Institut für Informatik
AG Autonome Robotik

Masterarbeit

Loop Closure in TSDF basiertem SLAM

Patrick Hoffmann

Oktober 2022

Erstgutachter: Prof. Dr. Mario Porrmann
Zweitgutachter: Alexander Mock

Zusammenfassung

Die vorliegende Arbeit beschäftigt sich mit der Implementierung und Konzeption einer Lösung zur Detektion von **Schleifenschlüssen** in einem auf **Truncated Signed Distance Funtion (TSDF)** basierenden **Simultaneaous Localization and Mapping (SLAM)** Ansatz und der nachfolgenden Optimierung der Robotertrajektorie und TSDF-Karte. Zur Optimierung der Trajektorie des Robters nach der Identifikation eines Schleifenschlusses kommt die Bibliothek **GTSAM** zum Einsatz. Es wird evaluiert ob und unter welchen Voraussetzungen eine Nachbearbeitung auf Basis einer fertigen TSDF Karte mit zugehöriger initialer Trajektorie möglich ist und zusätzlich der Einsatz in einem inkrementellen SLAM Ansatz geprüft. Darüber wird untersucht, inwiefern ein Teilupdate der TSDF basierten Karte möglich ist.

Abstract

This paper deals with the implementation and design of a solution for the detection of **loop closures** in a **Truncated Signed Distance Function (TSDF)** based **Simultaneaous Localization and Mapping (SLAM)** approach and the subsequent optimization of the robot trajectory and TSDF map. The **GTSAM** library is used to optimize the robot's trajectory after identifying a loop closure. It is evaluated whether and under which conditions a postprocessing based on a finished TSDF map with associated initial trajectory is possible and additionally the use in an incremental SLAM approach is examined. Furthermore, it will be investigated to what extent a partial update of the TSDF based map is possible.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Diese Arbeit beschäftigt sich mit der Konzeption und Implementation einer Lösung zum *Loop Closure (Schleifenschluss)* Problem in *Truncated Signed Distance Fields* (TSDF) basiertem *Simulatneous Localization and Mapping* (SLAM). Diese Arbeit setzt auf den Implementationen und Konzepten von Eisoldt et al. [4] auf, die einen TSDF basierten, hardware-beschleunigten SLAM Ansatz (*HATSDF-SLAM*) implementiert und dessen Effizienz und Funktionalität verfiziert haben.

Unter SLAM versteht man den Prozess der Erstellung einer Karte einer unbekannten Umgebung, bei gleichzeitiger Positionsbestimmung innerhalb dieser unbekannten Umgebung. Das SLAM Problem ist ein *Henne-Ei-Problem*, da eine vollständige Karte benötigt wird, um die Pose eines Roboters akkurat bestimmen zu könnnen, während auf der anderen Seite eine akkurate Pose benötigt wird, um eine vollständige Karte aufbauen zu können. Für die Lösung dieses Problems gibt es sowohl in 2D als auch 3D verschiedene Lösungsansätze, die in Abschnitt 1.1 grob dargestellt sind. Die in diesem Exposé vorgestellte Arbeit beschränkt sich auf 6D SLAM. Unter einem Schleifenschluss versteht man in der mobilen Robotik den Prozess, geschlossene Kreise in abgelaufen Pfaden durch die im SLAM-Prozess erschlossene, zunächst unbekannte Umgebung, zu identifizieren. Nach der Identifikation dieser Kreise können Fehler, die sich im Kartierungsprozess akkumuliert haben, nachträglich berichtigt werden. Dies sorgt für eine konsistenteren Pfad und damit verbunden für eine akkuratere Karte. Als Basis für dieses Problem dient der GraphSLAM Ansatz von Borrmann et al. [3], die einen Pose-Graphen erstellen, in den aufeinanderfolgende Positionen während der Kartierung nacheinander eingefügt werden. Zur Registierung aufeinanderfolgender Datenscans verwenden Borrmann et al. den *Pairwise Iterative Closest Point Algorithm* (Pairwise ICP). Zur Detektion eines Schleifenschlusses wird eine einfache Distanz-Heuristik verwendet. Wichtige Voraussetzung für die Optimierung der Posen im Graphen ist eine Assoziation der Posen mit den zugehörigen Umgebungsdaten. Im Falle von Borrmann et al. sind dies jeweils die zum Zeitpunkt aufgenommenen Punktwolken. Diese Assozation ist bei TSDF basiertem SLAM, das eine diskretisierte TSDF-Karte verwendet, nicht möglich, solange nicht zusätzlich die entsprechenden Punktdaten abgespeichert werden. Wird ein Schleifenschluss detektiert, müsste dann jedoch die gesamte Karte neu erstellt werden. Dies ist speicher- und

zeitaufwändig. Dieses Problem markiert den Hauptbeitrag dieser Arbeit. Es gilt zu bestimmen, welcher Teil der TSDF Karte mit welcher Pose im Graph assoziiert werden kann, um anschließend die bereits existierende Karte zu modifizieren und zu optimieren.

1.1 Stand der Forschung

Simulataneous Localization and Mapping (SLAM) ist eines der größten Forschungsgebiete in der mobilen Robotik. Die Forschungen zu diesem Problem reichen zurück bis vor die Jahrtausendwende. Einer der bekanntesten Lösungen zur Registierung dreidimensionaler Daten ist der ICP Algorithmus nach Besl & McKay [2], der in einer Abwandlung ebenfalls von Borrmann et al. im GraphSLAM [3] verwendet wird. Bis heute gibt es zahlreiche Implementationen dieses Algorithmus.

Erste Forschungen mit TSDF basiertem SLAM stammen aus dem letzten Jahrzehnt. Izadi et al. [5] nutzen ein TSDF-Voxelgrid und eine Kinect-Tiefenkamera, um Umbegungen zu kartieren, während ein Nutzer die Kinect Kamera durch die Umbebung schwenkt. Whelan et al. [9] optimieren diesen Ansatz durch Nutzung eines Ringbuffers zur Kartierung großer Umgebungen. Im Gegensatz zu genannten TSDF Verfahren, nutzen Eisoldt et al. [4] einen Hardware beschleunigten TSDF basierten SLAM Ansatz, sowie einen 3D-Laserscanner anstelle einer Tiefenkamera.

Borrmann et al. [3] schufen in ihrer Arbeit eine gute Grundlage für die Integration von Schleifenschlüssen in SLAM-Verfahren auf Basis von Pose-Graphen, die bis heute vielfach verwendet wird.

Insert more tsdf based approaches from [8]

1.2 Motivation

Ziel und Zweck dieser Arbeit ist die Lösung des Schleifenschluss Problems für TSDF basierte SLAM Verfahren. Als Basis dient der Ansatz von Eislodt et al. [4], der - obgleich performant und funktional - wie die meisten SLAM Verfahren bei der Kartierung großer Umgebungen stark anfällig für Drift ist. Um diesem Problem entgegen zu wirken, soll durch die Integration von Schleifenschlüssen in TSDF-Karten die Kartierung größerer Umgebungen ermöglicht werden. Um dieses Ziel zu erreichen, gilt es Datenassoziationen zwischen Posen und Zellen in der diskretisierten TSDF Karte herzustellen. In einer ersten Implementation soll dazu mittels Ray-Tracing eines simulierten Laserscans und der Detektion von Schnittpunkten mit der TSDF-Karte eine passende Indizierung der Zellen ermöglicht werden.

Diese Arbeit soll als Proof-of-concept für weitere Arbeiten auf diesem Gebiet dienen. Dazu werden die entwickelten Ansätze in einem Post-Processing Schritt auf die vom Eisoldt et al. [4] erstellte Karte, so wie den Pfad, den es noch zu extrahieren gilt, angewandt.

In einem Ausblick gilt es zu analysieren, ob die Implementationen sinnvoll beschleunigt werden können, um sie in ein Live Karierungssystem, wie HATSDF-Slam einbauen zu können, um zur Laufzeit Schleifen zu erkennen und die Karte zu optimieren.

Die Implementation dieses Prototyps wird ausschließlich in Software realisiert, allerdings erfolgt eine Untersuchung des Software-Prototyps auf Potenziale zur Hardware-Beschleunigung um

Raum für Optimierungen im Rahmen zukünftiger Arbeiten zu eröffnen.

TODO: Update der Motivation? Untersuchung, ob das Loop Closing als Post-Processing angewandt werden kann, gegebenenfalls Integration in inkrementellen SLAM Möglichkeit schaffen mit beliebigen SLAM Verfahren koexistieren zu können (LC) Schaffen einer API? Untersuchen welche Art des Kartenupdates sinnvoll ist und welches nicht Untersuchen, ob der Aufwand die Karte im Teil zu updaten gerechtfertigt ist, Untersuchung von Fallstricken

1.3 Ansatz

Beschreiben, wie der initiale Aufbau dieser Arbeit ist:

1. Herangehensweise, wie soll das Problem gelöst werden? 2. Offenheit des Themas klar herausstellen 3. Mögliche Fallback's aufzeigen, falls sich herausstellt, dass das Thema so in der Bearbeitungszeit nicht möglich ist 4. Deutlich machen, dass die erste Herangehensweise dokumentiert wird 5. Erklären, dass Fallstricke und Probleme erläutert werden 6. Lösungsansätze für zukünftige Arbeiten definieren 7. Kenntlich machen, warum das Problem ïnkrementelles Map Update"nicht in der Zeit lösbar gewesen wäre

1.4 Fallback

Kapitel 2

Bibliotheken und Frameworks

2.1 ROS Framework

- was ist ros, worfür wird es verwendet - kurz schreiben über publisher und subscriber - verwendete Komponenten - Visualisierung´

2.1.1 RViz

- Rviz als Visualisierungstool erlären - Beispiel für Visualisierung von Transformationen etc. - frames erklären

2.1.2 Bondcpp

- Verbund erklären - warum optimal für ein publisher subscriber szenario, bei dem keine Daten verloren gehen sollen

2.2 Eigen

-¿ grundlegende Datenstrukturen die verwendet werden -¿

2.3 HDF5

-¿ Repräsentation der TSDF Datenstruktur

2.4 Pointcloud Library

2.5 LVR2 Framework zur Oberflächenrekonstruktion

-¿ Evaluation des SLAM und des Updates der Kartenrepräsentation

Kapitel 3

Grundlagen

3.1 Mathematische Grundlagen

Diese Sektion beschreibt die wesentlichen mathematischen Konzepte, die in dieser Arbeit zur Verwendung kommen.

3.1.1 Konventionen

Transformationen werden im Folgenden folgendermaßen betitelt:

Konventionen einfügen

3.1.2 Koordinatensysteme

Ein wesentliches Konzept bei der Verarbeitung räumlicher Daten ist die Verwendung von Koordinatensystemen. Sie werden genutzt um die Positionen von Daten und Objekten im Raum zu beschreiben. Koordinatensysteme können für sich alleine stehen oder relativ zu anderen Koordinatensystemen. Im Dreidimensionalen besitzt ein Koordinatensystem drei verschiedene Achsen (x, y und z-Achse), die jeweils im 90 Grad Winkel zueinander ausgerichtet sind. Rotationen im Raum werden beschrieben als Rotationen um die jeweiligen Achsen. Welche Achse in welche Richtung zeigt ist nicht eindeutig definiert. Es gibt jedoch verschiedene Standards beziehungsweise Konventionen wie das links- oder rechtshändische Koordinatensystem. In ROS wird konventionell ein rechtshändisches Koordinatensystem, abgebildet in Abbildung 3.1 dargestellt. Dies wird im Folgenden ebenfalls als Standard verwendet.

In der Robotik kommt es häufig vor, dass verschiedene (bewegliche) Komponenten relativ zu einem globalen Bezugssystem oder relativ zueinander beschrieben werden müssen. Die Bewegung eines übergeordneten Bezugssystems kann implizit für eine Veränderung der relativ zu diesem Bezugssystem platzierten Systeme führen. Dies lässt sich anhand eines Arm-Roboters zeigen, der mehrere miteinander verbundene Gelenke hat. Bewegt sich ein Gelenk werden automatisch auch die am Arm weiter außen befindlichen Gelenke mitbewegt. Aus Sicht des bewegten, übergordneten Bezugssystems hat sich die Position der untergeordneten Gelenke nicht verändert, aus Sicht des globalen Bezugssystems, wie zum Beispiel dem Montierungspunkt des Roboters, allerdings schon.

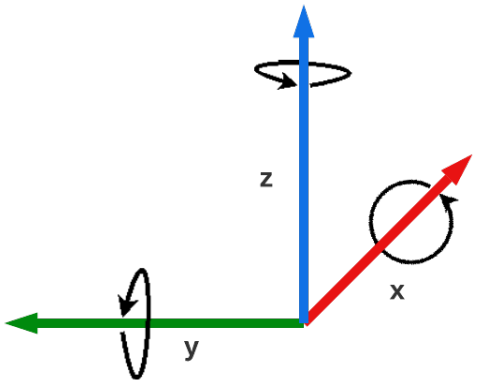

Abbildung 3.1: Schematische Darstellung der Konvention für Koordinatensysteme im ROS Framework. Die z-Achse zeigt nach oben, die x-Achse nach vorne und die y-Achse nach links. Die Rotation um die Achsen ist entsprechend der Konvention im Uhrzeigersinn.

In ROS werden Anhängigkeiten zwischen Bezugssystemen in einer Baumstruktur, genannt *transformation tree (tf-tree)* dargestellt. Die Wurzel dieser Baumstruktur ist das globale Bezugssystem, wie zum Beispiel der Ursprung einer globalen Karte oder der Startpunkt der Trajektorie eines Roboters. Das globale Bezugssystem kann beliebig gewählt werden. Auf diese Weise kann ein Koordinatensystem, welches relativ zu einem anderen gelegen ist im Baum als Kindknoten seinem Bezugssystem untergeordnet werden. Es wird nur die relative Transformation (s. Kapitel 3.1.2) zwischen den Systemen im Baum gespeichert. Dies hat den Vorteil, dass bei der Bewegung eines Systems die untergordneten Systeme nicht ebenfalls verändert werden müssen, da deren Transformationen relativ zum bewegten Bezugssystem angegeben sind und nicht global zur Wurzel des Baumes. Abbildung 3.2 zeigt ein Beispiel für einen Roboter mit mehreren voneinander abhängigen System.

Auch räumliche Daten wie zum Beispiel Punktenwolken aus Laserscannern können relativ zu verschiedenen Koordinatensystemen gesehen werden. So kann es nützlich sein die Punktwolke relativ zum Koordinatensystem des Scanners oder innerhalb des globalen Koordinatensystems zu betrachten. Um zwischen den Koordinatensystemen zu wechseln wird eine Koordinatensystem-transormation. Diese werden im folgenden Kapitel behandelt.

Transformationen

Eine Koordinatensystemtransformation ist ein Sonderfall einer mathematischen Transformation, die eine Menge X auf sich selbst abbildet:

$$f : X \to X \tag{3.1}$$

Eine Koordinatensystemtransformation beschreibt die Differenz zwischen zwei unterschiedlichen Koordinatensystemen und enthält sowohl die Translationsdifferenz als auch die Rotationsdifferenz. Zur Berechnung dieser Transformation zwischen zwei beliebigen Koordinatensystem C_1 und

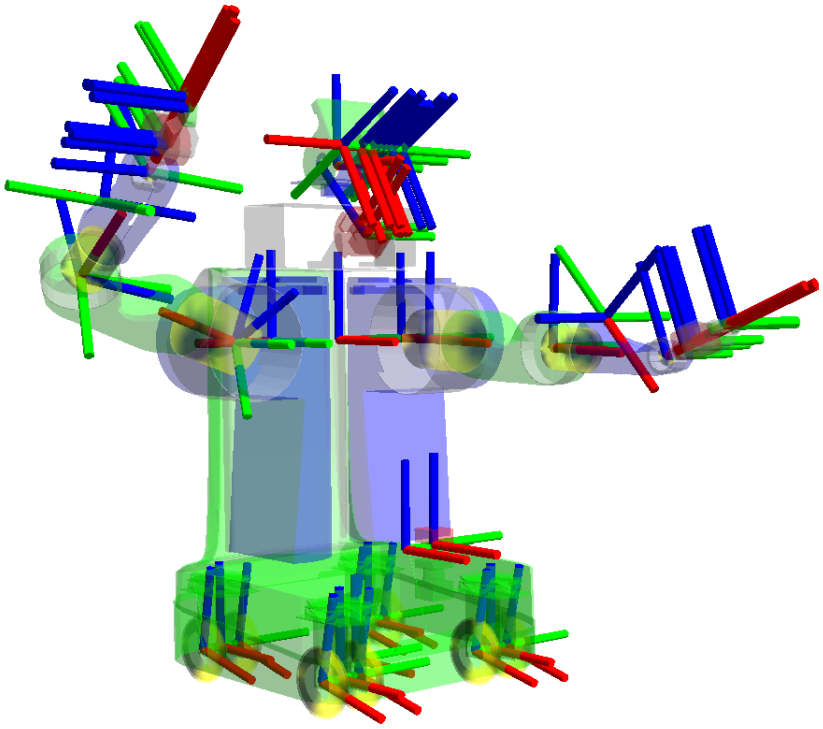

Abbildung 3.2: Schematische Darstellung eines Roboters und dessen beweglicher Teile. Die Ausrichtung der jeweiligen Gelenke wird mit einem lokalen Koordinatensystem beschrieben. Die globale Position einzelner Teile kann durch eine Verkettung der relativen Transformation in Richtung der Wurzel des Baumes bestimmt werden. Bild aus: [6]

C_2 wird die absolute Translation und Rotation beider Koordinatensysteme zum Ursrungskoordinatensystem, wie zum Beispiel den Urpsrung einer Umgebungskarte, benötigt. Durch diese Rotations und Translationskomponenten beschreibt sich die absolute Position und Rotation der Koordinatensysteme im Raum aus Sicht des Ursprungskoordinatensystems C_{MAP}. Diese absolute Position und Rotation ist die Transformation von den jeweiligen Koordinatensystemen ins Urpsrungskoordinatensystem.

Es existieren diverse Darstellungsweisen für Koodinatensystemtransformationen. Die intuitivste Weise der Darstellung ist die Darstellung als Vektor. In folgendem ist die Transformation vom Koordinatensystem C_X ins Koordinatensystem C_{MAP} dargestellt. Die Variablen t_i bezeichnen dabei die Translationskomponenten und die Variablen r_i die Rotationskomponenten um die jeweiligen Achsen.

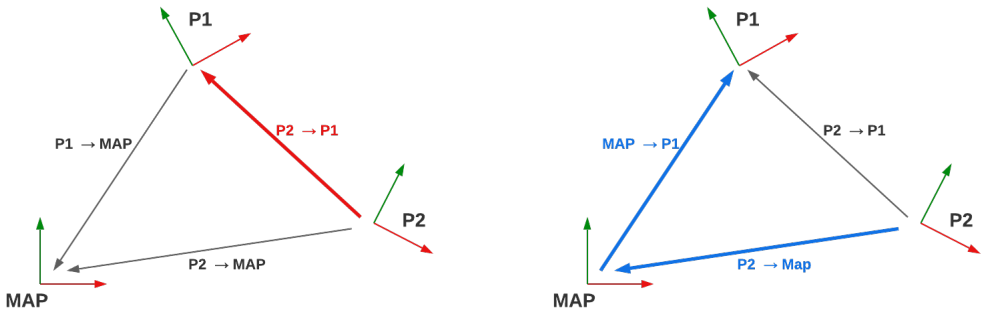

Abbildung 3.3: Schematische Darstellung der bestimmung einer Transformation zwischen zwei Roboter-Posen P_1 und P_2, hier zur Vereinfachung dargestellt in 2D. Gesucht ist die Transformation $(T_{P_2 \to P_1})$, dargestellt im linken Teil der Abbildung als roter Pfeil. Diese kann implizit bestimmt werden durch eine Verkettung der Transformationen $T_{P_2 \to MAP}$ und $T_{MAP \to P1}$, hier dargestellt im rechten Teil der Abbildung in blau. Die Transformation $T_{MAP \to P1}$ ist dabei nicht explizit gegeben. Sie kann berechnet werden durch eine Inversion der Transformation $T_{P1 \to MAP}$. Die finale Gleichung zur Berechnung der relativen Transformation ist dargestellt in Gleichung 3.3.

$$T_{C_X \to C_{MAP}} = \begin{pmatrix} t_x \\ t_y \\ t_z \\ r_x \\ r_y \\ r_z \end{pmatrix} \tag{3.2}$$

Neben der Vektordarstellung kann eine Transformation zusätzlich als eine $4x4$ Matrix dargstellt werden. Diese Darstellungsweise hat den Vorteil, dass die Transformation direkt per Matrixmultiplikation auf Daten wie um homogene Koordinaten erweiterte Punktdaten angewandt werden kann. Auch die direkt Kombination verschiedener Transformationen ist durch eine Matrixmultiplikation möglich. Hier gilt es zu beachten, dass Matrixmultiplikation nicht kommutativ ist und ein Vertauschen der Reihenfolge bei der Multiplikation zu unterschiedlichen Ergebnissen führen kann. Bei einer Verkettung von Transformationen durch Multiplikation wird die Matrix zuerst angewandt, welche am Ende der Multiplikation steht.

In dieser Arbeit werden Transformationen zum Beispiel verwendet um zu bestimmen, wie sich ein Roboter zwischen zwei Messungen bewegt hat. Diese Transformationen beschreiben relative Differenzen zwischen Roboterpositionen im Raum. Diese Roboterpositionen werden auch **Posen** genannt. Eine Pose ist dabei eine meist absolute Beschreibung der Translation und Rotation eines Roboters zu einem gewissen Zeitpunkt t aus Sicht des Ursprungskoordinatensystems wie zum Beispiel dem Map-Ursprung C_{MAP}.

TODO:

Abbildung 3.3 zeigt die Mathematik hinter der Berechnung der Posedifferenz exemplarisch. Es wird deutlich, dass eine gesuchte Transformation aus dem Koordinatensystem von Pose P_2 in das Koordinatensystem von Pose P_1 ($T_{P_2 \to P_1}$) gegeben ist durch:

$$(T_{P_2 \to P_1}) = (T_{P_1 \to MAP})^{-1} * T_{P_2 \to MAP} \tag{3.3}$$

Basierend auf den erläuterten mathematischen Grundlagen wird in Kapitel 3.3 die Grundlagen von SLAM, insbesondere von TSDF basierten SLAM Verfahren, erörtert. Zuvor werden in nachfolgender Sektion die Eigenschaften und Anwendungsbereiche der TSDF beschrieben.

3.2 TSDF

Zur Lösung des **Simulatneous Localization and Mapping (SLAM)** (siehe Kapitel 3.3) Problems in unbekannten Umgebungen wird im Regelfall eine Form der Kartenrepräsentation und die Algorithmik benötigt sich auf Basis der Kartenrepräsentation zu lokalisieren und diese im Anschluss aktualisieren zu können. Bei vielen Lösungsansätzen wie zum Beispiel einer inkrementellen **Registrierung (siehe Kapitel 3.3)** mit dem **Iterative Closest Point (ICP)** [1] oder **Generalized Iterative Closest Point** [7] Algorithmus werden als Kartenrepräsentation registrierte Punktwolken verwendet. Punktwolken stellen dabei keine geschlossenen Oberflächenrepräsentationen dar und benötigen viel Speicher im Gegensatz zu einigen geschlossenen Repräsentationen wie aus den Punktwolken generierten Dreiecksnetzen. Ein großer Nutzen einer solchen Repräsentation ist die vereinfachte Lokalisierung und Navigation auf Basis der Kartenrepräsentation.

Eine weitere Form der geschlossenen Oberflächenrepräsentation der Umgebung sind **Signed Distance Functions (SDF)**. Im Gegensatz zu Dreiecksnetzen sind die SDF implizite, volumetrische Beschreibung der Oberfläche [8]. Sie beschreiben die Oberfläche nicht direkt, sondern den Raum um die Oberfläche herum. Die **Signed Distance** ist die orthogonale metrische Distanz eines beliebigen Punktes p zur Oberfläche räumlicher Daten, wie zum Beispiel Punktwolken. Diese Distanz kann sowohl negativ, als auch positiv sein. Unterschieden wird zwischen dem Innenbereich und Außenbereich. Abbildung 3.4 zeigt ein zweidimensionales, schematisches Beispiel für eine diskretisierte TSDF nach Abtasten der Oberfläche einer unbekannten Umgebung durch einen Laserscanner.

Die **Truncated Signed Distance Funtion (TSDF)** ist eine Unterklasse der SDF. Sie betrachtet die Distanz zur Oberfläche nur bis zu einer maximalen Distanz $tsdf_{max}$, auch $tau\,(\tau)$) gennant[4]. Alle Werte die weiter von der Oberfäche entfernt oder unbekannt sind, erhalten als Wert *tau* selbst. Dies spart Rechenaufwand, da nur die Werte in direkter Nähe zur Oberfläche angepasst werden müssen. In Gebieten, in denen der TSDF-Wert *tau* entspricht, kann jedoch keine Aussage getroffen werden, wo die nächste Oberfläche ist, oder wie weit sie entfernt ist. Es ist lediglich bekannt, dass die betrachtete Position nicht in direkter Nähe zur Oberfläche befindlich und mindestens *tau* entfernt ist. Das Intervall möglicher Werte der TSDF ist:

$$I = [-\tau, \tau] := \{x \in \mathbb{R} | \tau > 0\} \tag{3.4}$$

TODO:

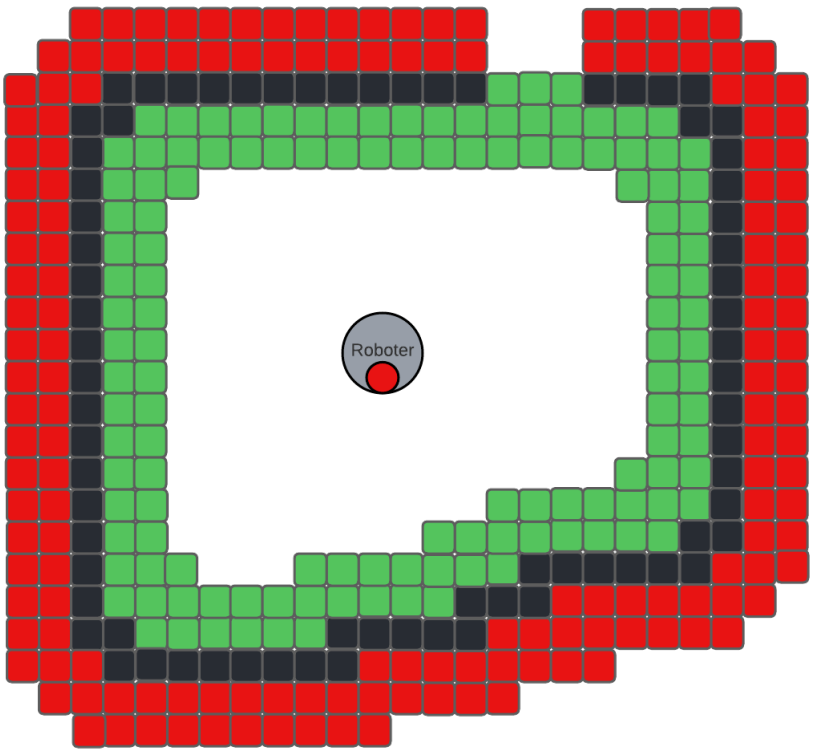

Abbildung 3.4: Schematische Darstellung einer diskretisierten 2D TSDF Karte. Abgebildet sind nur Voxel mit TSDF-Werten im Intervall $]-\tau, \tau[$. In der Mitte der Karte befindet sich ein Roboter mit einem Laserscanner (hier dargestellt in rot). Der Roboter befindet sich zum Beispiel in einem Raum. Der Innenbereich (des Raumes) mit positiven TSDF-Werten ist hier dargestellt in grün, der Außenbereich mit negativen TSDF-Werten in rot. Die Oberfläche, in deren Umgebung die TSDF-Werte nahezu Null sind, ist abgebildet in schwarz.

Über SDF und TSDF können kontinuierliche Karten erstellt werden. Da kontinuierliche Karten aber unendlich viel Speicherplatz benötigen, wird der Raum diskretisiert. Die Diskretisierung erfolgt durch eine Aufteilung der Umgebung in **Voxel** mit definierbarer, fester Seitenlänge v_{res} [9] [4]. Jeder Voxel enthält einen approximierten (T)SDF-Wert. Diese Form der Darstellung kann auch als pseudo-kontinuierlich angesehen werden, da durch eine Approximation über benachbarte Zellen ein approximierter TSDF-Wert für jeden beliebigen Raumpunkt berechnet werden kann. Dadurch sind TSDF basierte Karten ideal, um mittels des **Marching Cubes Algorithmus** [**?**] eine polygonale Netz-Repräsentation, wie zum Beispiel ein Dreiecksnetz generieren zu können. Dieses kann zum Beispiel als optische Referenz für die Qualität der Karte verwendet werden.

Eisoldt et al. [4] basieren ihren SLAM Ansatz auf einer diskreten, inkrementell erweiterten TSDF Karte. Zur Registierung (vergleiche Kapitel 3.3) verwenden sie ebenfalls die TSDF-Karte. Punktwolken werden dabei mit einer **Point-to-TSDF** Strategie an die TSDF Karte registiert [4]. Diese Arbeit beschäftigt sich mit einer möglichen Integration von Schleifenschlüssen in einen auf

einer TSDF Karte basierenden Ansatz wie vorgestellt von Eisoldt et al. [4]. Ziel ist die Korrektur von Fehlern bei der Registrierung und der damit Verbundenen Korrektur der TSDF-Karte. Dies ist in Kapitel 4 und 5 beschrieben.

Nachfolgende Sektion behandelt des Thema SLAM und gibt einen groben Überblick über einige SLAM Ansätze.

3.3 SLAM

Diese Sektion befasst sich mit den Grundlagen von SLAM, stellt heraus welche Varianten von SLAM Verfahren es gibt und wie sich TSDF basierte Verfahren, insbesondere der HATSDF-SLAM Ansatz von Eisoldt et al. **??** von diesen unterscheidet.

SLAM ist der Prozess der simultanen Generierung einer Karte einer unbekannten Umgebung und der Lokalisierung innerhalb dieser Karte beziehungsweise Umgebung. SLAM ist ein *Henne-Ei-Problem*, da auf der einen Seite eine vollständige Karte benötigt um die Pose des Roboters akkurat zu bestimmen, auf der anderen Seite allerdings eine akkurate Posenhistorie benötigt wird um eine gute Karte der Umgebung aufbauen zu können. Eine grobe Übersicht über existierende SLAM Verfahren liefert der Stand der Forschung in Kapitel **??**. Im Folgenden werden einige der genannten Verfahren erneut aufgegriffen und erläutert. Schlussendlich wird erklärt, welche der SLAM Verfahren für diese Masterarbeit von Interesse sind und wie sie genutzt werden.

Der **Incremental Closest Points (ICP)** Algorithmus nach Besl und McKay [1] ist ein Algorithmus zur **Registrierung** von Punktwolken. Als **Registierung** wird der Prozess der Zusammenführung von Punktwolken bezeichnet, die von unterschiedlichen Orten aus aufgenommen werden. Um Punktwolken möglichst gut zusammenzuführen, wird versucht die aus den Laserscans entstandenen Punktwolken maximal zu überlappen. Dieser Prozess wird auch **Scan Matching** genannt. Beim Scan Matching wird zwischen dem **Model** und dem **Scan** unterschieden. Als Scan bezeichnet werden die Daten, die an das Model registiert werden sollen. Das Ergebnis des Scan Matching ist eine Approximation der Transformation $T_{Scan \rightarrow Model}$ zwischen den Posen von denen aus die Punktwolken aufgenommen wurden. Damit untersucht werden kann, wie gut diese Approximation ist, liefern ICP und verwandte Algorithmen wie zum Beispiel **Generalized Incremental Closest Points** [7] ein Maß für die Genauigkeit der Approximation. Dies ist im Fall von ICP und GICP der sogenannte **Fitness-Score**. Er beschreibt die durchschnittliche quadrierte Distanz zwischen den **Nearest Neighbors (nächsten Nachbarn)** der Punktwolken nach Anwendung der approximierten Transformation $T_{Scan \rightarrow Model}$ der Scanpunktwolke in das Koordinatensystem der Modelpunktwolke. Er gibt dementsprechend an, wie groß die durschnittliche quadrierte Distanz eines Punktes aus der Modelpunktwolke zum euklidisch nächsten Punkt der transformierten Scanpunktwolke ist.

Sowohl ICP, als auch GICP sind inkrementelle Algorithmen, dass heißt sie nähern sich inkrementell einem Optimum immer weiter an. Dabei wird die approximiert Transformation jeweils um ein δT verändert. Fällt dieses δT in einer Iteration unter einen vom Benutzer gewählten Schwellwert, oder wird eine maximale Anzahl an Iterationen erreicht,bricht der Algorithmus ab und gibt die finale Transformation zurück. Genanntes Optimum ist dabei im Regelfall allerdings kein globales,

sondern lediglich ein lokales Optimum aus dem weder ICP noch GICP herauskommen, sobald sie hineingeraten. Aus diesem Grund gilt es die jeweiligen Ausgaben der Algorithmen zum Beispiel basierend auf dem resultierenden Fitness-Score zu analysieren.

Beide Algorithmen werden in Kapitel 3.4 und Kapitel 4 in Bezug auf die Identifikation von Loop-Closures evaluiert.

Varianten (auf die eingegangen wird):

ICP Graph-SLAM -¿ Global Relaxation

1. Grundlagen von SLAM beschreiben -¿ Varianten des SLAM -¿ Bezug zu HATSDF-SLAM 2. Vorraussetzungen (Repräsentationen für Posen (Pfad) und Umgebung) 3. Überleitungen in weitere Sektionen machen -¿ Loop Closure als mögliche Verbesserung des SLAM -¿ TSDF als Kartenrepräsentation -¿ auf Vorteile von TSDF eingehen (z.B. einfache Integration in Marching Cubes Algorithmus)

3.4 Loop Closure

Warum wird Loop Closure benötigt? Welchen Mehrwert gibt es? Wie wäre ein grundlegendes vorgehen? verweisen auf Loop-Closure Kapitel

Kapitel 4

Loop Closure

Diese Sektion baut auf den Grundlagen aus Sektion 3.4 auf.

4.1 Detektion

Beschreibung der Detektion (bildhaft), Erklärung von Kandidaten und Filterung Beschreibung des optionalen Sichtbarkeitskriteriums

4.2 Graphenoptimierung

Bezug zu GTSAM Library Beschreibung aufnehmen. Wichtigste verwendete Funktionen bennen. Verweis auf GTSAM Paper. Erklärung von Faktorgraphen und Faktoren. Erklärung der Genutzten datenstrukturen und optimizer. Erklärung von noise constraints (Unsicherheiten)

4.3 Optimierungen

1. Vorregistrierung

2. Filtern der Punktwolke

3. Unterschiedliche Scan-Matching Varianten 1. ICP 2. GICP 3. Kurz auf Teaser++ eingehen 4. VGICP

-¿ Analyse des Scan Matchings bezogen auf 1. Vorregistrierung, 2. LC Detektion Matching

4.4 Datensätze

Verwendete Datensätze und herausforderungen herausstellen

4.4.1 Hannover1

Herausforderungen:

- Datensatz allgemein: falsches Koordindatensystem -¿ Transformation beschreiben

- extrem fehlerbehaftete Rotation in zweitem Kreis (Lösung: Vorregistrierung ICP) (Scan-Matching bekommt extrem divergierte Wolken nicht mehr aufeinander)

- problematisch: fehlerhafte LC Optimierung durch schlechtes Scan Matching (Grund: Feature-armer Flur) -¿ mögliche (noch zu entwickelnde) Lösung: LC's auf Linien gesondert betrachten -¿ Identifikation eines LC auf Linien -¿ Betrachtung der Scan Matching Transformation - wenn aufGeraden kann die Transformation die beiden LC-Punkte nicht vollständig zusammen ziehen

4.4.2 Maps

Kapitel 5

Map Update

Kapitel 6

Ausblick

Literaturverzeichnis

[1] Besl, P.; McKay, N.: A Method for Registration of 3-D Shapes. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14 (1992), Februar, Nr. 2, S. 239–256

[2] Besl, Paul J.; McKay, Neil D.: Method for registration of 3-D shapes. In: *Sensor fusion IV: control paradigms and data structures* Bd. 1611 Spie, 1992

[3] Borrmann, Dorit; Elseberg, Jan; Lingemann, Kai; Nüchter, Andreas; Hertzberg, Joachim: Globally consistent 3D mapping with scan matching. In: *Robotics and Autonomous Systems* 56 (2008), Nr. 2, S. 130–142

[4] Eisoldt, Marc; Flottmann, Marcel; Gaal, Julian; Buschermöhle, Pascal; Hinderink, Steffen; Hillmann, Malte; Nitschmann, Adrian; Hoffmann, Patrick; Wiemann, Thomas; Porrmann, Mario: HATSDF SLAM – Hardware-accelerated TSDF SLAM for Reconfigurable SoCs. In: *2021 European Conference on Mobile Robots (ECMR)*, 2021

[5] Izadi, Shahram; Kim, David; Hilliges, Otmar; Molyneaux, David; Newcombe, Richard; Kohli, Pushmeet; Shotton, Jamie; Hodges, Steve; Freeman, Dustin; Davison, Andrew u. a.: Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In: *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 2011

[6] Schultz, Jarvis: *tf*. `http://wiki.ros.org/tf`. – zuletzt abgerufen am: 09.10.2022

[7] Segal, Aleksandr; Haehnel, Dirk; Thrun, Sebastian: Generalized-icp. In: *Robotics: science and systems* Bd. 2 Seattle, WA, 2009

[8] Werner, Diana; Al-Hamadi, Ayoub; Werner, Philipp: Truncated signed distance function: experiments on voxel size. In: *International Conference Image Analysis and Recognition* Springer, 2014

[9] Whelan, Thomas; Kaess, Michael; Fallon, Maurice; Johannsson, Hordur; Leonard, John; McDonald, John: Kintinuous: Spatially extended kinectfusion. (2012)

Endpage für Unterschrift einbauen