verfahren als Kombination aus Kartenerstellung und Lokalisierung bilden deshalb häufig die Datengrundlage für Navigationsverfahren. Die SLAM-Karten werden meist in zweidimensionale Belegungskarten (englisch: *occupancy map*) bzw. sogenannte Kostenkarten (englisch: *costmaps*) [8] umgerechnet. Insbesondere gegenüber 3D-Punktwolken sind diese deutlich weniger komplex und erlauben eine schnellere Wegplanung.

2.4 Robot Operating System

Das Robot Operating System (ROS) [9] ist ein Open-Spource Meta-Betriebssystem für Roboter. Es handelt sich also nicht um ein echtes Betriebssystem, sondern um eine Sammlung an Services und Implementierung von häufig genutzter Funktionalität für die Entwicklung von Robotik-Systemen. Als Basis werden meist Linux-Distributionen eingesetzt [24]. ROS stellt Implementierungen für Hardware Abstraktion, Low-Level Gerätesteuerung, Nachrichtenaustausch zwischen Prozessen und Paketverwaltung bereit. ROS gibt es in zwei Hauptversionen, die sich in einigen Grundsätzen unterscheiden. Im Rahmen dieser Arbeit wurde ROS 1 verwendet, weshalb sich alle Ausführungen auf diese Version beziehen.

ROS-Systeme setzten sich nach der ROS-Website [9] hauptsächlich aus folgenden Komponenten und Konzepten zusammen:

- **Nodes**: Nodes sind Prozesse, die Berechnungen durchführen. Als modulares System setzt sich ein Robotiksystem aus einer Vielzahl von Nodes zusammen, die je eigene abgegrenzte Aufgaben übernehmen.
- **Master**: Der ROS-Master stellt einen Namensdienst bereit und ermöglicht es so, dass Nodes sich gegenseitig finden und miteinander kommunizieren können.
- **Parameter Server**: Der Parameter Server speichert Daten basierend auf Schlüsseln an einem zentralen Ort.
- **Messages**: Nodes kommunizieren miteinander durch Nachrichtenaustausch (englisch: *message passing*). Messages sind Datenstrukturen, ähnlich zu Strukturen in der Programmiersprache C.
- **Topics**: Nachrichten (Messages) werden über das Publish/Subscribe Konzept ausgetauscht. Nodes senden Nachrichten, in dem sie sie veröffentlichen (englisch: *publish*). Dabei wird ein Kanal, genannt Topic, angegeben. Zum Empfangen dieser Nachrichten abonnieren (englisch: *subscribe*) Nodes dieses Topic beim Master. Der Master übernimmt dann die notwendigen Schritte, damit jede neue diesem Topic zugehörige Nachricht die entsprechenden Nodes erreicht. Es können beliebig viele Nodes auf dem selben Topic publishen und sich beliebig viele Nodes für ein Topic subscriben. Die Kommunikation zwischen den Nodes ist lose gekoppelt. Topics werden über einen eindeutigen Namen identifiziert, der beim Master registriert wird.

- **Services**: Als Alternative zur Publish/Subscribe-Kommunikation stellt ROS auch eine Request/Response-Kommunikation in Form von Services bereit. Nodes können einen Dienst (englisch: *Service*) unter einem eindeutigen Namen bereitstellen. Andere Nodes schicken Anfragen (englisch: *Requests*) an diesen Service und der bereitstellende Node schickt nach Bearbeitung eine Antwort (englisch: *Response*).
- **Bags**: Rosbags sind ein Datenformat zur Speicherung und Wiedergabe von Nachrichten. Damit lassen sich zum Beispiel Sensordaten als Datensatz aufnehmen und später wieder abspielen, um die Datenverarbeitung und Algorithmen testen und entwickeln zu können, ohne für jeden Durchlauf live Daten erzeugen zu müssen.

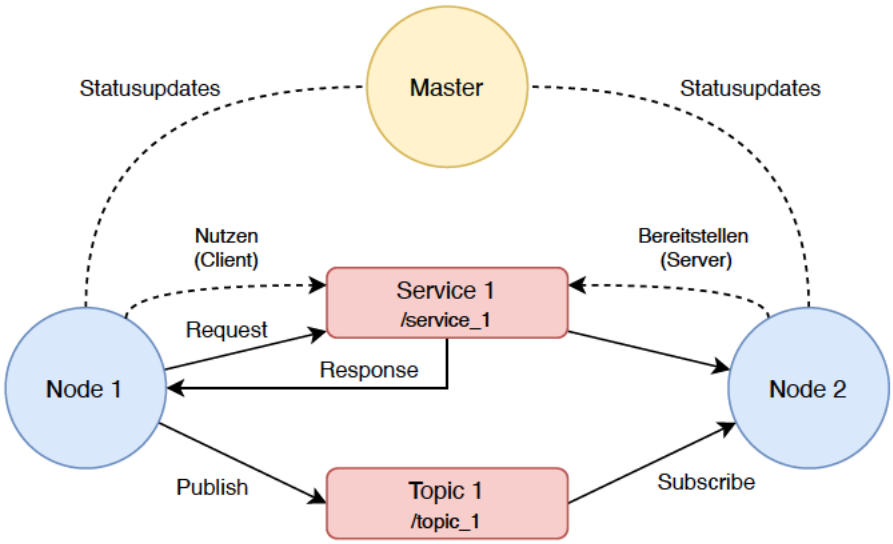


Abbildung 2.5: Blockdiagramm der Kommunikation in ROS in Anlehnung an [24]

In Abbildung 2.5 sind die wichtigsten Komponenten für die Kommunikation in ROS dargestellt. Der Master steht sinnbildlich über den Nodes und überwacht deren Status. Für eine beispielhafte Kommunikation stellt Node 2 als Server den Service *Service 1* mit dem eindeutigen Name *service_1* bereit. Node 1 schickt als Client eine Request an diesen Service und bekommt eine Response zurück. Die Kommunikation zwischen zwei Nodes über ein Topic ist im unteren Bereich dargestellt. Node 1 schickt dabei Nachrichten auf dem Topic *topic_1* und Node 2 hat dieses Topic abonniert.

3 Ausgewählte SLAM-Verfahren

In diesem Kapitel wird erklärt, nach welchen Kriterien SLAM-Verfahren für diese Arbeit ausgewählt wurden. Danach werden die drei Verfahren einzeln Vorgestellt und ihre Besonderheiten in den Kontext der Verwendung in Baumobstkulturen gesetzt.

3.1 Auswahlverfahren

Für den Vergleich und die Evaluation von 3D-SLAM-Verfahren wurden für diese Arbeit drei solcher Verfahren ausgewählt. Viele in der Literatur genannte SLAM-Verfahren wie GMapping [21] und Hector SLAM [27] sind reine 2D-Verfahren und werden in dieser Arbeit nicht behandelt. Für die Auswahl wurden folgende Auswahlkriterien festgelegt:

Freie Verfügbarkeit: das SLAM-Verfahren soll frei verfügbar und nutzbar sein. Lizenzierte und kommerzialisierte Verfahren wurden ausgeschlossen, um den organisatorischen Aufwand dieser Arbeit gering zu halten. Open-Source ist ebenfalls erwünscht, um nötigenfalls Anpassungen auf die Problemstellung vornehmen und das Verfahren im Detail analysieren zu können. Dies gibt außerdem die Möglichkeit, bei identifizierten Problemen in Bezug auf die verwendeten Datensätze in Fortsetzung an diese Arbeit spezifische Lösungen als Anpassung oder Erweiterung des SLAM-Verfahrens zu implementieren.

In ähnlicher Umgebung erprobt: das SLAM-Verfahren sollte in ähnlichen Umgebungen bereits erfolgreich verwendet worden sein. Dafür wurde gezielt nach Papern zu der Anwendung von SLAM-Verfahren in der Landwirtschaft, dem Obst- und dem Weinbau gesucht.

GNSS Integration: sowohl für die Navigation als auch bei der Baumanalyse ist es entscheidend, dass die SLAM-Karte georeferenziert ist. Dadurch wird z.B. die Navigation zu einem im UTM-Koordinatensystem angegebenen Wegpunkt ermöglicht. Dafür wird es als sinnvoll erachtet, dass das SLAM-Verfahren intern GNSS-Messungen

verarbeitet und in der Kartenerstellung berücksichtigt. Alternativ kann zwischen dem UTM-Koordinatensystem und dem Karten-Koordinatensystem ein Stützvektor über den Startpunkt festgelegt und die Ausrichtung der Karte über das Magnetometer der IMU nachjustiert werden. Die zweite Variante wird als ungenauer eingeschätzt.

ROS Unterstützung: das SLAM-Verfahren soll als ROS-Package verfügbar oder aus dem Quellcode zu einem ROS-Package kompilierbar sein. Dadurch wird gewährleistet, dass der mit ROS aufgezeichnete Datensatz (siehe Abschnitt 5.1) verarbeitet werden kann, ohne verfahrensunabhängige Störgrößen wie Timing-Probleme bei der Sensordatenverabeitung und Wiedergabe beachten zu müssen.

Einrichtung mit adäquatem Aufwand: da diese Arbeit einen eingeschränkten Arbeitsumfang besitzt, sollte bei der Einrichtung des SLAM-Verfahrens nach vertretbarem Aufwand eindeutig sein, dass das Verhalten und die Qualität der entstehenden Karten ausschließlich auf das Verfahren selber und nicht auf eine fehlerhafte Konfiguration zurückzuführen ist. Es soll vermieden werden, einzelne SLAM-Verfahren aufgrund eines Einrichtungsfehlers als ungeeignet für die in dieser Arbeit behandelten Problemstellung einzuschätzen.

Diese Kriterien schränken die Auswahl der SLAM-Verfahren teilweise stark ein und sind überwiegend sehr weich formuliert. Die daraus resultierende Auswahl wird nicht als allgemeingültig oder vollständig für die behandelte Problemstellung betrachtet. Für diese Arbeit wurden neben den drei ausgewählten auch andere Verfahren wie die 3D Variante von Cartographer [3] betrachtet und ausprobiert, es konnten jedoch keine wiederholbaren Ergebnisse ohne verfahrensinterne Fehlermeldungen erreicht werden. Dieses Verfahren wird daher in den nachfolgenden Kapiteln nicht weiter behandelt. Es ist daher anzunehmen, dass weitere 3D-SLAM-Verfahren für den geplanten Einsatz sehr gut geeignet sind, auch wenn hier kein Vergleich zu ihnen gezogen werden kann.

3.2 LIO-SAM

LIO-SAM ist ein graphbasiertes 3D-SLAM-Verfahren mit eng gekoppelter LiDAR- und Intertial-Odometrie durch Glättung und Kartierung (englisch: *tightly-coupled **L**iDAR **i**nertial **o**dometry via **s**moothing **a**nd **m**apping*) [34]. Es ist unter der BSD-3 Open-Source Lizenz verfügbar und als ROS-Package implementiert [7]. Neben den Hauptsensoren 3D-LiDAR und 9DOF-IMU werden auch GNSS-Sensoren direkt eingebunden, um

über GNSS-Odometrie die Kartierung zu verbessern und die Karte global zu referenzieren. Über die IMU-Daten wird die Bewegung des LiDAR-Sensors während des Scans geschätzt. Darauf basierend werden die einzelnen Entfernungsmessungen des LiDAR-Sensors über ihren individuellen Zeitstempel relativ zu der entsprechend geschätzten Position des LiDAR korrigiert. Dieser Prozess wird als *deskewing* bezeichnet. LiDAR-Odometrie wird über den ICP-Ansatz berechnet. Die Besonderheit von LIO-SAM ist die Verwendung von Schlüsselbildern (englisch: *keyframes*) und eines Schiebefensters (englisch: *sliding window*). Diese Techniken sind aus visuellen SLAM-Verfahren bekannt. Dies ermöglicht die Scan-Registrierung auf einem Ausschnitt der Karte anstatt der gesamten globalen Karte, was den Rechenaufwand reduziert und die Echtzeitfähigkeit steigert.

LIO-SAM wurde sehr erfolgreich in einer simulierten Umgebung aus dem Weinbau angewandt [22]. Auch in vielen anderen Umgebungen wie Städten und Parks wurde es erfolgreich eingesetzt [30].

Die Einrichtung von LIO-SAM verlief problemlos (siehe Abschnitt 4.1) und es konnten schnell erste Karten erstellt werden. Insgesamt erfüllt LIO-SAM damit die Auswahlkriterien für den Vergleich sehr gut. Insbesondere die GNSS-Integration ist hier positiv hervorzuheben.

3.3 RTAB-Map

RTAB-Map ist ein graphbasiertes SLAM-Verfahren, dass ursprünglich als visuelles Verfahren entwickelt wurde [29]. Der Name steht dabei für Echtzeit Aussehen-basierte Kartierung (englisch: ***R**eal-**T**ime **A**ppearance **B**ased **Map**ping*). Als Sensoren können Tiefenkameras (RGB-D), Stereokameras und LiDAR-Sensoren verwendet werden. Im Rahmen dieser Arbeit wurde ausschließlich ein LiDAR-Sensor verwendet. RTAB-Map setzt sich aus Schleifenschlusserkennung (englisch: *loop closure detection*) und Graphoptimierung zusammen. RTAB-Map ist unter einer eigenen Open-Source-Lizenz [10] und als vorkompiliertes ROS-Package verfügbar.

RTAB-Map wurde sehr erfolgreich mit simulierten Daten eines 3D-LiDAR zwischen Weinreben verwendet [22]. Zukünftig könnte die Kombination mit Stereokameras für die Anwendung in Baumobstkulturen interessant werden. Eine erfolgreiche Sensorfusion über dieses SLAM-Verfahren könnte vor allem für die Baumanalyse eine sehr gute Datengrundlage liefern.

Die Einrichtung von RTAB-Map verlief ohne Probleme (siehe Abschnitt 4.1). RTAB-Map erfüllt die Auswahlkriterien gut. Positiv hervorzuheben ist die Erzeugung von 2D-Occupancy-Grids, die direkt für die Navigation verwendet werden können. GNSS-Sensoren werden allerdings nicht direkt unterstützt.

3.4 HDL Graph SLAM

HDL Graph SLAM ist ein Echtzeit 6DOF-SLAM-Verfahren für 3D-LiDAR-Sensoren [28]. Das graphbasierte Verfahren verwendet Normal Distributions Transform (NDT) Scan Matching für LiDAR-Odometrie und Schleifenschlusserkennung (englisch: *loop closure detection*). Im Graphen können verschiedene Einschränkungen wie GNSS-Messungen, IMU-Lage und Ausrichtung sowie eine punktwolkenbasierte Bodenerkennung repräsentiert werden. Über die GNSS-Daten ist die entstehende Karte georeferenziert. HDL Graph SLAM ist unter der BSD-2 Open-Source Lizenz verfügbar und als ROS-Package implementiert [6].

4 Implementierung

In diesem Kapitel wird die Einrichtung der SLAM-Verfahren in einer ROS-Umgebung und die Implementierung eigener Werkzeuge für die Unterstützung der Evaluation beschrieben.

4.1 Einrichtung der SLAM Verfahren

Die drei ausgewählten SLAM-Verfahren wurden in einer ROS Noetic Umgebung unter Ubuntu 20.04 eingerichtet. Die Installation der Abhängigkeiten und das Kompilieren des Quellcodes bei LIO-SAM und HDL Graph SLAM ist sehr einfach. Für die einzelnen SLAM-Verfahren wurden ROS launch-Files erstellt, mit denen diese einfach gestartet werden können. Die Konfigurationen der Verfahren wurden entsprechend der Datensätze angepasst, sodass die ROS Topics der Sensoren richtig abonniert werden. Außerdem wurde die zuvor geschätzte magnetische Ablenkung durch den Traktor und den Aufbau in den dafür vorgesehenen Konfigurationen eingefügt.

Bei der Analyse des Quellcodes von LIO-SAM wurde festgestellt, dass es eine willkürliche Limitierung für das Einfügen von Knoten, die die GNSS-basierte Lokalisierung repräsentieren, in den Graphen gibt. Hier werden nur neue Messungen erlaubt, wenn die neue mehr als fünf Meter von der vorherigen Messung entfernt liegt. Diese Limitierung wurde aus dem Quellcode entfernt, um die Gewichtung der hochgenauen RTK-GNSS-basierten Lokalisierung nicht zu limitieren.

Außerdem wurde der Sensoraufbau als URDF-Datei modelliert, sodass zur Laufzeit ein entsprechender Transformationsbaum (englisch: *tf tree*) in ROS vorliegt (siehe Abbildung 4.1).

Die SLAM-Verfahren veröffentlichen selber die nötigen Transformationen zwischen dem Karten-Koordinatensystem (*map frame*) und dem Odometrie-Koordinatensystem (*odom*

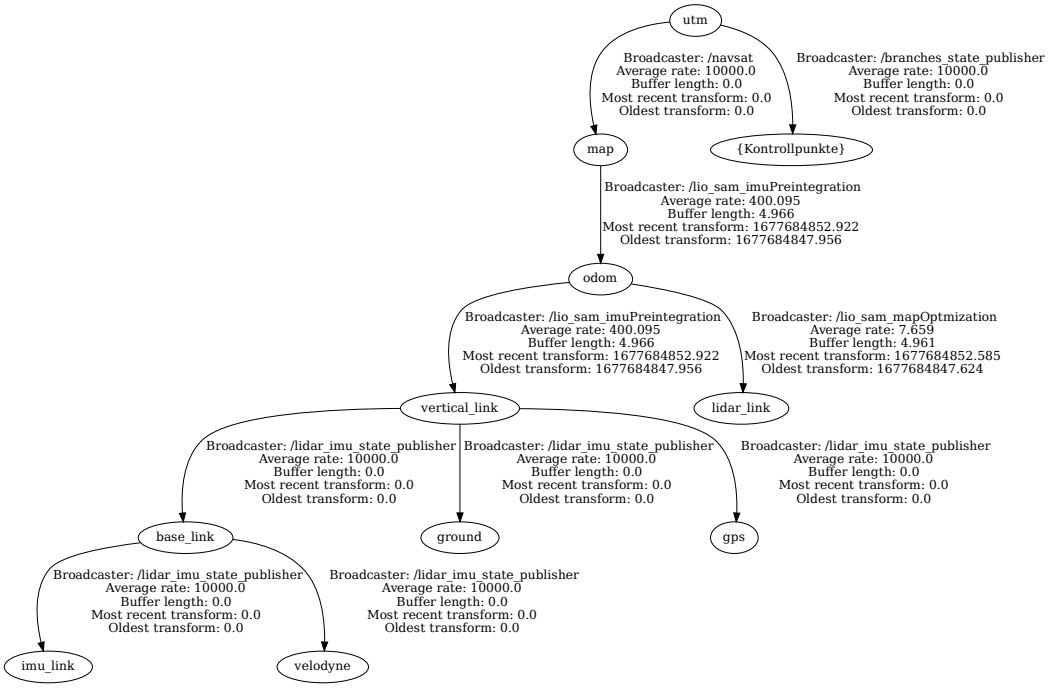


Abbildung 4.1: Transformationsbaum bei LIO-SAM

frame). Außerdem stellen sie die Transformation zwischen dem *odom frame* und der Basis des Sensoraufbaus bereit, in diesem Fall mit *vertikal link* bezeichnet. Dieser repräsentiert das vertikale Aluminiumprofil im Aufbau (siehe Abbildung 5.1). Im Falle von LIO-SAM werden diese Transformationen beispielsweise von dem Node *lio_ sam_ imuPreintegration* veröffentlicht.

Die Transformation zwischen dem globalen UTM-Koordinatensystem und dem Koordinatensystem der Karte wird bei LIO-SAM durch einen EKF-basierten Node bereitgestellt, der die Daten des GNSS-Empfängers und der IMU verarbeitet. Da RTAB-Map und HDL Graph SLAM diese Transformation in der Standardkonfiguration nicht bereitstellen, wurde die Implementierung von LIO-SAM für diese Verfahren kopiert. Dabei wird nur die Transformation für die Georeferenzierung der Karte für Analysezwecke in dieser Arbeit verwendet und die GNSS-basierte Odometrie, die bei LIO-SAM in die Graphoptimierung mit einfließt, in den SLAM-Verfahren ignoriert.

4.2 Zuschneiden von Punktwolken

Zur Evaluation der Punktwolkenqualität in Bezug auf die Baumanalyse wurde ein Tool entwickelt, mit dem globale Punktwolken, die Ausgabe von 3D-SLAM-Verfahren, zugeschnitten werden können. Dieses wurde als ROS-Node in Python implementiert und stellt zwei ROS-Services zur Verfügung. Der Node abonniert ein konfigurierbares Topic vom Typ *sensor_msgs::PointCloud2* und speichert die hier gesendeten Punktwolken zwischen.

Beim Aufruf des Services */pointcloud_processing/cut_lat_lon_circle_pcd* werden Zielkoordinaten im WGS84-Koordinatensystem, ein Radius in Metern und ein Dateipfad angegeben. Die Zielkoordinaten werden daraufhin über das UTM-Koordinatensystem in das Koordinatensystem der Punktwolke transformiert und die Punktwolke an dieser Position Zylinderförmig mit dem angegebenen Radius zugeschnitten. Der Zuschnitt beschränkt sich dabei auf die X- und Y-Achse, in der Z-Achse bleibt die volle Höhe der ursprünglichen Punktwolke erhalten. Der Ursprung der dadurch entstehenden Punktwolke wird auf die Zielkoordinaten geschoben. Abschließend wird die Punktwolke im pcd-Format an den angegebenen Dateipfad exportiert.

Der Service */pointcloud_processing/cut_location_circle_pcd* stellt eine sehr ähnliche Funktionalität bereit. Statt der Koordinaten muss hierbei allerdings ein Name angegeben werden. Hierfür lassen sich über ROS-Parameter benannte Koordinaten festlegen, sodass beim Aufruf des Services die WGS84-Koordinaten über den Namen abgerufen werden können. Dies erleichtert die häufige Verwendung der selben Koordinaten, wie im Fall der Analyse der einzelnen Äste im Abschnitt 5.2.

4.3 Odometriedifferenz Protokollierung

Zur Evaluation der Odometrie der SLAM-Verfahren wurde ein Tool entwickelt, mit dem sich der Abstand zweier Odometriequellen berechnen und aufzeichnen lässt. Dieses wurde als ROS-Node in Python implementiert und abonniert zwei konfigurierbare Odometrie-Topics. Das eine wird dabei als Referenz und das andere als zu überwachendes betrachtet. Bei jeder neuen Nachricht des zweiten Odometrie-Topics wird der Abstand zur aktuellsten zwischengespeicherten Referenzodometrie berechnet und mit einem Zeitstempel gespeichert. Es kann dabei zwischen einem 2D-Abstand, also in X und Y Achse, und einem

3D-Abstand gewählt werden. Die entstehende Zeitreihe wird als CSV-Datei exportiert und zusätzlich als Diagramm im PNG-Format abgespeichert.

5 Evaluation

Das Hauptziel dieser Arbeit ist die Evaluation der ausgewählten 3D-SLAM-Verfahren, um ihre Eignung für die Navigation und Pflanzenanalyse in Baumobstkulturen bewerten zu können. Dafür wurden Datensätze zwischen Apfelbäumen als ausgewählte Baumobstkultur aufgezeichnet. Diese wurden mit den SLAM-Verfahren verarbeitet und die resultierenden Positionsdaten und Punktwolken anhand der in Abschnitt 5.2 festgelegten Kriterien ausgewertet.

5.1 Datensätze

Für die Evaluation in dieser Arbeit wurden Datensätze von Apfelbäumen aufgenommen. Der Fokus lag dabei darauf, die Daten so aufzunehmen, als würden sie von einem Roboter oder für den Dauereinsatz an einem Standardtraktor konzipierten Sensoraufbau aufgezeichnet. Der in Abschnitt 5.1.1 beschriebene Aufbau wurde in Zusammenarbeit mit dem Fraunhofer IFAM erstellt und an die vom IFAM entwickelte SensorBox mit der nötigen Rechenhardware angeschlossen. Der Aufbau wurde an einem Traktor im Frontanbau montiert und durch die in Abschnitt 5.1.2 beschriebene Obstanlage gefahren. Dies wurde mehrfach mit den verschiedenen in Abschnitt 5.1.3 beschriebenen Modalitäten durchgeführt. Alle Sensoren wurden in ROS integriert und ihre Daten als Rosbag (siehe Abschnitt 2.4) aufgezeichnet.

5.1.1 Sensoraufbau

Der Sensoraufbau besteht aus einem LiDAR-Sensor, einer IMU und einem RTK-GNSS-Empfänger (siehe Abbildung 5.1). Der LiDAR-Sensor und die IMU sind in einem festen Abstand von 12,5 cm auf einem Aluminiumprofil befestigt. Dieses ist über ein verstellbares Gelenk mit einem vertikalen Profil verbunden. Darüber lässt sich die Neigung des

LiDAR-Sensors zusammen mit der IMU relativ zum Boden einstellen (siehe Abbildungen 5.2a und 5.2b). Dieser Aufbau vereinfacht die Konfiguration der SLAM-Verfahren, da meist eine Parallelität der Koordinatensysteme von IMU und LiDAR angenommen wird.

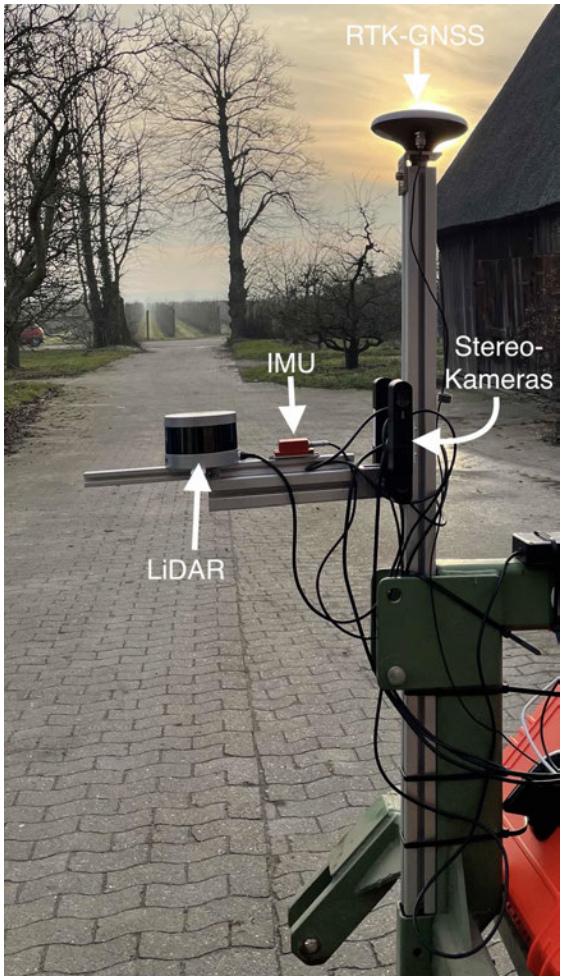


Abbildung 5.1: Sensoraufbau, befestigt an einem Anbaurahmen, der an beliebigen Traktoren aufgenommen werden kann. Horizontale Ausrichtung des LiDAR-Sensors.

In horizontaler Ausrichtung des LiDAR-Sensors, wie in Abbildung 5.1 gezeigt, befindet sich dieser etwa 130 cm über dem Boden. Am oberen Ende des vertikalen Profils ist eine RTK-GNSS-Antenne angebracht, die an einen RTK-GNSS-Empfänger in der SensorBox (unten rechts in Abbildung 5.1) vom Fraunhofer IFAM angeschlossen ist. Die Antenne

(a) 30° nach unten (b) vertikal

Abbildung 5.2: Sensoraufbau im Feld mit verschiedenen Ausrichtungen des LiDAR-Sensors, befestigt an einem Traktor

befindet sich bei horizontaler Ausrichtung des LiDAR-Sensors etwa 50 cm höher als der LiDAR-Sensor.

Zusätzlich befinden sich an dem Aufbau auf Höhe des Gelenks am vertikalen Profil zwei Stereokameras, die in Fahrtrichtung links und rechts ausgerichtet sind. Die Daten dieser Kameras wurden im Rahmen dieser Arbeit nicht ausgewertet, können aber in weiterer Forschung für visuelle SLAM-Verfahren oder Sensorfusion genutzt werden. Auch eine Baumanalyse auf Basis der Kamerabilder und Kombination mit den LiDAR-Messungen wird dadurch möglich.

Als LiDAR-Sensor wurde ein Velodyne Puck (VLP-16) [14] verwendet. Dabei handelt es sich um einen rotierenden ToF-LiDAR-Sensor mit 16 Ebenen. In der rotierenden Achse werden 360° abgedeckt, in der zweiten Achse 30° mit 1,875° Schritten zwischen den Ebenen. Um die relativ zum LiDAR-Sensor statischen Objekte wie die Kameras und den Traktor aus den Scans auszublenden, wurden die hinteren 90° des Sichtfeldes in der Software ausgeblendet. Der ROS-Treiber erlaubt die einfache Integration in ROS und veröffentlicht mit der Scanrate von 10 Hz die Scans als *sensor_msgs::PointCloud2*.

Die verwendete IMU ist das Modell MTi-30 von Xsens [15]. Die 9DOF-IMU liefert mit einer Frequenz von 400 Hz Beschleunigungs-, Gyroskop- und Magnetometerwerte über einen ROS-Treiber als *sensor_ msgs::Imu* Nachrichten. Die Werte des Magnetometers sind stark durch die Metallstruktur des Anbaurahmens und durch den Traktor beeinflusst. Die magnetische Ablenkung wird mit -0.46 Radiant abgeschätzt.

Bei dem verwendeten RTK-GNSS-Empfänger handelt es sich um einen u-blox ZED-F9P [13]. Als externe Antenne wurde eine Antenne von Ardusimple verwendet [2]. Diese Kombination verwendet die Frequenzbänder L1/L2/E5b und erreicht eine Positionierungsgenauigkeit von wenigen Zentimetern [23].

5.1.2 Eigenschaften der Obstanlage

Die Datensätze wurden alle in der selben Obstanlage aufgenommen. Die Bäume der Apfelsorte Elstar befinden sich im neunten Standjahr. Der Reihenabstand, also der Abstand zwischen zwei Baumreihen, beträgt 3,4 Meter. Der Baumabstand, also der Abstand zwischen den Bäumen innerhalb einer Reihe beträgt 1,2 Meter. Die Länge der Reihen beträgt etwa 207 Meter. Die Reihen sind mit minimalen Abweichungen parallel. Wie in Abbildung 5.3 zu sehen, wurden die Datensätze während der Winterruhe aufgezeichnet. Die Bäume sind entsprechend unbelaubt und auch die Knospen sind noch nicht aufgebrochen.

Abbildung 5.3: Unbelaubte Obstanlage, Sensoraufbau rechts

Die Baumhöhe beträgt etwa 3,5 Meter, wodurch der LiDAR-Sensor des Sensoraufbaus etwas niedriger als in der Mitte der Bäume entlang bewegt wird. So werden die Baumspitzen etwas schlechter als der untere Bereich der Bäume gescant, diese Gewichtung entspricht aber in etwa der Ertragsrelevanz der Baumregionen.

5.1.3 Modalitäten

Insgesamt wurden acht Datensätze mit verschiedenen Modalitäten aufgenommen. Bei den Aufnahmen wurden folgende Parameter verändert:

Ausrichtung des LiDAR-Sensors: Der LiDAR-Sensor wurde in drei verschiedenen Ausrichtungen befestigt:

- Die Horizontale Ausrichtung (siehe Abbildung 5.1) entspricht einer gewöhnlichen Ausrichtung an autonomen Robotern wie dem AgBot [1]. In dieser Ausrichtung wird die maximale horizontale Reichweite in alle Richtungen erreicht. Insbesondere orthogonal zur Fahrtrichtung wird jedoch eine sehr geringe Höhe der Bäume abgedeckt. Mit dem verwendeten Velodyne Puck und dem Reihenabstand von 3,4 m ergibt sich ein Sichtfeld von etwa 90 cm in der Baummitte. Durch die Höhe des Sensors entspricht dies des Bereiches zwischen 85 cm und 175 cm über dem Boden.
- Die vertikale Ausrichtung (siehe Abbildung 5.2b) mit einem um 90° nach unten geneigten Sensor erreicht die vollständige Abdeckung der Baumhöhe in unmittelbarer Nähe zum Aufbau. Diese Ausrichtung ist angelehnt an die Aufbauten von [31] und [33]. Das Horizontale Sichtfeld beträgt bei dem verwendeten Velodyne Puck jeweils orthogonal zur Fahrtrichtung links und rechts ±15°. Für die Hinderniserkennung ist diese Ausrichtung also ungeeignet und es ist fraglich, ob so auch eine kombinierte Nutzung zur Pflanzenanalyse und Navigation erreicht werden kann, da die Hinderniserkennung durch Umfeldsensorik als essentiell für die Navigation eingestuft werden kann. In dieser Arbeit wird diese Ausrichtung trotzdem untersucht, da mit ihr in der Literatur die besten Ergebnisse für die Pflanzenanalyse erzielt wurden. Es soll gezeigt werden, ob die Verwendung des selben Sensors mit einer für diesen Zweck optimierten Ausrichtung zu einem so großen Unterschied in der Qualität der Messergebnisse führt, dass die Verwendung dedizierter Sensorik für die einzelnen Zwecke gerechtfertigt werden kann.

- Als Kompromiss zwischen der horizontalen und vertikalen Ausrichtung wurde eine 30° nach unten geneigte Ausrichtung gewählt. Diese erreicht bei dem verwendeten Velodyne Puck in Fahrtrichtung für die Hinderniserkennung eine theoretische Sichtweite von etwa 5 Metern in der obersten Ebene. Diese wird als untere Grenze der Sichtweite von Robotern mit niedriger Geschwindigkeit eingeschätzt. In etwa 2 Meter horizontaler Entfernung zum Sensor erreicht die unterste Ebene den Boden. Für die Hinderniserkennung wird also der Bereich zwischen 2 und 5 Metern vor dem Roboter ideal abgedeckt. Der Bereich hinter dem Roboter wird jedoch nicht erfasst. Außerdem wird so in geringer Entfernung zum Sensor die gesamte Baumhöhe erfasst, der untere Bereich in Fahrtrichtung und der obere Bereich entgegen der Fahrtrichtung. Diese Ausrichtung wird daher als guter Kompromiss zwischen den Zwecken der Navigation und der Baumanalyse angesehen. Für eine Hinderniserkennung nach hinten, wäre ein zweiter LiDAR-Sensor notwendig, der wiederum gute Daten für die Baumanalyse liefern könnte.

Vor und nach dem Entfernen einzelner Äste: Um den Detailgrad der durch die SLAM-Verfahren erstellten Punktwolken zu evaluieren, wurden in der Obstanlage fünf Äste ausgewählt, markiert und herausgeschnitten. Einige Datensätze wurden nach dem Markieren und vor dem Entfernen aufgenommen, einige andere nach dem Entfernen. Die markierten Äste sind in Abbildung 5.4 zu sehen.

(a) Ast 1

(b) Ast 2

(c) Ast 3

(d) Ast 4

(e) Ast 5

Abbildung 5.4: Ausgewählte Äste vor dem Entfernen

Die Äste haben an der Schnittstelle am Stamm einen Durchmesser von 1,5 bis 3 cm und sind zwischen 80 und 120 cm lang. Die Äste befinden sich etwa mittig in den Baumreihen 1 und 2 (siehe Abbildung 5.5). Bei einer idealen Kartierung sollte der Unterschied zwischen den Punktwolken vor und nach dem Entfernen exakt den abgeschnittenen Ästen entsprechen.

Fahrgeschwindigkeit: die meisten Datensätze wurden mit einer Fahrgeschwindigkeit zwischen 3 und 4 km/h aufgenommen. Diese Geschwindigkeit sollte von schnelleren Robotern in Obstanlagen erreicht werden können. Diese Geschwindigkeit ist ein guter Kompromiss zwischen Zeitaufwand für die Erstellung der Datensätze, Schonung der Sensorik vor Schlaglöchern und realistischen Bedingungen für den Dauerbetrieb. Für Pflegearbeiten wie der mechanischen Baumstreifenpflege beträgt die Geschwindigkeit ca. 2 km/h. Diese Geschwindigkeit ist außerdem realistisch für Roboter mit niedriger Leistung. Für die meisten Arbeiten mit Fahrzeugen in der Obstanlage wie dem Pflanzenschutz beträgt die Fahrgeschwindigkeit etwa 7 km/h. Hier wird ein größerer Einfluss der Verarbeitung von Messzeitpunkten bei SLAM-Verfahren erwartet, da beispielsweise während eines vollständigen LiDAR-Scans bei 10 Hz fast 20 cm zurückgelegt werden.

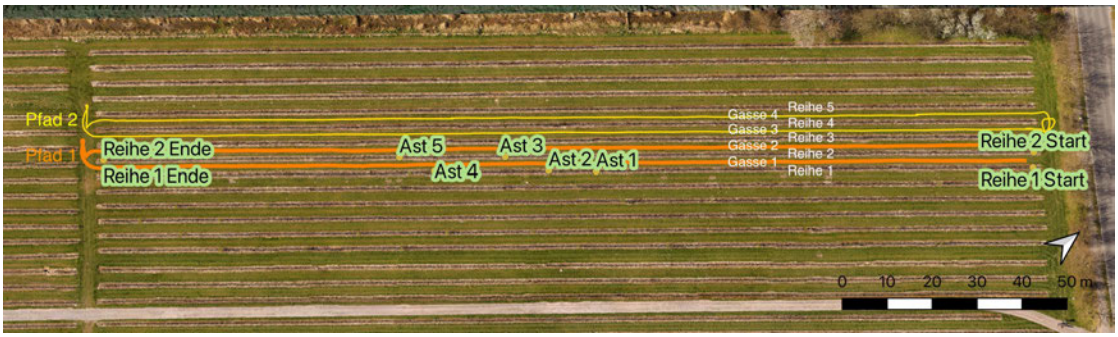


Abbildung 5.5: Karte Versuchsanlage

Aufgrund der nassen Witterung vor der Datenaufnahme und der daraus resultierenden schlechten Befahrbarkeit wurden die Datensätze nicht alle auf dem selben Pfad aufgenommen. Wie in Tabelle 5.1 zu erkennen, wurden die meisten Datensätze auf Pfad 1 aufgenommen. Dieser führt, wie in Abbildung 5.5 zu erkennen, zwischen den Baumreihen 1 und 2, also in Gasse 1, nach Südwesten. Nach einem Wendemanöver führt dieser Pfad zwischen Baumreihe 2 und 3, also in Gasse 2, wieder nach Nordosten in Richtung des Startpunktes. Die sechs Durchfahrten dieser Gassen waren gegeneinander um einige Zentimeter versetzt, um die Grasnabe zu schonen. Die Datensätze H1, H2, 30_1, 30_2,

V1 und V2 (siehe Tabelle 5.1) führen also entlang der selben Baumreihen, allerdings nicht an den exakt selben Positionen.

Kürzel	LiDAR Ausrichtung	Fahrgeschwindigkeit	Äste entfernt	Pfad
H1	Horizontal	3-4 km/h	Nein	1
H2	Horizontal	3-4 km/h	Ja	1
30_1	30° nach unten	3-4 km/h	Nein	1
30_2	30° nach unten	3-4 km/h	Ja	1
V1	Vertikal	3-4 km/h	Nein	1
V2	Vertikal	3-4 km/h	Ja	1
L	Horizontal	ca. 2 km/h	Ja	2
S	Horizontal	ca. 7 km/h	Ja	2

Tabelle 5.1: Übersicht aller Datensätze

Ebenfalls aufgrund der Befahrbarkeit wurden die beiden Datensätze L und S auf Pfad 2 aufgenommen. Dieser führt zwischen den Baumreihen 3 und 4, also in Gasse 3, nach Südwesten. Nach einem Wendemanöver führt dieser Pfad zwischen den Baumreihen 4 und 5, also in Gasse 4, wieder nach Nordosten zurück in Richtung des Startpunktes.

Um den Unterschied vor und nach dem Entfernen der Äste analysieren zu können, wurden Datensätze mit den Ausrichtungen Horizontal (H), Vertikal (V) und 30° nach unten geneigt (30) jeweils vor und nach dem Entfernen aufgezeichnet. Das Datensatzkürzel setzt sich aus dem Ausrichtungskürzel und der Nummer 1 für vorher und 2 für nachher zusammen. Die Fahrgeschwindigkeit betrug hierbei 3 bis 4 km/h und wurde nicht variiert, um die Auswirkungen der Ausrichtung zu isolieren. Um die Auswirkungen der Fahrgeschwindigkeit zu analysieren, wurden die Datensätze L und S mit 2 km/h bzw. 7 km/h aufgenommen. Die Parameter der Datensätze sind in Tabelle 5.1 gegenübergestellt. Auf weitere Permutationen wurde aufgrund der Befahrbarkeit und des Zeitaufwandes in der Aufnahme und Auswertung verzichtet.

5.2 Evaluationskriterien

Für die Evaluation der SLAM-Verfahren wurden folgende Kriterien ausgewählt:

Offensichtliche Fehler: die SLAM-Karte wird auf offensichtliche Fehler untersucht. Treten solche Fehler auf, kann die Karte keine realistische Repräsentation der Obstbäume

sein. Die übrigen Kriterien werden in diesem Fall nicht weiter überprüft. Als offensichtliche Fehler werden folgende erwartete Eigenschaften eingestuft:

- Grobe Struktur mit langen Baumreihen nicht erkennbar: besteht die Karte aus einer ungeordneten Ansammlung von Punkten, beispielsweise in Kugelform, liegt ein Fehler vor
- Kreuzende Baumreihen: die echten Baumreihen sind nahezu parallel und bestehen aus einer Reihe von Einzelbäumen. Nähern sich die Baumreihen an, fusionieren zu einer Reihe oder kreuzen sich, liegt ein Fehler vor.
- Doppelte Reihen: in den Baumreihen sind die Bäume einzeln gepflanzt. Ihre Nachbarn sind entweder in der selben Reihe oder in der parallelen Reihe mit einem Mindestabstand von 3 Metern. Befinden sich in der Karte zwei Bäume in einem Abstand von weniger als 3 Metern orthogonal zur Reihenrichtung oder sind zwei Baumreihen dicht nebeneinander oder ineinander erkennbar, liegt ein Fehler vor.

Relative Abstände: hierfür werden Abstände innerhalb der SLAM-Karte und der echten Welt gemessen und verglichen. Dafür wurden die Positionen der Stämme der ersten und letzten Bäume in Reihe 1 und Reihe 2 mit RTK-GNSS vermessen (siehe Abbildung 5.6). Diese Positionen sind in Abbildung 5.5 jeweils mit *Reihe X Start* bzw. *Reihe X Ende* beschriftet. Die Abstände zwischen den Reihen am Anfang und am Ende wurden mit einem Gliedermaßstab überprüft und weichen um $\pm 1cm$ ab.

Aus den Positionen der Bäume ergeben sich folgende Abstände:

- Länge Reihe 1 (LR1): 206,64 Meter
- Länge Reihe 2 (LR2): 206,81 Meter
- Reihenabstand Anfang (RAA): 3,39 Meter
- Reihenabstand Ende (RAE): 3,39 Meter

Diese Abstände werden in den Punktwolken entsprechend von Stamm zu Stamm gemessen, sofern die Stämme eindeutig erkennbar sind.

Abweichung der Odometrie: die Genauigkeit der Lokalisierung als Bestandteil von SLAM wird untersucht. Dabei wird im Zeitverlauf der Abstand zwischen der SLAM-basierten Lokalisierung (Odometrie) relativ zu einer RTK-GNSS-basierten Lokalisierung

Abbildung 5.6: Vermessung des Anfangs von Reihe 1 mit einem RTK-GNSS-Vermessungsstab

ermittelt und aufgezeichnet. Die Genauigkeit der RTK-GNSS Lokalisierung wird großzügig mit $\pm 5cm$ abgeschätzt [23]. Der Abstand beschränkt sich hierbei auf die X- und Y-Achse, die Höhe wird ignoriert. Dafür wird das in Abschnitt 4.3 vorgestellte Tool verwendet.

Absolute Abweichung der Position der entfernten Äste: mit Hilfe des in Abschnitt 4.2 vorgestellten Tools zum Zuschneiden von Punktwolken werden die Bäume mit den entfernten Ästen aus der globalen Karte extrahiert. Das Zentrum des Zuschnitts bildet der mit Hilfe von RTK-GNSS vermessene entsprechende Schnittpunkt des Astes. In der Teilpunktwolke wird der entsprechende Ast visuell gesucht und der Abstand des Schnittpunktes des Astes zum Ursprung gemessen. Daraus ergibt sich die absolute Abweichung des Astes in der Karte zum Ast in der realen Welt in einem globalen Koordinatensystem. Der Radius, mit dem die Punktwolke zugeschnitten wird, wird anhand der Positionierungsabweichung (siehe oben) zum Zeitpunkt des Vorbeifahrens an den Ästen bestimmt. Es wird angenommen, dass bei einer Abweichung der Positionierung zum globalen Koordinatensystem auch die entsprechenden Äste mit einem ähnlichen Abstand in

der Karte eingetragen werden. Voraussetzung für die Bestimmung dieser Abweichung ist ein ausreichender Detailgrad der Punktwolke (siehe unten), um die Äste identifizieren zu können.

Detailgrad der Bäume: zur Bewertung des Detailgrades der Bäume werden die Teilpunktwolken an den Positionen der entfernten Äste analysiert. Es werden 0 bis 10 Punkte vergeben. Bei 0 Punkten ist der Baum nicht als Baum zu erkennen und beispielsweise nur ein gleichmäßig gefüllter Quader. Bei 10 Punkten sind alle Äste auch mit einem Durchmesser unter 1 cm gut erkennbar. In diesem Fall wird die Punktwolke mit den Bildern der Stereokamera verglichen, um Unterschiede im Verlauf der Äste und Triebspitzen zu suchen. Bei 5 Punkten ist der Stamm und Äste mit mindestens 5 cm Stärke eindeutig erkennbar. Ab 5 Punkten werden mit doppelten Reihen (siehe oben) vergleichbare Strukturen gesucht, bei denen davon auszugehen ist, dass einzelne Äste mehrfach in der Karte eingetragen wurden.

Explizit nicht betrachtet wird die für die SLAM-Verfahren nötige Rechenleistung oder der Speicherbedarf. Auch das Verhalten bei großen Karten bzw. sehr langen Datensätzen mit vielen Baumreihen wird nicht untersucht.

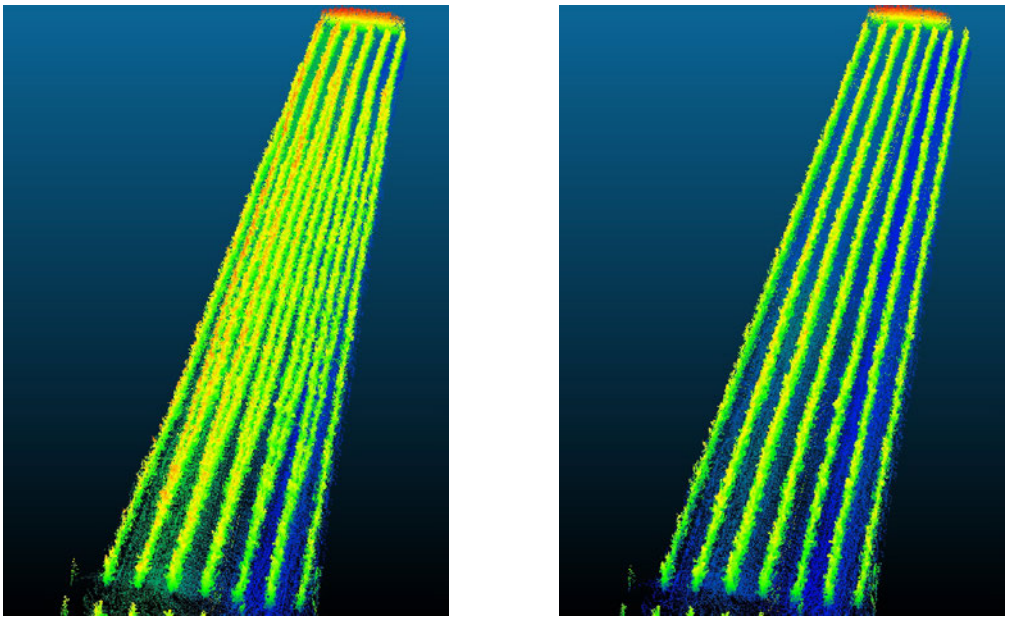
5.3 Durchführung

Die als Rosbag aufgezeichneten Datensätze wurden in der in Abschnitt 4.1 eingerichteten ROS-Umgebungen abgespielt und das Verhalten der SLAM-Verfahren beobachtet. Die entstehenden Karten wurden im pcd-Format exportiert und mit Hilfe des Programms CloudCompare visualisiert. In diesem Programm wurden zusätzlich, wenn die Struktur der Punktwolken in Ordnung war, die in Abschnitt 5.2 beschriebenen Abstände gemessen. Außerdem wurden die in Abschnitt 5.2 beschriebenen Schritte zur Analyse der Odometrieabweichung, des Detailgrades und der Abweichung der Positionen der Äste durchgeführt. Die Evaluation wird im folgenden je SLAM-Verfahren erläutert.

5.3.1 LIO-SAM

LIO-SAM erzeugt in der Standard-Konfiguration mit Datensätzen, bei denen der LiDAR-Sensor horizontal ausgerichtet ist (H1, H2, L, S) in groben Zügen korrekte Karten der Obstanlage. Nach dem Wendemanöver am Ende der Gasse (siehe Abbildung 5.5), also

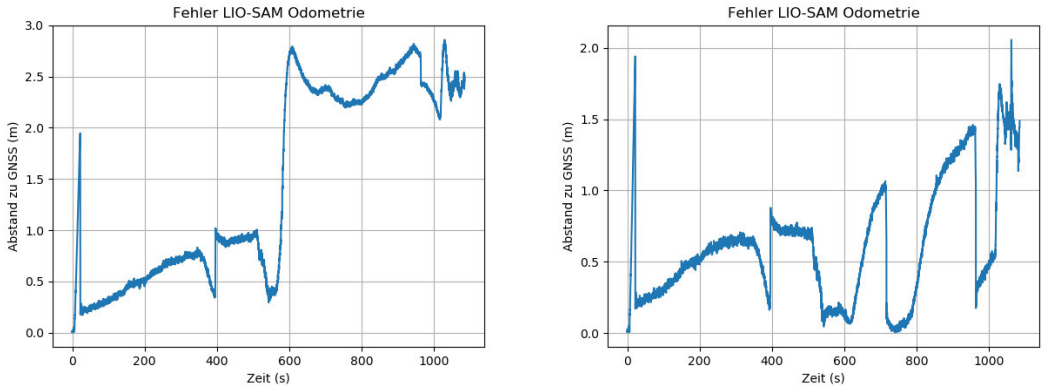
auf dem Rückweg, wird jedoch bei allen Datensätzen die Schleifenschlusserkennung ausgelöst, wodurch die Position des Aufbaus nach und nach immer weiter in der Gasse des Hinwegs geschätzt wird. Die Endposition wird dann in etwa auf der Startposition geschätzt, obwohl sich der Aufbau auf der anderen Seite der Baumreihe befindet. Die dabei entstehende Karte weist deshalb kreuzende Reihen auf (siehe Abbildung 5.7a) und am nördlichen Vorgewende sind sechs Baumreihen erkennbar, während am südlichen Vorgewende sieben erkennbar sind.



(a) aktivierte Schleifenschlusserkennung (b) deaktivierte Schleifenschlusserkennung

Abbildung 5.7: Schrägansicht nach Norden der mit LIO-SAM erstellten Karten des Datensatzes H1 mit und ohne Schleifenschlusserkennung

Wird die Schleifenschlusserkennung in der Konfiguration vollständig deaktiviert, treten diese Fehler nicht auf und es entsteht eine realistische Karte der Obstanlage (siehe Abbildung 5.7b). In diesem Fall sind an beiden Vorgewenden sieben Baumreihen zu erkennen. Dieses Verhalten ist ebenfalls in der Abweichung der Odometrie zu erkennen. Bei Datensatz L beispielsweise wächst, wie in Abbildung 5.8a zu sehen, der Abstand zwischen der LiDAR-basierten Odometrie und der GNSS-basierten Odometrie langsam auf ca. 1 Meter an. Nach etwa 600 Sekunden, kurz nach dem Wendemanöver steigt diese Abweichung auf ca. 2,5m an. In diesem Moment wird ein Schleifenschluss erkannt und in der Graphoptimierung entsprechend verrechnet.

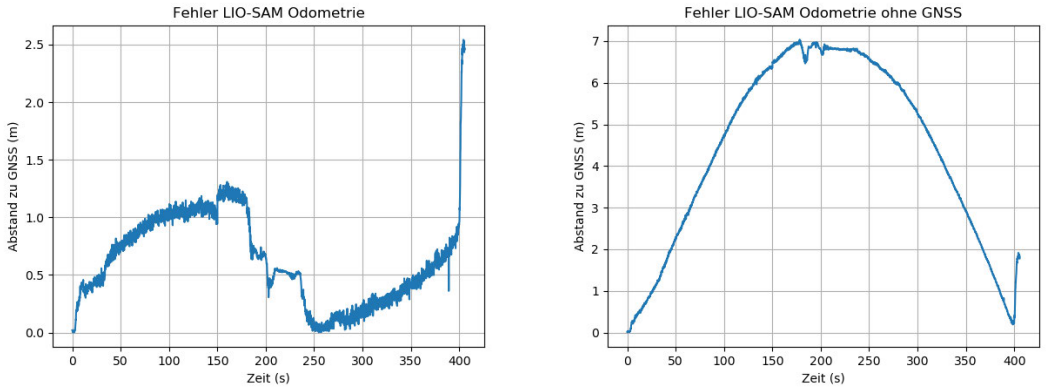


(a) aktivierte Schleifenschlusserkennung (b) deaktivierte Schleifenschlusserkennung

Abbildung 5.8: Fehler der LiDAR-basierten Odometrie für den Datensatz L mit und ohne Schleifenschlusserkennung

Wird die Schleifenschlusserkennung deaktiviert, ist dieser Sprung nicht erkennbar (siehe Abbildung 5.8b). Die ab Sekunde 610 und Sekunde 800 anwachsenden Peaks, sind auf ein Problem mit dem Bezug der Korrekturdaten für RTK-GNSS zurückzuführen. Durch zeitweises Fehlen der Korrekturdaten ging die Genauigkeit der RTK-basierten Odometrie zurück, weshalb diese Daten nur eingeschränkt als Ground Truth anzusehen sind. Zum Zeitpunkt, an dem die Korrekturdaten wieder vorliegen, fällt die Abweichung stark ab, weil die Genauigkeit der GNSS-Odometrie wieder hoch ist und die neue GNSS-Positionierung in die Graphoptimierung einfließt. Dieser Effekt ist ebenfalls bei aktivierter Schleifenschlusserkennung ab Sekunde 800 zu sehen (siehe Abbildung 5.8a). Bei Sekunde 600 unterstützt die Ungenauigkeit der GNSS-Positionierung die Gewichtung der Schleifenschlusserkennung. Für die übrigen Datensätze wurde daher die Schleifenschlusserkennung deaktiviert. Diese Einstellung wird im folgenden mit *LIO-SAM no loop closure* (kurz: *LIO NL*) bezeichnet. In Tabelle 5.2 ist diese Konfiguration mit *NL* bezeichnet.

Die Auswirkungen der Berücksichtigung von der GNSS-Positionierung in LIO-SAM wurde ebenfalls untersucht. Bei Datensatz H1 entsteht ohne GNSS-Einfluss eine Karte, die die grobe Struktur der Obstanlage gut abbildet. Die Baumreihen in der Karte weisen jedoch eine starke Krümmung auf, die in den realen Baumreihen nicht zu finden ist. Diese Krümmung resultiert in einer starken Zunahme der Odometrieabweichung bis zum Wendemanöver und einer identischen Abnahme bis zum Startpunkt zurück (siehe Abbildung 5.9b). Die maximale Abweichung beträgt etwa 7 Meter.



(a) Mit GNSS (b) Ohne GNSS

Abbildung 5.9: Fehler der LiDAR-basierten Odometrie für den Datensatz H1 mit und ohne Berücksichtigung der GNSS-basierten Odometrie, jeweils mit deaktivierter Schleifenschlusserkennung

Wird GNSS berücksichtigt, sind die Baumreihen gerade und der Fehler der LiDAR-basierten Odometrie erreicht einen maximalen Wert von etwa 1,3 Meter (siehe Abbildung 5.9a). Die starke Abweichung am Ende des Datensatzes, wie sie auch ohne GNSS-Berücksichtigung auftritt, ist auf einen Fehler der GNSS-basierten Odometrie zurückzuführen. Für diese kommt ein EKF zum Einsatz, der am Ende des Datensatzes die letzte Bewegung in die Zukunft extrapoliert. Der starke Anstieg am Ende kann also ignoriert werden.

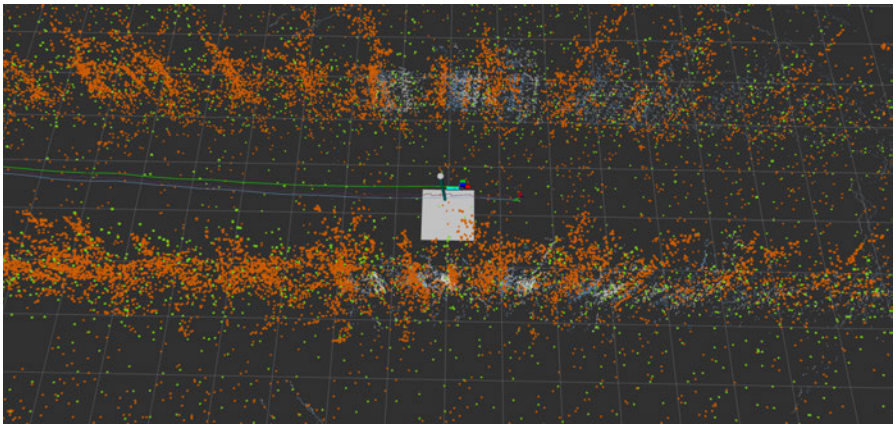


Abbildung 5.10: Momentaufnahme von LIO-SAM im Datensatz H1, Schrägansicht nach Südosten auf einen etwa 15 m langen Abschnitt der Baumreihen 1 und 2, Verlauf der LiDAR-Odometrie in gelb und GNSS-Odometrie in blau

Der Fehler der LiDAR-basierten Odometrie besteht dabei fast ausschließlich aus der Abweichung parallel zu den Baumreihen. Dies ist in Abbildung 5.10 gut erkennbar. Der Verlauf der LiDAR-Odometrie liegt mit Abweichung von ca. 10 cm auf dem Verlauf der GNSS-Odometrie. Letztere liegt im Datensatz H1 in Gasse 1 nahezu konstant etwa einen Meter weiter entlang der Baumreihen in der Fahrgasse. Dieses Verhalten ist auch im Verlauf der Odometrieabweichung in Abbildung 5.9a zwischen Sekunde 50 und Sekunde 180 gut erkennbar.

Bei den Datensätzen 30_1, 30_2, V1 und V2, also bei nicht horizontaler LiDAR-Ausrichtung, entstehen bei der Verarbeitung mit LIO-SAM keine sinnvollen Karten, die die Obstanlage abbilden. Schon nach wenigen Sekunden wird der Aufbau nicht mehr sinnvoll in der Karte lokalisiert und die Positionsannahme springt mehrere Meter umher. Bei dem Datensatz V2 wird der Aufbau für etwa 15 Sekunden mit geringen Abweichungen entlang der Baumreihe auf der Stelle lokalisiert (siehe Abbildung 5.11), obwohl sich der Aufbau, wie man an dem Verlauf der GNSS-Odometrie sehen kann, bereits etwa 15 Meter in die Gasse hinein bewegt hat. Danach springt die Lokalisierung ebenfalls mehrere Meter und die Karte wird durch viele nicht sinnvolle Elemente erweitert.

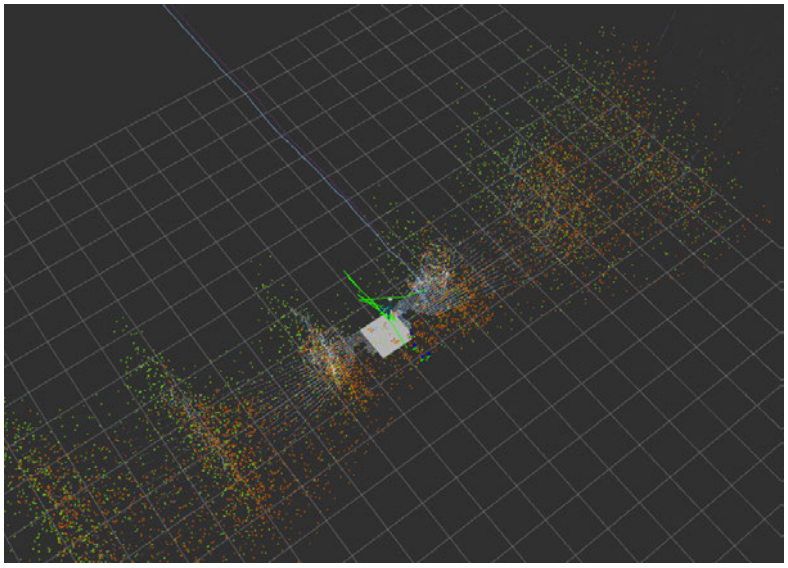


Abbildung 5.11: Momentaufnahme von LIO-SAM im Datensatz V2, Schrägansicht nach Südwesten, Verlauf der LiDAR-Odometrie in gelb und GNSS-Odometrie in blau

Diese Evaluationsergebnisse sind in Tabelle 5.2 zusammengefasst. Die Konfiguration ist dabei mit *Konfig.* abgekürzt. *Normal* bezeichnet die Standardkonfiguration von LIO-SAM, *NL* die veränderte Konfiguration mit deaktivierter Schleifenschlusserkennung.

Daten-satz	Konfig.	Fehler	Abstände	Odometrie	Position	Details
H1	Normal	Kreuzende Reihen	-	2,5 m	-	-
L	Normal	Kreuzende Reihen	-	2,5 m	-	-
H1	Kein GNSS	Krümmung	-	7 m	-	5
H1	NL	Keine	LR1: -0,36% LR2: -0,20% RAA: 0,00% RAE: -1,77%	Anfang: 0,5 m Mitte: 1,3 m Ende: 0,75 m	-	5
H2	NL	Keine	LR1: -0,40% LR2: -0,34% RAA:-0,45% RAE:-1,2%	Anfang: 0,5 m Mitte: 1,3 m Ende: 0,75 m	-	5
L	NL	Keine	LR1: -0,21% LR2: -0,27% RAA:0,00% RAE:-0,81%	Anfang: 0,75m Mitte: <0,3 m Ende: 1,5 m	-	6
S	NL	Keine	LR1: -0,34% LR2: -0,41% RAA:-0,70% RAE:-0,20%	Anfang: 1,5m Mitte: <0,5m Ende: 2 m	-	5
30_1	NL	Fehlende Struktur	-	>25 m Abbruch	-	-
30_2	NL	Fehlende Struktur	-	>25 m Abbruch	-	-
V1	NL	Fehlende Struktur	-	>25 m Abbruch	-	-
V2	NL	Fehlende Struktur	-	>25 m Abbruch	-	-

Tabelle 5.2: Übersicht LIO-SAM Evaluationsergebnisse

Die Spalte *Fehler* beinhaltet die in Abschnitt 5.2 beschriebenen offensichtlichen Fehler. Die Abweichung der in den Kriterien definierten Abstände wird in der Spalte *Abstände*

als relative Abweichung angegeben, wenn diese in der Karte eindeutig gemessen werden konnten. Die Spalte *Odometrie* beinhaltet die maximale Abweichung der LiDAR-basierten Odometrie zur GNSS-Referenz. Die Spalte *Position* beinhaltet die absolute Positionsabweichung der entfernten Äste, wenn diese eindeutig in den zugeschnittenen Punktwolken lokalisiert werden konnten. Die Spalte *Details* beinhaltet die Bewertung des Detailgrades der Bäume von 0 bis 10. Bei allen Datensätzen wurde kein ausreichender Detailgrad erreicht, um die entfernten Äste sicher zu lokalisieren.

5.3.2 RTAB-Map

RTAB-Map erzeugt bei keinem der aufgezeichneten Datensätze eine sinnvolle Karte der Obstanlage. Bei den Datensätzen H1 und H2 verläuft der Hinweg zwischen den Baumreihen 1 und 2 problemlos, etwa 20 Meter nach dem Wendemanöver driftet die Positionsschätzung jedoch in einem Winkel von etwa 45° stark nach Norden. Dieses Verhalten ist in Abbildung 5.12 gut zu erkennen. Zuvor bilden sich in der Karte Doppelreihen, die ebenfalls im Vordergrund gut zu erkennen sind.

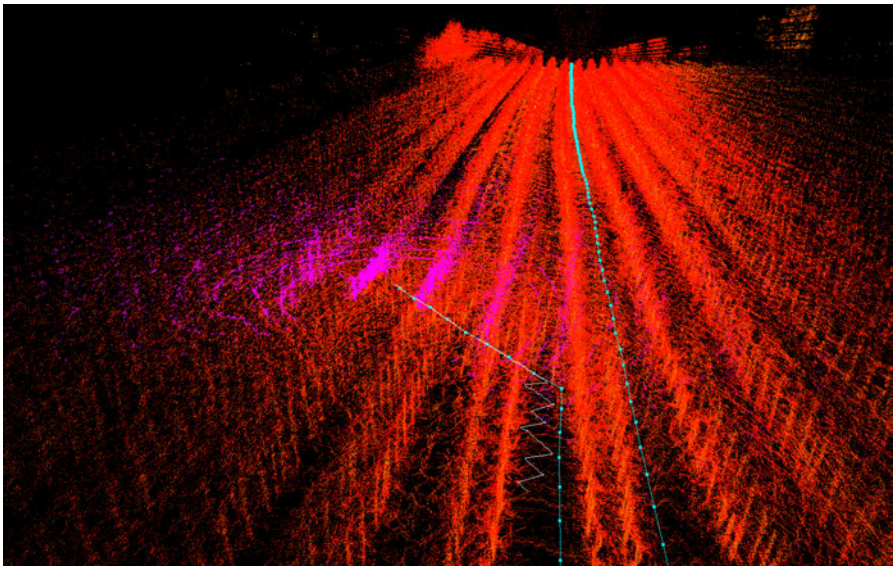


Abbildung 5.12: Moment des Fehlers bei RTAB-Map

Dieses Verhalten spiegelt sich auch in der Abweichung der LiDAR-basierten Odometrie zur GNSS-basierten Odometrie wieder. Wie in Abbildung 5.13 zu erkennen, steigt diese Abweichung bei Datensatz H1 im Zeitverlauf bis auf etwa 10 Meter an. Ab Sekunde 200, also beim Wendemanöver am Ende der Gasse, geht diese Abweichung wieder zurück. Der

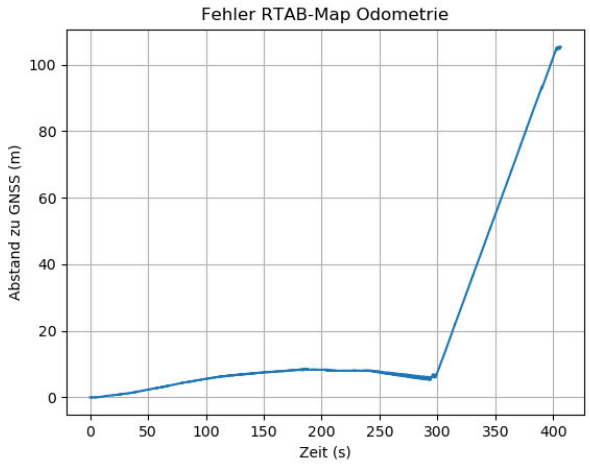


Abbildung 5.13: Abweichung der LiDAR-basierten Odometrie von RTAB-Map zur GNSS-basierten Odometrie bei dem Datensatz H1

Verlauf ähnelt bis zu Sekunde 280 stark dem Verlauf von LIO-SAM ohne Berücksichtigung der GNSS-Daten (siehe Abbildung 5.9b). An diesem Zeitpunkt beginnt der zuvor beschriebene Fehler. Auffällig ist hierbei, dass die Positionsschätzung zuvor in einer Art Sägezahnmuster mit etwa 1,5 Meter Amplitude in die selbe Richtung wandert und zurückspringt, in die dann der endgültige Drift erfolgt. Diese fehlerhafte Positionsschätzung ist nicht auf LiDAR-Scan-Matching zurückzuführen, da die Ausrichtung der Scans mit der Karte zu diesem Zeitpunkt übereinstimmt.

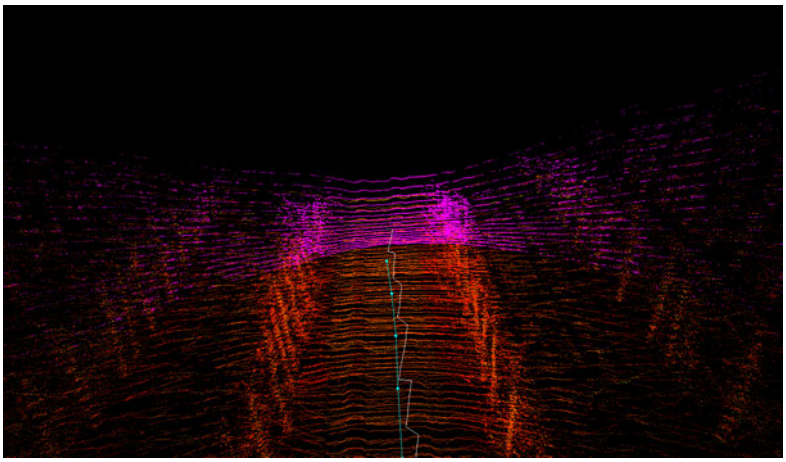


Abbildung 5.14: Momentaufnahme von RTAB-Map bei dem Datensatz 30_1, aktuellster Scan in blau, Karte in rot

Bei den Datensätzen 30_1 und 30_2 tritt dieses Verhalten nahezu identisch auf. Der Fehler tritt jedoch etwas später auf. In Abbildung 5.14 ist gut die Sichtweite des LiDAR-Sensors bei einer Neigung von 30° nach unten sichtbar.

Bei Datensatz S wird kurz nach dem Start die Bewegung des Aufbaus quer zu den in den einzelnen Scans erkennbaren Baumreihen geschätzt, sodass eine Karte mit einer zufälligen Verteilung von Punkten entsteht. Diese ist für jegliche weitere Analysen ungeeignet.

Bei Datensatz L entsteht zunächst eine Karte, in der die Baumreihen grob zu erahnen sind. Dabei weicht die Ausrichtung der Karte etwa 60° von der Realität ab. Nach etwa einem drittel des Datensatzes kehrt sich jedoch die Richtung der Bewegungsschätzung um, wodurch viele Artefakte in der Karte entstehen. Nach dem Wendemanöver ist keine sinnvolle Struktur mehr zu erkennen.

Datensatz	Fehler	Abstände	Odometrie	Position	Details
H1	Doppelreihen Kreuzende Reihen	-	Anfang: 1 m Mitte: 10 m Ende: >50 m	-	2
H2	Doppelreihen Kreuzende Reihen	-	Anfang: 1 m Mitte: 10 m Ende: >50 m	-	2
L	Fehlende Struktur Kreuzende Reihen	-	>25 m	-	2
S	Fehlende Struktur	-	>25 m	-	2
30_1	Doppelreihen Kreuzende Reihen	-	Anfang: 1 m Mitte: 10 m Ende: >50 m	-	2
30_2	Doppelreihen Kreuzende Reihen	-	Anfang: 1 m Mitte: 10 m Ende: >50 m	-	2
V1	Fehlende Struktur	-	>25 m Abbruch	-	-
V2	Fehlende Struktur	-	>25 m Abbruch	-	1

Tabelle 5.3: Übersicht RTAB-Map Evaluationsergebnisse

Bei Datensatz V1 tritt ein ähnliches Verhalten wie bei LIO-SAM auf. Nach wenigen Sekunden springt die Positionsschätzung mehrere Meter umher und es entsteht eine Karte

mit vielen zufällig verteilten Punkten. Bei Datensatz V2 sind die Baumreihen grob zu erahnen, diese sind allerdings nicht gerade.

Alle mit RTAB-Map erzeugten Karten sind sehr dünn, sodass sich die grob zu erahnenden Bäume nur aus etwa 100 Punkten zusammensetzen.

Die Evaluationsergebnisse sind in Tabelle 5.3 zusammengefasst. Die Bedeutung der Spalten ist dabei identisch zu Tabelle 5.2.

5.3.3 HDL Graph SLAM

Mit HDL Graph SLAM werden bei den Datensätzen mit Horizontaler LiDAR-Ausrichtung (H1, H2, L und S) Karten erstellt, die die grobe Struktur der Obstanlage gut abbilden (siehe Abbildung 5.15). Innerhalb der Gassen werden viele Punkte eingetragen, obwohl

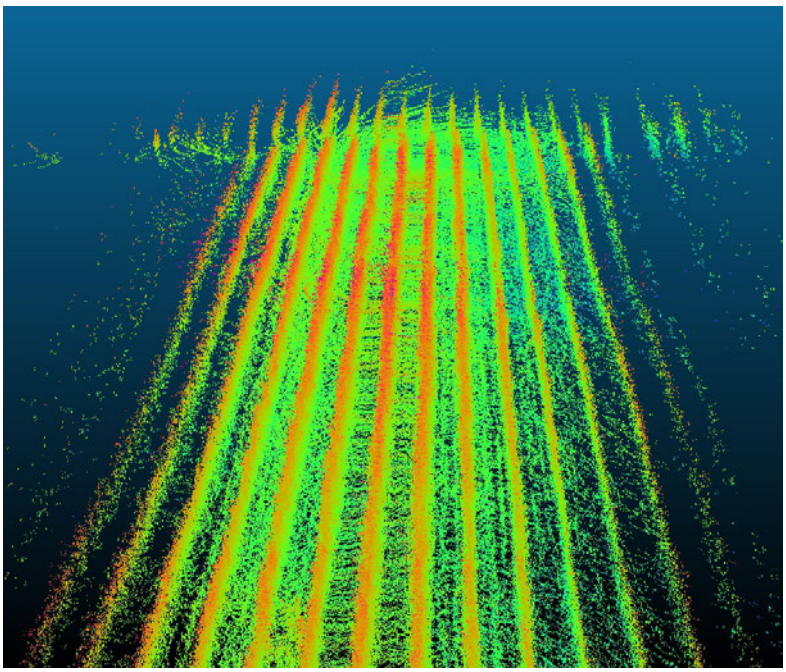


Abbildung 5.15: Schrägansicht der Karte von HDL Graph SLAM des Datensatzes H2, Blick nach Südosten

sich bei der Datenaufnahme dort keine Hindernisse befunden haben und auch in den einzelnen Scans solche Artefakte nicht zu erkennen sind. Die Baumreihen weisen eine geringe Krümmung auf, die in der Realität nicht zu finden ist.

Bei der Erstellung der Karte weicht die LiDAR-basierte Odometrie teilweise stark von der GNSS-basierten Odometrie ab (siehe Abbildung 5.16). Dabei ist auffällig, dass der Zeitverlauf ähnlich zum Verhalten von LIO-SAM ohne GNSS-Berücksichtigung (siehe Abbildung 5.9b) ist. Im Gegensatz zu LIO-SAM ohne GNSS besteht diese Abweichung allerdings fast ausschließlich entlang der Baumreihen. Dieses Verhalten entspricht eher dem Verhalten von LIO-SAM mit GNSS-Berücksichtigung (siehe Abbildung 5.10). Im Gegensatz zu LIO-SAM ist diese Abweichung nahezu linear von der Entfernung zum Startpunkt abhängig. Der große Peak am Anfang der Zeitreihe ist darauf zurückzuführen, dass die Karte zu Beginn nicht in die richtige Himmelsrichtung ausgerichtet ist. Durch die korrekte Lokalisierung in der falsch ausgerichteten Karte entsteht ein großer Abstand zur realen GNSS-basiert gemessenen Position. Dieser Fehler wird von HDL Graph SLAM nach kurzer Zeit korrigiert und die Karte in die richtige Richtung gedreht.

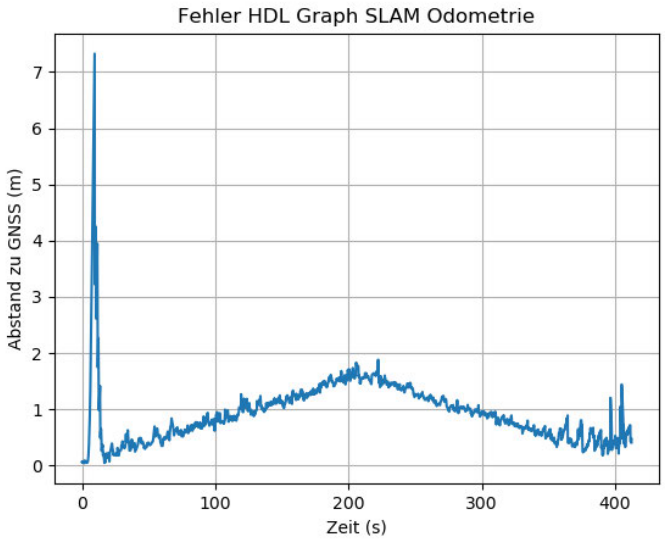


Abbildung 5.16: Odom distance HDl H2

Der Detailgrad der Bäume ist gering. Die Bäume bestehen aus sehr vielen nahezu zufällig verteilten Punkten, sodass keine einzelnen Äste erkennbar sind (siehe Abbildung 5.17). Auch der Stamm ist nicht eindeutig erkennbar. Die Position eines Baumes lässt sich am gedanklichen Schwerpunkt der zugehörigen Punkte schätzen.

Aufgrund eines Fehlers im Dateiexport sind die Karten von HDL Graph SLAM nicht ganz vollständig und das nordöstliche Vorgewende fehlt. Dieses Problem konnte nicht gelöst werden, weshalb die Messungen der Abstände LR1, LR2 und RAA nicht durchgeführt

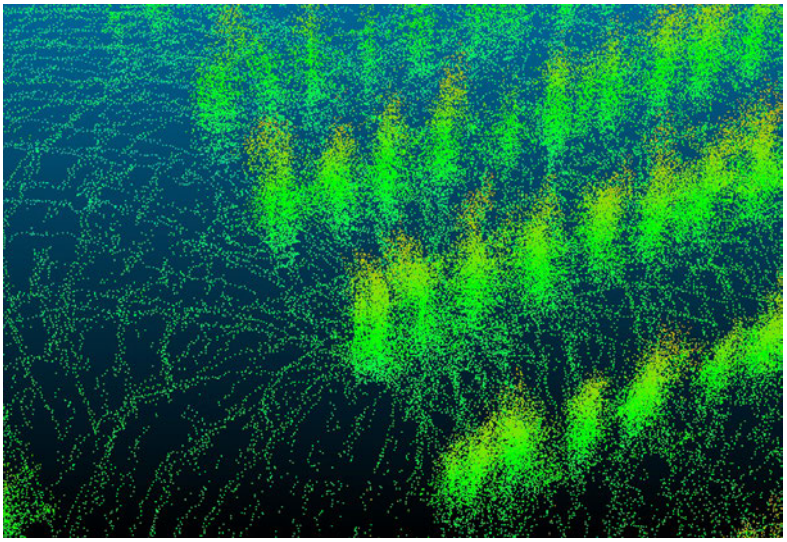


Abbildung 5.17: Detailansicht der Karte von HDL Graph SLAM des Datensatzes H1, südliches Vorgewende, Blick nach Norden

werden konnten. Die Messung des Abstandes RAE war aufgrund des niedrigen Detailgrades der Bäume und nicht eindeutig erkennbarem Stamm nur eingeschränkt möglich. Für die Längen der Baumreihen LR1 und LR2 wird angenommen, dass diese in der Größenordnung des Odometriefehlers zum Zeitpunkt des Wendemanövers kürzer sind, da die Lokalisierung relativ zur Karte sehr gut ist und der Odometriefehler nahezu vollständig entlang der Baumreihe besteht. Es wird daher angenommen, dass LR1 und LR2 in der HDL Graph SLAM Karte etwa einen Meter kürzer als in der Realität sind.

Bei den Datensätzen mit 30° geneigtem LiDAR-Sensor (Datensatz 30_1 und 30_2) entstehen bis zur Mitte des Datensatzes sehr ähnliche Karten, wie bei der horizontalen Ausrichtung. Auf dem Weg durch die zweite Fahrgasse zurück zum Startpunkt entsteht gewissermaßen eine zweite Ebene von Baumreihen unterhalb der bestehenden Karte. Dadurch ist eine weitere Analyse dieser Karten weder möglich noch sinnvoll. Dieses Verhalten ist in der Odometrieabweichung nicht zu erkennen, da diese nur in der X- und Y-Achse gemessen wird, der Fehler liegt aber in der Z-Achse.

Bei den Datensätzen mit vertikal ausgerichtetem LiDAR-Sensor (Datensatz V1 und V2) wird die Bewegung des Aufbaus durch die Obstanlage nicht erkannt. Ähnlich zu dem Verhalten von LIO-SAM (siehe Abbildung 5.11) wird der Aufbau immer an der selben Stelle lokalisiert. Es entsteht keine sinnvolle Karte.

Datensatz	Fehler	Abstände	Odometrie	Position	Details
H1	Artefakte in der Gasse	LR1: ca. -0,5% LR2: ca. -0,5% RAA: - RAE: -3,8%	Anfang: 1,8 m Mitte: 2,5 m Ende: 1,5 m	-	2
H2	Artefakte in der Gasse	LR1: ca. -0,5% LR2: ca. -0,5% RAA: - RAE: -3,2%	Anfang: <0,5 m Mitte: 1,8 m Ende: <0,5 m	-	2
L	Artefakte in der Gasse	LR1: ca. -0,5% LR2: ca. -0,5% RAA: - RAE: -3,5%	Anfang: 1,3m Mitte: 2,5 m Ende: 1 m	-	2
S	Artefakte in der Gasse	LR1: ca. -0,5% LR2: ca. -0,5% RAA: - RAE: -3,2%	Anfang: 0,5m Mitte: 1,8m Ende: <0,5 m	-	2
30_1	zweite Ebene	-	Anfang: 1,3m Mitte: 2,5 m Ende: 1 m	-	-
30_2	zweite Ebene	-	Anfang: 1,3m Mitte: 2,5 m Ende: 1 m	-	-
V1	auf der Stelle	-	>50 m Abbruch	-	-
V2	auf der Stelle	-	>50 m Abbruch	-	-

Tabelle 5.4: Übersicht HDL Graph SLAM Evaluationsergebnisse

Die Evaluationsergebnisse für HDL Graph SLAM sind in Tabelle 5.4 zusammengefasst. Die Bedeutung der Spalten ist dabei identisch zu Tabelle 5.2.

6 Diskussion

In diesem Kapitel werden die Evaluationsergebnisse aus Kapitel 5 diskutiert und anhand dessen anschließend die Eignung der ausgewählten SLAM-Verfahren für die Zwecke der Navigation und der Pflanzenanalyse eingeschätzt.

6.1 Allgemein

Insgesamt zeigt sich in den Evaluationsergebnissen, dass die Erstellung von realistischen Karten von Obstanlagen mit Hilfe von 3D-SLAM-Verfahren mit Einschränkungen möglich ist. Die besten Ergebnisse werden mit LIO-SAM ohne Schleifenschlusserkennung bei horizontaler Ausrichtung des LiDAR-Sensors erreicht. Mit RTAB-Map entstehen bei keinem der verwendeten Datensätze realistische Karten der Obstanlage. In den mit HDL Graph SLAM erstellten Karten finden sich viele Artefakte zwischen den Baumreihen, sofern die grobe Struktur die Obstanlage realistisch abbildet. Keines der vorgestellten und verwendeten SLAM-Verfahren erzeugt Karten, in denen der Detailgrad der Bäume ausreicht, um einzelne Äste eindeutig zu identifizieren und zu lokalisieren.

Ebenfalls konnte keines der Verfahren bei den Datensätzen mit 30° nach unten ausgerichtetem LiDAR-Sensor sinnvolle Karten erzeugen, wobei bis zum Wendemanöver in der Mitte der Datensätze teilweise gute Ansätze erkennbar sind. Auch bei den Datensätzen mit vertikaler LiDAR-Ausrichtung werden keine sinnvollen Karten erstellt. In diesem Fall ist gut zu erkennen, dass die verwendeten ICP-Algorithmen durch die repetitiven Strukturen der Baumreihen die Bewegung der Sensoren relativ zu diesen nicht korrekt erfassen. Der Vorteil der besseren Höhenauflösung der Bäume in unmittelbarer Umgebung bei geneigtem LiDAR-Sensor kann mit den verwendeten SLAM-Verfahren also nicht ausgenutzt werden.

Werden realistische Karten der Obstanlage erzeugt, zum Beispiel mit Hilfe von LIO-SAM bei horizontaler LiDAR-Ausrichtung, ist die Lokalisierung in der Karte sehr genau.

Die Lokalisierung zu einem globalen Koordinatensystem wie dem UTM-System weicht jedoch bis zu zwei Meter ab. Die Baumreihen in der Karte sind dadurch beispielsweise in derselben Größenordnung kürzer, als in der Realität. Diese Abweichung ist relativ betrachtet gering, hat jedoch in der Praxis als absolute Abweichung eine Bedeutung.

Diese Verzerrung des Karten-Koordinatensystems zum UTM-Koordinatensystem scheint bei HDL Graph SLAM meist linear, bei LIO-SAM ist sie vermutlich unregelmäßig. Im Rahmen dieser Arbeit wurde dieser Aspekt nicht weiter untersucht. Es ist jedoch davon auszugehen, dass bei zunehmender Kartengröße diese Verzerrung zu zunehmenden Differenzen zwischen dem Koordinatensystem der Karte und dem UTM-System führt. In dieser Arbeit wurden nur einzelne Baumreihen mit einer Länge von etwa 200 Metern betrachtet, bei einem Einsatz dieser Verfahren in Robotik- oder Kartierungsanwendungen über mehrere Kilometer oder Hektar wird eine nennenswerte absolute Abweichung erwartet.

Um diese Verzerrung zu reduzieren, könnte die RTK-GNSS-basierte Lokalisierung im UTM-Koordinatensystem in allen Verfahren stärker berücksichtigt werden. Bei LIO-SAM konnte der große Einfluss der GNSS-Berücksichtigung gezeigt werden. Trotzdem erreicht auch in diesem Verfahren die LIO-SAM eigene Positionierungsgenauigkeit nicht die Größenordnung der Genauigkeit der einzelnen berücksichtigten Komponenten RTK-GNSS und LiDAR-Scan-Matching.

6.2 Eignung für Navigation

Die Navigation auf Basis der realistischen Karten von LIO-SAM und HDL Graph SLAM wird als möglich eingeschätzt. Dafür könnten einfache 2D-Occupancy Grids aus den 3D-Punktwolken erzeugt werden, in dem alle Bereiche mit einer Mindestanzahl von Punkten in einer Höhe über einem Schwellwert von beispielsweise 1 Meter als belegt markiert werden. Der Vorteil von RTAB-Map, direkt solche Occupancy Grids zu erzeugen, kann aufgrund der fehlerhaften Kartierung nicht ausgenutzt werden. Bei den Karten von HDL Graph SLAM wird ein sinnvoller Umgang mit den Artefakten in den Fahrgassen benötigt. Mit einem naiven Ansatz würden viele Bereiche der Fahrgasse als belegt markiert werden, obwohl sich dort keine Hindernisse befinden. Der Detailgrad der Bäume wird als ausreichend für die Erstellung von Occupancy Grids eingeschätzt.

Die Korrekte Lokalisierung der Verfahren LIO-SAM und HDL Graph SLAM relativ zur Karte reicht voraussichtlich für eine lokale Hinderniserkennung und Hindernisvermeidung durch einen lokalen Planer aus. Die Abweichung der Lokalisierung zum UTM-Koordinatensystem könnte jedoch zu Problemen mit der globalen Pfadplanung führen. Werden die Zielkoordinaten für die Navigation im UTM-System angegeben, führt eine Transformation in das Koordinatensystem der Karte zu dieser entsprechenden Abweichung. Bei einer Größenordnung von einem Meter werden nur geringfügige Probleme erwartet. Bei einer Abweichung von mehreren Metern könnten jedoch die Zielkoordinaten in der realen Welt in einer anderen Fahrgasse liegen als in der Karte. Die Navigation zu diesen Koordinaten würde also dazu führen, dass ein lokal unüberwindbares Hindernis zwischen der realen Zielkoordinate und der von der Navigation angestrebten Position liegt. Die vorgestellten Verfahren sind also nur eingeschränkt für die globale Navigation basierend auf UTM-Zielkoordinaten geeignet. Die Angabe von Zielkoordinaten in einem globalen Koordinatensystem wie dem UTM-System wird im Kontext des Obstanbau als sinnvoll eingeschätzt.

6.3 Eignung für Pflanzenanalyse

In den realistischen Karten von LIO-SAM und HDL Graph SLAM wird die Messung von groben Parametern wie der Baumhöhe und dem Kronenvolumen als realistisch eingeschätzt. LIO-SAM erreicht den höchsten Detailgrad der vorgestellten Verfahren. Auch in den besten Karten sind einzelne Äste nicht eindeutig erkennbar, wodurch die Messung von Triebwachstum beispielsweise nicht möglich ist. In dieser Arbeit wurde durch das Entfernen einzelner Äste regionales negatives Wachstum simuliert. Dieses konnte in keiner der erstellten Karten eindeutig wiedergefunden werden.

Wie auch in der Navigation wird ein großes Problem in der Verzerrung der Karte im Verhältnis zu einem globalen Koordinatensystem gesehen. Für die Analyse der einzelnen Bäume ist es nötig, ihre Position in der Karte auf ihre Identität abzubilden. Dafür wird es als sinnvoll erachtet, die Position der Bäume in einem globalen Koordinatensystem wie dem UTM-System zu speichern und abrufen zu können. Dieses Vorgehen würde eine Kombination mit vielen weiteren Verfahren zur Erfassung der Baumeigenschaften ermöglichen. Durch die Abweichung der Koordinaten in der Karte zu den UTM-Koordinaten kann es zu einer Verwechslung der Bäume kommen. Bei einem Baumabstand von 1,2 Metern reicht eine Abweichung 1,2 Metern entlang der Baumreihe aus, um einen Baum

mit seinem direkten Nachbarn zu verwechseln. Die Abweichung der Lokalisierung war bei den vorgestellten Verfahren fast vollständig entlang der Baumreihe und lag in dieser Größenordnung.

Die vorgestellten Verfahren sind daher nur sehr eingeschränkt für die Pflanzenanalyse in Baumobstkulturen geeignet.

7 Fazit und Ausblick

In dieser Arbeit konnte gezeigt werden, dass die ausgewählten 3D-SLAM-Verfahren in Baumobstkulturen verwendet werden können, um realistische Karten zu erstellen. Die Verwendung der Karten und die Genauigkeit der Lokalisierung reicht jedoch nur eingeschränkt für die Zwecke der Navigation und in geringem Maße für die Zwecke der Pflanzenanalyse aus. Grund dafür ist zum einen der niedrige lokale Detailgrad, bei dem einzelne Äste nicht oder nur teilweise erkennbar sind, und die Abweichung der SLAM-basierten Lokalisierung zu einem globalen Referenzkoordinatensystem wie dem UTM-System, durch die Verzerrung der Karte.

Diese Verzerrung und der Umgang damit ist näher zu untersuchen. Dabei wird vor allem Potential in der stärkeren Gewichtung der RTK-GNSS-basierten Lokalisierung gesehen, um die Verzerrung während der Kartierung deutlich zu verringern. Hinweise darauf gibt die Verwendung von GNSS-Daten in LIO-SAM. Bei der in dieser Arbeit verwendeten Konfiguration und Einrichtung wird die Transformation zwischen dem Karten-Koordinatensystem und dem globalen Koordinatensystem ausschließlich durch eine Verschiebung und Rotation realisiert. Sollte die Verzerrung der Karte durch Anpassung der SLAM-Verfahren nicht eliminiert werden können, ist die Erstellung einer Verzerrungsmatrix während der Erstellung der Karte durch mit RTK-GNSS gemessene Stützpunkte denkbar, sodass die Transformation zwischen den beiden Koordinatensystemen mit dieser Matrix korrekt durchgeführt werden kann.

Sollte die Verzerrung in Zukunft stark verringert werden und der Detailgrad gesteigert werden, wird eine Baumanalyse auf Basis von mit 3D-SLAM-Verfahren erstellten Karten denkbar. Besonderes Potential wird in der Messung von Wachstum und der Kombination mit externen Datenquellen zu den Bäumen basierend auf den Geokoordinaten der Einzelbäume erwartet. Sollten die Baumpositionen in der Karte und in einem globalen Koordinatensystem im Bereich von wenigen Dezimetern abweichen, ist der Einsatz eines Stammerkennungsalgorithmus denkbar. In dieser Größenordnung ist es möglich, den einzelnen Baum anhand des Stammes eindeutig zu identifizieren und zu lokalisieren, sodass

auch Karten mit großem Zeitabstand und einer entsprechenden Verschiebung zueinander exakt übereinandergelegt werden können, um beispielsweise Triebwachstum zu messen.

Ein weiterer Vorteil der minimalen Verzerrung ist der einfache Wechsel zwischen einer RTK-GNSS-basierten und LiDAR-basierten Navigation. Wenn beide Lokalisierungen nahezu identische Daten liefern, lassen sich beispielsweise für Umfeldsensorik nicht erkennbare Hindernisse wie Sperrzonen oder Gewässerufer auf Basis weiterer Datenquellen wie Datenbanken in die Karte integrieren und von der Navigation verarbeiten. Außerdem kann das Fehlen von Korrekturdaten für RTK-GNSS zeitweise überbrückt werden, indem die Lokalisierung auf Basis der Lokalisierung durch Scan Matching und der ungenauen GNSS-Messungen fortgesetzt werden. Beim aktuellen Stand der Technik muss in diesem Fall üblicherweise angehalten werden, oder die Navigation wird ungenau.

In dieser Arbeit wurden ausschließlich unbelaubte Bäume betrachtet. Diese bieten im Vergleich zu belaubten Bäumen wesentlich weniger Reflektionsfläche für die LiDAR-Sensoren und erzeugen dadurch häufig spärliche Scans. Es ist daher unklar, ob sich die in dieser Arbeit gewonnenen Erkenntnisse auch auf belaubte Obstbäume übertragen lassen.

Die geneigte und vertikale Ausrichtung des LiDAR-Sensors hat sich als ungeeignet für die verwendeten SLAM-Verfahren herausgestellt. Es ist jedoch denkbar, dass eine stärkere Gewichtung von RTK-GNSS-basierter Lokalisierung auch mit dieser Sensorkonfiguration zu einer höheren Qualität der Karten führt. Alternativ ist das Aufaddieren von Scans basierend auf externer Lokalisierung wie bei [33] denkbar.

Insbesondere bei der Pflanzenanalyse wird außerdem viel Potential in der Sensorfusion vor allem mit bildgebenden Verfahren und Stereokameras erwaret. Dieses Vorgehen hat sich bereits in anderen Arbeiten in einem ähnlichen Umfeld bewährt [19].

Durch die eingeschränkte Auswahl von 3D-SLAM-Verfahren lassen sich die Ergebnisse dieser Arbeit nur bedingt verallgemeinern. Es ist denkbar, dass andere nicht betrachtete Verfahren deutlich bessere Ergebnisse sowohl in der Lokalisierung als auch im Detailgrad der Karte erzielen.

Literaturverzeichnis

[1] *AgXeed AgBot*. https://www.agxeed.com/our-solutions/agbot-2-055w3/. – zugegriffen: 29.05.2023

[2] *Ardusimple GNSS Multiband antenna*. https://www.ardusimple.com/product/survey-gnss-multiband-antenna/. – zugegriffen: 05.06.2023

[3] *Cartographer - Algorithm walkthrough* . https://google-cartographer-ros.readthedocs.io/en/latest/algo_walkthrough.html. – zugegriffen: 05.06.2023

[4] *Fendt 200 V/F/P Vario*. https://www.fendt.com/de/landmaschinen/traktoren/fendt-200-vfp-vario. – zugegriffen: 29.05.2023

[5] *Global Navigation Satellite Systems (GNSS)*. https://www.bkg.bund.de/DE/Observatorium-Wettzell/Messverfahren/Global-Navigation-System/global-navigation-system_cont.html. – zugegriffen: 29.05.2023

[6] *HDL Graph SLAM GitHub*. https://github.com/koide3/hdl_graph_slam. – zugegriffen: 05.06.2023

[7] *LIO-SAM GitHub*. https://github.com/TixiaoShan/LIO-SAM. – zugegriffen: 05.06.2023

[8] *ROS - costmap_2d*. http://wiki.ros.org/costmap_2d. – zugegriffen: 29.05.2023

[9] *ROS - Introduction*. https://wiki.ros.org/ROS/Introduction. – zugegriffen: 29.05.2023

[10] *RTAB-Map GitHub*. https://github.com/introlab/rtabmap/. – zugegriffen: 05.06.2023

[11] *RTK-GNSS Rover - Basestation*. https://commons.wikimedia.org/wiki/File:Real_time_kinematic.svg. – zugegriffen: 29.05.2023

[12] *SAPOS-Dienste im Überblick*. https://www.lgln.niedersachsen.de/sapos/sapos-r-dienste-im-uberblick-179039.html. – zugegriffen: 29.05.2023

[13] *u-blox ZED-F9P*. https://www.u-blox.com/en/product/zed-f9p-module. – zugegriffen: 05.06.2023

[14] *Velodyne Puck*. https://velodynelidar.com/products/puck/. – zugegriffen: 29.05.2023

[15] *Xsens MTi User Manual - Download*. https://www.xsens.com/hubfs/Downloads/usermanual/MTi_usermanual.pdf. – zugegriffen: 05.06.2023

[16] Bechar, Avital: *Innovation in Agricultural Robotics for Precision Agriculture A Roadmap for Integrating Robots in Precision Agriculture: A Roadmap for Integrating Robots in Precision Agriculture*. 01 2021. – ISBN 978-3-030-77035-8

[17] Behroozpour, Behnam ; Sandborn, Phillip A. M. ; Wu, Ming C. ; Boser, Bernhard E.: Lidar System Architectures and Circuits. In: *IEEE Communications Magazine* 55 (2017), Nr. 10, S. 135–142

[18] Chakraborty, Momtanu ; Khot, Lav R. ; Sankaran, Sindhuja ; Jacoby, Pete W.: Evaluation of mobile 3D light detection and ranging based canopy mapping system for tree fruit crops. In: *Computers and Electronics in Agriculture* 158 (2019), S. 284–293. – URL https://www.sciencedirect.com/science/article/pii/S0168169918314972. – ISSN 0168-1699

[19] Chen, Mingyou ; Tang, Yunchao ; Zou, Xiangjun ; Huang, Zhaofeng ; Zhou, Hao ; Chen, Siyu: 3D global mapping of large-scale unstructured orchard integrating eye-in-hand stereo vision and SLAM. In: *Computers and Electronics in Agriculture* 187 (2021), S. 106237. – URL https://www.sciencedirect.com/science/article/pii/S0168169921002544. – ISSN 0168-1699

[20] Grewal, Mohinder S. ; Andrews, Angus P. ; Bartone, Chris G.: *GNSS Measurement Errors*. S. 249–292. In: *Global Navigation Satellite Systems, Inertial Navigation, and Integration*, 2020

[21] Grisetti, Giorgio ; Stachniss, C. ; Burgard, Wolfram: 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

[22] Hroob, Ibrahim ; Polvara, Riccardo ; Mellado, Sergi M. ; Cielniak, Grzegorz ; Hanheide, Marc: Benchmark of visual and 3D lidar SLAM systems in simulation environment for vineyards. In: *CoRR* abs/2107.05283 (2021). – URL https://arxiv.org/abs/2107.05283

[23] Janos, Daniel ; Kuras, Przemysław ; Ortyl, Łukasz: Evaluation of low-cost RTK GNSS receiver in motion under demanding conditions. In: *Measurement* 201 (2022), S. 111647. – URL https://www.sciencedirect.com/science/article/pii/S0263224122008570. – ISSN 0263-2241

[24] Joseph, L.: *Robot Operating System (ROS) for Absolute Beginners: Robotics Programming Made Easy*. Apress, 2018 (For professionals by professionals). – URL https://books.google.de/books?id=0y1dDwAAQBAJ. – ISBN 9781484234051

[25] Kerry, Ruth ; Escolà, Alexandre ; Mulla, David ; Gregorio Lopez, Eduard ; Llorens Calveras, Jordi ; Lopez, Ainara ; De Castro, Ana ; Biswas, Asim ; Hopkins, Austin ; Stenberg, Bo ; Tisseyre, Bruno ; Minasny, Budiman ; Campillo Torres, Carlos ; Lopez-Molina, C. ; Jaren, Carmen ; Yang, Chenghai ; Fidelis, Chris ; Pérez-Roncal, Claudia ; Cribben, Curtis ; Miao, Yuxin: *Sensing Approaches for Precision Agriculture*. 11 2021. – ISBN 978-3-030-78430-0

[26] Khatoon, Shahida ; Ibraheem: Autonomous Mobile Robot Navigation by Combining Local and Global Techniques. In: *International Journal of Computer Applications* 37 (2012), Nr. 3

[27] Kohlbrecher, Stefan ; Meyer, Johannes ; Graber, Thorsten ; Petersen, Karen ; Klingauf, Uwe ; Von Stryk, Oskar: Hector Open Source Modules for Autonomous Mapping and Navigation with Rescue Robots, 01 2014, S. 624–631. – ISBN 978-3-662-44467-2

[28] Koide, Kenji ; Miura, Jun ; Menegatti, Emanuele: A portable three-dimensional LIDAR-based system for long-term and wide-area people behavior measurement. In: *International Journal of Advanced Robotic Systems* 16 (2019), 02

[29] Labbé, Mathieu ; Michaud, François: RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. In: *Journal of Field Robotics* 36 (2019), Nr. 2, S. 416–446

[30] Lee, Jinwon ; Park, Geonhyeok ; Cho, Ikhyeon ; Kang, Keundong ; Pyo, Daehyun ; Cho, Soohyun ; Cho, Minwoo ; Chung, Woojin: ODS-Bot: Mobile Robot Navigation for Outdoor Delivery Services. In: *IEEE Access* 10 (2022), S. 107250–107258

[31] Palleja, T. ; Tresanchez, M. ; Teixido, M. ; Sanz, R. ; Rosell, J.R. ; Palacin, J.: Sensitivity of tree volume measurement to trajectory errors from a terrestrial LIDAR scanner. In: *Agricultural and Forest Meteorology* 150 (2010), Nr. 11, S. 1420–1427. – URL https://www.sciencedirect.com/science/article/pii/S0168192310001942. – ISSN 0168-1923

[32] Riley, Stuart ; Talbot, Nick ; Kirk, Geoffrey: New system for RTK performance evaluation, 02 2000, S. 231 – 236. – ISBN 0-7803-5872-4

[33] Saha, Kowshik ; Tsoulias, Nikos ; Zude-Sasse, M.: Assessment of measurement uncertainty when using 2D mobile laser scanner to estimate tree parameters. In: *Landtechnik* 75 (2020), 12, S. 270–277

[34] Shan, Tixiao ; Englot, Brendan ; Meyers, Drew ; Wang, Wei ; Ratti, Carlo ; Daniela, Rus: LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* IEEE (Veranst.), 2020, S. 5135–5142

[35] Siciliano, Bruno (Hrsg.) ; Khatib, Oussama (Hrsg.): *Springer Handbook of Robotics*. Berlin, Heidelberg : Springer, 2008. – URL http://dx.doi.org/10.1007/978-3-540-30301-5. – ISBN 978-3-540-23957-4

[36] Zude, Manuela ; Peeters, Aviva ; Selbeck, Jörn ; Käthner, Jana ; Gebbers, Robin ; Bengal, Alon ; Hetzroni, Amots ; Jaeger-Hansen, Claes ; Griepentrog, Hans-Werner ; Pforte, Florian ; Rozzi, Paolo ; Torricelli, Alessandro ; Spinelli, Lorenzo ; Ünlü, Mustafa ; Kanber, Riza: Methoden für die präzise obstbauliche Produktion. In: *LANDTECHNIK* 67 (2012), Okt., Nr. 5, S. 338–341. – URL https://www.landtechnik-online.eu/landtechnik/article/view/2012-67-5-338-341

A Anhang

A.1 Quellcode der Odometriedifferenz Protokollierung

```
import rospy
import random
from nav_msgs.msg import Odometry, Path
from tf2_geometry_msgs import PoseStamped
import tf2_ros
import matplotlib.pyplot as plt
import math
import csv

class OdomDistance:
    def __init__(self):
        self.tf2Buffer = tf2_ros.Buffer()
        self.tf2listener = tf2_ros.TransformListener(self.tf2Buffer)
        self.ref_topic = rospy.get_param('~ref_topic', '/odom')
        self.topic = rospy.get_param('~topic', '/odom')
        self.csv_filename = rospy.get_param('~csv_file',
                                            'odom_distance.csv')
        self.fig_filename = rospy.get_param('~png_file',
                                            'odom_distance.png')
        self.use_z = rospy.get_param('~use_z', False)
        self.plot_title = rospy.get_param('~plot_title',
                                          f"Fehler {self.topic}")
        self.ref_sub = rospy.Subscriber(self.ref_topic, Odometry,
                                        self.ref_cb, queue_size=10)
        print(f"Subscribing to reference odom {self.ref_topic}")
        self.ref_sub = rospy.Subscriber(self.topic, Odometry,
```

```
                                                self.odom_cb, queue_size=10)
        print(f"Subscribing to reference odom {self.topic}")
        self.ref_coords = None
        self.timestamps = []
        self.distances = []
        self.start_time = None

    def ref_cb(self, msg: Odometry):
        z = msg.pose.pose.position.z
        if not self.use_z:
            z = 0
        self.ref_coords = (msg.pose.pose.position.x,
                           msg.pose.pose.position.y, z)
        self.ref_frame = msg.header.frame_id

        #print(f"{len(self.coords)} Points", end="\r", flush=True)

    def odom_cb(self, msg: Odometry):
        if self.ref_coords is None:
            return
        try:
            if self.tf2Buffer.can_transform(self.ref_frame,
                           msg.header.frame_id, rospy.Time(0)):
                pose = PoseStamped()
                pose.header = msg.header
                pose.pose = msg.pose.pose
                in_ref_frame = self.tf2Buffer.transform(pose,
                                self.ref_frame).pose.position
                if not self.use_z:
                    in_ref_frame.z = 0
                distance = math.sqrt(
                        (self.ref_coords[0] - in_ref_frame.x) ** 2 +
                        (self.ref_coords[1] - in_ref_frame.y) ** 2 +
                        (self.ref_coords[2] - in_ref_frame.z) ** 2
                        )
                if self.start_time is None:
```

```
                    self.start_time = msg.header.stamp.to_sec()
                self.distances.append(distance)
                self.timestamps.append(msg.header.stamp.to_sec()-
                                       self.start_time)
                print(f"Current distance: {distance:.2f} m",
                      end="\r", flush=True)
            else:
                print(f"Can't transform from {self.ref_frame} to
 {msg.header.frame_id}", end="\r", flush=True)
        except:
            print(f"Can't transform from {self.ref_frame} to
 {msg.header.frame_id}", end="\r", flush=True)

    def save_file(self):
        print(f"\nSaving figure to {self.fig_filename}")
        fig, ax = plt.subplots()
        ax.grid(True)
        ax.plot(self.timestamps, self.distances, label=f"
{self.topic}")
        ax.set_ylabel('Abstand zu GNSS (m)')
        ax.set_xlabel('Zeit (s)')
        ax.set_title(self.plot_title)
        plt.savefig(self.fig_filename)
        header = ['time', self.topic]
        with open(self.csv_filename, 'w', encoding='UTF8') as f:
            writer = csv.writer(f)

            writer.writerow(header)
            for i in range(len(self.timestamps)):
                writer.writerow([self.timestamps[i],
                                 self.distances[i]])

if __name__ == '__main__':
    rospy.init_node(f'odom_distance_{random.randint(0,10000)}')
    i = OdomDistance()
    rospy.spin()
```

```
    i.save_file()
```

A.2 Quellcode Punktwolkenzuschnitt

```
import open3d as o3d
import sensor_msgs.point_cloud2 as pc2
import rospy
import numpy as np
import ctypes
import struct
import math
import tf
import tf2_ros
from sensor_msgs.msg import PointCloud2
from pointcloud_processing.srv import CutLatLonCirclePCD,
                                      CutLatLonCirclePCDResponse,
                                      CutLatLonCirclePCDRequest,
                                      CutLocationCirclePCD,
                                      CutLocationCirclePCDResponse
from geometry_msgs.msg import Point, PointStamped

import utm

class PointCloudCutter():
    def __init__(self):
        self.cut_lat_lon_circle_pcd = rospy.Service(
            '/pointcloud_processing/cut_lat_lon_circle_pcd',
            CutLatLonCirclePCD, self.cut_lat_lon_circle_pcd)
        self.cut_lat_lon_circle_pcd = rospy.Service(
            '/pointcloud_processing/cut_location_circle_pcd',
            CutLocationCirclePCD, self.cut_location_circle_pcd)
        self.t = tf.TransformListener()
        self.tf2Buffer = tf2_ros.Buffer()
        self.tf2listener = tf2_ros.TransformListener(self.tf2Buffer)
        self.locations = rospy.get_param(
```

```
                                '/pointcloud_cutter/locations', [])
        self.pc_topic = rospy.get_param(
            '/pointcloud_cutter/pc_topic',
            '/lio_sam/mapping/map_global')
        self.pc_sub = rospy.Subscriber(self.pc_topic, PointCloud2,
                                       self.on_new_pc)
        self.pc_map_pub = rospy.Publisher(f"{self.pc_topic}/map",
                                          PointCloud2, queue_size=1)
        self.raw_xyz = np.empty((0, 3))
        self.raw_rgb = np.empty((0, 3))
        self.no_data = True

    def cut_circle(self, lat, lon, radius):
        if(not self.t.canTransform('utm', self.pc_frame,
                                   rospy.Time(0))):
            return (False,
                f"No transform between 'utm' and '{self.pc_frame}'")
        (easting, northing, zone, _) = utm.from_latlon(lat, lon)
        target = PointStamped()
        target.header.frame_id = 'utm'
        target.point = Point(x=easting, y=northing)

        pose_transformed = self.t.transformPoint(self.pc_frame,
                                                 target)
        center_x = pose_transformed.point.x
        center_y = pose_transformed.point.y
        print(f"Center: ({center_x},{center_y})")
        xyz = np.array([[0,0,0]])
        rgb = np.array([[0,0,0]])
        counter = 0
        checked = 0
        min_dist = 999999999999
        for idx, point in enumerate(self.raw_xyz):
            dist = math.dist([point[0], point[1]],
                             [center_x, center_y])
            if(dist<=radius):
```

```
                xyz = np.append(xyz, [[point[0]-center_x,
                                       point[1]-center_y, point[2]]],
                                       axis = 0)
                rgb = np.append(rgb, [self.raw_rgb[idx]], axis = 0)
                counter += 1
            if(dist< min_dist):
                min_dist = dist
            checked += 1
        print(f"Checked␣{checked}␣points,␣minimum␣{min_dist}␣m")
        return (True, (xyz, rgb, counter))

    def cut_lat_lon_circle_pcd(self, req: CutLatLonCirclePCDRequest):
        if self.no_data:
            return CutLatLonCirclePCDResponse(False,
                        f"No␣data␣on␣topic␣'{self.pc_topic}'␣so␣far")
        (sucess, result) = self.cut_circle(req.lat, req.lon,
                                           req.radius)
        if not sucess:
            return CutLatLonCirclePCDResponse(False, result)
        return CutLatLonCirclePCDResponse(True, )

    def cut_location_circle_pcd(self, req):
        if self.no_data:
            return CutLocationCirclePCDResponse(False,
                      f"No␣data␣on␣topic␣'{self.pc_topic}'␣so␣far")
        self.locations = rospy.get_param(
                             '/pointcloud_cutter/locations', [])
        locations = list(filter(
                        lambda loc: loc['name'] == req.location,
                            self.locations))
        if len(locations) <= 0:
            return CutLocationCirclePCDResponse(False,
                f"Location␣'{req.location}'␣not␣found")
        (sucess, result) = self.cut_circle(locations[0]['lat'],
                                       locations[0]['lon'], req.radius)
        if not sucess:
```

```
            return CutLocationCirclePCDResponse(False, result)
        (xyz, rgb, counter) = result
        (sucess, msg) = self.save_pcd(req.pcdPath, xyz, rgb)
        return CutLocationCirclePCDResponse(sucess,
                f"Saved PCD {msg}, {counter} Points")

    def save_pcd(self, name, xyz, rgb):
        out_pcd = o3d.geometry.PointCloud()
        out_pcd.points = o3d.utility.Vector3dVector(xyz)
        out_pcd.colors = o3d.utility.Vector3dVector(rgb)
        filename = f"/home/david/Documents/HAW/BA/{name}.pcd"
        o3d.io.write_point_cloud(filename, out_pcd)
        return (True, filename)

    def on_new_pc(self,data : PointCloud2):
        #self.lock.acquire()
        gen = pc2.read_points(data, skip_nans=True)
        int_data = list(gen)
        self.raw_xyz = np.empty((len(int_data), 3))
        self.raw_rgb = np.empty((len(int_data), 3))
        self.pc_frame = data.header.frame_id

        for idx, x in enumerate(int_data):
            test = x[3]
            s = struct.pack('>f' ,test)
            i = struct.unpack('>l',s)[0]
            pack = ctypes.c_uint32(i).value
            r = (pack & 0x00FF0000)>> 16
            g = (pack & 0x0000FF00)>> 8
            b = (pack & 0x000000FF)
            # prints r,g,b values in the 0-255 range
            # x,y,z can be retrieved from the x[0],x[1],x[2]
            self.raw_xyz[idx] = [x[0],x[1],x[2]]
            self.raw_rgb[idx] = [r,g,b]
        self.no_data = False
        self.orig_pc = data
```

```
    def timer_publish_pc(self, e):
        if(self.no_data):
            return
        if(self.tf2Buffer.can_transform('map', self.pc_frame,
                                        rospy.Time(0))):
            self.pc_map = self.tf2Buffer.transform(self.orig_pc,
                                                   'map')
            self.pc_map_pub.publish(self.pc_map)

if __name__ == "__main__":
    rospy.init_node('pointcloud_cutter')
    PointCloudCutter()
    rospy.spin()
```

Glossar

Altes-Land Das Obstanbaugebiet Altes-Land liegt am südlichen Ufer im Urstromtal der Elbe westlich von Hamburg.

ESTEBURG ESTEBURG ist ein Markenname des Obstbauversuchsring des Alten Landes e.V. (OVR). Mit diesem Name wird das Obstbauzentrum Jork (Kompetenzzentrum für den norddeutschen Obstbau) mit den Organisationen OVR, Obstbauversuchsanstalt der LWK Niedersachsen (OVA) und der Öko-Obstbau Norddeutschland Versuchs- und Beratungsring e.V. (ÖON) bezeichnet..

Fraunhofer IFAM Fraunhofer-Institut für Fertigungstechnik und Angewandte Materialforschung (IFAM).

Gasse Der Bereich zwischen zwei Baumreihen wird als Fahrgasse oder auch in Kurzform Gasse bezeichnet.

Vorgewende Als Vorgewende wird im Obstanbau der Bereich an den Enden der Baumreihen bezeichnet, der den Verkehr quer zu den Baumreihen und Wendemanöver ermöglicht. Dieser Bereich ist je nach verwendeten Maschinen zwischen 7 und 12 Meter breit..

Erklärung zur selbstständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Hamburg	10. Juni 2023	[redacted]
Ort	Datum	Unterschrift im Original