UNIVERSITÄT OSNABRÜCK

Institut für Informatik
AG Wissensbasierte Systeme

Bachelorarbeit

Global konsistente 3D Kartierung am Beispiel des Botanischen Gartens in Osnabrück

Dorit Borrmann
Jan Elseberg

06. Oktober 2006

Erstgutachter: Prof. Dr. Joachim Hertzberg
Zweitgutachter: Prof. Dr. Werner Brockmann

Zusammenfassung

Für einen autonomen mobilen Roboter, der eine zunächst unbekannte Umgebung durchquert, um dort verschiedene Aufgaben zu erledigen, ist es unter Umständen von Vorteil, eine Karte dieser Umgebung erstellen zu können. Dies wird häufig dadurch bewerkstelligt, separat aufgezeichneten Teilaufnahmen der Umwelt, wie zum Beispiel Laserscans, aneinander zu fügen. Bei einer sequentiellen Vorgehensweise, also wenn eine neue Aufnahme immer nur an die vorherige angepasst wird, führen kleine Fehler in der Regel zu einem großen globalen Fehler. Um ein inkonsistentes Modell der Umgebung zu vermeiden, haben Lu und Milios [18] einen Algorithmus zur gleichzeitigen Zusammenführung von 2D-Laserscans sowie von Odometriedaten zu einer global konsistenten Karte entwickelt. Hierbei werden die Posen des Roboters als Zufallsvariablen modelliert, für die, durch die Maximum-Likelihood Methode, Schätzer gefunden werden.

Die Formulierung des Algorithmus für 2-dimensionale Daten wird in dieser Arbeit erweitert, um den Algorithmus auch für 3D-Laserscans nutzbar zu machen. Der Übergang von zwei auf drei Dimensionen führt dabei gleich drei neue Freiheitsgrade ein, weshalb das Potential für Fehler besonders in der Orientierung der einzelnen Scans erheblich erhöht wird. Die essentielle Arbeitsweise des Algorithmus wird jedoch unangetastet bleiben. Weiterhin werden wir Experimente an ausgewählten Datensätzen durchführen um auf die potentiellen Probleme und Vorteile dieses Ansatzes einzugehen.

Abstract

For an autonomous mobile robot to enter an unknown environment and start performing work is a difficult task. One way to make this task easier is the ability to build a map of the unknown environment. This is often achieved by matching separately collected partialviews, for example laser scans, of the environment. In a sequential procedure, where each new scan is matched to the previous one, small errors usually add up to a large global error. To avoid an inconsistent environmental model, Lu and Milios [18] developed an algorithm for simultaneously adding 2D laser scans and odometry measurements to a globally consistent map. The robot poses are modeled as random variables, which are approximated by maximum likelihood estimation.

In this thesis, the formulation of the algorithm for 2-dimensional data is extended for use with 3D laser scans. The extension to three dimensions leads to three additional degrees of freedom, increasing the potential of errors, especially in the orientation of the scans. Nevertheless, the essential working of the algorithm will be unchanged. To address the potential advantages and disadvantages of this approach, we will present experiments on selected data sets in the last portion of this thesis.

Danksagung

Herzlich danken möchten wir Herrn Prof. Dr. Hertzberg, der uns durch seine Vorlesungen im Bereich der *künstlichen Intelligenz* und der *Robotik* inspiriert und somit das Interesse für diese Arbeit geweckt hat.

Ebenfalls gilt unser Dank Kai Lingemann und Andreas Nüchter, die uns während unserer Arbeit zu jeder Zeit für Fragen bereit standen und uns mit fachlicher Kompetenz betreut haben.

Des Weiteren danken wir unseren Familien, die uns immer unterstützt haben und uns unser Studium ermöglicht haben.

Inhaltsverzeichnis

Abbildungsverzeichnis

Symbolverzeichnis

$\mathbb{N}$	Die Menge der natürlichen Zahlen
$\mathbb{R}$	Die Menge der reellen Zahlen
$\mathbb{H}$	Die Menge der Quaternionen, auch Hamilton Menge genannt
M	Referenzscan beim Scanmatching
D	Scan, der an Referenzscan angepasst werden soll
m, d	Punkte aus M und D
$w_{i,j}$	Gewichte
(R, t)	Transformation um Rotationsmatrix R und Translation um $t = (t_x, t_y)$
$P(x)$	Wahrscheinlichkeit
$P(x\|y)$	bedingte Wahrscheinlichkeit
s_t	Roboterpose zum Zeitpunkt t
o_t	Beobachtung/Sensormessung zum Zeitpunkt t
a_t	Aktion zum Zeitpunkt t
o^t, a^t	Alle Sensormessungen oder Aktionen bis zum Zeitpunkt t
$\text{dist}(x, y)$	Abstand zwischen x und y
$O(f)$	O-Notation, $O(f) := \{g : \mathbb{N} \to \mathbb{N}\ \exists c > 0 \exists n_0 \in \mathbb{N} \forall n > n_0 : g(n) \leq c \cdot f(n)\}$
$q = [a, v]$	Quaternion $q = a + bi + cj + dk$ mit $v = (b, c, d)$ und $a, b, c, d \in \mathbb{R}$
q^*	Konjugiertes Quaternion zu q mit $q^* = a - bi - cj - dk$
$\|\|a\|\|$	Betrag des Vektors a
$\langle a, b \rangle$	Das Skalarprodukt zweier Vektoren a und b
A^T	Die Transponierte einer Matrix A
A^{-1}	Die Inverse der Matrix A
$\mathbf{A}$	Eine Matrix bestehend aus Submatrizen $A_{i,j}$
$R_{\theta_x, \theta_y, \theta_z}$	Eine $3D$-Rotationsmatrix definiert durch die Eulerwinkel θ_x, θ_y und θ_z

R_q	Eine $3D$-Rotationsmatrix definiert durch das Quaternion q
I	Identitätsmatrix
I_d	Die $d \times d$ Identitätsmatrix
$\mathbf{I}_i$	Eine Identitätsmatrix, deren einzelne (i,j)-Submatrizen, für alle j, negative Identitätsmatrizen $-I_d$ sind. Sie hat also die Form $$\mathbf{I}_i = \begin{pmatrix} I_{d(i-1)} & 0 & 0 \\ -I_d & \dots & -I_d \\ 0 & 0 & I_{d(n-i+1)} \end{pmatrix}.$$
$\mathbf{I_D}$	Obere rechte Dreiecksmatrix bestehend aus Identitätsmatrizen I_d. Sie hat also die Form $$\mathbf{I_D} = \begin{pmatrix} I_d & \dots & I_d \\ & \ddots & \vdots \\ 0 & & I_d \end{pmatrix}$$
$\mathbf{H}$	Eine gerichtete Inzidenzmatrix, bestehend aus Identitätsmatrizen
$\mathbf{C}$	Eine blockdiagonale Matrix, bestehend aus den Kovarianzen $C_{i,j}$
$\bar{\mathbf{D}}$	Ein zusammengesetzter Vektor, bestehend aus den Posedifferenzen $\bar{D}_{ij}$
$(X_0, \dots, X_n)$	Eine Menge von $n+1$ Knoten in einem Graphen, die jeweils eine Roboterpose repräsentieren.
$\bar{X}_0$	Ein Ankernoten, also die Pose auf die sich alle Posen beziehen
$V = \begin{pmatrix} P \\ O \end{pmatrix}$	Die Pose eines Roboters mit der Position P und der Orientierung O
$\bar{V}$	Die Schätzung der Pose V
$\Delta V = \bar{V} - V$	Der Fehler zwischen der wahren Pose V und ihrer Schätzung V
$\oplus$	Transformations-Operation auf zwei Posen
$V \oplus D$	Die Endpose eines Roboters nach der Fahrt von V um D
$\ominus$	Inverse Operation zu $\oplus$ mit der Posedifferenz als Resultat
$\otimes$	Komposition zweier Orientierungen
$\oslash$	Die inverse Komposition zweier Orientierungen
$\frac{\partial f}{\partial v}$	Ableitung der Funktion f nach v
$\nabla_V(F)$	Der Grad der Funktion $F = (f_1, f_2, \dots, f_n)$ nach den einzelnen Komponenten von $V = (v_1, v_2, \dots, v_k)$, also die $n \times k$ Matrix bestehend aus allen $\frac{\partial f_i}{\partial v_j}$
$\square$	Ende eines Beweises

Kapitel 1

Einleitung

Die präzise Kartierung einer Umgebung ist ein in vielseitiger Hinsicht wichtiges Hilfsmittel für einen autonomen mobilen Roboter. Um sich zum Beispiel in einer zuvor kartierten Umgebung zu lokalisieren, ist ein global konsistentes und akkurates Modell der Umgebung notwendig. Diese Modellierung wird in der Regel dadurch erreicht, dass nacheinander aufgenommene Laserscans zusammengefügt werden, um so die Position des zukünftigen Kartenstücks in der Karte zu finden. Durch fehlerhafte Odometrie-Daten und Messfehler werden Ungenauigkeiten in diese Positionsschätzung eingebracht, die sich mit fortschreitender Zeit aufaddieren und die Karte somit zunehmend unbrauchbarer machen. Dies wird verstärkt durch Probleme beim Zusammenfügen der Scans, da alle bislang bekannten Verfahren selbst fehlerbehaftet sind. Eine mögliche Abhilfe schaffen Informationen über eine vom Roboter gefahrene Schleife. Nachdem so eine Schleife gefunden ist, kann der kumulative Fehler durch verschiedenste Methoden auf die beteiligten Scans verteilt werden. Dies führt zu einer konsistenten, aber nicht unbedingt zu einer korrekten Karte.

Eine dieser Methoden ist der Schätzungsalgorithmus für global konsistente Kartierung von Lu und Milios (LUM) [18]. Hier wird zunächst ein Netz von Laserscans und den dazugehörigen Posedifferenzen gebildet und aus diesem anschließend ein lineares Gleichungssystem, mit dem die Positionen der Laserscans im globalen Koordinatensystem berechnet werden.

1.1 Problemdefinition

Das in dieser Arbeit behandelte Problem ist die Umgebungskartierung mit einem autonomen mobilen Roboter. Ausgerüstet mit einem 3D-Laserscanner soll der Roboter eine unbekannte Umgebung befahren und aus den Laserscandaten eine Karte erstellen. Die Intention liegt darin, die einzelnen Laserscans so aneinander zu fügen, dass sie eine global konsistente Karte bilden um gegebenenfalls einen Teil eines Weltmodells darzustellen. Nicht Teil dieser Arbeit ist es, durch beispielsweise Extraktion von Objekten ein Weltmodell auf einem höheren Level zu bilden. Der Schwerpunkt liegt nicht in der Interpretation der Laserscandaten, sondern in der global konsistenten Aneinanderreihung derselben. Dabei werden anders als in früheren Ansätzen neben zweidimensionalen auch dreidimensionale Daten verarbeitet.

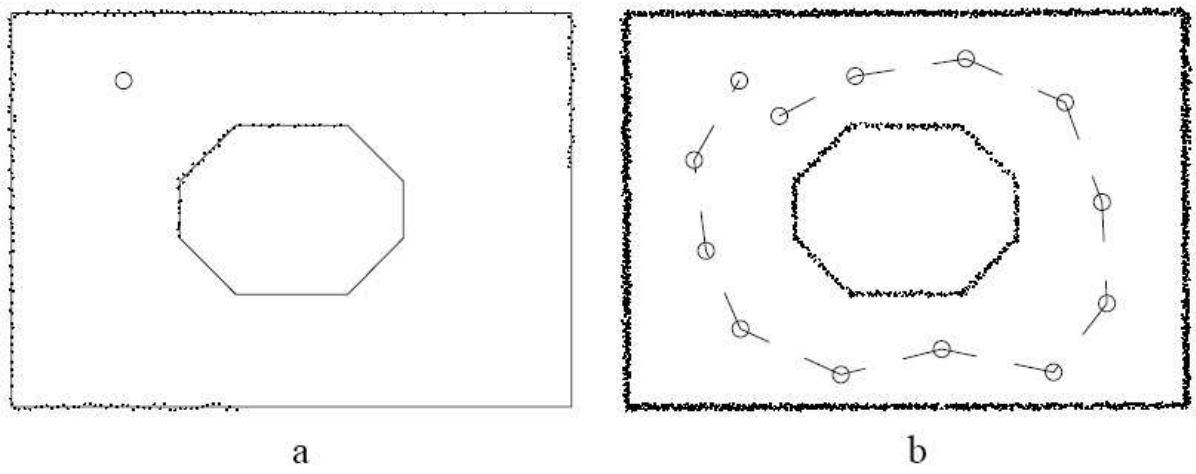


Abbildung 1.1: Erstellen eines Weltmodells aus Laserscans. (a) Ein 360° Laserscan in einer simulierten Umgebung. (b) Ein Modell bestehend aus mehreren Laserscans. Die Scanposen sind durch die kleinen Kreise markiert. Die Linien an den Kreisen indizieren den Roboterpfad. Quelle: [18]

Ein Laserscan ist eine Menge von Entfernungswerten, gemessen von einer Roboterpose. Diese Laserscans können auf einem mobilen Roboter zur Lokalisierung in bekannten Umgebungen benutzt werden. Eine alternative Verwendung finden sie in der Kartierung. Ein jeder dieser Laserscans, von einem Roboter an unterschiedlichen Positionen aufgenommen, repräsentiert einen kleinen Teil der Welt, der von der aktuellen Pose des Roboters aus erfassbar ist. Durch das Zusammenfügen mehrerer Scans lässt sich somit ein vollständigeres Modell einer größeren Umgebung erstellen. Abbildung 1.1 zeigt beispielhaft die Konstruktion eines Weltmodells aus mehreren Scans.

Die Schwierigkeit besteht nun in der Bestimmung der korrekten Posen für die Anordnung der Scans. Vordefinierte Landmarken sind in einer unbekannten Umgebung nicht vorhanden und Odometriedaten liefern zu ungenaue Ergebnisse, die sich mit zunehmender zurückgelegter Strecke noch verschlechtern. Aus diesem Grund lassen sich relative Posen zwischen den einzelnen Scans nur schwer bestimmen.

Eine mögliche Herangehensweise an dieses Problem besteht in der inkrementellen Registrierung neuer Scans. Unter der Registrierung eines Laserscans versteht man, dass ein Scan in ein Modell eingefügt und dort über seine Pose integriert wird, so dass er ein Teil des Modells wird. Bei inkrementeller Registrierung werden Scans nacheinander in das Modell integriert. Soll ein neuer Scan eingefügt werden, wird er an einen vorherigen Scan oder an das globales Modell angepasst. Die Integration in das Modell geschieht häufig über Durchschnittsbildung oder Verwendung eines Kalman-Filters (vgl. Kapitel 2.2.2). Dieser Ansatz führt jedoch zu Inkonsistenzen, da einzelne Bereiche des Modells unabhängig voneinander verändert werden. Eine nachträgliche Behebung der Inkonsistenzen ist nur möglich, wenn die Integration nicht permanent geschieht.

Um Inkonsistenzen zu vermeiden, muss eine nachträgliche Änderung bereits registrierter Scans durch Speicherung der kompletten Datensätze inklusive der geschätzten Posen ermöglicht wer-

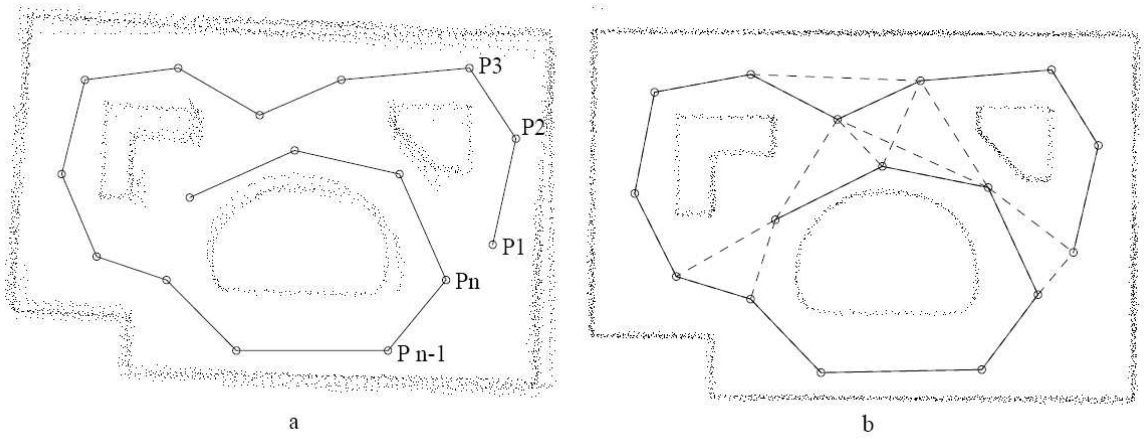


Abbildung 1.2: Beispiel konsistenter Kartierung. (a) Schlechte originale Anordnung der Scans als Folge von aufsummierten Posefehlern. (b) Gewünschte konsistente Darstellung der Karte nach Verarbeitung mit Hilfe eines Netzes von Verbindungen zwischen den Posen. Gestrichelte Linien repräsentieren durch Scanmatching erlangte Korrespondenzen, durchgezogenen Linien Verbindungen aus Odometriedaten. Quelle: [18]

den. Des Weiteren wird eine systematische Methode benötigt, um detektierte Fehler in der Poseschätzung auf bereits registrierte Scans zu verteilen und die notwendigen Korrekturen durchzuführen.

Abbildung 1.2 verdeutlicht dieses Problem. Betrachte man die im linken Bild dargestellte Roboterfahrt beginnend bei Punkt P_1. An jedem Knoten des Graphen wird ein Laserscan aufgezeichnet. Es wird deutlich, dass der Roboter am Punkt P_n wieder den Bereich des ersten Scans erreicht. Fügt man nun die Scans der Reihe nach aneinander, P_2 an P_1, P_3 an P_2 usw., tritt beim n-ten Scan das Problem auf, dass man diesen sowohl an Scan $n-1$ als auch an den ersten Scan anpassen kann. Aufgrund der Fehler während des Scanmatchings werden sich die Posen für P_n, die durch Anpassung an die beiden unterschiedlichen Scans bestimmt werden, unterscheiden. Entscheidet man sich nun für eine der beiden Poseschätzungen, entsteht eine Inkonsistenz bezüglich des anderen Scans. Eine naheliegende Variante verwendet den gewichteten Durchschnitt der beiden Schätzungen. Dies führt jedoch wiederum dazu, dass die Beziehung $P_{n-1}P_n$ inkonsistent mit der vorhergegangenen Schätzung ist. Folglich muss auch P_{n-1} und anschließend nacheinander alle anderen Posen des Roboterpfads bezüglich der Änderungen erneuert werden.

Das Ausmaß des Fehlers vergrößert sich mit zunehmender Länge des Roboterpfads. Bearbeitet man alle Scanposen einer Roboterfahrt mit paarweisem Scanmatching, führt das in der Regel zu einem komplexen Netz aus Beziehungen zwischen den einzelnen Roboterposen, in dem viele Konflikte vorliegen. Um diese zu lösen, bedarf es eines Verfahrens, das alle Konflikte bestmöglich löst. Das rechte Bild in Abbildung 1.2 zeigt wie eine konfliktfreie Lösung des Problems für das bereits erwähnte Beispiel aussehen könnte. Es ist das Ergebnis des Algorithmus von Lu und Milios. In dem abgebildeten Netz wird unterschieden zwischen zwei Arten von Beziehungen. Verbindungen die allein aus Odometriemessungen stammen, sind als schwache Verbindungen

definiert, während jene, die durch Scanmatching bestimmt werden, als starke Verbindungen bezeichnet werden.

Dieser Algorithmus ist ein Verfahren zur global konsistenten Registrierung von mehreren Laserscans, ist jedoch auf zweidimensionale Daten beschränkt. Die Idee besteht darin, neben den Punktmengen im lokalen Koordinatensystem des jeweiligen Laserscans einen Graph aus räumlichen Beziehungen zwischen den einzelnen Roboterposen zu verwalten, die durch Odometriedaten oder Ergebnisse von Scanmatchingverfahren erlangt werden. Die lokalen Koordinatensysteme werden jeweils durch die Roboterpose zum Zeitpunkt der Erfassung der Punkte definiert. Ziel ist es, mit Hilfe des Beziehungsgraphen die Roboterposen in einem globalen Koordinatensystem derart anzuordnen, dass sie ein Modell der Umgebung liefern. Dabei werden die Roboterposen als Variablen angesehen, deren optimaler Wert durch ein lineares Gleichungssystem berechnet wird. Konsistenz wird dadurch gesichert, dass alle Kanten des Graphen gleichzeitig berücksichtigt werden.

1.2 Wissenschaftlicher Beitrag

Da der Algorithmus von Lu und Milios bislang nur für zweidimensionale Laserscans formuliert ist, besteht das Ziel dieser Arbeit darin, ihn für 3D-Daten zu erweitern. Dafür muss insbesondere die Linearisierung der Posedifferenz-Gleichung angepasst werden. Die weiteren Berechnungen sollten sich lediglich in der Dimension unterscheiden. Das Problem bei der Erweiterung der Posedifferenz-Gleichung liegt in der üblichen Darstellung von Transformationen als Matrizen. Während im Zweidimensionalen eine Rotation einzig um eine Rotationsachse durchgeführt wird, setzt sich eine Rotation in der höheren Dimension aus Rotation um drei Achsen zusammen. Dies erfordert eine Multiplikation dreier Rotationsmatrizen um die Gesamttransformation zu erhalten und führt zu einer komplizierten Darstellung der Transformation.

Ausgangspunkt der Anwendung ist die Roboterplattform Kurt3D (vgl. Kapitel 6.2), welche mit einem SICK-Laserscanner ausgestattet ist, der mit Hilfe eines Servos bewegt werden kann um dreidimensionale Scans aufzuzeichnen. Die Scans werden momentan mit dem iterativen Algorithmus der nächsten Punkte (engl.: *iterative closest points* (ICP), vgl. Kapitel 2.1.1) aneinandergefügt. Anstatt die aus diesem Prozess entstehenden korrespondierenden Punktpaare zur direkten Minimierung des Abstandes zweier Scans zu verwenden, können mit ihnen Kovarianzmatrizen und die linearisierten Posedifferenzen errechnet werden. Mit diesen werden dann in geschlossener Form die Posen der einzelnen Scans bestimmt. Die Aufgabe besteht also darin, diese Berechnungen zu implementieren und in das bestehende System zu integrieren. Durch Tests sollen anschließend die Funktion des Algorithmus und eventuelle Optimierungsmöglichkeiten überprüft werden. Letztendlich sollen Teile des Botanischen Gartens in Osnabrück gescannt und global konsistent kartiert werden, so dass die Karte visualisiert werden kann.

1.3 Aufbau der Arbeit

Kapitel 1: Im ersten Kapitel wird ein kurze Zusammenfassung des Problems der Umgebungskartierung mit einem Roboter gegeben.

Kapitel 2: Hier werden unterschiedliche Scanmatching-Verfahren erläutert. Insbesondere wird auf den ICP-Algorithmus eingegangen, der auch als Grundlage für die Experimente in Kapitel 6 dient.

Kapitel 3: In diesem Kapitel erfolgt eine Beschreibung der unterschiedlichen Möglichkeiten, 3D-Transformationen mathematisch zu formulieren.

Kapitel 4: Kapitel 4 stellt den mathematischen Kern der Arbeit dar. Hier wird das probabilistisch formulierte Optimierungsproblem der Erstellung global konsistenter Karten, das von Lu und Milios für Posen in einer Ebene formuliert wurde, erläutert und dieses Problem und dessen Lösung auf drei Raumdimensionen erweitert. Die Optimierung erfolgt unter Verwendung der Posedifferenzen zwischen einzelnen Scans und deren Kovarianzen.

Kapitel 5: Im fünften Kapitel beschreiben wir Möglichkeiten, die für die Optimierung benötigten Poseschätzungen und Kovarianzen aus den Messwerten einer Roboterplattform nach einer Roboterfahrt zu extrahieren. Dies geschieht sowohl für den zweidimensionalen als auch für den dreidimensionalen Fall.

Kapitel 6: In diesem Kapitel wird die Funktionsweise des in den vorhergehenden Kapiteln aus mathematischer Sicht vorgestellten Vorgehens verdeutlicht und durch Experimente evaluiert.

Kapitel 7: Schlussendlich wird im letzten Kapitel eine Zusammenfassung der Ergebnisse geliefert, in der auch darauf eingegangen wird, ob der implementierte Algorithmus eine Verbesserung des bereits vorhandenen Scanmatching-Verfahrens ist und worin seine Einschränkungen liegen.

Kapitel 2

Scanmatching

Eine wichtige Aufgabe autonomer mobiler Roboter ist die Umgebungskartierung. Dies ist beispielsweise nützlich, wenn ein Roboter eine unbekannte Umgebung explorieren soll, was im bekannten Einsatzgebiet der Rettungsrobotik Verwendung findet. Ein beliebtes Verfahren ist das Aufzeichen von Laserscans mit anschließendem Aneinanderfügen per Scanmatching [26] um eine Karte zu bilden. Die generelle Idee des Scanmatchings, das eines der Schlüsselprobleme in der Computer-Vision ist, liegt darin, mehrere separat aufgenommene Laserscans so aneinander zu fügen, dass sie eine korrekte, zusammenhängende Darstellung der abgebildeten Umgebung liefern.

Ein Laserscanner funktioniert derart, dass ein Laserstrahl in unterschiedliche Richtungen ausgestrahlt und von dort befindlichen Objekten reflektiert wird. Durch die benötigte Zeit bis zum Wiedereintreffen des reflektierten Strahls beim Laserscanner lässt sich relativ genau die zurückgelegte Distanz bestimmen. Unter Einbeziehung des Ausgangswinkels des Laserstrahls liefert dies die genaue Position des Hindernisses im lokalen Koordinatensystem des Laserscanners. Die so erlangten Distanzinformationen von Oberflächen ergeben eine Punktwolke, aus der sich ein Abbild der Umgebung konstruieren lässt (vgl. 2.1). Im Folgenden werden wir eine Menge von Distanzwerten, die von einer Roboterpose aus aufgenommen wurden, als Laserscan bezeichnen.

In der Forschung wird häufig ein SICK-Laserscanner verwendet. Die Richtung, in die der Laserstrahl ausgesendet wird, lässt sich bei vielen Laserscannern durch rotierende Spiegel manipulieren, so auch bei dem für diese Arbeit verwendeten SICK-LMS200-Laserscanner. Er verfügt über einen Öffnungswinkel von 180 Grad und eine Auflösung zwischen 1.0 und 0.25 Grad. Der systematische Fehler wird mit 15 mm und der statistische Fehler mit 5 mm angegeben [29]. Zur Aufnahme von dreidimensionalen Scandaten ist der hier verwendete Scanner auf einen Aufbau montiert und kann mit Hilfe eines Servo-Motors um eine horizontale Achse gedreht werden, womit ein vertikaler Öffnungswinkel von zirka 90 Grad erreicht wird.

In den 3D-Punktwolken, die ein Laserscanner erzeugt, lassen sich Strukturen erkennen. Ziel des Scanmatchings ist es nun, gleiche Strukturen in teilweise überlappenden Laserscans zu erkennen und dadurch die Laserscans aneinander zu fügen. Dafür wird ausgehend von einer Initialschätzung versucht, mittels unterschiedlicher Algorithmen zumeist iterativ ein lokales Minimum zu finden. Einer der bekanntesten Algorithmen ist der ICP-Algorithmus (*iterative closest*

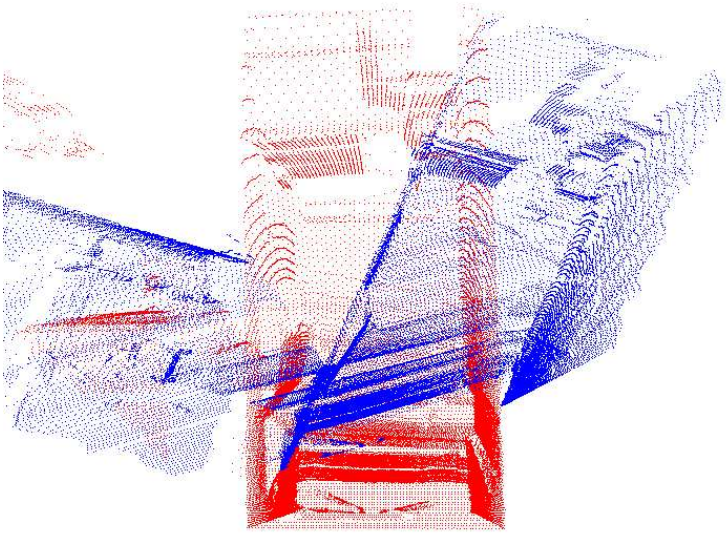


Abbildung 2.1: Punktwolken von zwei Laserscans.

point) [3], bei dem aus zwei Laserscans Punktpaare gesucht werden und der Abstand zwischen ihnen minimiert wird. Andere Algorithmen bestimmen die Distanz zwischen zwei Scans durch vorherige Extraktion von Linien oder Ebenen.

2.1 Paarweises Scanmatching

2.1.1 Der ICP-Algorithmus

Der Iterative-Closest-Point-Algorithmus von Besl und McKay [3] aus dem Jahr 1992 ist eine der ersten und seitdem eine der meist verwendeten Methoden zur Registrierung von 3D-Laserscans. Die Idee besteht darin, zu einer Menge von Punkten aus einer Form den jeweils nächsten aus einem korrespondierenden Gebilde zu finden. Auf diese Weise wird eine Transformation bestimmt, mit der die zweite Figur bewegt werden muss, um die Distanz zwischen den korrespondierenden Punktpaaren zu minimieren. Dieser iterative Algorithmus konvergiert schnell zu einem lokalen Minimum und erfordert keinerlei Vorverarbeitung der zu registrierenden Daten. Bereits nach wenigen Iterationsschritten wird ein gutes Ergebnis erzielt. Die Daten können aus Punktmengen, Polylinien, Kurven oder Flächen bestehen und es müssen weder Merkmale extrahiert, noch Ableitungen gebildet werden, da allein mit den Punkten des jeweiligen Objekts gearbeitet wird. Flächen, Polylinien und Kurven werden gewöhnlich durch Punkte gesampelt. Ein Problem ist, dass der Algorithmus häufig nur zu einem lokalen Minimum konvergiert. Dies kann zu fehlerhafter Registrierung führen.

Im Folgenden wird genauer auf die Verarbeitung von Laserscandaten eingegangen, da diese in der Robotik zu den bedeutendsten und verlässlichsten Sensordaten gehören. Die grundlegende

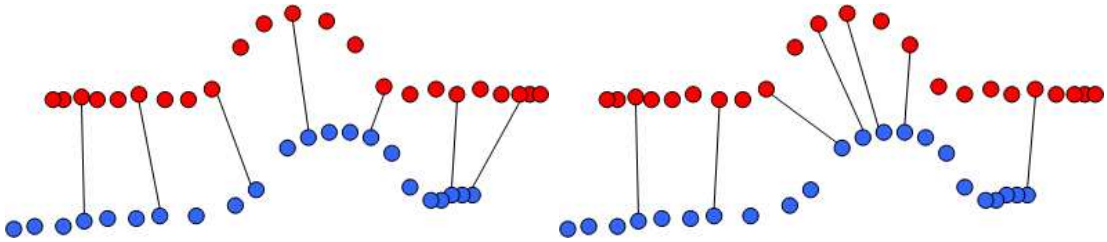


Abbildung 2.2: Verschiedene Matching-Möglichkeiten von zwei Punktmengen.

Vorgehensweise besteht aus den folgenden sechs Schritten:

1. Wähle aus den Punkten eines Laserscans n Punkte aus.
2. Suche aus einem überlappenden Scan den nächsten korrespondierenden Punkt.
3. Gewichte die Korrespondenzen nach ihrer Zuverlässigkeit.
4. Streiche alle Punktpaare, deren Distanz von der mittleren Distanz deutlich abweicht.
5. Wende die Fehlerfunktion auf die Punktpaare an.
6. Minimiere die Fehlerfunktion.

In zahlreichen ICP-Variationen werden einer oder mehrere dieser Schritte optimiert, um die Laufzeit zu verringern, Stabilität zu vergrößern, die Toleranz bei Verrauschen und Ausreißern zu erhöhen oder eine schlechtere Anfangsschätzung zu erlauben [26].

Auswahl der Punkte: Ein Laserscan besteht aus einer großen Anzahl von Punkten. Die Berücksichtigung all dieser Punkte beim Scanmatching würde zu einem enormen Zeitaufwand führen. Deswegen ist es aus Zeitgründen sinnvoll, die Punktmengen zu reduzieren. Dazu gibt es mehrere Ansätze.

So bietet sich die Möglichkeit, eine zufällige oder uniforme Auswahl von Punkten jedes Laserscans für die Berechnung zu verwenden. Dies führt jedoch zu unerwünschten Effekten, wenn einige wichtige Details unberücksichtigt bleiben.

Abbildung 2.2 zeigt zwei Punktmengen und zwei Möglichkeiten die Punkte dieser Mengen für das Scanmatching auszuwählen. Auf der linken Seite wird jeder dritte Punkt der roten Punktmenge ausgewählt und ein korrespondierender Punkt aus der blauen Punktmenge gesucht. Aus der kleinen Ausbuchtung, der signifikanten Stelle der Punktmenge, wird auf diese Weise nur ein einziger Punkt ausgewählt. Aus solch einer Auswahl folgt unter Umständen, dass der Algorithmus nur langsam konvergiert oder zwei Scans falsch aneinandergefügt werden. Sinnvoller scheint es zu sein, wie in der linken Abbildung, mehrere Punkte aus der Ausbuchtung zu wählen und stattdessen einige der Punkte der geraden Linie zu vernachlässigen. Dies kann besonders von Bedeutung sein, wenn die charakteristischen Stellen eines Laserscans in weiter Entfernung

liegen, da mit zunehmender Entfernung auch die Abstände zwischen aufeinanderfolgenden Punkten größer werden, oder wenn große Fehler bei der Poseschätzung vorliegen, beziehungsweise die Scans stark verrauscht sind.

Gelfand et al. haben eine Methode entwickelt, Punkte so auszuwählen, dass der ICP möglichst schnell konvergiert [11]. Zu viele Punkte aus instabilen Regionen, die keine signifikanten Eigenschaften aufweisen, führen dazu, dass der Algorithmus zwei zu matchende Scans hin- und herschiebt, ohne dabei die korrekte Pose zu erreichen oder gar ohne überhaupt zu konvergieren. Dies resultiert aus durchgängig geringen Fehlerwerten in diesen Bereichen. Um dies zu vermeiden, wählen Gelfand et al. die Punktpaare so aus, dass sie aus möglichst stabilen Regionen stammen. Dazu bestimmen sie zunächst die Kovarianzen einer kleinen Menge von Punkten aus der Eingabemenge, um anhand dieser stabile Stellen zu bestimmen, aus denen dann vorzugsweise Punkte ausgewählt werden, mit dem Ziel stabile Verbindungen zu erhalten, die alle Freiheitsgrade behandeln.

Finden von korrespondierenden Punktpaaren: Der wichtigste Bestandteil des ICP, aber auch der aufwendigste, ist das Finden von korrespondierenden Punktenpaaren. Die einfachste Variante ist es, ausgehend von einem Punkt des ersten Scans und unter Berücksichtigung der Poseschätzung, die Distanz zu jedem Punkt des zweiten Scans zu berechnen und den Punkt mit der geringsten Distanz auszuwählen.

Um die Laufzeit des Auswahlverfahrens von $O(n^2)$ bei n Punkten zu verringern, gibt es mehrere Ansätze. Betrachtet man die Punkte als Einträge, die durch Schlüssel, die x-, y- und z-Koordinaten, definiert sind, lassen sie sich durch besondere Speicherstrukturen besser verwalten und somit die Suche deutlich beschleunigen [9].

Eine Möglichkeit ist die Unterteilung des Raumes der Schlüssel in kleine, gleich große Zellen. Ausgehend von einem Eintrag kann man nun spiralförmig nach außen die nächste Zelle mit einem Eintrag suchen. Diese Suche minimiert die Anzahl der durchsuchten Punkte, benötigt aber dennoch große Mengen an Speicherplatz und Rechenzeit, besonders bei Schlüsselmengen von großer Dimension. Mit Cluster-Techniken lassen sich Verfahren entwickeln, bei denen sich die Anzahl der betrachteten Einträge weiter verringern lässt. Eine weitere Möglichkeit ist es, die Schlüssel in linearen Listen zu verwalten und nur jene Punkte zu durchsuchen, deren Schlüssel in einem Intervall um die Schlüssel des gegebenen Punktes liegen. Die Laufzeit bei n Punkten ist abhängig vom verwendeten Suchverfahren für die Listen und ist nach Friedman et al. proportional zu $3n^{-\frac{2}{3}}$ bei dreidimensionalen Punkten [9].

Des Weiteren kann man die Punkte in Quad- oder Octrees verwalten [9]. Darauf aufbauend entwickelten Bentley et al. einen k-dimensionalen Baum, den kd-tree [2]. Ein kd-tree ist ein binärer Baum, der zur Speicherung von Objekten mit k verschiedenen Schlüsseln dient. Jeder Knoten des Baumes speichert ein Objekt. Dabei ist der Baum derart strukturiert, dass in jeder Ebene alternierend nach einem der k Schlüssel diskriminiert wird. In dem linken Unterbaum eines Knotens P in Ebene i befinden sich somit nur Objekte, die bezüglich der Eigenschaft i mod k kleiner sind als P, in dem rechten Unterbaum nur solche, die größer sind.

Abbildung 2.3 zeigt Punkte in einer zweidimensionalen Ebene und eine dazugehörige Repräsentation als kd-tree. Durch die Verwendung dieser Darstellung lässt sich nun die Suche nach einem

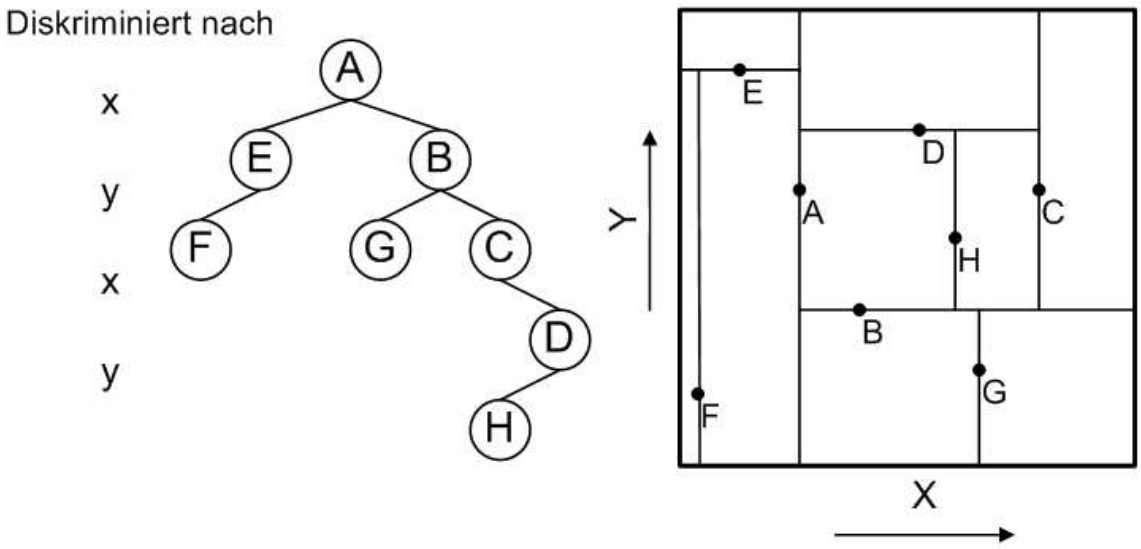


Abbildung 2.3: Beispiel eines 2d-trees.

korrespondierenden Punkt zu P als Form einer rekursiven Baumsuche beschleunigen [9], wie in Algorithmus 1 verdeutlicht wird.

In jedem Rekursionsschritt steigt man eine Ebene im Baum herab und verkleinert somit die Menge der in Frage kommenden Punkte. Auf diese Weise wird auch das Suchintervall verkleinert. Man betrachtet den Knoten, an dem man sich zur Zeit befindet, und entscheidet, auf welcher Seite sich Punkt P befinden würde. Der Wert des diskriminierenden Schlüssels bildet die neue Grenze des Intervalls. So wird der Suchraum fortschreitend eingeschränkt. Erreicht man den letzten Eintrag des Baums, erhält man somit einen nahen Punkt.

Zur Überprüfung, ob dieser Punkt Q der gesuchte Punkt ist, steigt man im Baum wieder hinauf und überprüft, ob eine Kugel mit dem Abstand zwischen P und Q als Radius die Intervallgrenzen des nicht betrachteten Teilbaums überlappt. Ist dies der Fall, muss auch dieser rekursiv nach einem näheren Punkt durchsucht werden, andernfalls steigt man weiter hinauf bis letztendlich die Wurzel erreicht wird.

Die Laufzeit des Suchalgorithmus ist abhängig von der Anzahl der Knoten, der Dimension der Punkte, aber auch vom Aufbau des kd-trees. Je ausgeglichener der Baum ist und je gleichmäßiger diskriminiert wird, desto schneller lässt sich der Baum durchsuchen.

Friedman et al. beschreiben die Optimierung eines kd-trees [9], in dem die Suche in Laufzeit von $O(\log n)$ durchgeführt werden kann. Dabei wird beim Einfügen in den Baum bei jedem Knoten, der kein Blatt ist, als Diskriminator der Schlüssel gewählt, dessen Werte am Weitesten verteilt sind und als Grenze der Median der Schlüsselwerte. Somit wird die Baumtiefe verringert sowie die Wahrscheinlichkeit, dass bei einem gefundenen Punkt in einem anderen Unterbaum noch ein näherer Punkt liegt. Zusätzlich zur Laufzeit der Suche bestimmt noch die Zeit für den Aufbau

Algorithm 1 Suche nach nächstem Punkt in einem kd-tree

```
Input: Baum b, Punkt p
Output: nächster Punkt n
  if b ist Blatt then
    return Punkt in b
  end if
  if p gehört in linken Teilbaum then
    Punkt q1 = suche( linkerSohn, p )
    if Abstand( p, q1 ) < Abstand( p, rechterSohn ) then
      Punkt q2 = suche( rechterTeilbaum, p )
    end if
    if Abstand ( p1, q1 ) < Abstand( p, q2 ) then
      return q1
    else
      return q2
    end if
  else
    Punkt q1 = suche( rechterSohn, p )
    if Abstand( p, q1 ) < Abstand( p, linkerSohn ) then
      Punkt q2 = suche( linkerTeilbaum, p )
    end if
    if Abstand ( p1, q1 ) < Abstand( p, q2 ) then
      return q1
    else
      return q2
    end if
  end if
```

des Baumes, die in $O(kn \log n)$ liegt, die Performanz der Bestimmung von Punktpaaren.

Leider ist aber nicht immer der nächste Punkt auch der Beste, deswegen gibt es auch Varianten, die Punkte nach Kompatibilität bewerten, und zusätzlich zur Distanz die Farbe, die Normalen oder andere Merkmale wie Krümmung oder Ableitungen berücksichtigen [26]. Dies verlangt aber die Existenz von solchen Merkmalen und ist somit für Laserscans ohne Vorverarbeitung nicht möglich.

Eliminieren von Ausreißern: Ein einfaches Auswählen einer kleinen Punktmenge aus den Laserscans kann dazu führen, dass das Scanmatching-Verfahren an Robustheit verliert, wenn Ausreißern auf diese Weise ein zu großes Gewicht beigemessen wird. Deswegen ist das Eliminieren von Ausreißern ein wichtiger Schritt zur Verbesserung des ICP-Algorithmus. Um dieses zu erreichen, erarbeiteten Chetverikov et al. den so genannten Trimmed ICP (TrICP) [6], der auf Basis des Least Trimmed Squares Verfahrens (LTS) [24] Ausreißer erkennt und eliminiert. Im Gegensatz zu klassischen Distanzberechnungen, funktioniert das LTS-Verfahren auch bei einer

großen Anzahl von Ausreißern, wie sie bei zwei unterschiedlichen Laserscans häufig vorkommt, da die Toleranzgrenze nicht durch die Ausreißer mitbestimmt wird.

Um die Punktmenge D passend zur Punktmenge M zu transformieren, wird zuerst für jeden Punkt aus D der nächste korrespondierende Punkt aus M gesucht und die Distanz zum nächsten Punkt aus M definiert als

$$\text{dist}_i(R,t) = \left\| \arg\min_{m \in M} \|m - R \cdot d_i + t\| - R \cdot d_i + t \right\|,$$

mit der Rotation R und Translation t, die nötig sind, um D auf M zu transformieren.

Beginnend mit einer Anfangstranformation (R,t) werden die Distanzen dist_i dann quadriert und absteigend sortiert. Anschließend werden die N_{do} Paare mit den geringsten Distanzen ausgewählt, wobei N_{do} die Anzahl der Punkte aus dem überlappenden Bereich von M und D ist. Diese Anzahl wird als bekannt vorrausgesetzt, kann aber auch mit dem TrICP-Algorithmus bestimmt werden.

Ist keine der Abbruchbedingungen erfüllt, setzt man S_{LTS}, einen anfangs sehr groß gewählten Schwellwert, auf $S_{LTS}\prime$, die Summe der N_{po} ausgewählten dist_i^2, berechnet die optimale Bewegung (R,t), die S_{LTS} minimiert, und wendet sie auf D an. Dies wird so oft wiederholt, bis entweder die maximale Anzahl an Iterationen erreicht ist, $S_{LTS}\prime/N_{do}$ ausreichend klein ist oder sich nur noch in geringem Maße verändert.

In Tests zeigen Chetverikov et al., dass dieses Verfahren selbst bei relativ schlechter Anfangsschätzung, z. B. relativer Rotation von bis zu 30 Grad, zu einer Verbesserung führt, da Punkte aus dem nicht überlappenden Bereich die Transformation nicht beeinflussen.

Gewichten der Korrespondenzen nach ihrer Zuverlässigkeit: Bei gering überlappenden Punktwolken oder Punktwolken mit geringer Dichte, kann es durch Auswahl der Punkte und Elimination von Ausreißern leicht dazu kommen, dass die Anzahl der Punkte zu klein wird um sinnvoll damit arbeiten zu können. Um in diesem Fall trotzdem weiterzuarbeiten, bietet es sich an, anstatt die Punkte zu ignorieren, die Punktpaare nach ihrer Zuverlässigkeit zu gewichten mit $w_{i,j}$ als dem Gewicht des Punktpaares (d_i, m_j).

Im ungewichteten Fall setzt man $w_{i,j}$ auf 1, wenn Punkt m_j aus M ein korrespondierender Punkt zu d_i aus D ist, ansonsten ist $w_{i,j}$ 0. Andernfalls kann man Punktpaare mit zunehmender Distanz geringer gewichten mit

$$w_{i,j} = 1 - \frac{\text{dist}(d_i, m_j)}{\text{dist}_{max}}.$$

Eine weitere Methode wäre, mit

$$w_{i,j} = 1 - \frac{|\text{dist}(d_i, m_j) - \text{dist}_{median}|}{\text{dist}_{max}}$$

nach dem Median der Abstände zu gewichten. Aber es sind auch andere Möglichkeiten erdenklich, wie zum Beispiel die Gewichtung nach der Kompatibilität der Normalen mit

$$w_{i,j} = n_{d_i} \cdot n_{m_j},$$

wobei diese Berechnung aber eine vorherige Bestimmung der Normalen voraussetzt.

Minimierung der Fehlerfunktionen: Die beiden meist verwendeten Fehlerfunktionen sind die Punkt-zu-Punkt-Fehlerfunktion, die auch in der ICP-Version implementiert ist, die in Kapitel 6.2 zum Vergleich herangezogen wird [21] und die Punkt-zu-Ebene-Fehler-Funktion, wie sie von Chen und Medioni implementiert wird [5].

Bei ersterer wird die Fehlerfunktion

$$E(R,t) = \sum_{j=0}^{|M|} \sum_{i=0}^{|D|} w_{i,j} \left\| m_j - (Rd_i + t) \right\|^2 \tag{2.1}$$

gebildet, die den Abstand zwischen einem Punkt m_i aus M und seinem korrespondierenden Punkt d_i aus D angibt, wobei d_i um R rotiert und um t verschoben ist. Die Fehlerfunktion soll minimiert werden, um eine optimale Transformation (R,t) zu finden, die alle d_i möglichst gut auf ihre korrespondieren Punkte bewegt. Ersetzt man die Korrespondenzmatrix durch einen Vektor, der nur die Punktpaare beinhaltet, vereinfacht sich die Fehlerfunktion (2.1) zu

$$E(R,t) \approx \frac{1}{n} \sum_{i=1}^{n} \left\| m_i - (Rd_i + t) \right\|^2$$

mit $n = \sum_{i=0}^{|D|} \sum_{j=0}^{|M|} w_{i,j}$, wobei $w_{i,j}$ entweder 0 oder 1 ist.

Zur Minimierung der Fehlerfunktion sind verschiedene Methoden bekannt. Auf der Basis von Quaternionen (siehe Kapitel 3.4) bestimmt sich die Rotation, die Gleichung 2.1 für alle ausgewählten Punkte $m' = m_i - c_m$ und $d' = d_i - c_d$ minimiert, durch die Kreuz-Kovarianz-Matrix

$$N = \begin{pmatrix} S_{xx}+S_{yy}+S_{zz} & S_{yz}+S_{zy} & S_{zx}+S_{xz} & S_{xy}+S_{yx} \\ S_{yz}+S_{zy} & S_{xx}-S_{yy}-S_{zz} & S_{xy}+S_{yx} & S_{zx}+S_{xz} \\ S_{zx}+S_{xz} & S_{xy}+S_{yz} & -S_{xx}+S_{yy}-S_{zz} & S_{yz}+S_{zy} \\ S_{xy}+S_{yx} & S_{yz}+S_{zy} & S_{zx}+S_{xz} & -S_{xx}-S_{yy}+S_{zz} \end{pmatrix}$$

mit $S_{xx} = \sum_{i=1}^{n_d} \sum_{j=1}^{n_m} w_{i,j} m'_{jx} d'_{ix}$, $S_{xy} \sum_{i=1}^{n_d} \sum_{j=1}^{n_m} w_{i,j} m'_{jx} d'_{iy} \ldots$.

Der größte Eigenwert von N ist die Quaternionen-Darstellung der gesuchten Rotation R [33]. Die zugehörige Translation ist $t = c_m - Rc_d$, wobei die Schwerpunkte

$$c_d = \frac{1}{n} \sum_{i=1}^{n} d_i \qquad \text{und} \qquad c_m = \frac{1}{n} \sum_{i=1}^{n} m_i$$

durch die Mittelwerte aller Punkte aus M und D bestimmt werden.

Eine alternative Bestimmung des Minimums, die auf Singulärwertzerlegung (SVD) basiert, wird von Nüchter et al. [21] verwendet. Bei dieser Methode ist die Rotation gegeben durch $R = VU^T$. V und U erhält man durch die Singulärwertzerlegung $H = U\Lambda V^T$ der Matrix

$$H = \begin{pmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{pmatrix}$$

mit $S_{xx} = \sum_{i=1}^{n} m'_{ix} d'_{ix}$, $S_{xy} = \sum_{i=1}^{n} m'_{ix} d'_{iy} \ldots$.

Weitere bekannte Methoden zur Minimierung der Gleichung verwenden Dualzahlquaternionen oder Polarzerlegung [17].

Die Punkt-zu-Ebene-Fehler-Funktion von Chen und Medioni [5] berücksichtigt anstelle der Distanz zwischen zwei Punkten den Abstand zwischen einem Punkt der Punktmenge D und einer Fläche der Menge M als Fehlerfunktion. Auf diese Weise erreicht man, dass der Fehler sich nicht vergrößert, wenn der eine Scan auf einer Ebene liegend verschoben wird. Somit können die Punkte einer Ebene nicht verhindern, dass der Scan weiterbewegt wird, obwohl er sich nicht in der richtigen Position befindet. Dies ist von Nutzen, wenn die abgebildete Umgebung eine ebene Fläche mit nur wenigen Erhebungen darstellt.

Der Algorithmus verändert sich leicht im Vergleich zur Punkt-zu-Punkt-Version. Für jeden ausgewählten Punkt d_i aus D wird zuerst die Flächennormale n_{d_i} errechnet. Danach wird der Schnittpunkt m_i der Linie, die durch d_i und n_{d_i} bestimmt ist, und der Punktmenge M gesucht und die Tangentenebene $S_i = \{s | n_{q_i} \cdot (q_i - s) = 0\}$ von M in m_i gebildet. Die zu minimierende Fehlerfunktion ist dann:

$$E = \sum_{i=1}^{n} ((Rd_i + t - m_i) \cdot n_{m_i})^2. \tag{2.2}$$

Da diese Gleichung nicht in geschlossener Form lösbar ist, wenden Rusinkievicz und Levoy [25,26] eine Linearisierung an, unter der Annahme von kleinen Rotationen und der Annäherung von $\sin\theta$ mit θ und $\cos\theta$ mit 0, wodurch Gleichung (2.2) zu

$$E \approx \sum_{i=1}^{n} ((m_i - d_i) \cdot n_{d_i} + r \cdot (m_i \times d_i) + t \cdot n_{d_i})^2$$

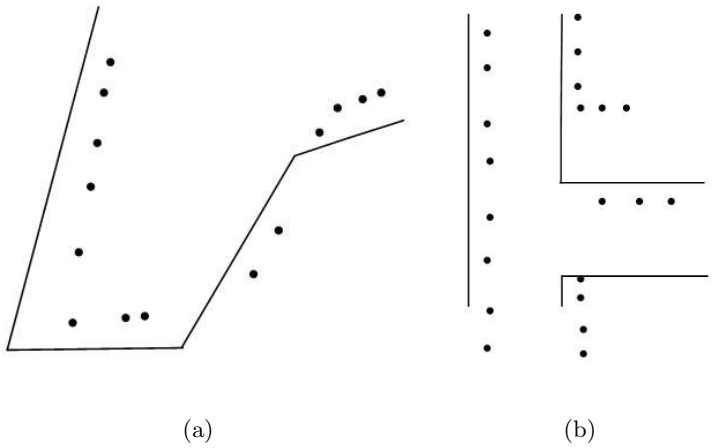
wird, einem linearen Gleichungssystem, das sich nach der Transformation $r = (\theta_x, \theta_y, \theta_z)^T$ und $t = (x, y, z)^T$ auflösen lässt.

2.1.2 Der Cox-Algorithmus

Der in seiner ursprünglichen Version zum Zuordnen von Laserscan-Punkten zu Referenzlinien, aus beispielsweise einem a priori Umgebungsmodell entwickelte Algorithmus von Cox [7] eignet sich auch als Scanmatcher [16] und wird auch als solcher verwendet. Für die Verwendung zum Scanmatching liegt der einzige Unterschied in der Definition des Modells. Anstatt eine gegebene Karte der Umgebung als Referenz zu benutzen, wird vielmehr aus dem Referenzscan eine Linienkarte extrahiert. Anders als beim *ICP* wird nun zu jedem Punkt des einen Scans, nicht der nächste Punkt aus dem Referenzscan bestimmt, sondern das nächste Liniensegment.

Der Algorithmus besteht dann aus folgenden vier Schritten:

1. Finde zu jedem Punkt aus Scan D das nächste Liniensegment aus dem Referenzscan M.
2. Bestimme die Transformation (R, t), die auf D angewandt die quadrierte Gesamtdistanz aller Punkte aus D zu ihren Referenzlinien aus M minimiert.



(a) (b)

Abbildung 2.4: Beispiel von Laserscan-Punkten und einem Linienmodells.

3. Wende (R, t) auf D an.
4. Wiederhole Schritt 1-3, bis die Distanz ausreichend klein ist. Die Komposition aller Transformationen aus Schritt 3 ist die Gesamttransformation, welche die beiden Scans optimal aneinanderfügt.

Abbildung 2.4(a) zeigt beispielhaft die Punkte eines Scans und die Linien eines Referenzscans. Der Scan dessen Punkte zu sehen ist, ist leicht gegen den Uhrzeigersinn gedreht und nach rechts oben verschoben. Betrachtet man nun die Punkte, so erkennt man, dass für die meisten von ihnen das am nächsten liegende Liniensegment auch die korrekte Referenzlinie ist. Es erscheint also offensichtlich, dass die Minimierung der Abstände durch eine quadratische Fehlerfunktion auch zu einem guten Ergebnis führen wird.

Jedoch wird immer nur der Abstand zur Referenzlinie minimiert. Die Position auf der Linie spielt dabei keine Rolle. Wenn der initiale Fehler groß ist oder die Punkte ungünstig verteilt sind, zum Beispiel viele am Ende von Liniensegmenten liegen, kann es passieren, dass die Zuordnung zu Referenzlinien fehlerhaft ist und der Algorithmus langsam, gar nicht oder mit einem nicht zufriedenstellenden Ergebnis konvergiert.

In Abbildung 2.4(b) erkennt man eventuell auftretende Probleme. Durch Anwendung des Algorithmus werden diese beiden Scans vermutlich falsch aneinandergefügt. Die Punkte der waagerechten Linien bewirken, dass der durch Punkte dargestellte Scan ein wenig nach rechts verschoben wird, so dass diese Punkte mit den Linien übereinstimmen. Die Punkte der unteren waagerechten Linie werden jedoch vermutlich nach oben gezogen, da sie näher an dem oberen waagerechten Liniensegment liegen, während die Punkte der oberen waagerechten Linie näher an der senkrechten Linie liegen und somit dieser zugeordnet würden.

Für die Minimierung der Fehlerfunktion in Schritt zwei werden die Liniensegmente zu Geraden erweitert. Wir bezeichnen die Referenzgerade eines jeden Punktes d_i mit $L_i = (u_i, r_i)$, wobei $u_i =$

$(u^x, u^y)^T$ ein zur Referenzlinie orthogonaler Einheitsvektor ist und $r_i = u_i \cdot z_j$ das Skalarprodukt von d_i mit jedem Punkt z_j der Referenzlinie.

Bezeichnen wir die Roboterpose mit p_{robot}, dann wird die Translation $b = (R, t)$, bestehend aus einer Rotation R des gesamten Scans um θ gegen den Uhrzeigersinn um die Roboterpose mit anschließender Translation um $t = (t_x, t_y)$, beschrieben als

$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} (d_i - p_{robot}) + p_{robot} + t.$$

Dies lässt sich approximieren durch

$$\theta \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} (d_i - p_{robot}) + p_{robot} + t. \tag{2.3}$$

Unter Verwendung von (2.3) lässt sich die Distanz eines Punktes d_i zu seiner Referenzlinie beschreiben als

$$\text{dist}(d_i, L_i) = \begin{pmatrix} u_i^x \\ u_i^y \\ u_i^d \end{pmatrix} \cdot b - r_i + u_i \cdot d_i$$

mit

$$u_i^d = u_i \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} (d_i - p_{robot}),$$

woraus sich folgende quadratische Fehlerfunktion ergibt:

$$\begin{aligned} E(b) &= \sum_{i=1}^{n} \text{dist}(d_i, L_i)^2 \\ &= (Xb - Y)^T (Xb - Y) \end{aligned}$$

mit

$$X = \begin{pmatrix} u_1^x & u_1^y & u_1^d \\ \dots & \dots & \dots \\ u_n^x & u_n^y & u_n^d \end{pmatrix} \quad \text{und} \quad Y = \begin{pmatrix} r_1 - u_1 \cdot d_1 \\ \dots \\ r_n - u_n \cdot d_n \end{pmatrix}.$$

Die Minimierung dieser Fehlerfunktion erreicht man durch die Ableitung

$$\frac{\partial E}{\partial b}\left(\hat{b}\right) = 0,$$

was als Lösung die Transformation

$$\hat{b} = \left(X^T X\right)^{-1} X^T Y$$

ergibt, die den Fehler verringert. Die Anwendung von $\hat{b}$ auf Scan D führt zu einer verbesserten Anordnung der Scans. Wiederholtes Anwenden des Algorithmus führt in endlicher Zeit zu einem

Minimum von E. Die benötigte Laufzeit ist, wie auch beim ICP-Algorithmus, $O(k \cdot n \cdot l)$ bei k Iterationen, n Punkten in D und l Linien in M. Hinzu kommt der Aufwand für die Extraktion der Linien, der abhängig von der Struktur des Scans (ungeordnet, geordnet) im Best-Case in $O(n)$ und im Worst-Case in $O(n^2)$ liegt.

Ein Problem des Cox-Algorithmus ist, dass er nicht ohne weiteres erweiterbar ist für dreidimensionale Punktwolken. Eine Möglichkeit besteht darin, anstelle der Referenzlinien Referenzebenen zu verwenden. Dies würde aber einen erheblichen Aufwand für die Extraktion dieser Ebenen aus einem Scan bedeuten.

2.1.3 Scanmatching mit Histogrammen

Eine weitere Methode zum Scanmatching, die für zweidimensionale Scans ausgelegt ist, verwendet Histogramme [16, 37]. Dafür wird jeder Scan als Winkelhistogramm und später auch als x/y-Histogramm repräsentiert.

Um ein Winkelhistogramm zu erstellen, müssen die Scanpunkte als Vektoren interpretiert werden, das bedeutet eine Darstellung der Punkte als Ortsvektoren im lokalen Koordinatensystem des Scans mit Ursprung in der Position des Roboters und der Roboterorientierung als Koordinatenachsen. Durch diese Interpretation ist es möglich die Differenz zwischen zwei aufeinanderfolgenden Scanpunkten zu bilden. Im Winkelhistogramm werden nun die Winkel der Differenzvektoren abgetragen. Der Winkel am Punkt p_i ergibt sich folglich aus dem Vektor vom Punkt p_i zum Punkt p_{i+1} und beträgt

$$\alpha_i = \left(\frac{x_{i+1} - x_i}{y_{i+1-y_i}} \right).$$

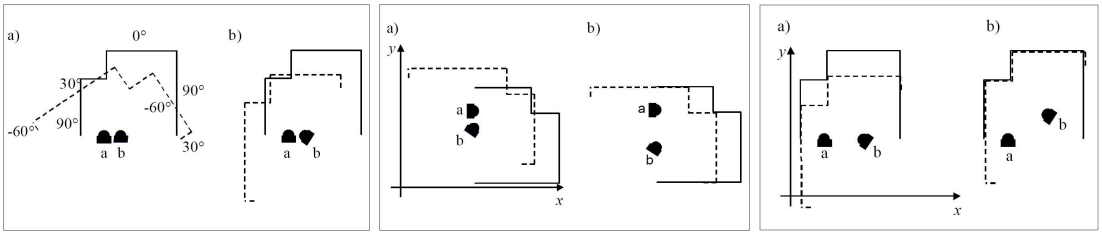
Ein solches Histogramm zeigt Maxima bei den Richtungen, welche den Scan dominieren, z. B. bei den Wänden eines Raumes. Das globale Maximum θ_0 wird als die Hauptrichtung des Scans bezeichnet. Eine wichtige Eigenschaft ist die Invarianz gegenüber Rotation. Die einzige qualitative Veränderung ist eine horizontale Verschiebung des Winkelhistogramms. Die Verteilung der Maxima und Minima bleibt davon unberührt.

Aus diesem Grund lässt sich aus der Struktur zweier Winkelhistogramme von aufeinanderfolgenden Scans die Rotation des Roboters bestimmen, indem die Phasenverschiebung in Rotation umgerechnet wird. Dies geschieht beispielsweise mit der Kreuzkorrelationsfunktion

$$K_j(H_M, H_D) = \sum_{i=1}^{n} H_M(i) H_D((i+j) \mod n+1),$$

wobei H_M und H_D die (zirkulären) Histogramme zu den Scans M und D sind und $|H_M| = |H_D| = n$ die Größe der Histogramme ist. j ist die Verschiebung zwischen den beiden Histogrammen und entspricht dem Winkel $\Delta\theta$, um den M und D verschieden sind.

Aus einem Winkelhistogramm lässt sich zwar die Rotation ablesen, nicht aber die Translation. Um diese zu bestimmen, werden im zweiten Schritt x/y-Histogramme erstellt. Dazu werden



(a) Rotation (b) x-Translation (c) y-Translation

Abbildung 2.5: Die drei Schritte des Scanmatchings mit Histogrammen. Quelle: [23]

zuerst beide Scans parallel zur x-Achse an der Hauptrichtung ausgerichtet und somit der Unterschied zwischen ihnen korrigiert, indem Scan M um $-\theta_0$ und Scan D um $-\theta_0 + \Delta\theta$ gedreht wird.

Die bei Winkelhistogrammen natürlich vorliegende Zirkularität wird durch eine Faltung mit $f(x, y) = (x \mod size_x, y \mod size_y)$ erzeugt. Anschließend werden zwei Histogramme erstellt. Das x-Histogramm repräsentiert die Scanpunkte als diskrete Verteilung der x-Koordinaten, das y-Histogramm die y-Koordinaten. Daraus lassen sich die nötigen Translationen in x- und y-Richtung über die Kreuzkorrelationsfunktion bestimmen.

Ein Problem der Histogramme ist die Genauigkeit. Belegt man sie mit einer zu feinen Skalierung, führt das gegebenenfalls dazu, dass verrauschte Messungen die Histogramme stark verändern und Minima oder Maxima nicht eindeutig erkennbar sind. Eine sehr grobe Einteilung wiederum verhindert eine genaue Berechnung der anzuwendenden Tranformationen. Da die Existenz von Minima und Maxima essentiell ist für die Funktionalität des Matchings, bekommt die Wahl der entsprechenden Skalierung einen besonderen Stellenwert. Verfügen die Scans über keine klare Richtung, schlägt das Matching fehl.

Des Weiteren ergibt sich auch hier das Problem, dass die Erweiterung auf eine dritte Dimension nicht trivial ist. Thrun et al. [13] verwenden dennoch Histogramme zur Verbesserung ihres dreidimensionalen Scanmatching Algorithmus. Ihr System verfügt über einen horizontal und einen vertikal befestigten Laserscanner. Um Ebenen zu extrahieren werden Histogramme von den Daten beider Scanner angefertigt.

Für die Verwendung mit einem 3D-Laserscanner, der durch Drehung Dreidimensionalität erreicht sind geometrische Transformationen notwendig um horizontale und vertikale Linien korrekt zu identifizieren.

2.1.4 Merkmalsbasiertes Scanmatching

Für die Lokalisierung eines autonomen mobilen Roboters existieren Methoden, die besonders in bekannten Umgebungen sehr effektiv sind. Anhand bestimmter Landmarken kann ein Roboter schnell erkennen, wo er sich in einer ihm bekannten Umgebung befindet. Dazu greift er häufig

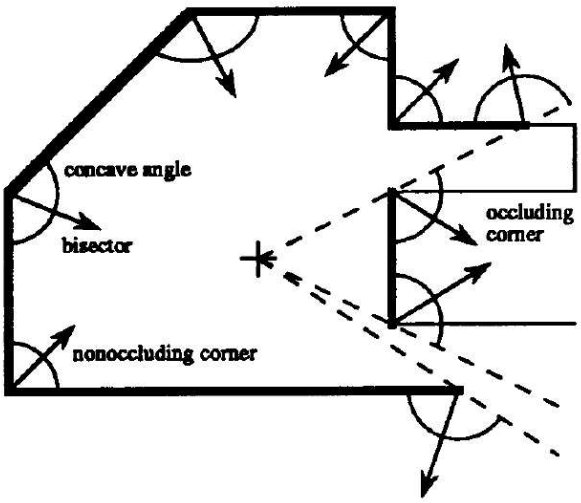


Abbildung 2.6: Ecken als Merkmale. Quelle: [28]

auf Kamerabilder zurück. In ähnlicher Form existieren aber auch Verfahren mit Laserscans, die auch zur Kartierung verwendbar sind [16].

Shaffer et al. verwenden merkmalsbasiertes Scanmatching bei Roboterfahrten in Minen [28]. Als Merkmale werden aus einem jeden Scan Ecken und Liniensegmente detektiert. Es wird unterschieden zwischen sichtbaren Ecken, die aus zwei sichtbar aufeinandertreffenden Liniensegmenten bestehen, und verdeckten Ecken, bei denen ein Liniensegment Teile eines anderen verdeckt. Eine jede Ecke wird durch die vier Parameter x, y, *konkaver Winkel* und *Bisektoren-Orientierung* definiert. Abbildung 2.6 stellt diese Parameter dar.

Liniensegmente werden durch ihre Länge, ihre Orientierung und und ihre Entfernung zum Ursprung definiert.

Der Matchingprozess basiert auf einer gut approximierten Roboterpose und Merkmalen mit geometrischem Informationsreichtum. Es werden Korrespondenzen zwischen den Merkmalen des aktuellen Scan und denen des Referenzscans bestimmt. Dabei werden zunächst jene Merkmale, die eindeutig mit einem Merkmal des Referenzscans übereinstimmen, gesucht und anschließend auch solche, die unter Verwendung einer Toleranzgrenze Ähnlichkeit zu den Merkmalen des Referenzscans haben. Die anzuwendende Tranformation ist die, welche die meisten eindeutigen und nicht eindeutigen Paare aufeinanderzieht.

Zur Bestimmung der exakten Transformation wird wiederum eine Fehlerfunktion minimiert, um die Distanzen zwischen den gefundenen Merkmalspaaren zu minimieren. Für jedes Merkmalspaar lässt sich nach der iterativen Gauss-Newton-Methode ein Fehlerterm

$$E = \begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{pmatrix} + \begin{pmatrix} \frac{\partial e_1}{\partial t_x} & \frac{\partial e_1}{\partial t_y} & \frac{\partial e_1}{\partial \theta} \\ \frac{\partial e_2}{\partial t_x} & \frac{\partial e_2}{\partial t_y} & \frac{\partial e_2}{\partial \theta} \\ \frac{\partial e_3}{\partial t_x} & \frac{\partial e_3}{\partial t_y} & \frac{\partial e_3}{\partial \theta} \\ \frac{\partial e_4}{\partial t_x} & \frac{\partial e_4}{\partial t_y} & \frac{\partial e_4}{\partial \theta} \end{pmatrix} \cdot \begin{pmatrix} d_x \\ d_y \\ d_\theta \end{pmatrix}$$

aufstellen mit den Fehlerwerten e_i, die für Linien die Differenz in der Orientierung und dem Abstand zum Ursprung sind und für Ecken die Differenz in den x- und y-Koordinaten. d_x, d_y und d_θ sind die Transformationsdifferenzen zwischen den aufeinanderfolgenden Iterationen. In geschlossener Form lässt sich eine Lösung für die Gesamttransformation finden, die die Distanzen zwischen allen Merkmalspaaren minimiert.

Stamos et al. [30] verwenden merkmalsbasiertes Scanmatching anhand von Linien und Ebenen für dreidimensionale Laserscans. Die Objekte, die aus einem Laserscan extrahiert und detektiert werden, sind planare Bereiche P, deren äußere und innere Grenzen B_{in} und B_{out}, und die Grenzlinien L_{in} und L_{out}. Jede dieser Grenzlinien wird definiert durch ein Fünftupel $(p_{start}, p_{end}, p_{id}, n, p_{size})$, bestehend aus Startpunkt p_{start}, Endpunkt p_{end}, der zugehörigen Ebene, repräsentiert durch einen eindeutigen Bezeichner p_{id}, der Normalen der Ebene n und der Größe der Ebene p_{size}, gegeben durch die Anzahl der Punkte auf der Ebene, dem Abstand zum Ursprung und der Flächennormalen.

Der paarweise Matching-Algorithmus arbeitet mit einer Menge von Grenzlinien und den entsprechenden Flächen der beiden Scans. In einem ersten Schritt werden korrespondierende Linienpaare gesucht und jene Paare herausgefiltert, deren Längen- oder Flächenverhältnisse nicht in einem vordefinierten Bereich liegen. Im nächsten Schritt werden alle Paare in lexikographischer Reihenfolge sortiert. Beginnend beim ersten Linienpaar wird eine passende Transformation (R, t) bestimmt, welche die Anzahl der Paare bestimmt, deren Entfernung durch (R, t) in einem bestimmten Toleranzbereich liegt. Nachdem diese Berechnung für alle Punktpaare durchgeführt wurde, wird die Transformation ausgewählt, die Korrespondenzen zwischen den meisten Punktpaaren rechtfertigt und mit diesen Punktpaaren der gewichtete Algorithmus der kleinsten Quadrate durchgeführt, um die optimale Tranformation zu bestimmen.

Wünstel et al. [39] verwenden ein ähnliches Verfahren und fügen aus Laserscans extrahierte Ebenen zusammen. Dabei machen sie sich zu Nutzen, dass viele Umgebungen von geraden Flächen dominiert sind. Sie detektieren in Laserscans Wände und analysieren die Beziehungen zwischen diesen. Für jeden Punkt wird der Normalenvektor und die Projektion auf eine durch Nachbarpunkte erstellte Ebene bestimmt. Mittels Singulärwertzerlegung wird dann die zugehörige Ebene berechnet. Durch Vergleich mit Nachbarregionen werden Ebenen des Scans gesucht. Anschließend werden analog zu dem Verfahren von Stamos et al. aus zwei Scans korrespondierende vertikale Ebenen gesucht und mit diesen eine Transformation bestimmt, die den Abstand zwischen den Paaren minimiert.

Der Vorteil von merkmalsbasiertem Scanmatching ist, dass die Wahrscheinlichkeit, ein lokales Minimum anstelle eines globalen zu erreichen, deutlich sinkt. Der Nachteil ist allerdings, dass die Extraktion von Merkmalen sehr zeitaufwändig sein kann.

2.2 Global konsistentes Scanmatching

Die bislang in diesem Kapitel beschriebenen Verfahren arbeiteten alle mit jeweils zwei Laserscans. In den meisten praktischen Anwendungen hat man es aber mit deutlich mehr Scans zu tun. Dabei ist es natürlich möglich, das gleiche Verfahren mehrfach anzuwenden und die

Scans nacheinander zu registrieren, indem man die Tranformation des i-ten Scans auch auf alle nachfolgenden anwendet. Dieses kann jedoch zu unerwünschten Effekten führen. Da ein Scanmatching nie zu perfekten Ergebnissen führt, sondern in jedem Schritt Fehler aufweist, wird beim Weiterreichen der Änderungen auch der Fehler fortgepflanzt. Wenn sich nun mehrere Scans überlappen, kann die Summierung dazu führen, dass ein Scan zwar an seinen Vorgänger gut angepasst ist, aber im Vergleich zu anderen Scans sehr fehlerhaft angeordnet wird.

Global konsistentes Scanmatching ist der Versuch, eine Menge von Scans so aneinander zu fügen, dass jeder Scan zu jedem anderen richtig positioniert ist. Hierbei gibt es verschiedene Ansätze, die, je nach Art der Anwendung, unterschiedlich gute Ergebnisse liefern.

Bei Umgebungen, in denen jeder Scan unabhängig von der Reihenfolge der Aufzeichnung eine Überlappung mit einer großen Anzahl von anderen Scans aufweist, erweisen sich leichte Modifikatoren eines paarweisen Scanmatchings als hilfreich [26]. Eine Möglichkeit ist es, einen Scan als Ankerpunkt auszuwählen. Alle anderen Scans werden per Scanmatching an diesen Anker angepasst. Dieses Verfahren verlangt jedoch, dass der Ankerscan einen großen Teil der gesamten abzubildenden Umgebung umfasst. Manchmal erreicht man das durch einen besonderen, einen zylindrischen Scan, der ein ganzes Objekt umfasst, für Kartierung ist dies jedoch wenig praktikabel.

Alternativ ist es möglich, jeden Scan an der Vereinigung aller vorherigen auszurichten. Dies führt allerdings in vielen Fällen zu einem enormen zusätzlichen Rechenaufwand. Außerdem wird so nicht unbedingt das Auftreten von Inkonsistenzen vermieden, wenn nur der letzte Scan in die bestmögliche Position gebracht wird, seine Vorgänger aber unverändert bleiben. Wird auf diese Weise an mehreren Stellen des Modells gearbeitet, tritt auch hier inkonsistentes Verhalten auf.

Deswegen entsteht die Notwendigkeit, auch die vorherigen Posen anzupassen. Eine mögliche Struktur ist in Abbildung 2.7 zu sehen. Betrachtet man die zu erstellende Karte als globales Modell, bestehend aus allen Laserscans und deren Posen, müssen beim Hinzufügen eines jeden Scans X_i zum Modell mehrere Schritte durchgeführt werden. Zuerst wird X_i ins globale Koordinatensystem transformiert und an das Modell gematcht. Die neu hinzugewonnenen Informationen werden verarbeitet und die restlichen Scans des Modells ebenfalls angepasst.

Eine ähnliche Struktur ist auch beim HILARE Projekt verwendet worden, einem der ersten Projekte, das sich mit Konsistenzen bei der dynamischen Modellierung befasst hat [18]. Dort wurden aus Laserscandaten Objekte segmentiert. Jeder Scan ist über die Pose mit dem globalen Modell assoziiert und erhält zusätzlich einen Wert, der die Unsicherheit der Pose dieses Scans angibt. Wird ein neuer Scan dem Modell hinzugefügt, wird er an die bisherigen Scans angepasst. Sobald ein Objekt aus einem früheren Scan wiedererkannt wird, werden die entsprechenden Posen mit Durchschnittsbildung erneuert. Ist die Posewahrscheinlichkeit des älteren Scans geringer als die des aktuell einzufügenden Scans, was der Normalfall ist, wenn der aktuelle Scan zu einem späteren Zeitpunkt der Roboterfahrt aufgenommen wurde, wird die spätere Pose korrigiert und der Fehler auf die vorherigen Scans verteilt, mit geringerem Effekt, je niedriger der Index des Scans ist.

Besonders bei Umgebungskartierung, welche die Exploration von unbekanntem Gelände beinhaltet, sind besondere Voraussetzungen gegeben. Hier werden die Scans anhand des Roboterpfads aneinander gereiht. Jeder Scan verfügt über Überlappungen mit einer geringen Anzahl

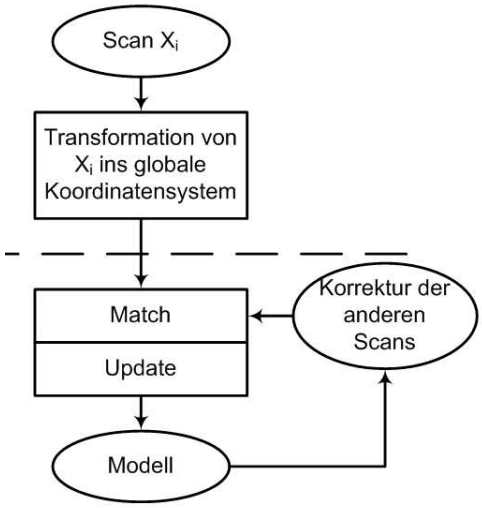


Abbildung 2.7: Struktur eines inkrementellen global konsistenten Scanmatchingalgorithmus.

von Vorgängern und Nachfolgern. Gemeinsamkeiten mit zeitlich weiter entfernt liegenden Scans sind eher selten. In diesem Fall bedeutet es einen erheblichen unnötigen Aufwand, alle Scans zu berücksichtigen. Vielmehr ist es sinnvoll, gezielt nach Schleifen zu suchen, nach Roboterpositionen, die sich im Bereich der Pose eines früheren Scans befinden.

Nüchter et al. [21, 31] benutzen ein Verfahren speziell für eine große Schleife, das heißt eine Roboterfahrt, bei der Start- und Endpunkt beinahe identisch sind. Der Algorithmus erkennt eine geschlossene Schleife, indem er überprüft, ob Übereinstimmungen des aktuellen Scans mit dem ersten vorliegen. Ist dies der Fall, wird der berechnete Fehler über alle Scans der Schleife gleichverteilt. Des Weiteren wird ein von Pulli [22] vorgestelltes Verfahren in leicht veränderter Form angewendet. Eine Schlange wird mit dem ersten Scan initialisiert, dessen fixierte Pose von nun an die Basis für das globale Koordinatensystem bildet. Dann werden folgende Schritte durchgeführt, bis die Schlange leer ist:

1. Der erste Scan wird aus der Schlange entfernt. Er ist nun der aktuelle Scan und bildet die Punktmenge D.

2. Ist der aktuelle Scan nicht der Startscan, werden alle Nachbarscans, jene Scans, die mit ihm mehr als p Punktpaare haben, ausgewählt, um die Punktmenge M zu bilden. D wird mit dem ICP-Algorithmus an M angepasst.

3. Wenn das Ergebnis des zweiten Schritts eine Transformation des aktuellen Scans erfordert, wird diese angewandt und alle Nachbarscans, die sich nicht bereits in der Schlange befinden, hinzugefügt.

Arbeiten von Durrant-Whyte beschäftigen sich ebenfalls mit diesem Problem [18]. In diesen Arbeiten besteht das Umgebungsmodell aus einer Menge von räumlichen Beziehungen zwischen Objekten. Mit einem probabilistischen Fusionsalgorithmus, ähnlich dem Kalman-Filter (vgl.

2.2.2), werden neue Messungen, in diesem Fall neue Scans, in das a priori Modell integriert. Die Konsistenz des Modells wird durch explizite Verbindungen zwischen den Schleifen des Netzes gewahrt.

2.2.1 Closing the Loop

Ein Problem, das maßgeblich an der Schwierigkeit von global konsistentem Scanmatching beteiligt ist, ist die Frage, wie man zwei zueinander gehörende Scans erkennt, allgemeiner gesagt, wie man erkennt, wann der Roboter sich wieder in der Nähe einer bereits einmal befahrenen Position befindet, denn die meisten Scanmatchingverfahren benötigen eine ausreichend gute Anfangsschätzung um überhaupt funktionieren zu können. Dafür ist es notwendig, den Roboter zu jeder Zeit lokalisieren zu können, in anderen Worten, den zurückgelegten Weg des Roboters ausreichend gut nachvollziehen zu können.

Die einfachste Methode hierfür ist die Verwendung von Odometrie. Mittels der Radumdrehungen des Roboters kann man den zurückgelegten Weg eines Roboters aufzeichnen und somit Roboterposen identifizieren, die nah aneinander liegen. Leider unterliegen Odometriedaten großen Fehlern. Durch wechselnden Untergrund oder Kurvenfahrten wird es schwieriger, gute Ergebnisse zu erzielen. Eine Abhilfe schafft dort die Kombination mit einem Drehratensensor (Gyroskop). Durch ein Gyroskop lässt sich bei konstanter Geschwindigkeit der Winkel der Drehung bestimmen. Besonders gute Messungen erzielt man bei tatsächlichen Drehungen, dort wo die Odometrie häufig fehlschlägt. Auf gerader Strecke sind Odometriedaten zuverlässiger, da ein Gyroskop einem Drift unterliegt. Durch Kombination der beiden Messungen lassen sich gute Ergebnisse erzielen.

Eine Alternative bietet die Verwendung von GPS-Daten. Diese sind nicht so fehleranfällig und verfügen über den deutlichen Vorteil unabhängig von einem Anfangswert zu sein. Das bedeutet insbesondere, dass die Schätzung nicht mit zunehmender Zeit schlechter wird, da sich die Fehler nicht aufsummieren. Nachteile liegen jedoch in der lokalen Ungenauigkeit und in der schlechten Verfügbarkeit, besonders in Gebäuden.

Verfahren, die weniger stark von Poseschätzungen abhängig sind, basieren auf der Erkennung von Merkmalen. Newman et al. verwenden das ATLAS-System [19], das auf Basis eines SLAM-Algorithmus einen Graph aus den lokalen Karten und ihren Verbindungen erstellt. Jede lokale Karte wird dabei mit einer Unsicherheit belegt, sowie auch jede Kante mit der Unsicherheit der Transformation belegt wird, mit der eine Karte auf die andere transformiert werden kann.

Das Auffinden von Schleifen geschieht dabei durch einen Matching-Algorithmus. Bei jeder neuen lokalen Karte, die zum Graphen hinzugefügt wird, wird über die Unsicherheit aus dem Graphen bestimmt, welcher Knoten den neuen überlappen könnte. Dabei gilt für nicht direkt verbundene Knoten jeweils der kürzeste Weg nach dem Dijkstra-Shortest-Path-Algorithmus.

Aus den so erreichten Kartenpaaren wird eine Liste von Merkmalen erstellt. Nun werden zuerst aus jeder Liste jene Merkmale entfernt, die einem anderen Merkmal derselben Karte entsprechen könnten. Anschließend werden aus den beiden Listen zweier potentiell korrespondierender Karten Paare gesucht. Für jedes dieser Paare wird eine Tranformation bestimmt, welche die Karten

aufeinander abbildet. Aus allen Tranformationen wird die Kovarianz für die beste Transformation bestimmt, welche in den Graphen eingetragen wird. Eine Kante gilt so lange als unbestätigt, bis sie durch eine weitere Korrespondenz bestätigt wird.

Mögliche Merkmale sind hier beispielsweise Linien oder Ecken aus Laserscans aber auch durch andere Sensoren erlangten Daten wie visuelle Merkmale aus Kamerabildern.

2.2.2 Probabilistisches Scanmatching

Eine sehr gebräuchliche Methode, Karten zu erstellen, arbeitet auf probabilistischer Basis [34], da Kartierung in der Robotik durch Unsicherheiten und Sensorrauschen charakterisiert ist.

Allgemein durch einen Bayes Filter formuliert, wird dann die Schätzung einer Karte M und einer Roboterpose s_t zum Zeitpunkt t gegeben durch

$$P(s_t, M|o^t, a^t) = \mu P(o_t|s_t, M) \int P(s_t|s_{t-1}, a_t) P(s_{t-1}, M|o^{t-1}, a^{t-1}) ds_{t-1}. \tag{2.4}$$

Dabei sind o_t die Beobachtungen oder Sensormessungen (Kamera, Laserscanner, usw.) zum Zeitpunkt t und a_t Aktionen, beispielsweise Bewegungskommandos oder Odometriemesswerte, im Zeitintervall $[t-1, t)$. Mit $o^t = o_1, o_2, ..., o_t$ und $a^t = a_1, a_2, ..., a_t$ wird die Menge alle Beobachtungen, beziehungsweise Aktionen bis zum Zeitpunkt t bezeichnet. Da angenommen wird, dass die Welt statisch ist, existiert kein Zeitindex für M.

Wichtig für die Schätzung (2.4) ist das Wahrnehmungsmodell $P(o|s, M)$, das in probalistischer Art und Weise das Generieren von Sensordaten von einer Roboterpose s und einer Karte M beschreibt. Des Weiteren benötigt man das Bewegungsmodell $P(s|a, s')$, das für die Wahrscheinlichkeit steht, dass eine im Zustand s' ausgeführte Aktion a zum Zustand s führt. Diese beiden Modelle sind allgemein in der Robotik zeitunabhängig. Diese Darstellung lässt sich jedoch nicht ohne weiteres in eine Implementation überführen, da die Wahrscheinlichkeitsverteilung über den Raum der Karten und Zustände unendlich viele Dimensionen hat. Aus diesem Grund ist es notwendig Zusatzannahmen zu machen, um wahrscheinlichkeitsbasierte Algorithmen zur Kartierung zu entwickeln.

Tabelle 2.1 gibt einen kurzen Überblick über einige der wichtigsten unterschiedlichen Ansätze auf diesem Gebiet. Das Feld Repräsentation gibt an, auf welche Weise die Karte repräsentiert wird. Bei der Unsicherheit wird unterschieden zwischen Maximum-Likelihood Methoden, bei denen eine einzige Schätzung für die Karte aufrechterhalten wird und der a posteriori Verteilung bei der die Karte zusammen mit ihrer Wahrscheinlichkeit charakterisiert wird. Die Konvergenzeigenschaften der Algorithmen werden ebenso aufgezeigt, wobei die Konvergenz des EM mit schwach bewertet wird, da sie auf lokale Karten beschränkt ist. Weitere Felder beschreiben, ob der Algorithmus zu lokalen Minima neigt, ob er inkrementell und somit eventuell in Echtzeit ausgeführt werden kann und ob die exakten Posen des Roboters bekannt sein müssen. Es folgen die Arten des Sensorrauschens, die Fähigkeit Schleifen abzubilden und die maximale Dimensionalität der generierbaren Karten. Die letzten beiden Felder geben an, ob ein Algorithmus mit unbekannten Korrespondenzen, wie mehreren ähnlichen Objekten in einer Umgebung, umgehen kann undob Rohdaten verarbeitet werden können oder eine Vorverarbeitung nötig ist.

Tabelle 2.1: Versuch die wichtigsten Kartierungsalgorithmen zu vergleichen. Quelle [34]

	Kalman	**EM**	**Incremental ML**	**Hybrid**	**Occupancy Grids**
Repräsentation	Landmarken	Hindernispunkte	Landmarken oder Gitterkarten	Hindernispunkte	Belegtheitsgitter
Unsicherheit	a posteriori Posen und Karte	Maximum-Likelihood Karte	(lokale) Maximum-Likelihood Karte	Maximum-Likelihood Karte	a posteriori Karte
Konvergenz	stark	schwach?	nein	nein	stark
Lokales Minimum	nein	ja	ja	ja	nein
Inkrementell	ja	nein	ja	ja	ja
Posen benötigt	nein	nein	nein	nein	ja
Sensorrauschen	normalverteilt	beliebig	beliebig	beliebig	beliebig
Kann Schleifen abbilden	ja	ja	nein	ja (nicht verschachtelt)	unbekannt
Kartendimensionalität	$\sim 10^3$	unbegrenzt	unbegrenzt	unbegrenzt	unbegrenzt
Korrespondenzen	nein	ja	ja	ja	ja
verarbeitet Rohdaten	nein	ja	ja	ja	ja

Kalman-Filter

Die Kartierung mit einem Kalman-Filter unterliegt den Annahmen, dass das Bewegungsmodell und das Wahrnehmungsmodell abgesehen von einem gaussverteilten Fehler linear sind und auch die initiale Unsicherheit gaussverteilt ist.

Der Standard Kalman-Filter [16] ist ein stochastischer Prozess über eine Folge von Zustandsvariablen $x_t \in \mathbb{R}^n$ mit den Messungen $o_t \in \mathbb{R}^m$, das über die Eingaben $a_t \in \mathbb{R}^l$ gelenkt wird. Eingaben sowie Beobachtungen sind fehlerbehaftet. Gesucht wird nun eine optimale Schätzung des tatsächlichen Zustands x_t.

Unter Verwendung der Zustandsübergangsmatrizen $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times l}$ und $H \in \mathbb{R}^{m \times n}$ und dem normalverteilten Rauschen $w \in \mathbb{R}^m$ und $v \in \mathbb{R}^m$ gilt:

$$\begin{aligned} x_t &= Ax_{t-1} + Ba_{t-1} + w_{t-1} \\ o_t &= Hx_t + v_t. \end{aligned}$$

Da die Roboterpose in der Regel jedoch nicht durch eine lineare Funktion gegeben ist, wird für den erweiterten Kalman-Filter (EKF) eine lineare Approximation durchgeführt:

$$\begin{aligned} x_t = f(x_{t-1}, a_{t-1}, w_{t-1}) &\approx \tilde{x}_t + A_t(x_{t-1} - \hat{x}_{t-1}) + W_t w_{t-1} \\ o_t = h(x_t, v_t) &\approx \tilde{o}_t + H_t(x_t - \hat{x}_t) + V_t v_t. \end{aligned}$$

Dabei ist $\hat{x}_t \in \mathbb{R}$ eine a priori Schätzung des Zustandes x_t, abhängig von den vorhergegangenen Zuständen und $\tilde{x}_t$ eine a posteriori Schätzung nach gegebener Messung o_t. Um fehlende Werte für das Rauschen zu berücksichtigen, verwendet man in der Praxis folgende Approximation:

$$\begin{aligned} \tilde{x}_t &= f(\tilde{x}_{t-1}, a_{t-1}, \mathbf{0}) \\ \tilde{o}_t &= h(\tilde{x}_t, \mathbf{0}). \end{aligned}$$

Mit $\mathbf{0}$ wird ein Vektor $(0, ..., 0)^T$ der passenden Dimension bezeichnet. Die weiteren Bausteine der Approximation sind die Jakobi-Matrizen der partiellen Ableitungen von f und h nach x und w:

$$\begin{aligned} A_t^{[i,j]} &= \frac{\partial f^{[i]}}{\partial x^{[j]}}(\hat{x}_{t-1}, a_{t-1}, \mathbf{0}) = \begin{pmatrix} \frac{\partial f^{[1]}}{\partial x^{[1]}} & \cdots & \frac{\partial f^{[1]}}{\partial x^{[n]}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f^{[n]}}{\partial x^{[1]}} & \cdots & \frac{\partial f^{[n]}}{\partial x^{[n]}} \end{pmatrix} \\ W_t^{[i,j]} &= \frac{\partial f^{[i]}}{\partial w^{[j]}}(\hat{x}_{t-1}, a_{t-1}, \mathbf{0}) \\ H_t^{[i,j]} &= \frac{\partial h^{[i]}}{\partial x^{[j]}}(\hat{x}_t, \mathbf{0}) \\ V_t^{[i,j]} &= \frac{\partial h^{[i]}}{\partial w^{[j]}}(\hat{x}_t, \mathbf{0}). \end{aligned}$$

Für den erweiterten Kalman-Filter bedeutet das, dass die Aktualisierung des EKF, das heißt, die Vorhersage des a priori Zustandes über die Gleichungen:

$$\begin{aligned} \tilde{x} &= f(\hat{x}_{t-1}, a_{t-1}, \mathbf{0}) \\ \tilde{P}_t &= A_t P_{t-1} A_t^T + W_t Q_{t-1} W_t^T \end{aligned}$$

gegeben wird. Die Aktualisierung des EKF bei neuer Messung, die Korrektur der a posteriori Schätzung, geschieht durch:

$$\begin{aligned} K_t &= \tilde{P}_t H_t^T (H_t \tilde{P}_t H_t^T + V_t R_t V_t^T)^{-1} \\ P_t &= (I - K_t H_t)\tilde{P}_t \\ \hat{x}_t &= \tilde{x}_t + K_t(o_t - \tilde{o}_t). \end{aligned}$$

Das Rauschen wird über die Kovarianzmatrizen Q_t und R_t bestimmt mit $w_t \sim \aleph(\mathbf{0}, Q_t)$ und $v_t \sim \aleph(\mathbf{0}, R_t)$. Die Kalman-Gewinnmatrix K_t minimiert die a priori Fehlerkovarianzmatrix $\tilde{P}$. Dadurch ergibt sich die a posteriori Schätzung $\hat{x}_t$ mit der Fehlerkovarianz P als Linearkombination aus der a priori Schätzung $\tilde{x}_t$ und der gewichteten Differenz zwischen Messung und Schätzung der Observation o_t.

Das Ergebnis dieser Filterung ist beweisbar optimal, im Sinne einer Minimierung der Fehlerkovarianz des vorhergesagten Zustandes.

Der Vorteil des Kalman-Filter-Ansatzes liegt in der Fähigkeit des Kalman-Filters, eine komplette a posteriori Schätzung online zu erstellen und dabei die Unsicherheit der Karte zu behalten. Ein Nachteil liegt in der Annahme von normalverteilten Fehlern, was dazu führt, dass Kalman-Filter nicht mit unbekannten Korrespondenzen umgehen können, da dieses ein multimodales Fehlermodell verlangt.

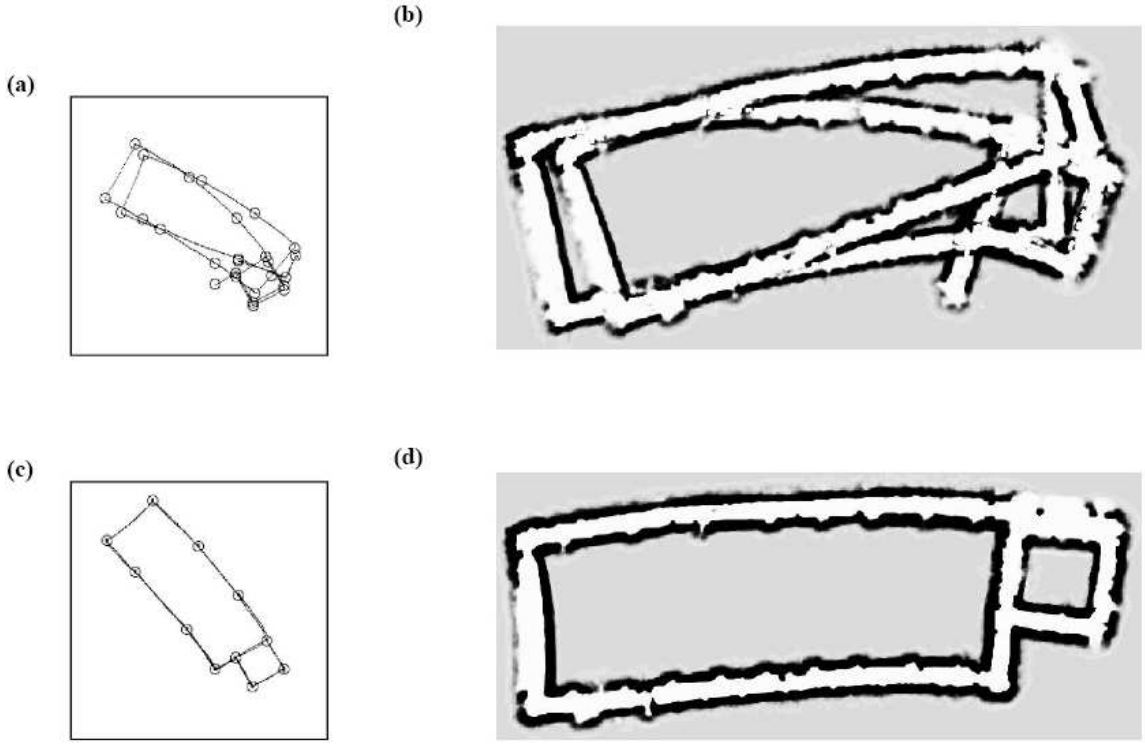


Abbildung 2.8: (a) Rohdaten einer zyklischen Umgebung mit nicht unterscheidbaren Landmarken. (c) Mit EM angepasster Roboterpfad. (b) und (d) Belegtheitsgitterkarten der Roboterpfade aus (a) und (c). Quelle: [34]

Erwartungsmaximierung

Erwartungsmaximierung ist eine bekannte Alternative zum Kalman-Filter, die durch die Unterteilung in zwei Schritte, den Erwartungs- und den Maximierungsschritt, charakterisiert wird [27]. Das Prinzip liegt darin, in jedem Zeitschritt die erwarteten Werte für unbekannte Größen („versteckte“ Variable) zu berechnen und diese Schätzungen so zu behandeln, als wären sie tatsächliche Beobachtungen.

Der EM-Algorithmus für alle gesammelten Daten d^t und alle „versteckten“ Variablen s_τ liefert die Karte

$$M^{[i+1]} = \underbrace{\arg\max_{M} \underbrace{\sum_{\tau} P(s_\tau | M^{[i]}, d^t) \log P(o_\tau | s_\tau, M)}_{Erwartungs-Schritt}}_{Maximierungs-Schritt} .$$

Der Erwartungs-Schritt setzt sich zusammen aus der log-Likelihood der Sensormessungen und dem Term $P(s_\tau | M^{[i]}, d^t)$, der das a posteriori der „versteckten“ Posen s_τ, gegeben der zum Zeitpunkt t vorhandenen Daten d^t ist, was der Schätzung der log-Likelihood der bereits verarbeiteten Daten entspricht. Dabei werden, auch im Fall $\tau < t$, für die Schätzung von s_τ alle im Intervall

$[1 \dots t]$ gesammelten Daten berücksichtigt. Dies hat zur Folge, dass einmal gemachte Schätzungen durch später erworbene Daten validiert werden und die Posen gegebenenfalls nachträglich verändert werden.

Der Maximierung-Schritt ist nun dafür verantwortlich, die aktuell beste Karte auszuwählen, die als Grundlage für den nächsten Schritt dient. Hierzu wird die Karte bestimmt, die unter den im E-Schritt berechneten Erwartungen die log-Likelihood der Sensormessungen $\log P(o_\tau|s_\tau, M)$ für alle Zeitpunkte τ und alle Posen s_τ maximiert.

Da für dieses Problem keine Lösung in geschlossener Form existiert, arbeiten viele Verfahren daran, für jede Position auf der Karte einzeln ein Ergebnis zu bestimmen [34]. Dies setzt jedoch eine endliche Anzahl an Positionen der Karte voraus, was zu einer weiten Verbreitung von Gitterkarten für die Kartierung mittels Erwartungsmaximierung geführt hat.

Ein großer Vorteil des EM-Ansatzes liegt in der Fähigkeit, Poseschätzungen nachträglich zu verändern. Durch die wiederholte Berechnung der a posteriori Posen können mehrere Hypothesen gleichzeitig aufrechterhalten werden und so auftretende Korrespondenzen bestätigt oder verworfen werden. Dies wird aus den in Abbildung 2.8 dargestellten Ergebnissen deutlich. Die Daten repräsentieren eine mit einem Roboter gefahrene Schleife. Aus den Sensordaten können die in Abbildung (a) skizzierten Landmarken extrahiert werden. Die fehlerhaften Odometriedaten erlauben es nicht, eine zusammenhängende Karte zu erstellen. Wird der Roboterpfad jedoch ein zweites Mal befahren und die Daten mit dem EM-Algorithmus bearbeitet, können Korrespondenzen zwischen den erkannten Landmarken gefunden werden und man erhält die in (c) und (d) gezeigten Ergebnisse.

Die Schwächen des Algorithmus liegen in der Laufzeit. Die verwendeten Wahrscheinlichkeiten müssen durch aufwendige Berechnungen immer neu bestimmt werden. Der Algorithmus kann somit nur offline angewandt werden und benötigt für eine Karte wie in dem Beispiel mehrere Stunden. Des Weiteren kann die Maximierung zu einem lokalen Maximum führen.

Hybride Ansätze

Hybride Kartierungsansätze basieren auf der „Inkrementellen Maximum-Likelihood Methode“ erweitert durch explizite Bestimmung der Unsicherheit in jedem Kartierungsschritt [12, 34].

Bei der „Inkrementellen Maximum-Likelihood Methode“ wird mit zunehmenden Laserdaten inkrementell, wie beim Maximierungsschritt des EM-Verfahrens, eine einzelne Karte aufrechterhalten, ohne jedoch dabei die restliche Unsicherheit weiter zu verfolgen. Die Maximum Pose s_t^* und Karte M_t^* erhält man durch Maximierung der Grenzwahrscheinlichkeit aus der vorherigen Karte und Pose und den neuen Beobachtungen mit

$$(M_t^*, s_t^*) = \arg\max_{M_t, s_t} P(o_t|M_t, s_t) P(s_t, M_t|a_t, s_{t-1}^*, M_{t-1}^*).$$

In der Praxis ist es ausreichend, die Posen s_t^* zu bestimmen, da die Karten direkt aus ihnen folgen. Mit diesem Einschrittverfahren ist es möglich, in Echtzeit eine Karte zu erstellen, was aber zur Folge hat, dass einmal geschätzte Posen nicht mehr verändert werden können. Dies führt auch dazu, dass zyklische Umgebungen nicht korrekt dargestellt werden.

Hybride Karten bestehen zusätzlich aus einer mit η normalisierten a posteriori Verteilung der Roboterposen s_t entsprechend dem Bayes-Filter:

$$P(s_t, o^t, a^t) = \eta P(o_t|s_t) \int P(s_t|a_t s_{t-1}) P(s_{t-1}|o^{t-1}, a^{t-1}) ds_{t-1}.$$

Auf diese Weise bestimmt der Roboter die Unsicherheit seiner Lokalisierung während der Fahrt. Erreicht er mit hoher Sicherheit die Nähe einer bereits angefahrenen Pose, wird der Fehler berechnet und auf die vorherigen Scans verteilt.

Dies entspricht einer schnellen Approximation des EM-Algorithmus, welche die beiden Schritte nur dann ausführt, wenn Diskrepanzen entdeckt werden. Eine große Schwäche liegt in der Ermangelung der Möglichkeit, das Verteilen der Fehler rückgängig zu machen, wenn sich dieses als falsch herausstellt, was zu einem kompletten Fehlschlagen der Kartenerstellung führen kann.

Belegtheitsgitterkarten

Die bislang behandelten Kartierungsalgorithmen arbeiten alle mit unbekannten Roboterposen, dem so genannten Simultaneous Localization and Mapping (SLAM) Problem. Zahlreiche Verfahren, die von bekannten Roboterposen ausgehen, verwenden Belegtheitsgitterkarten, teilweise in Verbindung mit den oben genannten Verfahren [34].

Das Problem, das bei Gitterkarten behandelt wird, ist die Verarbeitung verrauschter oder unvollständiger Sensordaten. Da es selbst bei bekannten Roboterposen schwierig ist, die Sensordaten derart zu interpretieren, dass sie korrekt darstellen, welche Bereiche der Umgebung belegt und welche nicht belegt sind.

Belegtheistgitterkarten für Entfernungsmesser (Sonar oder Laserscanner) generieren probabilistische Karten, um diese Ungenauigkeiten zu berücksichtigen. Diese Karten unterteilen den Raum der Karte in eine zwei- oder dreidimensionale Gitterstruktur und bestimmen für jede Zelle die Belegtheit. Dies lässt sich beispielsweise über einen binären Bayes-Filter erreichen. Die Belegtheit einer Zelle $m_{x,y}$ folgt dann aus

$$\frac{P(m_{x,y}|o^t, s^t)}{1 - P(m_{x,y}|o^t, s^t)} = \frac{P(m_{x,y}|o_t, s_t)}{1 - P(m_{x,y}|o_t, s_t)} \frac{1 - P(m_{x,y})}{P(m_{x,y})} \frac{P(m_{x,y}|o^{t-1}, s^{t-1})}{1 - P(m_{x,y}|o^{t-1}, s^{t-1})},$$

was alternativ in logarithmischer Form berechnet werden kann:

$$\log \frac{P(m_{x,y}|o^t, s^t)}{1 - P(m_{x,y}|o^t, s^t)} = \log \frac{P(m_{x,y}|o_t, s_t)}{1 - P(m_{x,y}|o_t, s_t)} + \log \frac{1 - P(m_{x,y})}{P(m_{x,y})} + \log \frac{P(m_{x,y}|o^{t-1}, s^{t-1})}{1 - P(m_{x,y}|o^{t-1}, s^{t-1})}.$$

Die Addition verringert die Rechenzeit und der Logarithmus vermeidet numerische Instabilitäten bei Wahrscheinlichkeiten nahe Null.

Die rekursive Struktur der Formel erlaubt eine inkrementelle Implementation, durch welche die Belegtheit der Gitterzellen an neue Sensordaten angepasst werden kann. Die beiden benötigten Wahrscheinlichkeiten sind die a priori Wahrscheinlichkeit für die Belegtheit einer Zelle $P(m_{x,y}|o_t, a_t)$ und das inverse Sensormodell $P(m_{x,y}|o_t, a_t)$, das, basierend auf einer einzigen

Sensormessung gemessen von einer Pose s_t, die Wahrscheinlichkeit dafür bestimmt, dass $m_{x,y}$ belegt ist. Das inverse Sensormodell wird dadurch charakterisiert, dass Zellen, die mit einer Messung übereinstimmen, mit großer Wahrscheinlichkeit, und jene zwischen der Pose des Roboters und dem Wert der Messung mit einer niedrigen Wahrscheinlichkeit belegt werden.

Die Beliebtheit von Belegtheitsgitterkarten resultiert aus ihrer Robustheit und der einfachen Implementierung. Jedoch werden die Probleme der Poseunsicherheit nicht behandelt. Ein weiterer Mangel ist die Annahme von unabhängigem Rauschen beim Bayes-Filter. Besonders bei stehendem Roboter sind Sensorfehler von Sonaren stark korreliert. Diese Schwäche wird in Implementationen umgangen, indem nur Messungen von fahrenden Robotern verwendet werden. Ein letztes Problem bildet die Annahme von Unabhängigkeiten zwischen einzelnen Gitterzellen, was unter Umständen zu minderwertigen Karten führt.

2.2.3 Global konsistentes Scanmatching in dieser Arbeit

In dieser Arbeit beschreiben wir ein Programm, welches global konsistente Kartenerstellung ermöglicht. Nach einer von Lu und Milios entwickelten Prozedur für zweidimensionale Laserscandaten [18] stellen wir ein System vor, in dem ein Netz aus Laserscans verwaltet wird. Zusätzlich zu den Laserscandaten werden die Roboterposen der Laserscans und die räumlichen Beziehungen zwischen den einzelnen Scans, die durch paarweises Scanmatching erlangt werden, durch das Netz repräsentiert. Durch ein Gleichungssystem, das durch die räumlichen Verhältnisse zwischen den Scans beschränkt ist, werden die Roboterposen so angepasst, dass alle Beziehungen möglichst gut beibehalten werden.

Kapitel 3

3D-Transformationen

3.1 Definition der Transformations-Operation

Im Folgenden werden die Translations- und Rotationsfunktionen als Operationen auf Posen definiert [18]. Eine Pose V sei hierbei definiert als $(P, O)^T$ mit einer Position P, dargestellt in kartesische Koordinaten und einer Orientierung O. Ein Roboter habe die Pose $V_a = (P_a, O_a)^T$ und fahre die Pose $V_b = (P_b, O_b)^T$ an, durch die Poseänderung $D = (P, O)^T$, dann sagen wir, dass V_a und D kombiniert werden, oder:

$$V_b = V_a \oplus D.$$

Sei R_a die Rotationsmatrix gegeben durch O_a, dann ist die Operation $\oplus$ definiert durch:

$$V_b = \begin{pmatrix} P_b \\ O_b \end{pmatrix} = \begin{pmatrix} P_a + R_a P \\ O_a \otimes O \end{pmatrix}. \tag{3.1}$$

Hierbei ist $O_a \otimes O$ die Komposition zweier Rotationsoperationen, also im Falle einer Darstellung durch Eulerwinkel stellt dies eine einfache Addition der Winkel dar. Die inverse Operation sei dementsprechend durch $\oslash$ gegeben.

Die inverse Transformations-Operation, das heißt die Operation $\ominus$, die angewandt auf zwei absolute Posen V_a und V_b die relative Posedifferenz D ergibt, ist:

$$D = V_b \ominus V_a.$$

Dies bedeutet nach Umformung der Gleichung (3.1):

$$D = \begin{pmatrix} P \\ O \end{pmatrix} = \begin{pmatrix} R_a^{-1}(P_b - P_a) \\ O_b \oslash O_a \end{pmatrix}.$$

Um die inverse Posedifferenz D_{ba} von $D_{ab} = V_a - V_b$ zu erhalten, wende man folgende Operation an:

$$D_{ba} = \ominus D_{ab} = 0 \ominus D_{ab}.$$

Ebenfalls wollen wir die Operation auf eine Pose V_a und einen Positionsvektor u definieren, dessen Ergebnis ein Positionsvektor u' ist:

$$u' = V_a \oplus u := P_a + R_a u.$$

Diese Operation ist insbesondere nützlich, um einen Punkt aus dem lokalen Koordinatensystem in das globale Koordinatensystem zu transformieren. In den nachfolgenden Abschnitten werden Posen sowohl in 3D als auch in 6D behandelt. An der grundlegenden Definition der Operation ändert dies allerdings nichts.

Im Falle einer planaren Sichtweise ist eine Pose $V = (x, y, \theta)$ ein Vektor bestehend aus zwei kartesischen Koordinaten und einem Winkel, um die Orientierung des Roboters festzulegen. Die Rotationsmatrix R_θ zu θ ist dann gegeben durch:

$$R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}.$$

Betrachten wir alle drei räumlichen Dimensionen, so wird die Pose zu einem 6-dimensionalen Vektor $V = (x, y, z, \theta_x, \theta_y, \theta_z)$ mit θ_x, θ_y und θ_z als Drehungen um die x-,y- bzw. z-Achsen. Diese und weitere Posen und deren Rotationsmatrizen werden im Folgenden besprochen.

3.2 Eulerwinkel

Eulerwinkel sind die klassische Art und Weise, Rotationen in 3 Dimensionen darzustellen. Nach dem eulerschen Rotationstheorem lassen sich alle Rotationen durch 3 Winkel repräsentieren. Sind also B, C und D die 3 Rotationen um die entsprechenden Winkel, so kann man die Gesamtrotation A darstellen als:

$$A = BCD.$$

Abhängig davon, um welche Achsen und in welcher Reihenfolge die Rotationen stattfinden, hat A eine andere Form.

Wir definieren die 6D-Pose V mit Hilfe der Eulerwinkel als:

$$V = \begin{pmatrix} x \\ y \\ z \\ \theta_x \\ \theta_y \\ \theta_z \end{pmatrix}.$$

Neben den kartesischen Koordinaten x, y und z für die Position geben die 3 Winkel θ_x, θ_y und θ_z die Orientierung an, wobei jeder Winkel den Grad der Rotation um eine der Hauptachsen angibt. Die einzelnen Rotationsmatrizen der Winkel sind wie folgt gegeben:

$$\begin{aligned} R_x &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{pmatrix} \\ R_y &= \begin{pmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & \\ -\sin\theta_y & 0 & \cos\theta_y \end{pmatrix} \\ R_z &= \begin{pmatrix} \cos\theta_y & -\sin\theta_y & 0 \\ \sin\theta_y & \cos\theta_y & 0 \\ 0 & 0 & 1 \end{pmatrix}. \end{aligned}$$

Daraus folgt die Gesamtrotationsmatrix $R = R_{\theta_x,\theta_y,\theta_z} = R_x R_y R_z$ als:

$$R = \begin{pmatrix} \cos\theta_y \cos\theta_z & -\cos\theta_y \sin\theta_z & \sin\theta_y \\ \cos\theta_z \sin\theta_x \sin\theta_y + \cos\theta_x \sin\theta_z & \cos\theta_x \cos\theta_z - \sin\theta_x \sin\theta_y \sin\theta_z & -\cos\theta_y \sin\theta_x \\ \sin\theta_x \sin\theta_z - \cos\theta_x \cos\theta_z \sin\theta_y & \cos\theta_z \sin\theta_x + \cos\theta_x \sin\theta_y \sin\theta_z & \cos\theta_x \cos\theta_y \end{pmatrix}.$$

Mit Hilfe dieser Rotationsmatrix können wir die Transformations-Operation $\oplus$ auf einer Pose $V_a = (x_a, y_a, z_a, \theta_{x_a}, \theta_{y_a}, \theta_{z_a})^T$ und einer Posedifferenz $D = (x, y, z, \theta_x, \theta_y, \theta_z)^T$ mit der resultierenden Pose $V_b = (x_b, y_b, z_b, \theta_{x_b}, \theta_{y_b}, \theta_{z_b})^T$ definieren:

$$\begin{aligned} V_b &= V_a \oplus D \\ &= \begin{pmatrix} P_b \\ O_b \end{pmatrix} = \begin{pmatrix} P_a + R_{\theta_{x_a},\theta_{y_a},\theta_{z_a}} P \\ O_a \otimes O \end{pmatrix} \\ P_b &= P_a + R_{\theta_{x_a},\theta_{y_a},\theta_{z_a}} P \\ &= \begin{pmatrix} x_a \\ y_a \\ z_a \end{pmatrix} + R_{\theta_{x_a},\theta_{y_a},\theta_{z_a}} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_b \\ y_b \\ z_b \end{pmatrix} \\ O_b &= O_a \otimes O \\ &= \begin{pmatrix} \theta_{x_a} + \theta_x \\ \theta_{y_a} + \theta_y \\ \theta_{z_a} + \theta_z \end{pmatrix} = \begin{pmatrix} \theta_{x_b} \\ \theta_{y_b} \\ \theta_{z_b} \end{pmatrix}. \end{aligned}$$

Analog lässt sich die inverse Operation $\ominus$ schreiben, als:

$$\begin{aligned}
D &= V_b \ominus V_a \\
&= \begin{pmatrix} P \\ O \end{pmatrix} = \begin{pmatrix} R^{-1}_{\theta_{x_a},\theta_{y_a},\theta_{z_a}}(P_b - P_a) \\ O_b \oslash O_a \end{pmatrix} \\
P &= R^{-1}_{\theta_{x_a},\theta_{y_a},\theta_{z_a}}(P_b - P_a) \\
&= R^{-1}_{\theta_{x_a},\theta_{y_a},\theta_{z_a}} \cdot \begin{pmatrix} x_b - x_a \\ y_b - y_a \\ z_b - z_a \end{pmatrix} \\
O &= O_b \oslash O_a \\
&= \begin{pmatrix} \theta_{x_b} - \theta_{x_a} \\ \theta_{y_b} - \theta_{y_a} \\ \theta_{z_b} - \theta_{z_a} \end{pmatrix}.
\end{aligned}$$

Drehungen mit Eulerwinkeln darzustellen hat jedoch einen nennenswerten Nachteil. Bei bestimmten Werten des zweiten Eulerwinkels, in diesem Fall bei $\theta_y = \pi/2$, tritt eine Singularität in der Rotationsmatrix auf.

Hierbei verschieben sich zwei der Drehachsen sozusagen aufeinander, was zur Folge hat, dass nachfolgende Drehungen in nur noch zwei Dimension möglich sind. Dieser Verlust eines Freiheitsgrades bei der Rotation wird mit „Gimbal Lock“ bezeichnet [8].

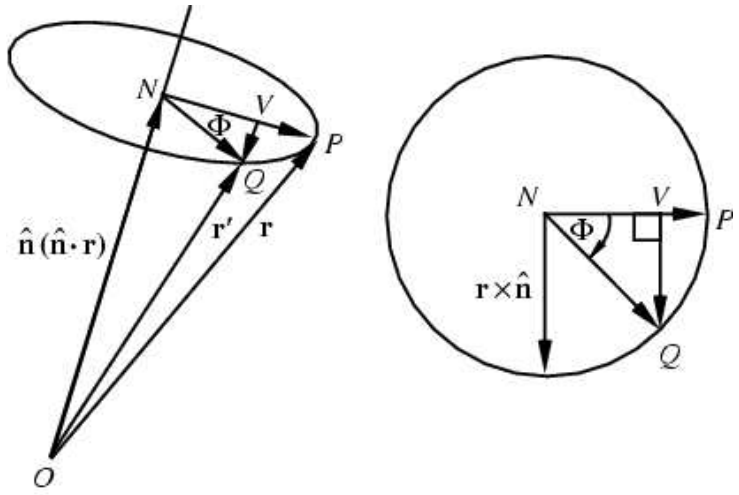


Abbildung 3.1: Rotation um eine beliebige Achse. Quelle [36]

3.3 Rotationsachse und Winkel

Eine weitere Möglichkeit, Rotationen darzustellen, ist es, eine einzige Rotationsachse und einen Drehwinkel gemeinsam als Repräsentation für die Orientierung zu nehmen (Abb. 3.1). Auch so lassen sich alle mögliche Rotationen repräsentieren. Haben wir eine Rotationsachse als einen normierten Vektor $n = (n_x, n_y, n_z)^T$ und einen Winkel θ, dann ist die Drehung von r nach r' gegeben durch:

$$\begin{aligned} p' &= \overrightarrow{0N} + \overrightarrow{NV} + \overrightarrow{VQ} \\ &= n(n \cdot r) + [r - n(n \cdot r)]\cos\theta + (r \times n)\sin\theta \\ &= p\cos\theta + n(n \cdot r)(1 - \cos\theta) + (r \times n)\sin\theta. \end{aligned}$$

Dies kann man auch mit Hilfe der Rotationsmatrix $R_{n,\theta}$ darstellen:

$$R_{n,\theta} = \mathbf{I}_3 + (nJ)\sin\theta + (nJ)^2(1 - \cos\theta)$$

$$nJ = \begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix}.$$

Die Matrix $R_{n,\theta}$ hat dann ausgeschrieben die Form:

$$R_{n,\theta} = \begin{pmatrix} (1-\cos\theta)n_x^2 + \cos\theta & (1-\cos\theta)n_x n_y + n_z\sin\theta & (1-\cos\theta)n_x n_z - n_y\sin\theta \\ (1-\cos\theta)n_x n_y - n_z\sin\theta & (1-\cos\theta)n_y^2 + \cos\theta & (1-\cos\theta)n_y n_z + n_x\sin\theta \\ (1-\cos\theta)n_x n_z + n_y\sin\theta & (1-\cos\theta)n_y n_z - n_x\sin\theta & (1-\cos\theta)n_z^2 + \cos\theta \end{pmatrix}.$$

Im nachfolgenden Abschnitt werden wir sehen, dass die Achse-Winkel-Darstellung zu der Darstellung mit Quaternionen äquivalent ist, daher verzichten wir hier auf eine zusätzliche Definition der Transformations-Operation.

3.4 Quaternionen

Quaternionen sind vierdimensionale hyperkomplexe Zahlen, die sich unter anderem auch dafür eignen, Rotationen im dreidimensionalen Raum darzustellen [14]. Sie entstehen aus den komplexen Zahlen, indem eine weitere Wurzel aus -1 mit der Bezeichnung j postuliert wird, die weder i noch $-i$ ist. Hieraus ergibt sich notwendigerweise ein weiteres Element $ij = k$, welches selbst auch wieder eine Wurzel aus -1 darstellt. Es gelten die folgenden Beziehungen:

$$\begin{aligned} i^2 &= j^2 = k^2 = ijk = -1 \\ ij &= k, & ji &= -k \\ jk &= i, & kj &= -i \\ ki &= j, & ik &= -j. \end{aligned}$$

Zusammen mit 1 bilden i,j und k eine Basis der Quaternionen $\mathbb{H}$, so bezeichnet nach ihrem Entdecker William Rowan Hamilton. Jedes Quaternion q aus $\mathbb{H}$ lässt sich also darstellen als $q = a + bi + cj + dk$. Somit definiert jedes 4-Tupel $(a, b, c, d)^T \in \mathbb{R}^4$ eindeutig ein Quaternion. Wir können also ein Quaternion auch in vektorieller Schreibweise notieren:

$$q = a + bi + cj + dk = [a, (b, c, d)] = [a, v] = \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix},$$

wobei jeder Eintrag eine reelle Zahl ist.

Wie in den komplexen Zahlen, so gibt es auch in den Quaternionen das Konzept der Konjugation eines Elementes. Das konjugierte Element zu dem Quaternion $q = a + bi + cj + dk = [a, v]$ wird bezeichnet mit q^* und ergibt sich durch die Negation der nicht-reellen Anteile, also als $q^* = a - bi - cj - dk = [a, -v]$. Mit dem konjugierten Element q^* zu q lässt sich der Betrag eines Quaternions beschreiben. Dafür werden wir das das Skalarprodukt benutzen, jedoch sind über den Quaternionen eine Vielzahl weiterer Multiplikationen definiert.

$$\|q\| = \sqrt{\langle q, q^* \rangle} = \sqrt{a^2 + b^2 + c^2 + d^2}.$$

Die nicht kommutative Multiplikation ergibt sich aus den bereits gegebenen Beziehungen, ist also mit $p = a_p + b_p i + c_p j + d_p k$ und $q = a_q + b_q i + c_q j + d_q k$ gegeben durch:

$$\begin{aligned} pq &= (a_p + b_p i + c_p j + d_p k)(a_q + b_q i + c_q j + d_q k) \\ &= (a_p a_q - b_p b_q - c_p c_q - d_p d_q) \\ &+ (a_p b_q + b_p a_q + c_p d_q - d_p c_q) i \\ &+ (a_p c_q - b_p d_q + c_p a_q + d_p b_q) j \\ &+ (a_p d_q + b_p c_q - c_p b_q + d_p a_q) k \\ &= [a_p, v_p][a_q, v_q] \\ &= [a_p a_q - v_p \cdot v_q, v_p \times v_q + a_p v_q + a_q v_p]. \end{aligned}$$

Von Bedeutung ist eine Teilmenge der Quaternionen, nämlich die der normierten Quaternionen. Dies sind alle Quaternionen q für die gilt $\|q\| = 1$. Diese Quaternionen besitzen die besondere Eigenschaft, dass das inverse Element q^{-1} von q, bezüglich des Skalarproduktes, das Konjugierte q^* ist. Denn es gilt:

$$\langle q, q^* \rangle = \|q\|^2 = 1.$$

Rotationen lassen sich in einfacher Form über normierte Quaternionen darstellen. Angenommen wir haben ein normiertes Quaternion $q = (a, b, c, d)^T = [a, v]$, das eine Rotation darstellen soll und einen Punkt $u = (x, y, z)^T \in \mathbb{R}^3$, den wir rotieren wollen. Um u zu rotieren, stellen wir ihn ebenfalls als Quaternion dar, so dass wir ihn nun als p bezeichen mit $p = (0, x, y, z)^T = [0, u]$. Man kann den rotierten Punkt p' folgendermaßen darstellen:

$$p' = qpq^*.$$

Warum dies einer Rotation gleichkommt, kann man sehen, nachdem man einige Umformungen gemacht hat. Für eine genauere Herleitung siehe [8].

$$\begin{aligned}
p' &= qpq^* \\
&= [a, v][0, u][a, v]^* \\
&= [a, v][0, u][a, -v] \\
&= [a, v][v \cdot r, au - u \times v] \\
&= [a(v \cdot u) - v \cdot (au - u \times v), sa(au - u \times v) + (v \cdot u)v + v \times (au - u \times v)] \\
&= [0, a^2 u + (v \cdot u)v - (v \cdot v)u + (v \cdot u)v + 2a(v \times u)] \\
&= [0, (s^2 - v \cdot v)u + 2(v \cdot u)v + 2a(v \times u)].
\end{aligned}$$

Schreiben wir das Einheitsquaternion q als $q = [\cos\theta, \sin\theta n]$, so ergibt sich:

$$\begin{aligned}
p' &= [0, (\cos^2\theta - \sin^2\theta(n \cdot n))u + 2((\sin\theta)n \cdot u)(\sin\theta)n + 2\cos\theta((\sin\theta)n \times u)] \\
&= [0, r\cos 2\theta + (1 - \cos 2\theta)(n \cdot u)n + (n \times u)\sin 2\theta].
\end{aligned}$$

Dies hat nun dieselbe Form wie die Rotation mit Achse und Winkel (man vergleiche mit Gleichung (3.2)). Möchte man also eine Rotation um die Achse n mit dem Winkel θ durch ein Quaternion q darstellen, so ist diese gegeben durch:

$$q = [\cos\frac{\theta}{2}, \sin\frac{\theta}{2}n].$$

Möchte man zusätzlich die Rotationsmatrix R_q zu dem Quaternion $q = (a, b, c, d)^T$ ermitteln, so gelingt dies durch die Betrachtung der einzelnen Komponenten von p':

$$p' = qpq^* = \begin{pmatrix} 0 \\ a^2x + b^2x - (c^2 + d^2)x + a(-2dy + 2cz) + 2b(cy + dz) \\ 2bcx + 2adx + a^2y - b^2y + c^2y - d^2y - 2abz + 2cdz \\ -2acx + 2bdx + 2aby + 2cdy + a^2z - b^2z - c^2z + d^2z \end{pmatrix} = \begin{pmatrix} 0 \\ x' \\ y' \\ z' \end{pmatrix}.$$

Dies lässt sich in eine einfache Matrixschreibweise mit der Rotationsmatrix R_q umformen:

$$u' = Ru = \begin{pmatrix} a^2+b^2-c^2-d^2 & 2bc-2ad & 2bd+2ac \\ 2ad+2bc & a^2-b^2+c^2-d^2 & 2cd-2ab \\ 2bd-2ac & 2ab+2cd & a^2-b^2-c^2+d^2 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}.$$

Nun können wir aus den kartesischen Koordinaten x, y, z und dem normierten Quaternion $q = (a, b, c, d)^T$ eine Pose V definieren.

$$V = \begin{pmatrix} P \\ q \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ q \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ a \\ b \\ c \\ d \end{pmatrix}.$$

Die Transformations-Operation $\oplus$ auf $V_a = (x_a, y_a, z_a, p_a, q_a, r_a, s_a)^T$ und $D = (x, y, z, p, q, r, s)^T$ ist dementsprechend definiert als:

$$\begin{aligned} V_b &= V_a \oplus D \\ &= \begin{pmatrix} P_b \\ q_b \end{pmatrix} = \begin{pmatrix} P_a + R_{q_a} P \\ q_a q \end{pmatrix} \\ P_b &= P_a + R_{q_a} P \\ &= \begin{pmatrix} x_a \\ y_a \\ z_a \end{pmatrix} + R_{q_a} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_b \\ y_b \\ z_b \end{pmatrix}. \end{aligned}$$

Damit gilt für die inverse Operation $\ominus$:

$$\begin{aligned} D &= V_b \ominus V_a \\ &= \begin{pmatrix} P \\ q \end{pmatrix} = \begin{pmatrix} R_{q_a}^{-1}(P_b - P_a) \\ q_b q_a^* \end{pmatrix} \\ P &= R^{-1}_{\theta_{x_a}, \theta_{y_a}, \theta_{z_a}}(P_b - P_a) \\ &= R_{q_a}^{-1} \cdot \begin{pmatrix} x_b - x_a \\ y_b - y_a \\ z_b - z_a \end{pmatrix}. \end{aligned}$$

3.5 Umwandlung der Repräsentationen

Unter Umständen ist es notwendig, zwischen verschiedenen Repräsentationen von Rotationen zu wechseln, um deren jeweilige Vorteile besser zu nutzen. Wie bereits erwähnt, ist die Umrechnung von einer Rotation um die normierte Achse $n = (n_1, n_2, n_3)^T$ mit dem Winkel θ zu dem Quaternion $q = (a, b, c, d)^T$ gegeben durch:

$$q = (\cos\frac{\theta}{2}, \sin\frac{\theta}{2}n_1, \sin\frac{\theta}{2}n_2, \sin\frac{\theta}{2}n_3)^T = [\cos\frac{\theta}{2}, \sin\frac{\theta}{2}n].$$

Die rückwärtige Umrechnung ergibt sich somit trivial als:

$$\begin{aligned} \theta &= 2\arccos a \\ n_1 &= \frac{b}{\sin\theta} \\ n_2 &= \frac{c}{\sin\theta} \\ n_3 &= \frac{d}{\sin\theta}. \end{aligned}$$

Um von den Eulerwinkeln $(\theta_x, \theta_y, \theta_z)^T$ zu dem korrespondierenden Quaternion $q = (a, b, c, d)^T$ zu gelangen, wenden wir die Multiplikation über den Quaternionen an. Zunächst stellen wir die einzelnen Rotationen der Eulerwinkel als Quaternionen dar und errechnen dann das resultierende Quaternion. Jeder Einzelne der Eulerwinkel repräsentiert eine Drehung um eine Achse, die einzelnen Quaternionen q_x,q_y und q_z lassen sich also in unserem Fall darstellen als:

$$q_x = \begin{pmatrix} \cos\frac{\theta_x}{2} \\ \sin\frac{\theta_x}{2} \\ 0 \\ 0 \end{pmatrix}, q_y = \begin{pmatrix} \cos\frac{\theta_y}{2} \\ 0 \\ \sin\frac{\theta_y}{2} \\ 0 \end{pmatrix}, q_z = \begin{pmatrix} \cos\frac{\theta_z}{2} \\ 0 \\ 0 \\ \sin\frac{\theta_z}{2} \end{pmatrix}.$$

Nun ist die Gesamtrotation gegeben durch das Quaternion q mit:

$$q = q_z q_y q_x = \begin{pmatrix} \cos\frac{\theta_x}{2}\cos\frac{\theta_y}{2}\cos\frac{\theta_z}{2} + \sin\frac{\theta_x}{2}\sin\frac{\theta_y}{2}\sin\frac{\theta_z}{2} \\ \sin\frac{\theta_x}{2}\cos\frac{\theta_y}{2}\cos\frac{\theta_z}{2} - \cos\frac{\theta_x}{2}\sin\frac{\theta_y}{2}\sin\frac{\theta_z}{2} \\ \cos\frac{\theta_x}{2}\sin\frac{\theta_y}{2}\cos\frac{\theta_z}{2} + \sin\frac{\theta_x}{2}\cos\frac{\theta_y}{2}\sin\frac{\theta_z}{2} \\ \cos\frac{\theta_x}{2}\cos\frac{\theta_y}{2}\sin\frac{\theta_z}{2} - \sin\frac{\theta_x}{2}\sin\frac{\theta_y}{2}\cos\frac{\theta_z}{2} \end{pmatrix}.$$

Dementsprechend lassen sich umgekehrt für ein Quaternion $q = (a, b, c, d)^T$ die dazugehörenden Eulerwinkel ermitteln:

$$\begin{pmatrix} \theta_x \\ \theta_y \\ \theta_z \end{pmatrix} = \begin{pmatrix} \arctan\frac{2(ab+cd)}{1-2(b^2+c^2)} \\ \arcsin 2(ac-bd) \\ \arctan\frac{2(ad+bc)}{1-2(c^2+d^2)} \end{pmatrix}.$$

Kapitel 4

Das Optimierungsproblem

In diesem Abschnitt diskutieren wir den von Lu und Milios vorgestellten allgemeinen Ansatz zur Findung von Roboterposen durch eine gegebene Menge an Posedifferenz-Messungen, die den Kanten in einem Netz aus Posen entsprechen. Das Problem wird zunächst probabilistisch formuliert. Anschließend wird eine Lösung des linearen Optimierungsproblems vorgestellt und gezeigt, dass diese anwendbar ist.

4.1 Definition des Schätzungs-Problems

Gegeben sei ein Graph mit $n+1$ Knoten $X_0, \ldots, X_n$ und einer Menge von Kanten. Jeder Knoten X_i ist ein d-dimensionale Posevektor dessen Wert unbekannt ist. Jede Kante $D_{i,j}$ korrespondiert zu einer gemessenen Posedifferenz zwischen 2 Knoten X_i und X_j. Die so genannte Posedifferenzgleichung $D_{i,j}$ ist eine Funktion von X_i und X_j. Von besonderer Bedeutung wird die lineare Posedifferenzgleichung $D_{i,j} = X_i - X_j$ sein. Die Beobachtung $\bar{D}_{i,j}$ von $D_{i,j}$ sei modelliert als $\bar{D}_{i,j} = D_{i,j} + \Delta D_{i,j}$, wobei $\Delta D_{i,j}$ ein gaussverteilte Zufallsvariable mit 0 als Erwartungswert und einer bekannten Kovarianz $C_{i,j}$ ist.

Ziel ist es nun, aus einer gegebenen Menge von $\bar{D}_{i,j}$ und $C_{i,j}$ die Werte der X_i zu ermitteln, so dass alle Messungen einfließen. Zusätzlich zu den Posevektoren sollen die zugehörigen Kovarianzen auf Basis der Kovarianzen der Posedifferenz ermittelt werden.
Um dies zu erreichen werden wir eine Art der Maximum-Likelihood Methode anwenden. Speziell geschieht dies über das, in der Statistik gebräuchliche, Mahalanobis Distanzmaß. Um die Gesamtwahrscheinlichkeit $P(D_{i,j}|\bar{D}_{i,j})$ über alle (i,j) zu maximieren, minimiere man die folgende Mahalanobisdistanz über alle Knoten X_i:

$$\mathbf{W} = \sum_{(i,j)} (D_{i,j} - \bar{D}_{i,j})^T C_{i,j}^{-1} (D_{i,j} - \bar{D}_{i,j}). \tag{4.1}$$

In der praktischen Anwendung für mobile Robotik stellen die Knoten X_i angefahrene Roboterposen dar, die je nachdem, ob man eine planare oder räumliche Sichtweise bevorzugt, in 3 bzw. 6 Dimensionen repräsentiert werden. In jedem Fall jedoch ist die Posedifferenzgleichung nicht linear, da die Orientierung des Roboters eine Rotationstransformation des Positionanteils der Pose erfordert. Die Schätzungen der Posedifferenzen können im Allgemeinen aus vielerlei Quellen, wie der Odometrie oder der Auswertung von Laserscans kommen.

Im Gegensatz zu einem kontinuierlichen Ansatz, in dem lediglich die aktuelle Pose des Roboters durch einen Kalman-Filter aufrecht erhalten wird, sind in diesem Ansatz alle Posen einer Änderung unterworfen, und alle Posedifferenzen gehen mit in die Berechnung ein.

Dies führt dazu, dass sich die ansonsten akkumulierenden Fehler verringern lassen und sich diese Methode somit gut dafür eignet um global konsistente Kartierung zu betreiben.

Für den Fall einer linearen Posedifferenzgleichung lässt sich eine geschlossene Lösung für eine optimale Schätzung aller Posen X_i und deren Kovarianzen angeben. Der relevantere Fall einer nicht-linearen Differenzgleichung lässt sich dann auf die lineare Lösung zurückführen, indem die Posedifferenz linearisiert wird.

4.2 Optimale Lösung für lineares Differenzmaß

In diesem Abschnitt gehen wir von dem Sonderfall der linearen Differenzgleichung an, das heißt: $D_{i,j} = X_i - X_j$. Die Vektoren X_i mit $i = 0, 1, \ldots, n$ seien d-dimensional und stellen die Knoten eines Graphen, beziehungsweise die Posen, dar. Jede gerichtete Kante von X_i nach X_j entspricht einer Posedifferenz $D_{i,j}$. Für jedes $D_{i,j}$ gibt es eine gaussverteilte Messung $\bar{D}_{i,j}$ um $D_{i,j}$ mit der invertierbaren Kovarianz $C_{i,j}$. Wir nehmen ohne Beschränkung der Allgemeinheit an, dass der Graph vollständig verbunden ist, also $k = \frac{n(n+1)}{2}$ Kanten hat. Sollte dies nicht der Fall sein und die Kante $D_{i,j}$ fehlen, so setze man das dazugehörende $C_{i,j}^{-1}$ gleich Null. Dies ändert *generell* nichts an der Lösbarkeit des Problems. Die Mahalanobisdistanz (4.1) ändert sich also folgendermaßen:

$$\mathbf{W} = \sum_{(0 \leq i < j \leq n)} (X_i - X_j - \bar{D}_{i,j})^T C_{i,j}^{-1} (X_i - X_j - \bar{D}_{i,j}). \tag{4.2}$$

Zu beachten ist, dass wie zuvor über alle Kanten des Graphen summiert wird, allerdings nehmen wir hier an, dass alle Kanten (i, j) so angeordnet sind, dass $i < j$ gilt. Ebenfalls ist es wichtig anzumerken, dass diese Gleichung nicht eindeutig lösbar ist, wenn man alle X_i mit $i = 0, 1, ..., n$ als freie Variablen ansieht. Daher benötigen wir einen Referenzpunkt oder Ankerknoten. Wir wählen ohne Beschränkung der Allgemeinheit X_0 als Referenzknoten mit $X_0 = 0$. Wir kennzeichnen einen Referenzknoten mit einem Balken $\bar{X}_0$. Somit bezeichnen die n freien Knoten $X_1, \ldots, X_n$ die Posen relativ zu $\bar{X}_0$.

Wir werden nun die Mahalanobis Gleichung in Matrixform darstellen. Sei dazu $\mathbf{X}$ die Konkatenation der X_1 bis X_n und $\mathbf{H}$ die Inzidenzmatrix des Graphen, wobei die Zeilen von $\mathbf{H}$ die Kanten und die Spalten die Knoten darstellen. $\mathbf{H}$ sei eine gerichtete Inzidenzmatrix, das heißt ihre Einträge bestehen nur aus 1,-1 und 0. Dann ist $\mathbf{D}$ die Verkettung der Positionsdifferenzen $D_{i,j} = X_i - X_j$, so dass gilt:

$$\mathbf{D} = \mathbf{HX}.$$

Sei dementsprechend $\bar{\mathbf{D}}$ die Verkettung der Observationen $\bar{D}_{i,j}$ und $\mathbf{C}$ die Kovarianz von $\bar{\mathbf{D}}$. Sollten die Messfehler unabhängig sein, so ist $\mathbf{C}$ eine blockdiagonale $nk \times nk$ Matrix deren Submatrizen die $C_{i,j}$ sind. Nun lässt sich ohne weiteres $\mathbf{W}$ umschreiben zu:

$$\mathbf{W} = (\bar{\mathbf{D}} - \mathbf{HX})^T \mathbf{C}^{-1} (\bar{\mathbf{D}} - \mathbf{HX}).$$

Der Lösungsvektor $\mathbf{X}$, welcher $\mathbf{W}$ minimiert ist wie folgt gegeben:

$$\mathbf{X} = (\mathbf{H}^T \mathbf{C}^{-1} \mathbf{H})^{-1} \mathbf{H}^T \mathbf{C}^{-1} \bar{\mathbf{D}}.$$

Die Kovarianz $\mathbf{C_X}$ dieser Lösung ist gegeben durch:

$$\mathbf{C_X} = (\mathbf{H}^T \mathbf{C}^{-1} \mathbf{H})^{-1}.$$

Bezeichnen wir $\mathbf{H}^T \mathbf{C}^{-1} \mathbf{H}$ mit $\mathbf{G}$, so setzen sich die einzelnen $d \times d$ Submatrizen $G_{i,j}$ von $\mathbf{G}$ folgendermaßen zusammen:

$$\begin{aligned} G_{i,i} &= \sum_{j=0}^{n} C_{i,j}^{-1} \\ G_{i,j} &= C_{i,j}^{-1} \qquad \text{für} \quad (i \neq j). \end{aligned}$$

$$\mathbf{G} = \begin{pmatrix} \sum_{j=0}^{n} C_{1,j}^{-1} & -C_{1,2}^{-1} & -C_{1,3}^{-1} & \dots & -C_{1,n}^{-1} \\ -C_{2,1}^{-1} & \sum_{j=0}^{n} C_{2,j}^{-1} & -C_{2,3}^{-1} & \ddots & \vdots \\ -C_{3,1}^{-1} & -C_{2,3}^{-1} & \ddots & \ddots & -C_{2,n}^{-1} \\ \vdots & \ddots & \ddots & \sum_{j=0}^{n} C_{n-1,j}^{-1} & -C_{n-1,n}^{-1} \\ -C_{n,1}^{-1} & \dots & -C_{n,2}^{-1} & -C_{n,n-1}^{-1} & \sum_{j=0}^{n} C_{n,j}^{-1} \end{pmatrix}. \tag{4.3}$$

Benennen wir den Ausdruck $\mathbf{H}^T \mathbf{C}^{-1} \bar{\mathbf{D}}$ mit $\mathbf{B}$ und beachten wir, dass gilt $\bar{D}_{i,j} = -\bar{D}_{j,i}$, so sind die einzelnen d-dimensionalen Subvektoren B_i zusammengesetzt aus:

$$B_i = \sum_{\substack{j=0 \\ j \neq i}}^{n} C_{i,j}^{-1} \bar{D}_{i,j}.$$

Die Lösungen für $\mathbf{X}$ bzw. $\mathbf{C_X}$ ergeben sich also direkt durch:

$$\mathbf{X} = \mathbf{G}^{-1} \mathbf{B}, \qquad \mathbf{C_X} = \mathbf{G^{-1}}.$$

4.3 Lösbarkeit des linearen Optimierungsproblems

Die Durchführung des Algorithmus steht und fällt mit der Invertierbarkeit von **G**. Im Folgenden wird gezeigt, dass, wenn wenige Bedingungen erfüllt sind, **G** die noch striktere Eigenschaft der positiven Definitheit besitzt. Der Beweis wird mit vollständiger Induktion über die Anzahl der Kanten k in einem Graphen mit $n+1$ Knoten erbracht. Zusätzlich setzen wir voraus, dass die Kovarianzen $C_{i,j}$ positiv definit sind. Um diesen Beweis einfacher zu gestalten wird vorher gezeigt, dass die Wahl des Referenzknoten in einem Graphen die positive Definitheit von **G** nicht beeinträchtigt.

Beweis: Sei **G** die positiv definite Matrix für einen Graphen mit $n+1$ Knoten $(\bar{X}_0, X_1, \ldots, X_n)$ und sei X_0 ohne Beschränkung der Allgemeinheit der dazugehörige Referenzknoten. Wechselt man den Referenzknoten von X_0 zu X_i so entsteht die Matrix $\mathbf{G}''$. Zu zeigen ist, dass $\mathbf{G}''$ positiv definit ist. **G** hat die Form wie in Gleichung (4.3), während sich $\mathbf{G}''$ von **G** lediglich in der i-ten Spalte/Zeile unterscheidet.

$$\begin{aligned} G''_{i,i} &= \sum_{j=1}^{n} C^{-1}_{0,j} \\ G''_{i,j} &= G'_{ji} = -C^{-1}_{0,j} \end{aligned}$$

Man erhält $\mathbf{G}''$ aus **G** indem man zunächst folgende Spaltenoperationen durchführt,

$$G'_{i,j} = -\sum_{k=1}^{n} G_{i,k},$$

und anschließend die $G'_{j,i}$ Submatrizen ähnlich transformiert:

$$\begin{aligned} G''_{i,i} &= -\sum_{k=1}^{n} G'_{k,i} = \sum_{k,l=1}^{n} G_{k,l} \\ G''_{j,i} &= -\sum_{k=1}^{n} G'_{k,i} = -\sum_{k=1}^{n} G_{k,i} \quad (i \neq j) \\ G''_{i,j} &= G'_{i,j} = -\sum_{k=1}^{n} G_{i,k}. \end{aligned}$$

Die erste Transformation entspricht der Linksmultiplikation von **G** mit der Matrix $\mathbf{I}_i$ deren $d \times d$ Submatrix $I_{i,j}$ negative Identitätsmatrizen der Dimension d sind. Also unterscheidet sich $\mathbf{I}_i$ von der Identitätsmatrix nur in dem i-ten Zeilenblock, der bei $\mathbf{I}_i$ vollständig mit negativen Identitäten besetzt ist. Wenn $I_{d(i-1)}$ und $I_{d(n-i+1)}$ Identitätsmatrizen der entsprechenden Dimensionen sind so lässt sich $\mathbf{I}_i$ schreiben als:

$$\mathbf{I}_i = \begin{pmatrix} I_{d(i-1)} & 0 & 0 \\ -I_d & \ldots & -I_d \\ 0 & 0 & I_{d(n-i+1)} \end{pmatrix}.$$

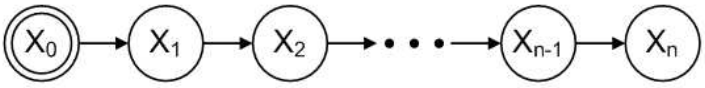


Abbildung 4.1: Minimalverbundener Graph mit $n+1$ Knoten.

Somit lässt sich $\mathbf{G}''$ formulieren als:

$$\mathbf{G}'' = \mathbf{I}_i \mathbf{G}.$$

Die zweite Transformation kommt dementsprechend der Rechtsmultiplikation mit $\mathbf{I}_i^T$ gleich, so dass:

$$\mathbf{G}'' = \mathbf{G}'\mathbf{I}_i^T = \mathbf{I}_i \mathbf{G} \mathbf{I}_i^T.$$

Da $\mathbf{I}_i$ invertierbar ist, folgt, dass $\mathbf{G}''$ positiv definit ist. □

Induktionsanfang $k = n$: Zunächst wird gezeigt, dass ein minimal verbundener Graph mit $n+1$ Knoten $(\bar{X_0}, \ldots, X_n)$ und k Kanten in einer invertierbaren Matrix $\mathbf{G}$ resultiert. Ein minimal verbundener Graph heißt in diesem Zusammenhang, das alle Knoten sequentiell angeordnet sind. Praktisch bedeutet dies, dass nacheinander angefahrene Positionen jeweils eine Kante bilden müssen (Abb. 4.1).

Ohne Beschränkung der Allgemeinheit sei X_0 der Ankerknoten, dann hat $\mathbf{G}$ die Form:

$$\mathbf{G} = \begin{pmatrix} C_{0,1}^{-1} + C_{1,2}^{-1} & -C_{1,2}^{-1} & 0 & \ldots & 0 \\ -C_{1,2}^{-1} & C_{1,2}^{-1} + C_{2,3}^{-1} & -C_{2,3}^{-1} & \ddots & \vdots \\ 0 & -C_{2,3}^{-1} & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & C_{n-2,n-1}^{-1} + C_{n-1,n}^{-1} & -C_{n-1,n}^{-1} \\ 0 & \ldots & 0 & -C_{n-1,n}^{-1} & C_{n-1,n}^{-1} \end{pmatrix},$$

mit

$$\begin{aligned} G_{i,i} &= C_{i-1,i}^{-1} + C_{i,i+1}^{-1} \\ G_{i,i+1} &= G_{i+1,i} = -C_{i,i+1}^{-1} \\ G_{i,j} &= 0 \qquad j+1 > i \qquad \text{oder} \qquad i+1 > j. \end{aligned}$$

Transformiert man nun die Matrix, indem auf die Submatrizen $G_{i,j}$ alle rechts liegenden Submatrizen addiert werden, entsteht:

$$G'_{i,j} = \sum_{k=j}^{n} G_{i,k}.$$

Nach einigen einfachen Umformungen ergeben sich für die Hauptdiagonale wie für die obere erste Nebendiagonale die Werte wie folgt:

$$\begin{aligned} G'_{i,i} &= G_{i,i} + G_{i,i+1} = C^{-1}_{i-1,i} + C^{-1}_{i,i+1} - C^{-1}_{i,i+1} = C^{-1}_{i-1,i} \\ G'_{i,i+1} &= G_{i,i+1}. \end{aligned}$$

Alle anderen Submatrizen fallen hingegen weg, denn:

$$\begin{aligned} G'_{i,j} &= G_{i,i-1} + G_{i,i} + G_{i,i-1} = 0 \qquad j < i \\ G'_{i,j} &= 0 \quad j > i+1. \end{aligned}$$

Dann ergibt sich die Matrix $\mathbf{G'}$ mit:

$$\mathbf{G'} = \begin{pmatrix} C^{-1}_{0,1} & -C^{-1}_{1,2} & 0 & \dots & 0 \\ 0 & C^{-1}_{1,2} & -C^{-1}_{2,3} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & C^{-1}_{n-2,n-1} & -C^{-1}_{n-1,n} \\ 0 & \dots & 0 & 0 & C^{-1}_{n-1,n} \end{pmatrix}.$$

Wenn zusätzlich auf $\mathbf{G'}$ eine ähnliche Transformation, jedoch jetzt vertikal durchgeführt wird, also

$$G''_{i,j} = \sum_{k=j}^{n} G'_{k,j},$$

hat dies offensichtlich die invertierbare Matrix G'' zur Folge.

$$\mathbf{G''} = \begin{pmatrix} C^{-1}_{0,1} & 0 & 0 & \dots & 0 \\ 0 & C^{-1}_{1,2} & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & C^{-1}_{n-2,n-1} & 0 \\ 0 & \dots & 0 & 0 & C^{-1}_{n-1,n} \end{pmatrix}.$$

Die entstehende blockdiagonale Matrix $\mathbf{G''}$ ist ebenfalls positiv definit, da sie aus positiv definiten Submatrizen besteht.
Die Transformationen, die auf $\mathbf{G}$ angewandt wurden, um $\mathbf{G''}$ zu erhalten, lassen sich mit Hilfe der oberen Dreiecksmatrix $\mathbf{I_D}$, die nur mit Identitätsmatrizen belegt ist, darstellen.

$$\begin{aligned} \mathbf{I_D} &= \begin{pmatrix} I_d & \dots & I_d \\ & \ddots & \vdots \\ 0 & & I_d \end{pmatrix} \\ \mathbf{G''} &= \mathbf{I_D G'} = \mathbf{I_D G I_D^T}. \end{aligned}$$

Da $\mathbf{I_D}$ invertierbar ist, folgt dass $\mathbf{G}$ positiv definit ist. □

Induktionsschritt $k \to k+1$**:** Sei $\mathbf{G}$ eine positiv definite Matrix, die zu einem Graphen mit $n+1$ Knoten $(\bar{X}_0, ..., X_n)$ und k Kanten korrespondiert. Zu beweisen ist nun, dass wenn dieser Graph um eine Kante $(X_i - X_j)$ erweitert wird, die resultierende Matrix $\mathbf{G}^*$ ebenfalls positiv definit ist. Die Kovarianz $C_{i,j}$ für diese Kante sei positiv definit. Zunächst transformieren wir, wie bereits gezeigt, den zugrundeliegenden Graphen so, dass der neue Referenzknoten X_i ist.

$$(\bar{X}_0, ..., X_j, ..., X_i, ..., X_n) \to (X_0, ..., X_j, ..., \bar{X}_i, ..., X_n)$$

Die zu diesem Graphen gehörende Matrix $\mathbf{G'}$ ist positiv definit. Da X_i nun der Referenzknoten ist, verändert das Hinzufügen der Kante $(X_i - X_j)$ an $\mathbf{G'}$ nur die Submatrix $G'_{j,j}$.

$$G'^*_{j,j} = G'_{j,j} + C^{-1}_{i,j}.$$

Die entstandene Matrix $\mathbf{G'}^*$ ist genau dann positiv definit, wenn gilt:

$$\mathbf{X}^T \mathbf{G'}^* \mathbf{X} > 0 \qquad \mathbf{X} \in \mathbb{R}^{d \cdot n} \qquad \mathbf{X} \neq 0.$$

Dies ist äquivalent zu

$$\sum_{k,l=1}^{n} X_k^T G'^*_{k,l} X_l > 0,$$

wobei X_k die d-dimensionalen Teilvektoren aus $\mathbf{X}$ sind. Offensichtlich gilt:

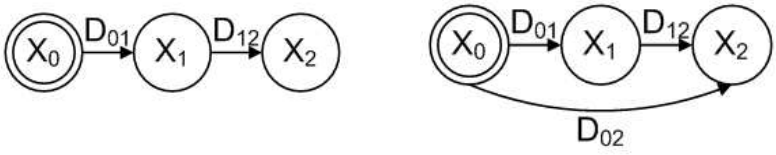
$$\begin{aligned}
\sum_{k,l=1}^{n} X_k^T G'^*_{k,l} X_l &= X_j^T G'^*_{j,j} X_j + \sum_{\substack{k,l=1 \\ k \neq l \neq j}}^{n} X_k^T G'_{k,l} X_l \\
&= X_j^T C^{-1}_{i,j} X_j + \sum_{k,l=1}^{n} X_k^T G'_{k,l} X_l \\
&= X_j^T C^{-1}_{i,j} X_j + \mathbf{X}^T \mathbf{G'} \mathbf{X} > 0.
\end{aligned}$$

Somit ist $\mathbf{G'}^*$ positiv definit. Außerdem lässt sich $\mathbf{G'}^*$ ohne weiteres in $\mathbf{G}^*$ transformieren indem man X_0 als neuen Referenzknoten wählt. Somit ist auch $\mathbf{G}^*$ positiv definit. □

Für das lineare Optimierungsproblem gibt es also demnach sicher eine Lösung, wenn folgende Bedingungen erfüllt sind:

1. Der Graph ist minimal verbunden.
2. Die Messfehler sind unabhängig.
3. Die einzelnen Kovarianzen sind positiv definit.

Bedingung 1 ist zwingend (siehe Induktionsanfang), allerdings praktisch keine Einschränkung, während sich die dritte Bedingung auch dazu erweitern lässt, dass alle Kovarianzen negativ definit sein können. Dann wäre $\mathbf{G}$ ebenfalls negativ definit und der Beweis analog durchführbar.



(a) Ein minimal verbundener Graph mit 3 Knoten und 2 Kanten

(b) Der minimale Graph erweitert zu einer geschlossenen Schleife

Abbildung 4.2: Beispiele einfacher Graphen.

4.4 Beispiele mit einfachen Graphen

Wir werden jetzt die Lösung des linearen Optimierungsproblem an einigen einfachen Beispielen verdeutlichen. Zunächst betrachten wir einen einfachen Graphen, wie in Abbildung 4.2(a), mit dem Ankerknoten $\bar{X}_0$. Seien zu jeder Kante die Poseschätzungen $\bar{D}_{0,1}$, $\bar{D}_{1,2}$ und deren Kovarianzen $C_{0,1}$ und $C_{1,2}$ gegeben. Wenn I_d die $d \times d$ Identitätsmatrix ist, dann ist die Matrix $\mathbf{G}$ für diesen Graphen gegeben durch:

$$\mathbf{G} = \mathbf{H}^T\mathbf{C}^{-1}\mathbf{H} = \begin{pmatrix} -I_d & I_d \\ 0 & -I_d \end{pmatrix} \begin{pmatrix} C_{0,1}^{-1} & 0 \\ 0 & C_{1,2}^{-1} \end{pmatrix} \begin{pmatrix} -I_d & 0 \\ I_d & -I_d \end{pmatrix} = \begin{pmatrix} C_{0,1}^{-1} + C_{1,2}^{-1} & -C_{1,2}^{-1} \\ -C_{1,2}^{-1} & C_{1,2}^{-1} \end{pmatrix}.$$

Der dazugehörige Vektor $\mathbf{B}$ hat dann die Form:

$$\mathbf{B} = \mathbf{H}^T\mathbf{C}^{-1}\bar{\mathbf{D}} = \begin{pmatrix} -I_d & I_d \\ 0 & -I_d \end{pmatrix} \begin{pmatrix} C_{0,1}^{-1} & 0 \\ 0 & C_{1,2}^{-1} \end{pmatrix} \begin{pmatrix} \bar{D}_{0,1} \\ \bar{D}_{1,2} \end{pmatrix} = \begin{pmatrix} -C_{0,1}^{-1}\bar{D}_{0,1} + C_{1,2}^{-1}\bar{D}_{1,2} \\ -C_{1,2}^{-1}\bar{D}_{1,2} \end{pmatrix}.$$

Die Lösung $\mathbf{X}$ des Gleichungsystem entspricht nun dem, was man intuitiv von einem so einfach strukturierten Graphen erwartet:

$$\mathbf{X} = \mathbf{G}^{-1}\mathbf{B} = \begin{pmatrix} \bar{D}_{0,1} \\ \bar{D}_{0,1} + \bar{D}_{1,2} \end{pmatrix}.$$

Also sind die Posen, welche der Algorithmus liefert, ganz einfach die Poseschätzungen, die zu Anfang vorlagen. Die Kovarianz C_X der ermittelten Lösung ist:

$$\mathbf{C_X} = \mathbf{G}^{-1} = \begin{pmatrix} C_{0,1} & C_{0,1} \\ C_{0,1} & C_{0,1} + C_{1,2} \end{pmatrix}.$$

Die Varianzen für die Posen sind die $d \times d$ Untermatrizen auf der Hauptdiagonalen, das heißt die Varianz der Pose X_1 ist $C_{0,1}$ während die Varianz von X_2 die Matrix $C_{0,1} + C_{1,2}$ ist.

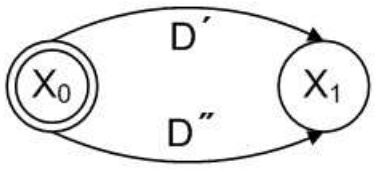


Abbildung 4.3: Ein einfacher Graph mit parallelen Kanten

Erweitert man denselben Graphen so, dass eine Kante von $\bar{X}_0$ nach X_2 hinzukommt, mit $\bar{D}_{0,2}$ der Schätzung der Posedifferenz und der dazugehörigen Kovarianz $C_{0,2}$, so verändern sich $\mathbf{G}$ und $\mathbf{B}$ folgendermaßen:

$$\mathbf{G} = \begin{pmatrix} C_{0,1}^{-1} + C_{1,2}^{-1} & -C_{1,2}^{-1} \\ -C_{1,2}^{-1} & C_{1,2}^{-1} + C_{0,2}^{-1} \end{pmatrix}$$

$$\mathbf{B} = \begin{pmatrix} -C_{0,1}^{-1}\bar{D}_{0,1} + C_{1,2}^{-1}\bar{D}_{1,2} \\ -C_{1,2}^{-1}\bar{D}_{1,2} - C_{0,2}^{-1}\bar{D}_{0,2} \end{pmatrix}.$$

Obwohl die Lösung für den zugrundeliegenden Graphen sehr einfach war, ist das Resultat durch das Schließen des Kreises um einiges komplizierter geworden:

$$\begin{aligned} \mathbf{X} &= \mathbf{G}^{-1}\mathbf{B} \\ &= \left(\begin{pmatrix} 1 & C_{0,1}^{-1} \end{pmatrix} \begin{pmatrix} C_{0,2}^{-1}C_{1,2}^{-1} \\ C_{0,2}^{-1} + C_{1,2}^{-1} \end{pmatrix} \right)^{-1} \\ &\quad \begin{pmatrix} C_{0,1}^{-1}(C_{0,2}^{-1} + C_{1,2}^{-1})\bar{D}_{0,1} - C_{0,2}^{-1}C_{1,2}^{-1}(\bar{D}_{0,2} + \bar{D}_{1,2}) \\ -C_{0,2}^{-1}C_{1,2}^{-1}\bar{D}_{0,2} + C_{0,1}^{-1}(-C_{0,2}^{-1}\bar{D}_{0,2} + C_{1,2}^{-1}(\bar{D}_{0,1} + \bar{D}_{1,2})) \end{pmatrix}. \end{aligned}$$

Ein weiterer interessanter Fall ist ein einfacher Graph mit 2 Knoten und Kanten, wie in Abbildung 4.3. $\mathbf{G}$ und $\mathbf{B}$ haben die einfache Form:

$$\begin{aligned} \mathbf{G} &= \left(C''^{-1} + C'^{-1}\right) \\ \mathbf{B} &= \left(C''^{-1}\bar{D}'' + C'^{-1}\bar{D}''\right). \end{aligned}$$

Hiermit ergibt sich die Pose X_1 und deren Varianz C_1 als:

$$\begin{aligned} X_1 &= \mathbf{G}^{-1}\mathbf{B} = (C''^{-1} + C'^{-1})^{-1}(C''^{-1}\bar{D}'' + C'^{-1}\bar{D}'') \\ C_1 &= \mathbf{G}^{-1} = (C''^{-1} + C'^{-1})^{-1}. \end{aligned}$$

Dies entspricht der Lösung, die auch ein Kalman-Filter liefert.

Kapitel 5

Ableitung der Posedifferenzen und Kovarianzen

In diesem Kapitel besprechen wir einige Möglichkeiten, für das lineare Optimierungsproblem geeignete Posedifferenzen und Kovarianzen zu erhalten. Hierzu werden zum einen Schätzungen aus der Odometrie und zum anderen Schätzungen aus dem Matchen zweier Laserscans herangezogen. Für beide Möglichkeiten werden jeweils 3D-Transformationsfunktionen und 6D-Transformationsfunktionen betrachtet.

5.1 Posedifferenzen aus Laserscans

Nehmen wir an, V_a und V_b seien zwei Posen aus einem Graphen, die durch eine starke Kante verbunden sind, das heißt es gibt eine relevante Überlappung der Laserscans. Aus dieser Überlappung gebe es eine Menge von m korrespondierenden Punktpaaren u_k^a, u_k^b mit $k = 1, \ldots, m$, die jeweils zu einem einzigen Punkt in der Umgebung des Roboters korrespondieren. Jeder Punkt u_k^a bzw. u_k^b ist in dem lokalen Koordinatensystem von V_a bzw. V_b gegeben. Ohne jegliche Fehler in den Messungen und der Auswahl eines Paares, würde ein Punktpaar folgende Gleichung erfüllen:

$$\Delta Z_k = V_a \oplus u_k^a - V_b \oplus u_k^b = 0.$$

Wenn nun Messfehler hinzukommen, kann man ΔZ_k als eine Zufallsvariable mit einer unbekannten Kovarianz C_k^Z ansehen. Wir stellen zunächst die Distanzfunktion $F_{ab}(V_a, V_b)$ zwischen zwei Posen als Summe über allen Punktpaaren auf.

$$F_{ab}(V_a, V_b) = \sum_{k=1}^{m} \|\Delta Z_k\|^2 = \sum_{k=1}^{m} \left\| V_a \oplus u_k^a - V_b \oplus u_k^b \right\|^2 = \sum_{k=1}^{m} \left\| (V_a \ominus V_b) \oplus u_k^a - u_k^b \right\|^2,$$

Also kann man F_{ab} als Funktion von der Posedifferenz $D' = V_a \ominus V_b$ sehen.

Um mit Hilfe von F_{ab} einen Summanden aus der linearen Mahalanobis-Distanz (4.2) darzustellen, linearisieren wir jedes ΔZ_k. Seien $\bar{V}_a$ und $\bar{V}_b$ Schätzungen von V_a und V_b mit den Fehlern ΔV_a und ΔV_b, so dass $\Delta V_a = \bar{V}_a - V_a$ und $\Delta V_b = \bar{V}_b - V_b$. Seien weiterhin u_k die globalen Koordinaten des Punktpaares (u_k^a, u_k^b), also $u_k \approx V_a \oplus u_k^a \approx V_b \oplus u_k^b$. Durch Taylor Expansion erster Ordnung von ΔZ_k nach V_a und V_b ergibt sich:

$$\begin{aligned}
\Delta Z_k &= V_a \oplus u_k^a - V_b \oplus u_k^b := F(V_a, V_b) \\
&\approx F(\bar{V}_a, \bar{V}_b) - [\nabla_{\bar{V}_a}(F(\bar{V}_a, \bar{V}_b))\Delta V_a - \nabla_{\bar{V}_b}(F(\bar{V}_a, \bar{V}_b))\Delta V_b] \\
&= \bar{V}_a \oplus u_k^a - \bar{V}_b \oplus u_k^b - [\nabla_{\bar{V}_a}(\bar{V}_a \oplus u_k^a)\Delta V_a - \nabla_{\bar{V}_b}(\bar{V}_b \oplus u_k^b)\Delta V_b].
\end{aligned}$$

Nehmen wir an, es gebe eine Matrixdekomposition von $\nabla_{\bar{V}_a}(\bar{V}_a \oplus u_k^a)$ und $\nabla_{\bar{V}_b}(\bar{V}_b \oplus u_k^b)$, derart, dass gilt :

$$\begin{aligned}
M_k H_a &= \nabla_{\bar{V}_a}(\bar{V}_a \oplus u_k^a) \\
M_k H_b &= \nabla_{\bar{V}_b}(\bar{V}_b \oplus u_k^b).
\end{aligned}$$

Zusätzlich gelte, dass M_k unabhängig von V_a und V_b ist, sowie dass H_a bzw. H_b unabhängig von u_k und H_b bzw. u_k und H_a sind.
Führen wir nun zusätzlich folgende Bezeichnungen ein,

$$\begin{aligned}
\bar{Z}_k &= \bar{V}_a \oplus u_k^a - \bar{V}_b \oplus u_k^b \\
D &= (H_a \Delta V_a - H_b \Delta V_b),
\end{aligned}$$

so lässt sich ΔZ_k darstellen als:

$$\begin{aligned}
\Delta Z_k &\approx \bar{V}_a \oplus u_k^a - \bar{V}_b \oplus u_k^b - M_k[H_a \Delta V_a - H_b \Delta V_b] \\
&= \bar{Z}_k - M_k D.
\end{aligned}$$

Nun ist D die linearisierte Posedifferenz, welche die alte Posedifferenz $D' = V_a \ominus V_b$ in F_{ab} ersetzen wird. Da D somit unabhängig von u_k ist, lässt sich für jedes Punktpaar k ein Fehler ΔZ_k formulieren. Die Fehlerfunktion F_{ab} kann jetzt leicht als eine Funktion von D geschrieben werden.

$$\begin{aligned}
F_{ab}(D) &= \sum_{k=1}^{m} \|\Delta Z_k\|^2 = \sum_{k=1}^{m} (\Delta Z_k)^T (\Delta Z_k) \\
&\approx \sum_{k=1}^{m} (\bar{Z}_k - M_k D)^T (\bar{Z}_k - M_k D).
\end{aligned}$$

Mit $\mathbf{Z}$, der Konkatenation aller $\bar{Z}_k$ und $\mathbf{M}$ der Aneinanderreihung aller M_k ist dies in Matrixform:

$$F_{ab}(D) \approx (\mathbf{Z} - \mathbf{M}D)^T (\mathbf{Z} - \mathbf{M}D).$$

Nach der Methode der kleinsten Quadrate wird diese Gleichung durch das $\bar{D}$ minimiert, das durch

$$\bar{D} = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{Z} \tag{5.1}$$

gegeben ist. Dieses $\bar{D}$ werden wir als Posedifferenzschätzung in dem globalen Optimierungsproblem verwenden.

Um die zugehörige Kovarianz C_D von $\bar{D}$ zu erhalten, betrachten wir den Fehler ΔZ_k. Wir machen die Annahme dass alle Messfehler unabhängig identisch normalverteilt um 0 mit der Varianz C_K^Z sind. Sei C_K^Z eine $d \times d$ Diagonalmatrix mit den Einträgen σ^2. Somit ist ein typischer Schätzer s^2 für σ^2 gegeben durch:

$$s^2 = (\mathbf{Z} - \mathbf{M}\bar{D})^T(\mathbf{Z} - \mathbf{M}\bar{D})/(2m-3) = \frac{F_{ab}(\bar{D})}{2m-3}.$$

Die Schätzung der Varianz von $\bar{D}$ ist demnach gegeben wie folgt:

$$C_D = \sigma^2(\mathbf{M}^T\mathbf{M})^{-1} \approx s^2(\mathbf{M}^T\mathbf{M})^{-1}. \tag{5.2}$$

Wir können nun eine Mahalanobisdistanz für die beiden Posen V_a und V_b aufstellen:

$$W_{ab} = (\bar{D} - D)^T C_D^{-1}(\bar{D} - D).$$

Diese Formulierung steht im Einklang mit dem linearen Optimierungsproblem aus Kapitel 4.2.

Im Verlaufe dieses Abschnittes haben wir grobe Annahmen über das Verhalten des Fehlers gemacht, um eine besonders einfach zu errechnende Kovarianzmatrix C_D zu erhalten. Allerdings kann der Messfehler in der Praxis durchaus systematisch sein und eine nichttriviale Varianz besitzen. Andere Schätzungen für die Kovarianzen können jedoch ebenfalls ohne große Schwierigkeiten benutzt werden.

5.1.1 3D Pose

Was bedeutet dies nun konkret für eine Pose, die gegeben ist durch $V = (x, y, \theta)$? Um dies zu beantworten, wenden wir uns zunächst der Transformations-Operation $\oplus$ zu. Wenn wir die im vorhergehenden Abschnitt dargestellten Rechnungen für diesen Spezialfall nachvollziehen wollen, müssen wir die Terme $\nabla_{\bar{V}_a}(\bar{V}_a \oplus u_k^a)$ und $\nabla_{\bar{V}_b}(\bar{V}_b \oplus u_k^b)$ bestimmen. Seien also dementsprechend die Posen $V_a = (x_a, y_a, \theta_a)$ und $V_b = (x_b, y_b, \theta_b)$ sowie die Punkte $u_k^a = (x_k^a, y_k^a)$ und $u_k^b = (x_k^b, y_k^b)$ gegeben. Zusätzlich sind $\bar{V}_a = (\bar{x}_a, \bar{y}_a, \bar{\theta}_a)$ und $\bar{V}_b = (\bar{x}_b, \bar{y}_b, \bar{\theta}_b)$ die Beobachtungen von V_a und V_b. Die Transformation von $u_k = (x_k, y_k) \approx V_a \oplus u_k^a$ ist nach Abschnitt 3.1 gegeben durch:

$$\begin{aligned} u_k &\approx \begin{pmatrix} x_a \\ y_a \end{pmatrix} + \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_k^a \\ y_k^a \end{pmatrix} \\ x_k &\approx x_a + x_k^a \cos\theta_a - y_k^a \sin\theta_a \\ y_k &\approx y_a + x_k^a \sin\theta_a + y_k^a \cos\theta_a. \end{aligned} \tag{5.3}$$

Somit folgt der Grad $\nabla_{\bar{V}_a}(\bar{V}_a \oplus u_k^a)$ als:

$$\nabla_{\bar{V}_a}(\bar{V}_a \oplus u_k^a) = \begin{pmatrix} 1 & 0 & -x_k^a \sin\bar{\theta}_a - y_k^a \cos\bar{\theta}_a \\ 0 & 1 & x_k^a \cos\bar{\theta}_a - y_k^a \sin\bar{\theta}_a \end{pmatrix}.$$

Für kleine ΔV_a gilt mit den Gleichungen (5.3):

$$\begin{aligned} \bar{y}_a - y_k &= -x_k^a \sin\bar{\theta}_a - y_k^a \cos\bar{\theta}_a \\ x_k - \bar{x}_a &= x_k^a \cos\bar{\theta}_a - y_k^a \sin\bar{\theta}_a. \end{aligned}$$

Also folgt:

$$\nabla_{\bar{V}_a}(\bar{V}_a \oplus u_k^a) = \begin{pmatrix} 1 & 0 & \bar{y}_a - y_k \\ 0 & 1 & x_k - \bar{x}_a \end{pmatrix}.$$

Nun kann ohne weiteres eine Matrixdekomposition $M_k H_a$ für $\nabla_{\bar{V}_a}(\bar{V}_a \oplus u_k^a)$ angegeben werden, die den Kriterien in Abschnitt 5.1 genügen:

$$M_k = \begin{pmatrix} 1 & 0 & -y_k \\ 0 & 1 & x_k \end{pmatrix}, \quad H_a = \begin{pmatrix} 1 & 0 & \bar{y}_a \\ 0 & 1 & -\bar{x}_a \\ 0 & 0 & 1 \end{pmatrix}.$$

Für $M_k H_b = \nabla_{\bar{V}_b}(\bar{V}_b \oplus u_k^b)$ lässt sich analog ein H_b finden:

$$H_b = \begin{pmatrix} 1 & 0 & \bar{y}_b \\ 0 & 1 & -\bar{x}_b \\ 0 & 0 & 1 \end{pmatrix}.$$

Denn offensichtlich gilt:

$$\begin{aligned} \Delta Z_k &\approx \bar{V}_a \oplus u_k^a - \bar{V}_b \oplus u_k^b - [\nabla_{\bar{V}_a}(\bar{V}_a \oplus u_k^a)\Delta V_a - \nabla_{\bar{V}_b}(\bar{V}_b \oplus u_k^b)\Delta V_b] \\ &= \bar{V}_a \oplus u_k^a - \bar{V}_b \oplus u_k^b - M_k[H_a \Delta V_a - H_b \Delta V_b]. \end{aligned}$$

Um die Posedifferenzschätzung $\bar{D} = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{Z}$ zu erhalten werden die $2m \times 3$ Matrix $\mathbf{M}$ und der $2m \times 1$ Vektor $\mathbf{Z}$ durch Konkatenation der M_k's bzw. der $\bar{Z}_k$'s konstruiert. $\mathbf{M}$ hat mit den Punkten $u_k = (x_k, y_k)$, für alle $k = 1, \ldots, m$ die Form:

$$\mathbf{M} = \begin{pmatrix} 1 & 0 & -y_1 \\ 0 & 1 & x_1 \\ \vdots & \vdots & \vdots \\ 1 & 0 & -y_m \\ 0 & 1 & x_m \end{pmatrix}.$$

Der Vektor $\mathbf{Z}$, bestehend aus den Observationen $\bar{Z}_k$ des Fehlers ΔZ_k, hat die Form:

$$\mathbf{Z} = \begin{pmatrix} (x_a + x_1^a \cos\theta_a - y_1^a \sin\theta_a) - (x_b + x_1^b \cos\theta_b - y_1^b \sin\theta_b) \\ (y_a + x_1^a \sin\theta_a + y_1^a \cos\theta_a) - (y_b + x_1^b \sin\theta_b + y_1^b \cos\theta_b) \\ \vdots \\ (x_a + x_m^a \cos\theta_a - y_m^a \sin\theta_a) - (x_b + x_m^b \cos\theta_b - y_m^b \sin\theta_b) \\ (y_a + x_m^a \sin\theta_a + y_m^a \cos\theta_a) - (y_b + x_m^b \sin\theta_b + y_m^b \cos\theta_b) \end{pmatrix}.$$

Mit Hilfe von $\mathbf{M}$ und $\mathbf{Z}$ lassen sich ohne weiteres mit den Gleichungen (5.1) und (5.2) die Observation der Posedifferenz $\bar{D}$ und deren Varianz C_D berechnen.

Insbesondere ist die Varianz C_D von Interesse, da wir nun zeigen, dass sie den Anforderung der positiven Definitheit genügt.

Beweis: $C_D = s^2(\mathbf{M}^T\mathbf{M})^{-1}$ ist genau dann positiv definit, wenn $(\mathbf{M}^T\mathbf{M})$ positiv definit ist, da $s^2 > 0$.

$$\mathbf{M}^T\mathbf{M} = \begin{pmatrix} m & 0 & \sum_{k=1}^{m} -y_k \\ 0 & m & \sum_{k=1}^{m} x_k \\ \sum_{k=1}^{m} -y_k & \sum_{k=1}^{m} x_k & \sum_{k=1}^{m} x_k^2 + y_k^2 \end{pmatrix}.$$

Die symmetrische Matrix $\mathbf{M}^T\mathbf{M}$ ist nach dem Hurwitz-Kriterium [4] genau dann positiv definit, wenn alle Determinanten der oberen linken Untermatrizen größer als Null sind. Offensichtlich erfüllen die oberen beiden Determinanten mit $m > 0$ und $m^2 > 0$ dieses Kriterium, also bleibt noch $\mathrm{Det}(\mathbf{M}^T\mathbf{M})$ mit :

$$\mathrm{Det}(\mathbf{M}^T\mathbf{M}) = m^2 \sum_{k=1}^{m} x_k^2 + y_k^2 - m\left(\sum_{k=1}^{m} y_k\right)^2 - m\left(\sum_{k=1}^{m} x_k\right)^2 \geq 0.$$

Da die Determinante trivialerweise nicht kleiner Null ist, bleibt noch zu Überprüfen ob der Gleichheitsfall eintritt. Nehmen wir dazu an, m sei größer oder gleich 2, das heißt es gibt mindestens zwei korrespondierende Punktpaare. Dies ist sinnvoll, da die Kovarianz C_D wegen $s^2 = \frac{F_{ab}}{2m-3}$ ansonsten ohnehin nicht existiert.

$$m \sum_{k=0}^{m} x_k^2 + y_k^2 - \left(\sum_{k=1}^{m} y_k\right)^2 - \left(\sum_{k=1}^{m} x_k\right)^2 = 0.$$

Es gilt:

$$\begin{aligned}
\mathrm{Det}(\mathbf{M}^T\mathbf{M}) &= m \sum_{k=1}^{m} x_k^2 + y_k^2 - \left(\sum_{k=1}^{m} y_k\right)^2 - \left(\sum_{k=1}^{m} x_k\right)^2 \\
&= m \sum_{k=1}^{m} x_k^2 + y_k^2 - \sum_{i,j=1}^{m} y_i y_j - \sum_{i,j=1}^{m} x_i x_j \\
&= (m-1) \sum_{k=1}^{m} x_k^2 + y_k^2 - \sum_{\substack{i,j=1 \\ i \neq j}}^{m} y_i y_j - \sum_{\substack{i,j=1 \\ i \neq j}}^{m} x_i x_j \\
&= \sum_{\substack{i,j=1 \\ i \neq j}}^{m} (x_i^2 + x_j^2 + y_i^2 + y_j^2) - \sum_{i=1}^{m} \sum_{j=i+1}^{m} 2y_i y_j - \sum_{i=1}^{m} \sum_{j=i+1}^{m} 2x_i x_j \\
&= \sum_{i,j=1}^{m} (x_i^2 + x_j^2 - 2x_i x_j + y_i^2 + y_j^2 - 2y_i y_j) \\
&= \sum_{i,j=1}^{m} ((x_i - x_j)^2 + (y_i - y_j)^2).
\end{aligned}$$

Dies ist nur dann gleich Null, falls alle Punkte u_k gleich sind, was ohnehin niemals der Fall sein sollte. Daher gilt:

$$\mathrm{Det}(\mathbf{M}^T\mathbf{M}) > 0.$$

Also ist die Kovarianz C_D positiv definit. □

5.1.2 6D Pose

Eulerwinkel

Für 6D-Pose gegeben durch $V = (x, y, z, \theta_x, \theta_y, \theta_z)$ lassen sich ähnliche Berechnungen anstellen. Seien die Poseschätzungen $\bar{V}_a = (\bar{x}_a, \bar{y}_a, \bar{z}_a, \bar{\theta}_{x_a}, \bar{\theta}_{y_a}, \bar{\theta}_{z_a})$ und $\bar{V}_b = (\bar{x}_b, \bar{y}_b, \bar{z}_b, \bar{\theta}_{x_b}, \bar{\theta}_{y_b}, \bar{\theta}_{z_b})$ von den Posen $V_a = (x_a, y_a, z_a, \theta_{x_a}, \theta_{y_a}, \theta_{z_a})$ und $V_b = (x_b, y_b, z_b, \theta_{x_b}, \theta_{y_b}, \theta_{z_b})$ im gleichen Sinne definiert. Ebenfalls gibt es m Punktpaare (u_k^a, u_k^b) mit $u_k^a = (x_k^a, y_k^a, z_k^a)$ und $u_k^b = (x_k^b, y_k^b, z_k^b)$, die zu jeweils einem Punkt $u_k = (x_k, y_k, z_k)$ korrespondieren. Die Korrelation $u_k \approx V_a \oplus a_k^a$ ist nach Abschnitt 3.2 gegeben durch:

$$
\begin{aligned}
x_k &\approx x_a - z_k^a \sin\theta_{y_a} + \cos\theta_{y_a}(x_k^a \cos\theta_{z_a} - y_k^a \sin\theta_{z_a}) && =: f_1(V_a)\\
y_k &\approx y_a + z_k^a \cos\theta_{y_a} \sin\theta_{x_a} + \cos\theta_{x_a}(y_k^a \cos\theta_{z_a} + x_k^a \sin\theta_{z_a})\\
&\quad + \sin\theta_{x_a} \sin\theta_{y_a}(x_k^a \cos\theta_{z_a} - y_k^a \sin\theta_{z_a}) && =: f_2(V_a)\\
z_k &\approx z_a - \sin\theta_{x_a}(y_k^a \cos\theta_{z_a} + x_k^a \sin\theta_{z_a})\\
&\quad + \cos\theta_{x_a}(z_k^a \cos\theta_{y_a} + \sin\theta_{y_a}(x_k^a \cos\theta_{z_a} - y_k^a \sin\theta_{z_a})) && =: f_3(V_a).
\end{aligned}
$$

Die nicht-trivialen relevanten Einträge von $\nabla_{\bar{V}_a}(\bar{V}_a \oplus u_k^a)$, also die partiellen Ableitungen nach den Winkeln sind nun:

$$
\begin{aligned}
\frac{\partial f_1}{\partial \theta_{x_a}}(\bar{V}_a) &= 0\\
\frac{\partial f_2}{\partial \theta_{x_a}}(\bar{V}_a) &= -\sin\bar{\theta}_{x_a}(y_k^a \cos\bar{\theta}_{z_a} + x_k^a \sin\bar{\theta}_{z_a})\\
&\quad + \cos\bar{\theta}_{x_a}(z_k^a \cos\bar{\theta}_{y_a} + \sin\bar{\theta}_{y_a}(x_k^a \cos\bar{\theta}_{z_a} - y_k^a \sin\bar{\theta}_{z_a}))\\
\frac{\partial f_3}{\partial \theta_{x_a}}(\bar{V}_a) &= -\cos\bar{\theta}_{x_a}(y_k^a \cos\bar{\theta}_{z_a} + x_k^a \sin\bar{\theta}_{z_a})\\
&\quad - \sin\bar{\theta}_{x_a}(z_k^a \cos\bar{\theta}_{y_a} + \sin\bar{\theta}_{y_a}(x_k^a \cos\bar{\theta}_{z_a} - y_k^a \sin\bar{\theta}_{z_a}))
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial f_1}{\partial \theta_{y_a}}(\bar{V}_a) &= -z_k^a \cos\bar{\theta}_{y_a} + \sin\bar{\theta}_{y_a}(y_k^a \sin\bar{\theta}_{z_a} - x_k^a \cos\bar{\theta}_{z_a})\\
\frac{\partial f_2}{\partial \theta_{y_a}}(\bar{V}_a) &= -\sin\bar{\theta}_{x_a}(z_k^a \sin\bar{\theta}_{y_a} + \cos\bar{\theta}_{y_a}(y_k^a \sin\bar{\theta}_{z_a}) - x_k^a \cos\bar{\theta}_{z_a})\\
\frac{\partial f_3}{\partial \theta_{y_a}}(\bar{V}_a) &= -\cos\bar{\theta}_{x_a}(z_k^a \sin\bar{\theta}_{y_a} + \cos\bar{\theta}_{y_a}(y_k^a \sin\bar{\theta}_{z_a}) - x_k^a \cos\bar{\theta}_{z_a})
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial f_1}{\partial \theta_{z_a}}(\bar{V}_a) &= -\cos\bar{\theta}_{y_a}(y_k^a \cos\bar{\theta}_{z_a} + x_k^a \sin\bar{\theta}_{z_a})\\
\frac{\partial f_2}{\partial \theta_{z_a}}(\bar{V}_a) &= -\sin\bar{\theta}_{x_a} \sin\bar{\theta}_{y_a}(y_k^a \cos\bar{\theta}_{z_a} + x_k^a \sin\bar{\theta}_{z_a}) + \cos\bar{\theta}_{x_a}(x_k^a \cos\bar{\theta}_{z_a} - y_k^a \sin\bar{\theta}_{z_a})\\
\frac{\partial f_3}{\partial \theta_{z_a}}(\bar{V}_a) &= -\cos\bar{\theta}_{z_a}(x_k^a \sin\bar{\theta}_{x_a} + y_k^a \cos\bar{\theta}_{x_a} \sin\bar{\theta}_{y_a}) + (y_k^a \sin\bar{\theta}_{x_a} - x_k^a \cos\bar{\theta}_{x_a} \sin\bar{\theta}_{y_a}) \sin\bar{\theta}_{z_a}.
\end{aligned}
$$

Diese lassen sich durch Ersetzungen in eine einfachere Form bringen, in der alle x_k^a,y_k^a und z_k^a verschwinden.

$$\begin{aligned}
\frac{\partial f_1}{\partial \theta_{x_a}}(\bar{V}_a) &= 0 \\
\frac{\partial f_2}{\partial \theta_{x_a}}(\bar{V}_a) &= z_k - \bar{z}_a \\
\frac{\partial f_3}{\partial \theta_{x_a}}(\bar{V}_a) &= \bar{y}_a - y_k \\
\frac{\partial f_1}{\partial \theta_{y_a}}(\bar{V}_a) &= (\bar{z}_a - z_k)\cos\bar{\theta}_{x_a} + (\bar{y}_a - y_k)\sin\bar{\theta}_{x_a} \\
\frac{\partial f_2}{\partial \theta_{y_a}}(\bar{V}_a) &= (x_k - \bar{x}_a)\sin\bar{\theta}_{x_a} \\
\frac{\partial f_3}{\partial \theta_{y_a}}(\bar{V}_a) &= (x_k - \bar{x}_a)\cos\bar{\theta}_{x_a} \\
\frac{\partial f_1}{\partial \theta_{z_a}}(\bar{V}_a) &= \cos\bar{\theta}_{y_a}((\bar{y}_a - y_k)\cos\bar{\theta}_{x_a} + (z_k - \bar{z}_a)\sin\bar{\theta}_{x_a} \\
\frac{\partial f_2}{\partial \theta_{z_a}}(\bar{V}_a) &= (x_k - \bar{x}_a)\cos\bar{\theta}_{x_a}\cos\bar{\theta}_{y_a} + (z_k - \bar{z}_a)\sin\bar{\theta}_{y_a} \\
\frac{\partial f_3}{\partial \theta_{z_a}}(\bar{V}_a) &= (\bar{x}_a - x_k)\cos\bar{\theta}_{y_a}\sin\bar{\theta}_{x_a} + (\bar{y}_a - y_k)\sin\bar{\theta}_{y_a}.
\end{aligned}$$

Somit lässt sich $\nabla_{\bar{V}_a}(\bar{V}_a \oplus u_k^a)$ also schreiben als:

$$\nabla_{\bar{V}_a}(\bar{V}_a \oplus u_k^a) = \begin{pmatrix} 1 & 0 & 0 & \frac{\partial f_1}{\partial \theta_{x_a}}(\bar{V}_a) & \frac{\partial f_1}{\partial \theta_{y_a}}(\bar{V}_a) & \frac{\partial f_1}{\partial \theta_{z_a}}(\bar{V}_a) \\ 0 & 1 & 0 & \frac{\partial f_2}{\partial \theta_{x_a}}(\bar{V}_a) & \frac{\partial f_2}{\partial \theta_{y_a}}(\bar{V}_a) & \frac{\partial f_2}{\partial \theta_{z_a}}(\bar{V}_a) \\ 0 & 0 & 1 & \frac{\partial f_3}{\partial \theta_{x_a}}(\bar{V}_a) & \frac{\partial f_3}{\partial \theta_{y_a}}(\bar{V}_a) & \frac{\partial f_3}{\partial \theta_{z_a}}(\bar{V}_a) \end{pmatrix}.$$

Nun finden wir eine Matrixdekomposition, welche die x_k,y_k und z_k von den Einträgen der Poseschätzung $\bar{V}_a$ trennt. Wenn man beachtet, dass die partiellen Ableitungen, Polynome ersten Grades in x_k,y_k und z_k sind, ist dies zu bewerkstelligen, indem in H_a der additive Anteil von dem multplikativen Anteil getrennt wird.

Setzen wir also M_k so fest, dass die additiven Elemente mit 1 multipliziert werden und die multiplikativen mit x_k,y_k und z_k dann entsteht die folgende oder eine ähnliche Matrix:

$$M_k = \begin{pmatrix} 1 & 0 & 0 & 0 & -y_k & -z_k \\ 0 & 1 & 0 & z_k & x_k & 0 \\ 0 & 0 & 1 & -y_k & 0 & x_k \end{pmatrix}.$$

Mit diesem M_k folgt H_a zwingend als:

$$H_a = \begin{pmatrix} 1 & 0 & 0 & 0 & \bar{z}_a \cos(\bar{\theta}_{x_a}) + \bar{y}_a \sin(\bar{\theta}_{x_a}) & \bar{y}_a \cos(\bar{\theta}_{x_a}) \cos(\bar{\theta}_{y_a}) - \bar{z}_a \cos(\bar{\theta}_{y_a}) \sin(\bar{\theta}_{x_a}) \\ 0 & 1 & 0 & -\bar{z}_a & -\bar{x}_a \sin(\bar{\theta}_{x_a}) & -\bar{x}_a \cos(\bar{\theta}_{x_a}) \cos(\bar{\theta}_{y_a}) - \bar{z}_a \sin(\bar{\theta}_{y_a}) \\ 0 & 0 & 1 & \bar{y}_a & -\bar{x}_a \cos(\bar{\theta}_{x_a}) & \bar{x}_a \cos(\bar{\theta}_{y_a}) \sin(\bar{\theta}_{x_a}) + \bar{y}_a \sin(\bar{\theta}_{y_a}) \\ 0 & 0 & 0 & 1 & 0 & \sin(\bar{\theta}_{y_a}) \\ 0 & 0 & 0 & 0 & \sin(\bar{\theta}_{x_a}) & \cos(\bar{\theta}_{x_a}) \cos(\bar{\theta}_{y_a}) \\ 0 & 0 & 0 & 0 & \cos(\bar{\theta}_{x_a}) & -\cos(\bar{\theta}_{y_a}) \sin(\bar{\theta}_{x_a}) \end{pmatrix}.$$

Man beachte besonders die obere rechte 3×3 Teilmatrix, deren Einträge die additiven Werte der respektiven partiellen Ableitungen sind. Die Einträge der unteren rechten 3×3 Matrix hingegen sind die Faktoren der Ableitungen, so verteilt, dass sie richtig mit den x_k,y_k und z_k aufeinandertreffen.

Die Matrixdekomposition $M_k H_b$ für $\nabla_{\bar{V}_b}(\bar{V}_b \oplus u_k^b)$ lässt sich gleichsam finden durch:

$$H_b = \begin{pmatrix} 1 & 0 & 0 & 0 & \bar{z}_b \cos(\bar{\theta}_{x_b}) + \bar{y}_b \sin(\bar{\theta}_{x_b}) & \bar{y}_b \cos(\bar{\theta}_{x_b}) \cos(\bar{\theta}_{y_b}) - \bar{z}_b \cos(\bar{\theta}_{y_b}) \sin(\bar{\theta}_{x_b}) \\ 0 & 1 & 0 & -\bar{z}_b & -\bar{x}_b \sin(\bar{\theta}_{x_b}) & -\bar{x}_b \cos(\bar{\theta}_{x_b}) \cos(\bar{\theta}_{y_b}) - \bar{z}_b \sin(\bar{\theta}_{y_b}) \\ 0 & 0 & 1 & \bar{y}_b & -\bar{x}_b \cos(\bar{\theta}_{x_b}) & \bar{x}_b \cos(\bar{\theta}_{y_b}) \sin(\bar{\theta}_{x_b}) + \bar{y}_b \sin(\bar{\theta}_{y_b}) \\ 0 & 0 & 0 & 1 & 0 & \sin(\bar{\theta}_{y_b}) \\ 0 & 0 & 0 & 0 & \sin(\bar{\theta}_{x_b}) & \cos(\bar{\theta}_{x_b}) \cos(\bar{\theta}_{y_b}) \\ 0 & 0 & 0 & 0 & \cos(\bar{\theta}_{x_b}) & -\cos(\bar{\theta}_{y_b}) \sin(\bar{\theta}_{x_b}) \end{pmatrix}.$$

Um die Posedifferenzschätzung $\bar{D}$ zu erhalten, werden ähnlich wie im $2D$-Fall, die $3m \times 6$ Matrix $\mathbf{M}$ und der $3m$ Vektor $\mathbf{Z}$ konstruiert.

$$\mathbf{M} = \begin{pmatrix} 1 & 0 & 0 & 0 & -y_1 & -z_1 \\ 0 & 1 & 0 & z_1 & x_1 & 0 \\ 0 & 0 & 1 & -y_1 & 0 & x_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 0 & 0 & -y_m & -z_m \\ 0 & 1 & 0 & z_m & x_m & 0 \\ 0 & 0 & 1 & -y_m & 0 & x_m \end{pmatrix}$$

$$\mathbf{Z} = \begin{pmatrix} \bar{V}_a \oplus u_1^a - \bar{V}_b \oplus u_1^b \\ \bar{V}_a \oplus u_2^a - \bar{V}_b \oplus u_2^b \\ \vdots \\ \bar{V}_a \oplus u_m^a - \bar{V}_b \oplus u_m^b \end{pmatrix}.$$

Wir wollen auch hier zeigen, dass die Varianz $C_D = s^2(\mathbf{M}^T\mathbf{M})^{-1}$ von $\bar{D}$ positiv definit ist.

Beweis: Auch in diesem Fall gilt $s^2 > 0$, daher reicht es aus zu zeigen, dass $\mathbf{M}^T\mathbf{M}$ positiv definit ist.

$$\mathbf{M}^T\mathbf{M} = \sum_{k=1}^{m} \begin{pmatrix} 1 & 0 & 0 & 0 & -y_k & -z_k \\ 0 & 1 & 0 & z_k & x_k & 0 \\ 0 & 0 & 1 & -y_k & 0 & x_k \\ 0 & z_k & -y_k & y_k^2+z_k^2 & x_k z_k & -x_k y_k \\ -y_k & x_k & 0 & x_k z_k & y_k^2+x_k^2 & y_k z_k \\ -z_k & 0 & x_k & -x_k y_k & y_k z_k & x_k^2+z_k^2 \end{pmatrix}.$$

$\mathbf{M}^T\mathbf{M}$ ist genau dann positiv definit, wenn gilt:

$$a^T\mathbf{M}^T\mathbf{M}a > 0, \qquad a \in \mathbb{R}^6, \quad a \neq 0.$$

Mit $a = (a_1, a_2, a_3, a_4, a_5, a_6)^T$ bedeutet dies:

$$\begin{aligned} a^T\mathbf{M}^T\mathbf{M}a = \sum_{k=1}^{m} & (a_1^2 + a_2^2 + a_3^2 + a_4^2(y_k^2+z_k^2) + a_5^2(x_k^2+y_k^2) + a_6^2(x_k^2+z_k^2) \\ + & 2x_k(a_2a_5 + a_3a_6) - 2y_k(a_3a_4 + a_1a_5) + 2z_k(a_2a_4 + a_1a_6) \\ - & 2a_4a_6x_ky_k + 2a_4a_5x_kz_k + 2a_5a_6y_kz_k). \end{aligned}$$

Dies lässt sich umformen zu:

$$a^T\mathbf{M}^T\mathbf{M}a = \sum_{k=1}^{m}(a_1 - a_5y_k - a_6z_k)^2 + (a_2 + a_4z_k + a_5x_k)^2 + (a_3 - a_4y_k + a_6x_k)^2 \geq 0. \quad (5.4)$$

Der Fall $\mathbf{M}^T\mathbf{M} = 0$ tritt nur auf, wenn alle Summanden gleich Null sind, dass heißt, wenn für alle u_k gilt:

$$\begin{aligned} a_5y_k + a_6z_k &= a_1 \\ a_4z_k + a_5x_k &= -a_2 \\ -a_4y_k + a_6x_k &= -a_3. \end{aligned}$$

Konstruieren wir mit $B = (a_1, -a_2, -a_3)^T$ und $\chi = (x, y, z)$

$$A = \begin{pmatrix} 0 & a_5 & a_6 \\ a_5 & 0 & a_4 \\ a_6 & -a_4 & 0 \end{pmatrix}, \quad (5.5)$$

die, zu dem Gleichungssystem äquivalente, folgende Gleichung in Matrixform:

$$A\chi = B. \quad (5.6)$$

A ist nicht invertierbar, hat also für alle $a \in \mathbb{R}^6$ den maximalen Rang 2. Der minimale Rang von A, bei dem die Gleichung erfüllt sein kann, ist jedoch nicht 0, da aus $a_4 = a_5 = a_6 = 0$ folgt, dass auch $a_1 = a_2 = a_3 = 0$, also $a = 0$ wäre. Es muss also zumindest einer der Werte a_4, a_5 und a_6 ungleich Null sein. Angenommen es gilt: $a_4 \neq 0$, dann lässt sich A ohne weiteres durch elementare Zeilen- und Spaltenoperationen in A' umwandeln.

$$A' = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & a_4 \\ 0 & -a_4 & 0 \end{pmatrix}.$$

Ähnliche Matrizen ergeben sich für $a_5 \neq 0$ und $a_6 \neq 0$. Dies bedeutet, dass, wenn die Gleichung (5.6) lösbar ist, gilt:

$$\mathrm{Rg}(A) = 2 \Leftrightarrow a \neq 0.$$

Der Rang von A ist also 2 und die Lösungsmenge der Gleichung hat die Dimension 1. Der Gleichheitsfall des Audruckes (5.4) tritt nur dann auf, wenn alle u_k in der Lösungsmenge von (5.6) liegen. Wenn also Punktpaare zu solchen Punkten gehören, die auf einer Geraden liegen, ist die Varianz C_D nicht positiv definit. Um sicherzustellen, dass C_D positiv definit ist, reicht es aus, 3 Punktpaare zu haben, die nicht kollinear sind. Dies sollte in der Regel bei sich überschneidenden $3D$-Scans ohnehin der Fall sein. □

Quaternionen

Sollten wir unsere Posen mithilfe von Quaternionen definiert haben, lässt sich ebenfalls eine äquivalente Formulierung finden. Haben wir also die Posen $V_a = (x_a, y_a, z_a, p_a, q_a, r_a, s_a)^T$ und $V_b = (x_b, y_b, z_b, p_b, q_b, r_b, s_b)^T$, deren Schätzungen $\bar{V}_a = (\bar{x}_a, \bar{y}_a, \bar{z}_a, \bar{p}_a, \bar{q}_a, \bar{r}_a, \bar{s}_a)^T$ und $\bar{V}_b = (\bar{x}_b, \bar{y}_b, \bar{z}_b, \bar{p}_b, \bar{q}_b, \bar{r}_b, \bar{s}_b)^T$ und zusätzlich m Punktpaare (u_k^a, a_k^b) mit $u_k^a = (x_k^a, y_k^a, z_k^a)$ und $u_k^b = (x_k^b, y_k^b, z_k^b)$, die zu den m Punkten $u_k = (x_k, y_k, z_k)$ korrespondieren, dann gilt:

$$
\begin{aligned}
u_k &\approx V_a \oplus u_k^a \\
x_k &\approx x_a + (p_a^2 + q_a^2 - r_a^2 - s_a^2)x_k^a + 2(q_a^2 r_a^2 - p_a^2 s_a^2)y_k^a + 2(p_a^2 r_a^2 + q_a^2 s_a^2)z_k^a && =: f_1(V_a) \\
y_k &\approx y_a + 2(q_a^2 r_a^2 + p_a^2 s_a^2)x_k^a + (p_a^2 - q_a^2 + r_a^2 - s_a^2)y_k^a + 2(r_a^2 s_a^2 - p_a^2 q_a^2)z_k^a && =: f_2(V_a) \\
z_k &\approx z_a + 2(q_a^2 s_a^2 - p_a^2 r_a^2)x_k^a + 2(p_a^2 q_a^2 + r_a^2 s_a^2)y_k^a + (p_a^2 - q_a^2 - r_a^2 + s_a^2)z_k^a && =: f_3(V_a)
\end{aligned}
$$

Die relevanten partiellen Ableitungen sind nach der Beseitigung von x_k^a,y_k^a und z_k^a nun:

$$
\begin{aligned}
\frac{\partial f_1}{\partial p_a}(\bar{V}_a) &= 2\bar{p}_a x_k + 2\bar{s}_a y_k - 2\bar{r}_a z_k - 2(\bar{p}_a \bar{x}_a + \bar{s}_a \bar{y}_a - \bar{r}_a \bar{z}_a) \\
\frac{\partial f_2}{\partial p_a}(\bar{V}_a) &= -2\bar{s}_a x_k + 2\bar{p}_a y_k - 2\bar{q}_a z_k - 2(\bar{s}_a \bar{x}_a + \bar{p}_a \bar{y}_a - \bar{q}_a \bar{z}_a) \\
\frac{\partial f_3}{\partial p_a}(\bar{V}_a) &= 2\bar{r}_a x_k - 2\bar{q}_a y_k + 2\bar{p}_a z_k - 2(\bar{r}_a \bar{x}_a - \bar{q}_a \bar{y}_a + \bar{p}_a \bar{z}_a) \\
\\
\frac{\partial f_1}{\partial q_a}(\bar{V}_a) &= 2\bar{q}_a x_k + 2\bar{r}_a y_k + 2\bar{s}_a z_k - 2(\bar{q}_a \bar{x}_a + \bar{r}_a \bar{y}_a + \bar{s}_a \bar{z}_a) \\
\frac{\partial f_2}{\partial q_a}(\bar{V}_a) &= -2\bar{r}_a x_k + 2\bar{q}_a y_k - 2\bar{p}_a z_k - 2(-\bar{r}_a \bar{x}_a + \bar{q}_a \bar{y}_a - \bar{p}_a \bar{z}_a) \\
\frac{\partial f_3}{\partial q_a}(\bar{V}_a) &= -2\bar{s}_a x_k + 2\bar{p}_a y_k + 2\bar{q}_a z_k - 2(-\bar{s}_a \bar{x}_a + \bar{p}_a \bar{y}_a + \bar{q}_a \bar{z}_a) \\
\\
\frac{\partial f_1}{\partial r_a}(\bar{V}_a) &= 2\bar{r}_a x_k - 2\bar{q}_a y_k + 2\bar{p}_a z_k - 2(\bar{r}_a \bar{x}_a - \bar{q}_a \bar{y}_a + \bar{p}_a \bar{z}_a) \\
\frac{\partial f_2}{\partial r_a}(\bar{V}_a) &= 2\bar{q}_a x_k + 2\bar{r}_a y_k + 2\bar{s}_a z_k - 2(\bar{q}_a \bar{x}_a + \bar{r}_a \bar{y}_a + \bar{s}_a \bar{z}_a) \\
\frac{\partial f_3}{\partial r_a}(\bar{V}_a) &= -2\bar{p}_a x_k - 2\bar{s}_a y_k + 2\bar{r}_a z_k - 2(-\bar{p}_a \bar{x}_a - \bar{s}_a \bar{y}_a + \bar{r}_a \bar{z}_a) \\
\\
\frac{\partial f_1}{\partial s_a}(\bar{V}_a) &= 2\bar{s}_a x_k - 2\bar{p}_a y_k - 2\bar{q}_a z_k - 2(\bar{s}_a \bar{x}_a - \bar{p}_a \bar{y}_a - \bar{q}_a \bar{z}_a) \\
\frac{\partial f_2}{\partial s_a}(\bar{V}_a) &= 2\bar{p}_a x_k + 2\bar{s}_a y_k - 2\bar{r}_a z_k - 2(\bar{p}_a \bar{x}_a + \bar{s}_a \bar{y}_a - \bar{r}_a \bar{z}_a) \\
\frac{\partial f_3}{\partial s_a}(\bar{V}_a) &= 2\bar{q}_a x_k + 2\bar{r}_a y_k + 2\bar{s}_a z_k - 2(\bar{q}_a \bar{x}_a + \bar{r}_a \bar{y}_a + \bar{r}_a \bar{z}_a).
\end{aligned}
$$

Nach demselben Prinzip wie bei den Euler-Winkeln lassen sich auch hier die Matrixdekompositionen $M_k H_a$ und $M_k H_b$ angeben mit:

$$
\begin{aligned}
M_k &= \begin{pmatrix} 1 & 0 & 0 & x_k & 0 & -z_k & y_k \\ 0 & 1 & 0 & y_k & z_k & 0 & -x_k \\ 0 & 0 & 1 & z_k & -y_k & x_k & 0 \end{pmatrix} \\
H_a &= \begin{pmatrix} I_3 & T_a \\ 0 & U_a \end{pmatrix} \\
H_b &= \begin{pmatrix} I_3 & T_b \\ 0 & U_b \end{pmatrix} \\
T_a^T &= \begin{pmatrix} (\bar{r}_a\bar{z}_a - \bar{p}_a\bar{x}_a - \bar{s}_a\bar{y}_a) & 2(\bar{q}_a\bar{z}_a - \bar{s}_a\bar{x}_a - \bar{p}_a\bar{y}_a) & 2(\bar{q}_a\bar{y}_a - \bar{r}_a\bar{x}_a - \bar{p}_a\bar{z}_a) \\ -2(\bar{q}_a\bar{x}_a + \bar{r}_a\bar{y}_a + \bar{s}_a\bar{z}_a) & (\bar{r}_a\bar{x}_a - \bar{q}_a\bar{y}_a + \bar{p}_a\bar{z}_a) & 2(\bar{s}_a\bar{x}_a - \bar{p}_a\bar{y}_a - \bar{q}_a\bar{z}_a) \\ 2(\bar{q}_a\bar{y}_a - \bar{r}_a\bar{x}_a - \bar{p}_a\bar{z}_a) & -2(\bar{q}_a\bar{x}_a + \bar{r}_a\bar{y}_a + \bar{s}_a\bar{z}_a) & 2(\bar{p}_a\bar{x}_a + \bar{s}_a\bar{y}_a - \bar{r}_a\bar{z}_a) \\ 2(\bar{p}_a\bar{y}_a - \bar{s}_a\bar{x}_a + \bar{q}_a\bar{z}_a) & 2(\bar{r}_a\bar{z}_a - \bar{p}_a\bar{x}_a - \bar{s}_a\bar{y}_a) & -2(\bar{q}_a\bar{x}_a + \bar{r}_a\bar{y}_a + \bar{r}_a\bar{z}_a) \end{pmatrix} \\
U_a &= \begin{pmatrix} 2\bar{p}_a & 2\bar{q}_a & 2\bar{r}_a & 2\bar{s}_a \\ 2\bar{q}_a & -2\bar{p}_a & 2\bar{s}_a & -2\bar{r}_a \\ 2\bar{r}_a & -2\bar{s}_a & -2\bar{p}_a & 2\bar{q}_a \\ 2\bar{s}_a & 2\bar{r}_a & -2\bar{q}_a & -2\bar{p}_a \end{pmatrix} \\
T_b^T &= \begin{pmatrix} 2(\bar{r}_b\bar{z}_b - \bar{p}_b\bar{x}_b - \bar{s}_b\bar{y}_b) & 2(\bar{q}_b\bar{z}_b - \bar{s}_b\bar{x}_b - \bar{p}_b\bar{y}_b) & 2(\bar{q}_b\bar{y}_b - \bar{r}_b\bar{x}_b - \bar{p}_b\bar{z}_b) \\ -2(\bar{q}_b\bar{x}_b + \bar{r}_b\bar{y}_b + \bar{s}_b\bar{z}_b) & 2(\bar{r}_b\bar{x}_b - \bar{q}_b\bar{y}_b + \bar{p}_b\bar{z}_b) & 2(\bar{s}_b\bar{x}_b - \bar{p}_b\bar{y}_b - \bar{q}_b\bar{z}_b) \\ 2(\bar{q}_b\bar{y}_b - \bar{r}_b\bar{x}_b - \bar{p}_b\bar{z}_b) & -2(\bar{q}_b\bar{x}_b + \bar{r}_b\bar{y}_b + \bar{s}_b\bar{z}_b) & 2(\bar{p}_b\bar{x}_b + \bar{s}_b\bar{y}_b - \bar{r}_b\bar{z}_b) \\ 2(\bar{p}_b\bar{y}_b - \bar{s}_b\bar{x}_b + \bar{q}_b\bar{z}_b) & 2(\bar{r}_b\bar{z}_b - \bar{p}_b\bar{x}_b - \bar{s}_b\bar{y}_b) & -2(\bar{q}_b\bar{x}_b + \bar{r}_b\bar{y}_b + \bar{r}_b\bar{z}_b) \end{pmatrix} \\
U_b &= \begin{pmatrix} 2\bar{p}_b & 2\bar{q}_b & 2\bar{r}_b & 2\bar{s}_b \\ 2\bar{q}_b & -2\bar{p}_b & 2\bar{s}_b & -2\bar{r}_b \\ 2\bar{r}_b & -2\bar{s}_b & -2\bar{p}_b & 2\bar{q}_b \\ 2\bar{s}_b & 2\bar{r}_b & -2\bar{q}_b & -2\bar{p}_b \end{pmatrix}.
\end{aligned}
$$

Um die Posedifferenzschätzung $\bar{D}$ und die Kovarianz C_D auszurechnen, wird die $3m \times 7$ Matrix $\mathbf{M}$ und der $3m$ Vektor $\mathbf{Z}$ benötigt. Diese haben die Form wie folgt:

$$
\mathbf{M} = \begin{pmatrix} 1 & 0 & 0 & x_1 & 0 & -z_1 & y_1 \\ 0 & 1 & 0 & y_1 & z_1 & 0 & -x_1 \\ 0 & 0 & 1 & z_1 & -y_1 & x_1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ 1 & 0 & 0 & x_m & 0 & -z_m & y_m \\ 0 & 1 & 0 & y_m & z_m & 0 & -x_m \\ 0 & 0 & 1 & z_m & -y_m & x_m & 0 \end{pmatrix}, \qquad \mathbf{Z} = \begin{pmatrix} \bar{V}_a \oplus u_1^a - \bar{V}_b \oplus u_1^b \\ \bar{V}_a \oplus u_2^a - \bar{V}_b \oplus u_2^b \\ \vdots \\ \bar{V}_a \oplus u_m^a - \bar{V}_b \oplus u_m^b \end{pmatrix}.
$$

Beweis: Um zu zeigen, dass die Kovarianz $C_D = s^2(\mathbf{M}^T\mathbf{M})^{-1}$ positiv definit ist, betrachten wir erneut $\mathbf{M}^T\mathbf{M}$.

$$\mathbf{M}^T\mathbf{M} = \sum_{k=1}^{m} \begin{pmatrix} 1 & 0 & 0 & x_k & 0 & -z_k & y_k \\ 0 & 1 & 0 & y_k & z_k & 0 & -x_k \\ 0 & 0 & 1 & z_k & -y_k & x_k & 0 \\ x_k & y_k & z_k & x_k^2+y_k^2+z_k^2 & 0 & 0 & 0 \\ 0 & z_k & -y_k & 0 & y_k^2+z_k^2 & -x_k y_k & -x_k z_k \\ -z_k & 0 & x_k & 0 & -x_k y_k & x_k^2+z_k^2 & -y_k z_k \\ y_k & -x_k & 0 & 0 & -x_k z_k & -y_k z_k & x_k^2+y_k^2 \end{pmatrix}.$$

Setzen wir dies in die Definition der positiven Definitheit ein, gilt mit dem reellwertigen Vektor $a = (a_1, a_2, a_3, a_4, a_5, a_6, a_7)^T \neq 0$ wie folgt:

$$\begin{aligned} a^T\mathbf{M}^T\mathbf{M}a = \sum_{k=1}^{m} & \quad (a_1 + a_4x_k + a_7y_k - a_6z_k)^2 \\ + & \quad (a_2 - a_7x_k + a_4y_k + a_5z_k)^2 \\ + & \quad (a_3 + a_6x_k - a_5y_k - a_4z_k)^2. \end{aligned}$$

Dies ist nun genau dann gleich Null, wenn es ein $(A, B) \neq 0$ gibt, mit dem für alle u_k gilt:

$$\begin{aligned} Au_k &= B \\ \begin{pmatrix} a_4 & a_7 & -a_6 \\ -a_7 & a_4 & a_5 \\ a_6 & -a_5 & a_4 \end{pmatrix} \cdot \begin{pmatrix} x_k \\ y_k \\ z_k \end{pmatrix} &= \begin{pmatrix} -a_1 \\ -a_2 \\ -a_3 \end{pmatrix}. \end{aligned}$$

Da aus $A = 0$ folgt, dass auch $B = 0$ gilt, sind nicht alle a_4, a_5, a_6 und a_7 gleich Null. Die Matrix A erhält genau dann den Rang Null, wenn die Determinante Null ist.

$$\begin{aligned} \mathrm{Det}(A) &= 0 \\ &= a_4^3 + a_4a_5^2 + a_4a_6^2 + a_4a_7^2 \\ &= a_4(a_4^2 + a_5^2 + a_6^2 + a_7^2). \end{aligned}$$

Dies gilt mit $A \neq 0$ nur, wenn $a_4 = 0$, also wenn A die folgende Form hat:

$$A = \begin{pmatrix} 0 & a_7 & -a_6 \\ -a_7 & 0 & a_5 \\ a_6 & -a_5 & 0 \end{pmatrix}.$$

Nun hat A die selbe Form wie in der Gleichung (5.5). Somit hat A mit $a_4 = 0$ den minimalen Rang 2, und, da die Matrix mit $a_4 \neq 0$ immer den vollen Rang hat, gilt auch in diesem Fall, dass die korrespondierenden Punktpaare nicht kollinear sein dürfen, damit C_D positiv definit ist. □

5.2 Posedifferenzen aus Odometrie

In diesem Abschnitt werden wir Messungen aus der Odometrie in die Form der zuvor veränderten Posedifferenzgleichung bringen. Dafür werden wir die unveränderte Posedifferenzgleichung für Odometriemessungen einer Taylor-Expansion unterziehen und den so entstehenden Term in eine Mahalanobisdistanz mit der neuen Posedifferenz umformen. Allerdings lässt sich das in dieser Weise aufgrund der komplexen Matrixdekomposition $M_k H_a$ im Falle einer $6D$-Pose nur mit Einschränkungen durchführen.

Nehmen wir an, zwischen den beiden Posen V_a und V_b existiere eine schwache Verbindung, dass heißt, es gibt Odometrieschätzungen $\bar{V}_a$ und $\bar{V}_b$ der Posen V_a und V_b. Auch hier haben wir zunächst die nicht-lineare Posedifferenzgleichung D', und deren Schätzung $\bar{D}'$ gegeben:

$$D' = V_a \ominus V_b.$$

Mit $\Delta D' = \bar{D}' - D'$ und der Varianz C' des Fehlers stellen wir die folgende Mahalanobisdistanzgeichung auf.

$$\begin{aligned} W_{ab} &= (\Delta D')^T C'^{-1} (\Delta D') \\ &= (\bar{D}' - D')^T C'^{-1} (\bar{D}' - D'). \end{aligned}$$

Ziel ist es, diese Gleichung in eine Form zu bringen, die der, der Posedifferenzen aus Abschnitt 5.1 gleich kommt. Wir müssen also eine Möglichkeit finden, eine Mahalanobisdistanz mit Hilfe der linearisierten Pose $D = \bar{H}_a \Delta V_a - \bar{H}_b \Delta V_b$ zu formulieren.

Seien $\Delta V_a = \bar{V}_a - V_a$ und $\Delta V_b = \bar{V}_b - V_b$ die Messfehler in den Posen, dann führen wir auf D' eine Tayler-Entwicklung erster Ordnung nach V_a und V_b durch.

$$\begin{aligned} \Delta D' &= \bar{D}' - D' \\ &= \bar{D}' - (V_a \ominus V_b) \\ &\approx \bar{D}' - (\bar{V}_a \ominus \bar{V}_b) + \nabla_{\bar{V}_a}(\bar{V}_a \ominus \bar{V}_b)\Delta V_a + \nabla_{\bar{V}_b}(\bar{V}_a \ominus \bar{V}_b)\Delta V_b. \end{aligned}$$

Wir führen folgende Bezeichnungen ein,

$$\begin{aligned} K_a^{-1} &:= \nabla_{\bar{V}_a}(\bar{V}_a \ominus \bar{V}_b) \\ K_b^{-1} &:= -\nabla_{\bar{V}_b}(\bar{V}_a \ominus \bar{V}_b), \end{aligned}$$

und postulieren mit den bereits bekannten H_a und H_b eine Bedingung wie folgt.

$$K_a K_b^{-1} = H_a^{-1} H_b.$$

Dies ergibt eingesetzt in $\Delta D'$:

$$\begin{aligned} \Delta D' &\approx \bar{D}' - (\bar{V}_a \ominus \bar{V}_b) + K_a^{-1}\Delta V_a - K_b^{-1}\Delta V_b \\ &= \bar{D}' - (\bar{V}_a \ominus \bar{V}_b) + K_a^{-1}(\Delta V_a - H_a^{-1} H_b \Delta V_b). \end{aligned}$$

Es liegt nun nahe, einen neuen Messfehler ΔD zu definieren, so dass gilt:

$$\begin{aligned}\Delta D &= -H_a K_a \Delta D' \\ &= H_a K_a((\bar{V}_a \ominus \bar{V}_b) - \bar{D}') + (H_a \Delta V_a - H_b \Delta V_b) \\ &= \bar{D} - D.\end{aligned}$$

Wobei $D = H_a \Delta V_a - H_b \Delta V_b$ die linearisierte Posedifferenz und $\bar{D} = H_a K_a((\bar{V}_a \ominus \bar{V}_b)$ die Schätzung der Posedifferenz ist. Die Kovarianz C von $\bar{D}$ ergibt sich dementsprechend aus C' mit:

$$C = H_a K_a C^{-1} K_a^T H_a^T.$$

Mit diesen Werten wird die Mahalanobisdistanz W_{ab} jetzt umgeformt zu:

$$W_{ab} \approx (\bar{D} - D)^T C^{-1} (\bar{D} - D).$$

5.2.1 3D Pose

Wir werden nun den Spezialfall von 3D-Posen behandeln. Nehmen wir an der Roboter fährt von der Pose $V_b = (x_b, y_b, \theta_b)^T$ zu der Pose $V_a = (x_a, y_a, \theta_a)^T$. Uns sind von der Odometrie die Schätzungen $\bar{V}_a = (\bar{x}_a, \bar{y}_a, \bar{\theta}_a)^T$ und $\bar{V}_b = (\bar{x}_b, \bar{y}_b, \bar{\theta}_b)^T$ der Posen mit den Fehlern ΔV_a und ΔV_b gegeben, so dass gilt $\Delta V_a = \bar{V}_a - V_a$ und $\Delta V_b = \bar{V}_b - V_b$.

Um die Taylorexpansion von D' nachzuvollziehen, betrachten wir die inverse Translations-Operation $V_a \ominus V_b$:

$$\begin{aligned}x &= (x_a - x_b)\cos\theta_b + (y_a - y_b)\sin\theta_b \\ y &= -(x_a - x_b)\sin\theta_b + (y_a - y_b)\cos\theta_b \\ \theta &= \theta_a - \theta_b.\end{aligned}$$

Wir benötigen also die Ableitungen K_a^{-1} und K_b^{-1} von D' nach V_a und V_b.

$$\nabla_{\bar{V}_a}(\bar{V}_a \ominus \bar{V}_b) = K_a^{-1} = \begin{pmatrix} \cos\bar{\theta}_b & \sin\bar{\theta}_b & 0 \\ -\sin\bar{\theta}_b & \cos\bar{\theta}_b & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\nabla_{\bar{V}_b}(\bar{V}_a \ominus \bar{V}_b) = -K_b^{-1} = \begin{pmatrix} -\cos\bar{\theta}_b & -\sin\bar{\theta}_b & -(\bar{x}_a - \bar{x}_b)\sin\bar{\theta}_b + (\bar{y}_a - \bar{y}_b)\cos\bar{\theta}_b \\ \sin\bar{\theta}_b & -\cos\bar{\theta}_b & -(\bar{x}_a - \bar{x}_b)\cos\bar{\theta}_b - (\bar{y}_a - \bar{y}_b)\sin\bar{\theta}_b \\ 0 & 0 & -1 \end{pmatrix}.$$

Besonders zu beachten ist $K_a K_b^{-1} = H_a^{-1} H_b$ mit:

$$K_a K_b^{-1} = H_a^{-1} H_b = H_{ab} = \begin{pmatrix} 1 & 0 & \bar{y}_b - \bar{y}_a \\ 0 & 1 & \bar{x}_a - \bar{x}_b \\ 0 & 0 & 1 \end{pmatrix}.$$

Somit ist also im 2D-Fall die Bedingung $K_a K_b^{-1} = H_a^{-1} H_b$ erfüllt und $\Delta D'$ lässt sich approximieren durch:

$$\begin{aligned} \Delta D' &= \bar{D}' - (V_a \ominus V_b) \\ &\approx \bar{D}' - (\bar{V}_a \ominus \bar{V}_b) + K_a^{-1}(\Delta V_a - H_{ab}\Delta V_b). \end{aligned}$$

Ebenso können wir den neuen Messfehler $\Delta D = -H_a K_a \Delta D'$ definieren und die Mahalanobisdistanz umschreiben zu:

$$W_{ab} \approx (\bar{D} - D)^T C^{-1} (\bar{D} - D).$$

Die Kovarianz C von $\bar{D}$ kann folgendermaßen aus C' errechnet werden.

$$C = H_a K_a C' K_a^T H_a^T.$$

Wenden wir uns nun der Kovarianz C' der Posedifferenz D' zu. Wir nehmen an, wir erhalten bei einem Positionswechsel des Roboters von der Odometrie die Werte α,β und L. Die Roboterplattform dreht sich also zunächst um einen Winkel α, bewegt sich anschließend um eine Distanz L und dreht sich am Ende der Strecke erneut um einen Winkel β. Die Varianz dieser Messwerte sei durch σ_α,σ_L und σ_β gegeben. Die Posedifferenz $D' = (x, y, \theta)^T$ lässt sich dann errechnen aus:

$$\begin{aligned} x &= L\cos\alpha \\ y &= L\sin\alpha \\ \theta &= \alpha + \beta. \end{aligned}$$

Daraus folgt die Kovarianz C' von D' als:

$$C' = J \begin{pmatrix} \sigma_\alpha^2 & 0 & 0 \\ 0 & \sigma_L^2 & 0 \\ 0 & 0 & \sigma_\beta^2 \end{pmatrix} J^T,$$

mit J, dem Grad von D':

$$J = \begin{pmatrix} -L\sin\alpha & \cos\alpha & 0 \\ L\cos\alpha & \sin\alpha & 0 \\ 1 & 0 & 1 \end{pmatrix}.$$

Für die Varianz C von D bedeutet dies also:

$$C = H_a K_a J \begin{pmatrix} \sigma_\alpha^2 & 0 & 0 \\ 0 & \sigma_L^2 & 0 \\ 0 & 0 & \sigma_\beta^2 \end{pmatrix} J^T K_a^T H_a^T.$$

Da H_a,K_a und J mit $L \neq 0$ trivialerweise invertierbar sind, folgt, dass C positiv definit ist.

5.2.2 6D Pose

Um Odometriemessungen auch für $6D$ Posen nutzbar zu machen, werden wir im Zuge dieses Kapitels annehmen, dass die Odometrie nichts über die Höhe z des Roboters und die Orientierungen θ_x und θ_y aussagt, diese also zwischen 2 Posen, soweit es die Odometrie betrifft, unverändert bleiben. Wir haben also eine Messung $\bar{D}'$ der Posedifferenz D' zwischen $V_a = (x_a, y_a, z_a, \theta_{x_a}, \theta_{y_b}, \theta_{z_b})^T$ und $V_b = (x_b, y_b, z_b, \theta_{x_b}, \theta_{y_b}, \theta_{z_b})^T$ gegeben, mit der Posedifferenzgleichung:

$$D' = V_a \ominus V_b.$$

Seien $\bar{V}_a = (\bar{x}_a, \bar{y}_a, \bar{z}_a, \bar{\theta}_{x_a}, \bar{\theta}_{y_b}, \bar{\theta}_{z_b})^T$ und $\bar{V}_b = (\bar{x}_b, \bar{y}_b, \bar{z}_a, \bar{\theta}_{x_b}, \bar{\theta}_{y_b}, \bar{\theta}_{z_b})^T$ die Schätzungen der Posen V_a und V_b mit den Fehlern ΔV_a und ΔV_b, so dass gilt $\Delta V_a = \bar{V}_a - V_a$ und $\Delta V_b = \bar{V}_b - V_b$.

Um auf die Posedifferenz $D' = (x, y, z, \theta_x, \theta_y, \theta_z)^T$ die Taylor-Expansion erster Ordnung anzuwenden, betrachten wir zunächst $V_a \ominus V_b$.

$$\begin{aligned}
x &= (x_a - x_b)\cos\theta_{y_b}\cos\theta_{z_b} + \cos\theta_{x_b}((z_a - z_b)\cos\theta_{z_b}\sin\theta_{y_b} + (y_a - y_b)\sin\theta_{z_b}) \\
&\quad + \sin\theta_{x_b}((y_a - y_b)\cos\theta_{z_b}\sin\theta_{y_b} + (-z_a + z_b)\sin\theta_{z_b}) && =: f_1 \\
y &= -(z_a - z_b)\cos\theta_{z_b}\sin\theta_{x_b} + ((-xa + x_b)\cos\theta_{y_b} + (-y_a + y_b)\sin\theta_{x_b}\sin\theta_{y_b})\sin\theta_{z_b} \\
&\quad + \cos\theta_{x_b}((y_a - y_b)\cos\theta_{z_b} + (-z_a + z_b)\sin\theta_{y_b}\sin\theta_{z_b}) && =: f_2 \\
z &= (z_a - z_b)\cos\theta_{x_b}\cos\theta_{y_b} + (y_a - y_b)\cos\theta_{y_b}\sin\theta_{x_b} + (-xa + x_b)\sin\theta_{y_b} && =: f_3 \\
\theta_x &= \theta_{x_a} - \theta_{x_b} \\
\theta_y &= \theta_{y_a} - \theta_{y_b} \\
\theta_z &= \theta_{z_a} - \theta_{z_b}.
\end{aligned}$$

Die Ableitung K_a^{-1} ist gegeben durch:

$$\nabla_{\bar{V}_a}(\bar{V}_a \ominus \bar{V}_b) = K_a^{-1} = \begin{pmatrix}
\frac{\partial f_1}{\partial x_a}(\bar{V}_a) & \frac{\partial f_1}{\partial y_a}(\bar{V}_a) & \frac{\partial f_1}{\partial z_a}(\bar{V}_a) & 0 & 0 & 0 \\
\frac{\partial f_2}{\partial x_a}(\bar{V}_a) & \frac{\partial f_2}{\partial y_a}(\bar{V}_a) & \frac{\partial f_2}{\partial z_a}(\bar{V}_a) & 0 & 0 & 0 \\
\frac{\partial f_3}{\partial x_a}(\bar{V}_a) & \frac{\partial f_3}{\partial y_a}(\bar{V}_a) & \frac{\partial f_3}{\partial z_a}(\bar{V}_a) & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}.$$

Die Einträge der Matrix sind gegeben durch:

$$\begin{aligned}
\frac{\partial f_1}{\partial x_a}(\bar{V}_a) &= \cos\bar{\theta}_{y_b}\cos\bar{\theta}_{z_b} \\
\frac{\partial f_2}{\partial x_a}(\bar{V}_a) &= -\cos\bar{\theta}_{y_b}\sin\bar{\theta}_{z_b} \\
\frac{\partial f_3}{\partial x_a}(\bar{V}_a) &= -\sin\bar{\theta}_{y_b} \\
\\
\frac{\partial f_1}{\partial y_a}(\bar{V}_a) &= \cos\bar{\theta}_{z_b}\sin\bar{\theta}_{x_b}\sin\bar{\theta}_{y_b} + \cos\bar{\theta}_{x_b}\sin\bar{\theta}_{z_b} \\
\frac{\partial f_2}{\partial y_a}(\bar{V}_a) &= \cos\bar{\theta}_{x_b}\cos\bar{\theta}_{z_b} - \sin\bar{\theta}_{x_b}\sin\bar{\theta}_{y_b}\sin\bar{\theta}_{z_b} \\
\frac{\partial f_3}{\partial y_a}(\bar{V}_a) &= \cos\bar{\theta}_{y_b}\sin\bar{\theta}_{x_b} \\
\\
\frac{\partial f_1}{\partial z_a}(\bar{V}_a) &= \cos\bar{\theta}_{x_b}\cos\bar{\theta}_{z_b}\sin\bar{\theta}_{y_b} - \sin\bar{\theta}_{x_b}\sin\bar{\theta}_{z_b} \\
\frac{\partial f_2}{\partial z_a}(\bar{V}_a) &= -\cos\bar{\theta}_{z_b}\sin\bar{\theta}_{x_b} - \cos\bar{\theta}_{x_b}\sin\bar{\theta}_{y_b}\sin\bar{\theta}_{z_b} \\
\frac{\partial f_3}{\partial z_a}(\bar{V}_a) &= \cos\bar{\theta}_{x_b}\cos\bar{\theta}_{y_b}.
\end{aligned}$$

Die Ableitung von $(\bar{V}_a \ominus \bar{V}_b)$ nach $\bar{V}_b$ ist dann:

$$\nabla_{\bar{V}_a}(\bar{V}_a \ominus \bar{V}_b) = -K_b^{-1} = \begin{pmatrix}
-\frac{\partial f_1}{\partial x_a}(\bar{V}_a) & -\frac{\partial f_1}{\partial y_a}(\bar{V}_a) & -\frac{\partial f_1}{\partial z_a}(\bar{V}_a) & \frac{\partial f_1}{\partial \theta_{x_b}}(\bar{V}_b) & \frac{\partial f_1}{\partial \theta_{y_b}}(\bar{V}_b) & \frac{\partial f_1}{\partial \theta_{z_b}}(\bar{V}_b) \\
-\frac{\partial f_2}{\partial x_a}(\bar{V}_a) & -\frac{\partial f_2}{\partial y_a}(\bar{V}_a) & -\frac{\partial f_2}{\partial z_a}(\bar{V}_a) & \frac{\partial f_2}{\partial \theta_{x_b}}(\bar{V}_b) & \frac{\partial f_2}{\partial \theta_{y_b}}(\bar{V}_b) & \frac{\partial f_2}{\partial \theta_{z_b}}(\bar{V}_b) \\
-\frac{\partial f_3}{\partial x_a}(\bar{V}_a) & -\frac{\partial f_3}{\partial y_a}(\bar{V}_a) & -\frac{\partial f_3}{\partial z_a}(\bar{V}_a) & \frac{\partial f_3}{\partial \theta_{x_b}}(\bar{V}_b) & \frac{\partial f_3}{\partial \theta_{y_b}}(\bar{V}_b) & \frac{\partial f_3}{\partial \theta_{z_b}}(\bar{V}_b) \\
0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & -1
\end{pmatrix},$$

Mit den folgenden partiellen Ableitungen:

$$\begin{aligned}
\frac{\partial f_1}{\partial \theta_{x_b}}(\bar{V}_a) &= \sin\bar{\theta}_{x_b}((-z_a + z_b)\cos\bar{\theta}_{z_b}\sin\bar{\theta}_{y_b} + (-y_a + y_b)\sin\bar{\theta}_{z_b}) \\
&\quad + \cos\bar{\theta}_{x_b}((\bar{y}_a - \bar{y}_b)\cos\bar{\theta}_{z_b}\sin\bar{\theta}_{y_b} + (-\bar{z}_a + \bar{z}_b)\sin\bar{\theta}_{z_b}) \\
\frac{\partial f_2}{\partial \theta_{x_b}}(\bar{V}_a) &= \cos\theta_{x_b}((-\bar{z}_a + \bar{z}_b)\cos\theta_{z_b} + (-\bar{y}_a + \bar{y}_b)\sin\theta_{y_b}\sin\theta_{z_b}) \\
&\quad + \sin\bar{\theta}_{x_b}((-\bar{y}_a + y_b)\cos\bar{\theta}_{z_b} + (\bar{z}_a - \bar{z}_b)\sin\bar{\theta}_{y_b}\sin\bar{\theta}_{z_b}) \\
\frac{\partial f_3}{\partial \theta_{x_b}}(\bar{V}_a) &= \cos\bar{\theta}_{y_b}((\bar{y}_a - \bar{y}_b)\cos\theta_{x_b} + (-\bar{z}_a + \bar{z}_b)\sin\bar{\theta}_{x_b})
\end{aligned}$$

$$
\begin{aligned}
\frac{\partial f_1}{\partial \theta_{y_b}}(\bar{V}_a) &= \cos\bar{\theta}_{z_b}((\bar{z}_a-\bar{z}_b)\cos\bar{\theta}_{x_b}\cos\bar{\theta}_{y_b}+(\bar{y}_a-\bar{y}_b)\cos\bar{\theta}_{y_b}\sin\bar{\theta}_{x_b}+(-\bar{x}_a+\bar{x}_b)\sin\bar{\theta}_{y_b})\\
\frac{\partial f_2}{\partial \theta_{y_b}}(\bar{V}_a) &= ((-\bar{z}_a+\bar{z}_b)\cos\bar{\theta}_{x_b}\cos\bar{\theta}_{y_b}+(-\bar{y}_a+\bar{y}_b)\cos\bar{\theta}_{y_b}\sin\bar{\theta}_{x_b}+(\bar{x}_a-\bar{x}_b)\sin\bar{\theta}_{y_b})\sin\bar{\theta}_{z_b}\\
\frac{\partial f_3}{\partial \theta_{y_b}}(\bar{V}_a) &= (-\bar{x}_a+\bar{x}_b)\cos\bar{\theta}_{y_b}+((-\bar{z}_a+\bar{z}_b)\cos\bar{\theta}_{x_b}+(-\bar{y}_a+\bar{y}_b)\sin\bar{\theta}_{x_b})\sin\bar{\theta}_{y_b}\\
\frac{\partial f_1}{\partial \theta_{z_b}}(\bar{V}_a) &= (-\bar{x}_a+\bar{x}_b)\cos\bar{\theta}_{y_b}\sin\bar{\theta}_{z_b}+\sin\bar{\theta}_{x_b}((-\bar{z}_a+\bar{z}_b)\cos\bar{\theta}_{z_b}+(-\bar{y}_a+\bar{y}_b)\sin\bar{\theta}_{y_b}\sin\bar{\theta}_{z_b})\\
&\quad+\cos\bar{\theta}_{x_b}((\bar{y}_a-\bar{y}_b)\cos\bar{\theta}_{z_b}+(-\bar{z}_a+\bar{z}_b)\sin\bar{\theta}_{y_b}\sin\bar{\theta}_{z_b})\\
\frac{\partial f_2}{\partial \theta_{z_b}}(\bar{V}_a) &= \cos\bar{\theta}_{z_b}((-\bar{x}_a+\bar{x}_b)\cos\bar{\theta}_{y_b}+((-\bar{z}_a+\bar{z}_b)\cos\bar{\theta}_{x_b}+(-\bar{y}_a+\bar{y}_b)\sin\bar{\theta}_{x_b})\sin\bar{\theta}_{y_b})\\
&\quad+((-\bar{y}_a+\bar{y}_b)\cos\bar{\theta}_{x_b}+(\bar{z}_a-\bar{z}_b)\sin\bar{\theta}_{x_b})\sin\bar{\theta}_{z_b}\\
\frac{\partial f_3}{\partial \theta_{z_b}}(\bar{V}_a) &= 0.
\end{aligned}
$$

Somit lässt sich $\Delta D'$ schreiben als:

$$
\begin{aligned}
\Delta' &= \bar{D}'-(V_a\ominus V_b)\\
&\approx \bar{D}'-(\bar{V}_a\ominus\bar{V}_b)+(K_a^{-1}\Delta V_a-K_b^{-1}\Delta V_b).
\end{aligned}
$$

Um wie zuvor den neuen Posefehler ΔD einzuführen, müssen wir die Substitutionen $\theta_{x_a}=\theta_{x_b}$ und $\theta_{y_a}=\theta_{y_b}$ durchführen. Dann gilt:

$$
K_aK_b^{-1}=\begin{pmatrix} I_3 & K\\ 0 & I_3\end{pmatrix}=H_{ab}=H_a^{-1}H_b.
$$

Dabei ist K gegeben durch:

$$
K=\begin{pmatrix}
0 & (\bar{z}_b-\bar{z}_a)\cos\bar{\theta}_{x_b}+(\bar{y}_b-\bar{y}_a)\sin\bar{\theta}_{x_b} & \cos\bar{\theta}_{y_b}((\bar{y}_b-\bar{y}_a)\cos\bar{\theta}_{x_b}+(\bar{z}_a-\bar{z}_b)\sin\bar{\theta}_{x_b})\\
\bar{z}_a-\bar{z}_b & (\bar{x}_a-\bar{x}_b)\sin\bar{\theta}_{x_b} & (\bar{x}_a-\bar{x}_b)\cos\bar{\theta}_{x_b}\cos\bar{\theta}_{y_b}+(\bar{z}_a-\bar{z}_b)\sin\bar{\theta}_{y_b}\\
\bar{y}_b-\bar{y}_a & (\bar{x}_a-\bar{x}_b)\cos\bar{\theta}_{x_b} & (\bar{x}_b-\bar{x}_a)\cos\bar{\theta}_{y_b}\sin\bar{\theta}_{x_b}+(\bar{y}_b-\bar{y}_a)\sin\bar{\theta}_{y_b}
\end{pmatrix}.
$$

Nun lässt sich mit $\Delta D=-H_a^{-1}K_a\Delta D'$ die Mahalanobisdistanz W_{ab} aufstellen.

$$
W_{ab}\approx(\bar{D}-D)^TC^{-1}(\bar{D}-D).
$$

Mit den bereits bekannten Bezeichnungen für D und $\bar{D}$ aus Abschnitt 5.2. Die Kovarianz C von $\bar{D}$ geht auch hier folgendermaßen aus C' hervor.

$$
C=H_aK_aC'K_a^TH_a^T.
$$

Wir wollen nun die Kovarianz von $D' = (x, y, z, \theta_x, \theta_y, \theta_z)^T$ herleiten. Dazu nehmen wir an, dass die Odometrie die Werte α,β und L mit derselben Bedeutung wie in Abschnitt 5.2.1 liefert. Zusätzlich haben wir ebenso die willkürlichen Schätzungen z',θ'_x und θ'_y der respektiven Koordinaten der Posedifferenz. Die Varianz dieser Werte sei modelliert durch σ_α,σ_β,σ_L,$\sigma_{z'}$,$\sigma_{\theta'_x}$ und $\sigma_{\theta'_y}$. D' ist durch folgende Beziehungen gegeben:

$$\begin{aligned} x &= L\cos\alpha \\ y &= L\sin\alpha \\ z &= z' \\ \theta_x &= \theta'_x \\ \theta_y &= \theta'_y \\ \theta_z &= \alpha + \beta. \end{aligned}$$

Dann ergibt sich die Varianz C' aus:

$$C' = J \begin{pmatrix} \sigma_\alpha^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_L^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{z'}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{\theta'_x}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{\theta'_y}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_\beta^2 \end{pmatrix} J^T,$$

wobei J die Jakobi-Matrix bestehend aus allen partiellen Ableitungen von D' ist.

$$J = \begin{pmatrix} -L\cos\alpha & \cos\alpha & 0 & 0 & 0 & 0 \\ L\cos\alpha & \sin\alpha & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Wenn wir zusätzlich die Tatsache repräsentieren wollen, dass die Odometrie nichts über die Werte z, θ_x und θ_y der Posedifferenz aussagt, so wählen wir einfach für die Varianzen $\sigma_{z'}$,$\sigma_{\theta'_x}$ und $\sigma_{\theta'_y}$ beliebig große Werte. Für die Lösung des Optimierungsproblems ist in der Regel nur die Inverse der Kovarianz von Bedeutung.

$$C'^{-1} = \begin{pmatrix} \frac{\cos^2\alpha}{\sigma_L^2} + \frac{(\sigma_\alpha^2+\sigma_\beta^2)\sin^2\alpha}{L^2\sigma_\alpha^2\sigma_\beta^2} & \frac{(L^2\sigma_\alpha^2\sigma_\beta^2-\sigma_L^2(\sigma_\alpha^2+\sigma_\beta^2))\cos\alpha\sin\alpha}{L^2\sigma_L^2\sigma_\alpha^2\sigma_\beta^2} & 0 & 0 & 0 & \frac{\sin\alpha}{L\sigma_\beta^2} \\ \frac{(L^2\sigma_\alpha^2\sigma_\beta^2-\sigma_L^2(\sigma_\alpha^2+\sigma_\beta^2))\cos\alpha\sin\alpha}{L^2\sigma_L^2\sigma_\alpha^2\sigma_\beta^2} & \frac{\sin^2\alpha}{\sigma_L^2} + \frac{(\sigma_\alpha^2+\sigma_\beta^2)\cos^2\alpha}{L^2\sigma_\alpha^2\sigma_\beta^2} & 0 & 0 & 0 & -\frac{\cos\alpha}{L\sigma_\beta^2} \\ 0 & 0 & \frac{1}{\sigma_{z'}^2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{\sigma_{\theta'_x}^2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\sigma_{\theta'_y}^2} & 0 \\ \frac{\sin\alpha}{L\sigma_\beta^2} & -\frac{\cos\alpha}{L\sigma_\beta^2} & 0 & 0 & 0 & \frac{1}{\sigma_\beta^2} \end{pmatrix}.$$

Machen wir nun einige Festlegungen wie folgt,

$$\frac{1}{\sigma_{z'}^2} = 0, \quad \frac{1}{\sigma_{\theta'_x}^2} = 0, \quad \frac{1}{\sigma_{\theta'_y}^2} = 0,$$

so modelliert dies die völlige Unbekanntheit dieser Werte. Allerdings hat dies ebenfalls die Folge, dass C'^{-1} nicht invertierbar und erst recht nicht positiv definit ist. Wird also diese Kovarianz verwendet, sollte dies nur mit Vorsicht getan werden und zum Beispiel nicht Bestandteil eines minimal verbundenen Graphen sein, da ansonsten das lineare Optimierungsproblem nicht lösbar wäre. Geht man jedoch von reellwertigen Varianzen $\sigma_{z'}$, $\sigma_{\theta'_x}$ und $\sigma_{\theta'_y}$ aus, so ist C' und auch $C = H_a K_a C' K_a^T H_a^T$ positiv definit und genügt vollkommen den Ansprüchen aus Kapitel 4.2.

5.3 Bedeutung der linearisierten Posedifferenzgleichung

In den vorangegangenen Abschnitten wurde die linearisierte Posedifferenzgleichung $D_{i,j}$ zwischen zwei Posen V_i und V_j eingeführt.

$$D_{i,j} = H_i \Delta V_i - H_j \Delta V_j.$$

Ebenfalls wurde eine Schätzung $\bar{D}_{i,j}$ von $D_{i,j}$ und die dazugehörige Kovarianz $C_{i,j}$ hergeleitet. Angenommen wir haben $n+1$ angefahrene Roboterposen $V_0, V_1, \ldots, V_n$, deren Schätzungen $\bar{V}_0, \bar{V}_1, \ldots, \bar{V}_n$ und somit alle $\bar{D}_{i,j}$ und $C_{i,j}$. Dann können wir das folgende lineare Optimierungsproblem aufstellen:

$$\begin{aligned} W &= \sum_{i,j} W_{i,j} \\ &= \sum_{i,j} (\bar{D}_{i,j} - D_{i,j})^T C_{i,j}^{-1} (\bar{D}_{i,j} - D_{i,j}). \end{aligned}$$

Dies hat dann den Vektor $\mathbf{X}$, mit der zugehörigen Kovarianz $\mathbf{C_X}$ zur Lösung. Die Einträge X_i von $\mathbf{X}$ sind jetzt allerdings nicht direkt die gesuchten Posen V_i, sondern stattdessen gilt: $X_i = H_i \Delta V_i$. Um also an die Pose V_i zu kommen, führt man folgende Aktualisierung durch:

$$V_i = \bar{V}_i - H_i^{-1} X_i.$$

Ebenso muss die Kovarianz C_i von V_i aktualisiert werden.

$$C_i = (H_i^{-1}) C_i^X (H_i^{-1})^T.$$

Wobei C_i^X die Kovarianz von X_i ist. Dies geht so aber nur unter der Annahme, dass $V_0 = 0$ ist. Sollte dies hingegen nicht der Fall sein, führt man anschließend noch die folgende Transformation durch:

$$\begin{aligned} V_i' &= V_0 \oplus V_i \\ C_i' &= K_0 C_i K_0^T. \end{aligned}$$

Gilt $V_0 = (x_0, y_0, \theta_0)^T$, so ist K_0 gegeben durch:

$$K_0 = \begin{pmatrix} \cos\theta_0 & -\sin\theta_0 & 0 \\ \sin\theta_0 & \cos\theta_0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Haben wir hingegen eine $6D$-Pose mit $V_0 = (x_0, y_0, z_0, \theta_{x_0}, \theta_{y_0}, \theta_{z_0})^T$, so gilt:

$$K_0 = \begin{pmatrix} R_{\theta_{x_0}, \theta_{y_0}, \theta_{z_0}} & 0 \\ 0 & I_3 \end{pmatrix},$$

wobei I_3 die Identitätsmatrix der Dimension 3 ist, während $R_{\theta_{x_0}, \theta_{y_0}, \theta_{z_0}}$ die Rotationsmatrix mit den Winkeln θ_{x_0},θ_{y_0} und θ_{z_0} ist.

Kapitel 6

Der Algorithmus von Lu und Milios

In diesem Kapitel beschreiben wir den Algorithmus für global konsistentes Scanmatching nach Lu und Milios. Dabei behandeln wir zuerst den Algorithmus und seine Funktionsweise als solches und gehen später auf die Implementierung ein. Der Algorithmus ist in seiner ursprünglichen Form nur für zweidimensionale Daten ausgelegt, in unserer Implementierung zusätzlich auch auf dreidimensionalen Daten anwendbar. Abschließend stellen wir einige Ergebnisse aus Experimenten vor und vergleichen diese mit dem ICP-Algorithmus.

Anhand der in Kapitel 5 abgeleiteten Posedifferenzen und Kovarianzen beschreiben wir hier eine praktische Anwendung des Schätzungsproblems aus Kapitel 4. Die Anwendung erstellt eine global konsistente Karte einer Roboterfahrt aus während der Fahrt aufgenommenen Laserscans. Im Programmablauf der Implementation bilden wir zuerst ein Netz aus den Roboterposen, an denen ein dreidimensionaler Laserscan aufgenommen wurde. Die initialen Poseschätzungen $\bar{V}_1, ...\bar{V}_n$ entnehmen wir den Odometriemessungen. Für jeden Link des Graphen berechnen sich die Posedifferenzenschätzungen $\bar{D}_{ij}$ aus den Poseschätzungen nach Gleichung (5.1) und ebenso die Kovarianzmatrizen $C_{i,j}$ entsprechend Gleichung (5.2). Aus den $\bar{D}_{i,j}$ und den $C_{i,j}$ bilden wir das Gleichungssystem $\mathbf{GX} = \mathbf{B}$ aus Abschnitt 4.2 und lösen dieses nach den Posevariablen $\mathbf{X}$ auf.

Um $\mathbf{G}$ und $\mathbf{B}$ zu erstellen, benötigen wir $C_{i,j}^{-1}$ und $C_{i,j}^{-1}\bar{D}_{i,j}$. In unserem Graphen aus Verbindungen zwischen den Scans kann man zwischen starken und schwachen Kanten unterscheiden. Die starken Kanten entstehen aus dem Matching von zwei Laserscans. Für diese Kanten lassen sich die Komponenten leicht berechnen mit $C_{i,j}^{-1} = (\mathbf{M}^T\mathbf{M}/s^2)$ und $C_{i,j}^{-1}\bar{D}_{i,j} = (\mathbf{M}^T\mathbf{Z})/s^2$. Im Falle einer schwachen Kante aus Odometriedaten ergibt sich die Lösung durch Multiplikation kleiner (im dreidimensionalen Fall (6×6)) Matrizen. In unserer Implementation verzichten wir dennoch auf die Berücksichtigung der schwachen Verbindungen, da diese bei dreidimensionalen Daten wenig aussagekräftig sind. Odometriewerte geben zwar Auskunft über die x- und y-Koordinaten sowie den Winkel θ_z, über die z-Koordinate und die beiden Winkel θ_x und θ_y vermögen sie aber keine Informationen zu geben. Die von uns vorgeschlagene Variante mit beliebig großen Varianzen für die nicht erfassten Werte (vgl. Kapitel 5.2.2) ist zwar eine Möglichkeit diese fehlenden Informationen zu umgehen, ist aber unserer Meinung nach nicht besonders sinnvoll. Auf diese Weise sind nur die Hälfte der Informationen über die Verbindung zweier Scans bekannt. Aus

diesen dennoch eine gute Transformation zu errechnen, erscheint uns als kaum möglich. Deswegen beschränken wir uns auf die Berechnungen mit den tatsächlich vorhandenen Daten aus dem Scanmatching.

Bei der Formulierung des Optimierungsproblems verwenden wir zur Linearisierung der Posedifferenzen eine lineare Approximation. Der Fehler dieser Approximation ist proportional zum Fehler der initialen Poseschätzung. Dies führt dazu, dass man wiederholter Anwendung des Algorithmus das Ergebnis noch verbessern kann. So kann bereits durch wenige Iterationen ein gutes Ergebnis erzielt werden.

6.1 Implementierung

6.1.1 Global konsistentes Scan Matching nach Lu und Milios

Repräsentation der Scans

Die Repräsentation der Scandaten ist in einer sehr einfach Form gehalten. Für jeden dreidimensionalen Laserscan existieren drei Dateien, von denen zwei die ursprünglichen Messwerte enthalten und auch nicht verändert werden. Die Änderungen, die durch den Algorithmus erreicht werden, werden einzig und allein in der dritten Datei eingetragen, um die Messwerte nicht zu verfälschen.

Die erste Datei enthält die Pose des Roboters mit der x-, y- und z-Koordinate und den Winkeln θ_x, θ_y, θ_z im Moment der Aufzeichnung des Laserscans. Diese Daten entsprechen der Odometrieschätzung des Roboters, die teilweise manuell oder mit ICP vorverarbeitet wurden. Der zweite Bestandteil sind die gescannten Punkte. Jeder Scanpunkt wird durch die x-, y-, und z-Koordinaten im lokalen Koordinatensystem, definiert durch die Roboterpose, repräsentiert. Der dritte Baustein enthält die nötigen Transformationen für den jeweiligen Scan. Für die Darstellung der Scans ist es notwendig, sie in ein globales Koordinatensystem zu übertragen. Dieses wird in unserem Fall durch das lokale Koordinatensystem des ersten Scans definiert. In der dritten Datei werden für jeden Scan alle Transformationen abgespeichert, die durch den Algorithmus berechnet werden. Dies ist zum einen die Anfangstransformation, die den Scan nach seiner Pose ausrichtet. Des Weiteren wird in jedem Iterationsschritt die Transformation abgespeichert, die den Scan von seiner Ursprungsposition in die aktuell berechnete bringt.

Bilden des Netzes

Ein für die Funktionalität des Algorithmus essentieller Bestandteil ist das Vorhandensein eines Netzes, das über die Verbindungen zwischen den einzelnen Scans Auskunft gibt. Wie in Kapitel 2.2.1 beschrieben ist es kein Leichtes, ein solches Netz aufzustellen und eine automatisierte Erstellung eines solchen erfordert einen eigenen Algorithmus. Deswegen ermöglicht unser Programm unterschiedliche Ansätze zum Bilden eines Graphen, der die Roboterfahrt nachvollzieht.

Sequentielles Matching Hier wird eine Roboterfahrt repräsentiert, in der keine Position zweimal angefahren wird. Das Matching wird sequentiell durchgeführt, das heißt, jeder Scan wird an seinen Vorgänger angepasst.

Sequentielles Matching mit Schleife Dieser Ansatz interpretiert eine Menge von Scans als ein große Schleife. Hier wird davon ausgegangen, dass Start- und Endpunkt einer Roboterfahrt nahezu identisch sind. Zusätzlich zu der sequentiellen Abarbeitung der Scans wird der erste Scan an den letzten angepasst.

Automatisiert erstelltes Netz Ein intuitiver Ansatz zum Erstellen eines Netzes ist die Erstellung anhand der Odometriedaten und der Größe des Bereichs, den ein Scan überdeckt. Wir approximieren den Bereich, den ein Scan überdeckt, durch eine Kugel, mit Mittelpunkt im Schwerpunkt der Punktemenge und dem Abstand zwischen Roboterposition und Mittelpunkt als Radius. Zwei Scans überlappen sich, wenn der Abstand der Schwerpunkte kleiner ist als die Hälfte der Summe ihrer Radien. In das Netz für das Matching werden alle Paare von überlappenden Scans eingefügt. Dieser Ansatz verlangt, dass die Schätzung der Roboterpose annähernd den wahren Werten entspricht, da sonst falsche Korrespondenzen zwischen den einzelnen Scans gefunden werden können.

Einlesen eines Netzes Als letzte Möglichkeit die Korrespondenzen zwischen den Scans für den Algorithmus zur Verfügung zu stellen ermöglichen wir eine externe Bestimmung des Netzes. Wie in Kapitel 2.2.1 beschrieben gibt es viele Möglichkeiten, auch mit anderen Sensoren, die ansonsten für den hier implementierten Algorithmus nicht von Bedeutung sind, Korrespondenzen zwischen Roboterposen zu finden. Um diese Informationen verwenden zu können, existiert eine Funktion zum Einlesen des Netzes aus einer Datei. Diese Datei ist in einer sehr einfachen Form gehalten und kann durch ein externes Programm geschrieben oder auch manuell erstellt werden.

Punktpaare

Die Bestimmung von Punktpaaren ist ein essentieller Bestandteil des Algorithmus. Die Qualität der Punktpaare beeinflusst maßgeblich das Ergebnis des Algorithmus. Da die Suche nach Punktpaaren der zeitaufwendigste Bestandteil ist, haben wir uns bei der Suche nach Punktpaaren für die schnellste Variante entschieden, der auch beim ICP-Algorithmus verwendeten Suche nach dem nächsten Punkt mit Hilfe eines kd-trees (vgl. Kapitel 2.1.1). Bei jedem Paar von überlappenden Scans wird für jeden Punkt aus dem aktuellen Scan ein korrespondierender Punkt aus dem Referenzscan gesucht. Ist die Distanz zum nächsten Punkt zu groß, wird das Punktpaar nicht berücksichtigt. Dafür wird aus den Punkten des Referenzscans ein kd-tree gebildet. Zur Optimierung des Baumes wird das in Kapitel 2.1.1 beschriebene Verfahren verwendet, in dem als diskriminierender Schlüssel die Koordinate mit der breitesten Verteilung gewählt wird und als Grenze der Mittelwert zwischen der Ober- und Untergrenze. In jedem Blatt befinden sich höchstens zehn Punkte. Die Suche nach dem nächsten Punkt verläuft ebenfalls wie in Kapitel 2.1.1 beschrieben.

Wir verzichten hier auf eine Reduktion der Punktpaare. Zur Beschleunigung des Algorithmus kann dies jedoch hilfreich sein oder bei geringerer vorhandener Rechnerleistung sogar notwendig werden.

Minimierung der Fehlerfunktion

Wie viele Optimierungsalgorithmen arbeitet auch der von uns verwendete auf Basis der Minimierung einer Fehlerfunktion. Die hier verwendete Fehlerfunktion entstammt dem in Kapitel 4 beschriebenen Schätzungsproblems mit

$$\mathbf{W} = (\bar{\mathbf{D}} - \mathbf{HX})^T \mathbf{C}^{-1} (\bar{\mathbf{D}} - \mathbf{HX}).$$

$\mathbf{W}$ setzt sich zusammen als Summe von Termen für jede Kante, bestehend aus der observierten Posedifferenz der beiden Endknoten der Kante und der Kovarianz dieser Observation. Die Minimierung von $\mathbf{W}$ ergibt sich durch

$$\mathbf{X} = (\mathbf{H}^T \mathbf{C}^{-1} \mathbf{H})^{-1} \mathbf{H}^T \mathbf{C}^{-1} \bar{\mathbf{D}} = \mathbf{G}^{-1} \mathbf{B}$$

Wobei sich $\mathbf{B}$ aus den Vektoren $B_i = \sum_{j=0;j\neq i}^{n} C_{i,j}^{-1} \bar{D}_{i,j}$ zusammensetzt und $\mathbf{G}$ eine Matrix aus den entsprechenden Kovarianzen ist (vgl. Kapitel 4). Die $\bar{D}_{i,j}$ sind die Posedifferenzen des i-ten und j-ten Scans, die sich anfangs aus den durch Odometriedaten oder deren Vorverarbeitung mit ICP bestimmten Posedaten des Scans ergibt und in späteren Iterationsschritten die transformierten Werte beinhaltet. Die Posedifferenz ergibt sich wie in Kapitel 5 beschrieben, aus den Differenzen aller Punktpaare der beiden Scans mit

$$\bar{D} = (\mathbf{M}^T \mathbf{M})^{-1} \mathbf{M}^T \mathbf{Z}.$$

Die Varianz von $\bar{D}$ wird approximiert durch

$$C_D \approx s^2 (\mathbf{M}^T \mathbf{M})^{-1}.$$

Da für den Algorithmus nur das Inverse C_D^{-1} benötigt wird, verwenden wir hier die direkte Berechnung durch

$$\begin{aligned} C_D^{-1} &= (\mathbf{M}^T \mathbf{M})/s^2 \\ C_D^{-1} \bar{D} &= (\mathbf{M}^T \mathbf{Z})/s^2. \end{aligned}$$

Gleichung (5.1.2) verdeutlicht, wie man $\mathbf{M}^T\mathbf{M}$ effizient ohne Matrixmultiplikation aufstellen kann. Eine ähnliche Vereinfachung existiert auch für die Berechnung von $\mathbf{M}^T\mathbf{Z}$ durch

$$\mathbf{M}^T \mathbf{Z} = \sum_{k=0}^{m} \begin{pmatrix} \Delta x_k \\ \Delta y_k \\ \Delta z_k \\ -z_k \cdot \Delta y_k + y_k \cdot \Delta z_k \\ -y_k \cdot \Delta x_k + x_k \cdot \Delta y_k \\ z_k \cdot \Delta x_k - x_k \cdot \Delta z_k \end{pmatrix}$$

mit

$$\begin{pmatrix} \Delta x_k \\ \Delta y_k \\ \Delta z_k \end{pmatrix} = \bar{Z}_k = \bar{V}_a \oplus u_k^a - \bar{V}_b \oplus u_k^b$$

und einer Annäherung der exakten Werte eines jeden Punktes durch

$$\begin{pmatrix} x_k \\ y_k \\ z_k \end{pmatrix} = u_k \approx (\bar{V}_a \oplus u_k^a + \bar{V}_b \oplus u_k^b)/2.$$

s^2 erhält man durch einfache Summierung von

$$\begin{aligned} s^2 = \sum_{k=0}^{m} & (\Delta x_k - (\bar{D}_0 - y_k \cdot \bar{D}_4 + z_k \cdot \bar{D}_5))^2 \\ + & (\Delta y_k - (\bar{D}_1 - z_k \cdot \bar{D}_3 + x_k \cdot \bar{D}_4))^2 \\ + & (\Delta z_k - (\bar{D}_2 + y_k \cdot \bar{D}_3 - x_k \cdot \bar{D}_5))^2. \end{aligned}$$

Durch Summierung der so erhaltenen Matrizen C_D^{-1} und $C_D^{-1}\bar{D}$ erhalten wir $\mathbf{B}$ und $\mathbf{G}$.

Um den Lösungsvektor $\mathbf{X}$ zu erhalten, muss nun $\mathbf{G}$ invertiert und mit $\mathbf{B}$ multipliziert werden. Dieser Schritt lässt sich nicht weiter beschleunigen.

Die Aktualisierung der Scanposen geschieht wie in Abschnitt 5.3 beschrieben. Der erste Scan liegt auf dem Nullpunkt des globalen Koordinatensystems. Alle anderen Scans müssen nun relativ zu diesem verschoben werden. Die neue Pose des i-ten Scans wird durch

$$V_i = \bar{V}_i - H_i^{-1} X_i$$

gegeben.

Aufgrund der Approximierung durch Linearisierung der Posedifferenzen sind die berechneten Posen kein optimales Ergebnis. Der Fehler der Approximation ist proportional zum Fehler der Anfangsschätzungen. Durch mehrfache Anwendung des Algorithmus können wir das Ergebnis verbessern. Dabei sind die neu erlangten Posen Grundlage für die Posedifferenzgleichungen. Die Kovarianzen einer jeden Verbindung kann man neu berechnen oder der in der vorangegangenen Iteration berechneten Kovarianzmatrix $\mathbf{C_X} = \mathbf{G}^{-1}$ entnehmen.

6.1.2 ICP

Zum Vergleich mit unserem Algorithmus verwenden wir zwei unterschiedliche ICP-Varianten, zum einen paarweises Scanmatching, zum anderen eine Form des Metascanmatchings. Beim paarweisen Scanmatching wird analog zur Netzstruktur in unserem Algorithmus jeder Scan an seinen Nachbarn angepasst. Dazu wird identisch zur Suche nach korrespondierenden Punkten beim global konsistenten Scanmatchingansatz aus den Punkten des Referenzscans ein kd-tree gebildet und für jeden Punkt des aktuellen Scans der nächste Punkt gesucht. Beim Matching

mit einem Metascan wird der kd-tree aus den Punkten aller bereits registrierten Scans gebildet. Nachdem alle Punktpaare gefunden sind, wird mit ihnen die Transformation bestimmt, die den Abstand minimiert und diese auf den aktuellen Scan angewandt. Zur Minimierung werden Fehlerfunktion und die Methode mit der Singulärwertzerlegung von Seite 14 werwendet. Das Bestimmen der Punktpaare mit anschließender Transformation wird wiederholt, bis ein ausreichend gutes Ergebnis erzielt oder die maximale Anzahl an Iterationen erreicht wird.

6.2 Experimenteller Aufbau

Um die Implementation und die Funktionalität des Algorithmus zu testen, führen wir einige Experimente mit simulierten Daten und mit reellen, von einem Roboter aufgenommenen Laserscandaten durch. Hierzu verwenden wir die Roboterplattform Kurt3D [15] und eine Simulation derselben Roboterplattform mittels der Simulationssoftware USARSIM [1]. Mit beiden Methoden lassen sich die 6-dimensionale Roboterpose (x, y, z, Roll-, Neig-, und Gierwinkel) und die Koordinaten der aufgenommenen Scanpunkte (in x-, y-, z-Koordinaten) erzeugen, die der Algorithmus als Eingabe benötigt.

6.2.1 Die Roboterplattform Kurt3D

Die Roboterplattform Kurt3D ist ein, aufbauend auf die Plattform KURT2 [38], ursprünglich am Fraunhofer Institut für Autonome Intelligente Systeme (AiS) entwickelter und nun an der Universität Osnabrück weiterentwickelter autonomer mobiler Roboter [20], der bei einer Länge von 45 cm, einer Breite von 33 cm und einer Höhe von 47 cm 22.6 kg wiegt. Die sechs großen Räder aus Gummi, von denen die mittleren nach außen versetzt sind, werden mit zwei 90W Motoren (kurzzeitig 200W) angetrieben und können sowohl glatten als auch unebenen Untergrund befahren. Da die vorderen und hinteren Räder ohne Profil gehalten sind, kann der Roboter auf der Stelle rotieren. Eine Batterieladung (28 NIMH-Akkus mit 4500 mAh) reicht für eine Betriebszeit von circa 4 Stunden. Den Kern des Roboters bildet ein Laptop mit einem Intel-Centrino-1400 MHz-Prozessor, 768 MB RAM und einem Linux Betriebssystem. Über eine CAN-Schnittstelle werden mit Hilfe eines eingebetteten 16-Bit CMOS Mikrocontrollers Steuersignale an die Motoren gesandt. Außerdem verfügt der Roboter über zwei bewegliche Logitech QuickCam 4000 Kameras, die durch LED-Ringe auch bei Dunkelheit einsatzfähig sind.

Für diese Arbeit von besonderer Bedeutung ist der angebrachte 3D-Laserscanner [32] (s. Abb. 6.2). Mit einer Akku-Ladung (Scanner: 17 W, 20 NiMH-Akkus mit 4500 mAh, Servo: 0.85 W, 4.8 V mit 4500 mAh) erreicht man eine Betriebszeit von bis zu fünf Stunden. Es handelt sich um einen SICK 2D Laserscanner (LMS-200), der über eine Halterung derart auf dem Roboter montiert ist, dass er um eine horizontale Achse rotieren kann. Dies ermöglicht die Erfassung der Umgebung mit einem horizontalen Öffnungswinkel von 180 Grad und einem vertikalen Öffnungswinkel von 90 Grad mit unterschiedlicher horizontaler (181, 361, 721) und vertikaler (128, 176, 256, 400, 500) Auflösung. Das Aufnehmen eines horizontalen Laserscans mit 181 Datenpunkten benötigt 13 ms. Mit zunehmender Punktzahl wächst die Zeit proportional. Für einen Scan mit 361 $\times$ 176 Datenpunkten ergibt das 4.5 s. Der 3D Laserscanner liefert die Entfernung

Abbildung 6.1: Links: Flur im AVZ-Gebäude Osnabrück real, rechts: simuliert.

der gemessenen Punkte sowie die reflektierte Lichtmenge zurück. Somit lassen sich detaillierte Bilder der Szene erstellen. Um die Genauigkeit des Sensors zu verstärken, werden 3D-Laserscans nur bei stehendem Roboter aufgenommen.

Robotersimulation mit der Simulationssoftware USARSIM

USARSIM [35] ist eine Simulationssoftware für Roboter in Katastrophen- und Rettungsszenarien und basiert auf den Mehrspieler Ego-Shooter Unreal Tournament 2003 oder 2004 [1]. Die von M. Lewis und J. Wang für Testszenarien des American National Institute of Standards (NIST) [35] entwickelte Software kommt auch in der RoboCup Rescue Real Robot Liga zum Einsatz. Aufgrund der Verbindung mit dem kommerziellen Computerspiel Unreal Tournament verfügt die Software über eine exzellente Grafik und real wirkende Simulation, wie derzeit bei Computerspielen der Standard ist. Es existiert eine Skriptsprache, mit der Roboter und ihr Verhalten individuell entwickelt werden können. Der im Lieferumfang enthaltene Unreal-Editor sowie das Open Source Programm Blender unterstützen den Benutzer beliebige Szenen realitätsnah zu erzeugen. Das Projekt Gamebots [10] bietet eine Modifikation von Unreal Tournament an, mit der die Kommunikation mit dem simulierten Roboter beinahe identisch zu der mit dem realen Roboter gestaltet werden kann. Als Erweiterung der Client-Server-Architektur des Spiels, bei der das Rendering von jedem Clienten, den einzelnen Spielern geleistet wird, während der Server einzig für die Koordination und Interaktion zuständig ist, kann mit Hilfe dieses Projekts die Steuerung des Agenten und die Sensordatenübertragung vom Spiel auf den Agenten mittels einer TCP/IP Verbindung durchgeführt werden. Die „Karma-Physics-Engine“ hilft bei der physikalisch korrekten Simulation von Gelenken, Motoren, Rädern, Federn und Scharnieren, indem diese als Bewegung von Starrkörpern simuliert werden, aus denen sich leicht komplexe Objekte zusammensetzen lassen.

Die Simulation des Roboters Kurt3D in USARSIM besteht aus einem im Unreal-Editor erstellten Gittermodell der Hardware, das in Abbildung 6.2 zu sehen ist. Um die Roboterkontrollsoftware alternativ mit der Simulationssoftware oder einem realen Roboter zu verwenden, wurde diese

Abbildung 6.2: Links: Kurt3D real, Rechts: Kurt3D simuliert.

durch Schnittstellen an beide Varianten angepasst.

Bislang integriert die Simulation folgende Komponenten:

- Der Motor, der den Roboter antreibt inklusive PWM-Signalen wird simuliert.
- Die Odometrie, d. h. die Radumdrehungen in Ticks, ist integriert.
- Ein Gyro bestimmt die aktuelle Neigung des Roboters.
- Eine Kamera liefert Umgebungsbilder. Diese werden von einem Kameraserver als Schnappschüsse von Unreal an den Treiber der Kamera weitergegeben.
- Ein Laserscanner, der 181 Distanz-Werte auf einer Ebene in der Umgebung vor dem Roboter liefert. Der Scanner ist drehbar montiert, so dass er sich vertikal neigen kann und somit genau wie der reale Scanner dreidimensionale Laserscans aufzeichnen kann.

Zur schnellen Erzeugung von Daten mit Hilfe der Simulationssoftware ist es möglich, die Simulation auf vier Computern durchzuführen:

1. Auf einem Computer läuft der Unreal Server, der alle Robotersensoren, außer der Kamera, simuliert.
2. Auf einem zweiten Computer wird die Kamera, über ein Programm, das Bilder von einem Unreal Betrachter Fenster aufnimmt, simuliert.
3. Das Kontrollprogramm des Kurt3D-Roboters läuft auf einem dritten Computer und erhält Motorsignale mit 100Hz und Laserscans mit 75Hz von Unreal.
4. Ein weiterer Computer beherbergt die Benutzerschnittstelle zur Steuerung des Roboters. Dieser Computer ist mit dem Computer verbunden, auf dem die Roboterkontrollsoftware läuft, nicht aber mit Unreal.

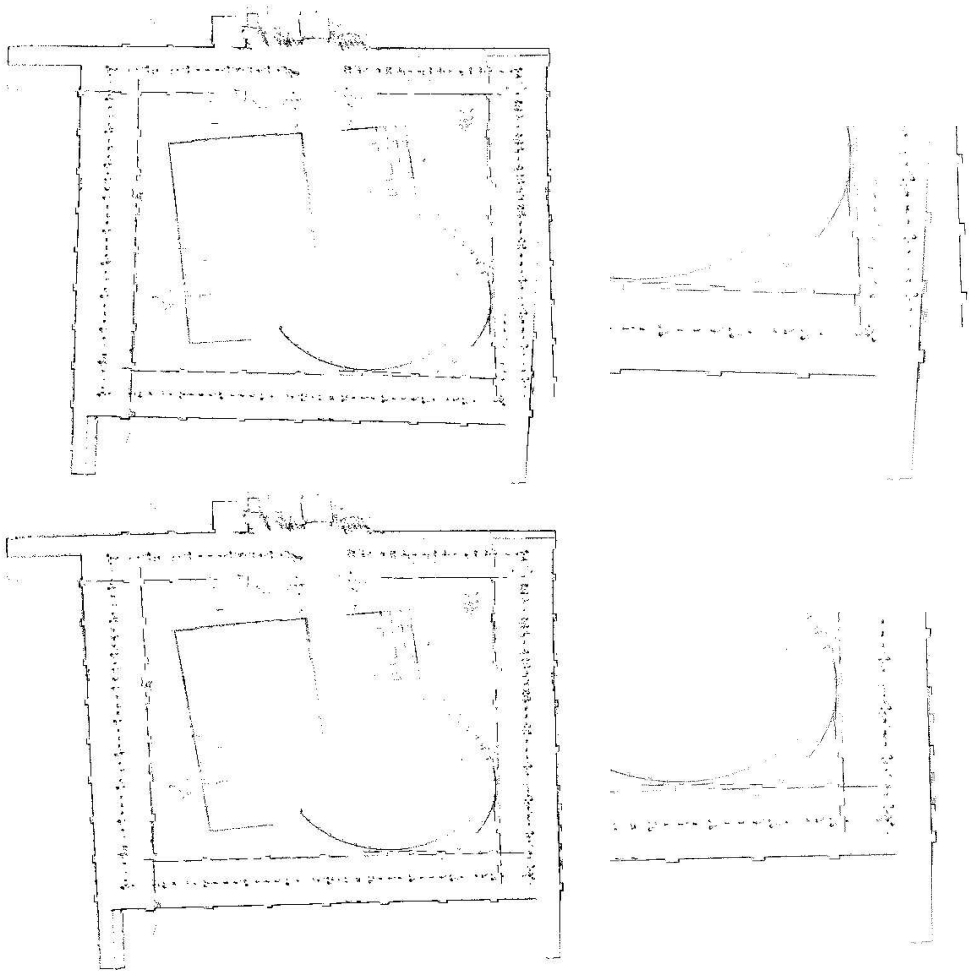


Abbildung 6.3: Ergebnisse von ICP (oben) und Lu/Milios (unten) am Beispiel von zweidimensionalen Daten.

6.2.2 Durchführung der Experimente

Versuche mit zweidimensionalen Daten

In diesem Abschnitt stellen wir die Ergebnisse einiger Versuche mit zweidimensionalen Daten vor. Obwohl die Implementation für dreidimensionale Daten ausgelegt ist, sollte der Algorithmus auch auf zweidimensionalen Daten arbeiten, indem bei sämtlichen Daten die fehlenden Werte auf Null gesetzt werden. Zum einen wollen wir dieses überprüfen, zum anderen bieten zweidimensionale Daten eindeutige Vorteile im Vergleich der Qualität des Ergebnisses. Aufgrund der geringen Datenmengen und der besseren Möglichkeiten zur Anzeige geben visuelle Vergleiche der Ergebnisse gut Aufschlüsse über die Funktion der Implementation.

Für unsere Experimente wählen wir einen Datensatz aus Schloss Dagstuhl. Abbildung 6.3 zeigt grafisch die Ergebnisse des Matchings des Datensatzes als Aufsicht. Die Fahrt des Roboters

beginnt in der unteren rechten Ecke und führt zunächst nach oben und dann den ganzen Flur entlang, bis zurück zur Startecke. Die oberen Abbildungen sind das Ergebnis des sequentiellen Matchings mit ICP, unten sieht man das Ergebnis unseres Algorithmus. Beide Versuche wurden mit 50 Iterationen durchgeführt. Auf der rechten Seite ist noch einmal ein vergrößertes Bild der Ecke, in der die Schleife geschlossen wird.

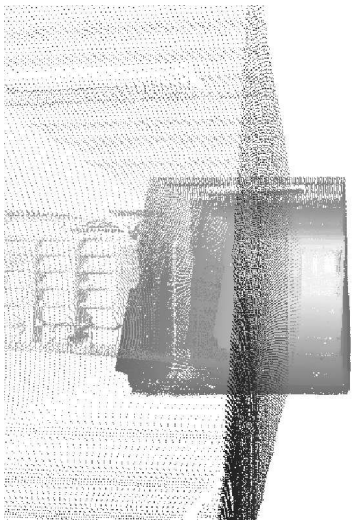
Bei der Betrachtung erkennt man, dass in beiden Versionen der Übergang zwischen aufeinanderfolgenden Scans glatt verläuft. Doch global betrachtet fällt beim ICP eine Aufsummierung kleiner Fehler auf, die dazu führt, dass zwischen dem letzten und dem ersten Laserscan keine korrekte Verbindung besteht. Bei der Implementation des Algorithmus von Lu und Milios hingegen wirkt die Ecke optisch gut zusammengefügt. Auch die für 6D-Posen implementierten Algorithmen lassen die Scans weiterhin in einer Ebene liegen.

Dreidimensionale Daten auf ebenem Untergrund

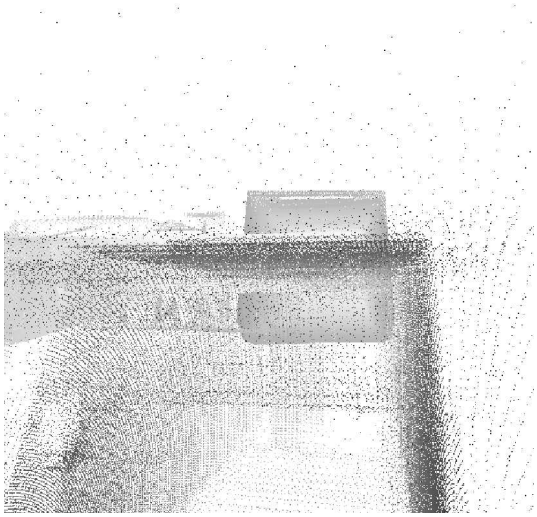
Der nächste Schritt unserer Experimente erweitert die Tests auf dreidimensionale Daten. Hierzu verwenden wir Datensätze aus dem AVZ-Gebäude der Universität Osnabrück. Zum einen verwenden wir einen mit USARSIM erzeugten Datensatz einer Laborumgebung aus dem fünften Stock, bestehend aus zwei Computerlaboren und einem angrenzenden Flur. Der zweite Datensatz wurde bei einer reellen Roboterfahrt im Flur des zweiten Stocks aufgenommen (s. Abb. 6.5). Da die Daten aus geschlossenen Gebäuden stammen, sind hier klare geometrische Strukturen vorhanden. Auf diese Weise lässt sich gut überprüfen, inwieweit die einzelnen Scans gut aneinandergefügt werden, indem betrachtet wird, inwiefern zusammengesetzte Wände und Fußböden Ebenen bilden.

Zur qualitativen Evaluation verwenden wir zunächst den Datensatz aus dem zweiten Stock. Abbildung 6.5 zeigt den Verlauf des Scanmtching mit dem LUM-Algorithmus. Man erkennt, dass der rechte Gang mit zunehmender Anzahl an Iterationen gerader wird. Gleichzeitig verringert sich der Versatz an der Schnittstelle. Eine genauere Betrachtung der Schnittstelle bietet Abbildung 6.4. Hier sieht man nun einen Vergleich zwischen dem ICP- und dem LUM-Algorithmus. Auf der linken Seite ist jeweils eine Ansicht von innerhalb des Flures, auf der rechten Seite ein Blick von außerhalb auf die Außenwand. Man erkennt, dass weder beim ICP noch bei unserer Implementation der letzte Scan gut an den ersten Scan angefügt ist. Doch während die Fehler unseres Algorithmus in der Ebene liegt und somit nur aus Schwächen bei der Korrektur besteht, liegt der ICP auch in der Höhe falsch, was im Vergleich zur Anfangsschätzung einen zusätzlichen Fehler bedeutet.

Eine quantitative Aussage über die Qualität des Matchings lässt sich anhand der Graphen in Abbildung 6.6(a) machen. Die Graphen bilden die Fehler des Matchings der beiden Datensätze ab. Die oberen Graphen betreffen die Daten aus dem zweiten Stock, die unteren Graphen die Daten aus dem fünften Stock. Abgetragen wird jeweils der Fehler der Pose jedes einzelnen Scans im Vergleich zur *Ground Truth*. Mangels echter *Ground Truth*-Posen, haben wir diese durch manuelles Matching erreicht. Die Graphen bieten einen Vergleich zwischen dem initialen Fehler der Anfangsschätzung und dem Fehler nach Matching per ICP beziehungsweise LUM.

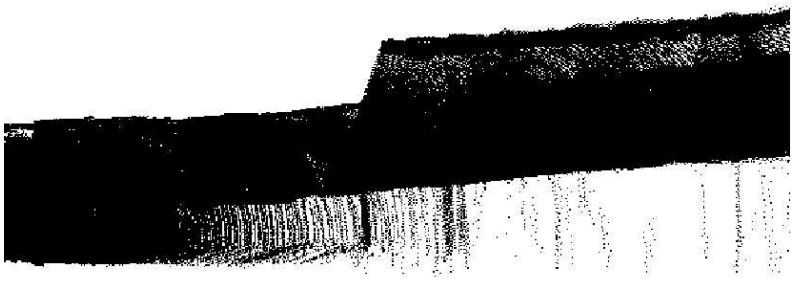


(a) Ansicht LUM



(b) Ansicht ICP

(c) Seitenansicht des Flurs nach Matching mit LUM



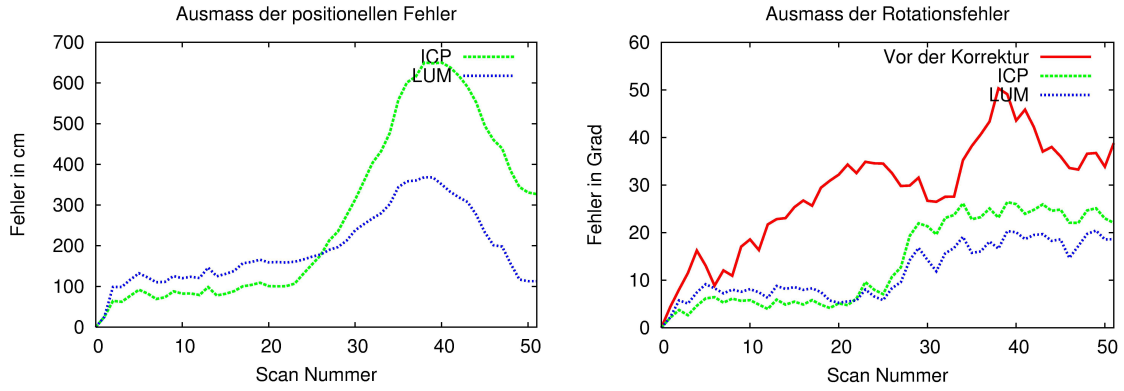
(d) Seitenansicht des Flurs nach Matching mit ICP

Abbildung 6.4: Vergleich zwischen dem global konsistenten Matching-Algorithmus (oben) und ICP und am Beispiel eines Flurs (unten).

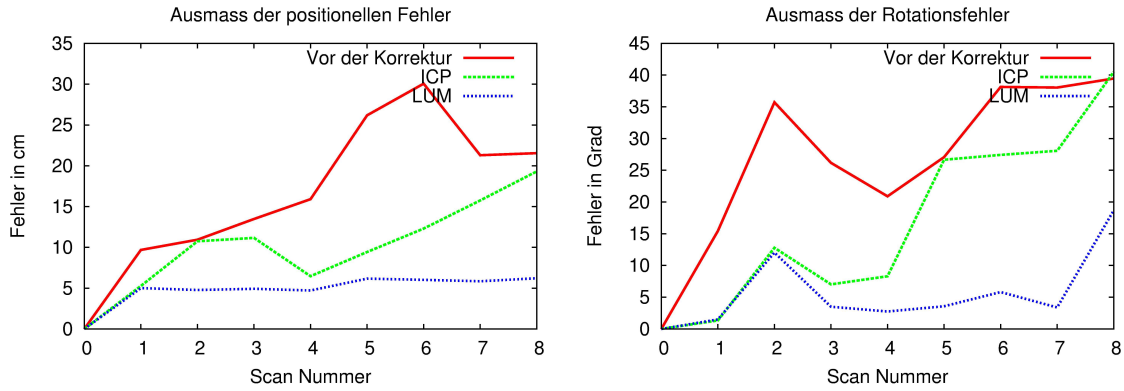
(a) Iteration 0 (b) Iteration 4 (c) Iteration 8

(d) Iteration 12 (e) Iteration 16 (f) Iteration 20

Abbildung 6.5: Exemplarischer Verlauf des Scanmatchings mit dem global konsistenten Scanmatching-verfahren nach Lu und Milios. Man sieht eine Aufsicht auf den anfangs stark verzerrten Flur im zweiten Stock des AVZ-Gebäudes der Universität Osnabrück. Im Verlauf glättet sich der linke Gang, während im rechten Gang der Anfang und das Ende der Roboterfahrt aufeinander gezogen werden.



(a) Zweiter Stock des AVZ-Gebäudes



(b) Laborumgebung im fünften Stock des AVZ-Gebäudes

Abbildung 6.6: Posefehler vor und nach dem Scanmatching.

Im linken Graphen ist der Fehler der Translation aufgetragen. Dieser wird als Distanz der errechneten Posen zu den *Ground Truth*-Posen berechnet. Hierbei erkennt man deutlich, dass bei den beiden Versuchen sowohl beim ICP als auch bei LUM der Fehler in der zweiten Hälfte größer ist. Jedoch ist beim LUM-Algorithmus erkennbar, dass der Fehler besser verteilt ist, da der gesamte Kurvenverlauf flacher ist. Der Rotationsfehler ist im rechten Graphen aufgeführt. Wenn die Orientierungsdifferenz mit der normierten Rotationsachse n und dem Drehwinkel r dargestellt wird, so ist der Rotationsfehler der Betrag des Drehwinkels. In diesem Vergleich liegen sowohl ICP als auch LUM deutlich unter dem initialen Rotationsfehler.

Zur Bewertung des Vergleichs sei erwähnt, dass der ICP lokal durchaus gute Ergebnisse erzielt. Die Schwäche liegt eindeutig in den globalen Ergebnissen, da sich ein Fehler in einem früheren Scan auf die späteren Scans auswirkt. Dies wird besonders deutlich bei den Laserscandaten aus der Laborumgebung im fünften Stock. Hier zeigt sich, dass das Matching mit ICP den

Abbildung 6.7: Kurt3D im Botanischen Garten. Links: Auf steinigem Untergrund. Rechts: Weg des Roboterpfads.

Anfangsfehler verringert, die Verbesserungen aber überwiegend die ersten Scans betreffen. Die späteren Scans weisen weiterhin einen großen Fehler auf. Bei dem globalen Ansatz nach Lu und Milios wird hingegen der Fehler stark verringert und auf alle Scans verteilt.

Der Botanische Garten

Zur Überprüfung der Funktionalität des Programms in allen Dimensionen haben wir eine Testreihe im Botanischen Garten durchgeführt. Die Besonderheit der dort aufgenommenen Scans liegt darin, dass alle Freiheitsgrade ausgenutzt werden. Anders als bei vorherigen Experimenten findet man im Botanischen Garten keinen ebenen Untergrund. Dies führt dazu, dass nicht nur die Scans dreidimensional sind, sondern auch die Roboterpose echt 6-dimensional ist, da z-Koordinate sowie die Winkel θ_x und θ_y nicht mehr fixiert sind und neben dem üblichen Rauschen auch tatsächliche Veränderungen aufweisen. Nachdem wir bereits festgestellt haben, dass unser Algorithmus diese Variablen nicht stark verändert, wenn keine Änderungen vorliegen, zeigen diese Experimente nun, dass andersherum tatsächlich notwendige Anpassungen auch durchgeführt werden. Abbildung 6.7 zeigt den Roboter auf einem steinigen Weg, wie er typisch ist für den Botanischen Garten. In einem Experiment haben wir den Roboter eine Schleife bestehend aus einem Stück dieses Weges und einem Schotterweg fahren lassen und Laserscans in Abständen von einigen Metern aufnehmen lassen. Auf diese Scans haben wir anschließend Scanmatching angewandt. Abbildung 6.8 zeigt die Ergebnisse des Matchings mit unserem Algorithmus. Bei Betrachtung des Fotos 6.7 sieht man, dass das Ergebnis den gefahrenen Pfad zufriedenstellend abbildet. ICP scheitert hieran komplett. Abbildung 6.9 zeigt den Vergleich der Kurve auf dem steinigen Weg. An der roten Markierung erkennt man, wo ICP mehrere Scan falsch aneinanderfügt.

Abbildung 6.8: Ergebnisse des Scanmatchings einer kleinen Schleife im Botanischen Garten mit LUM.

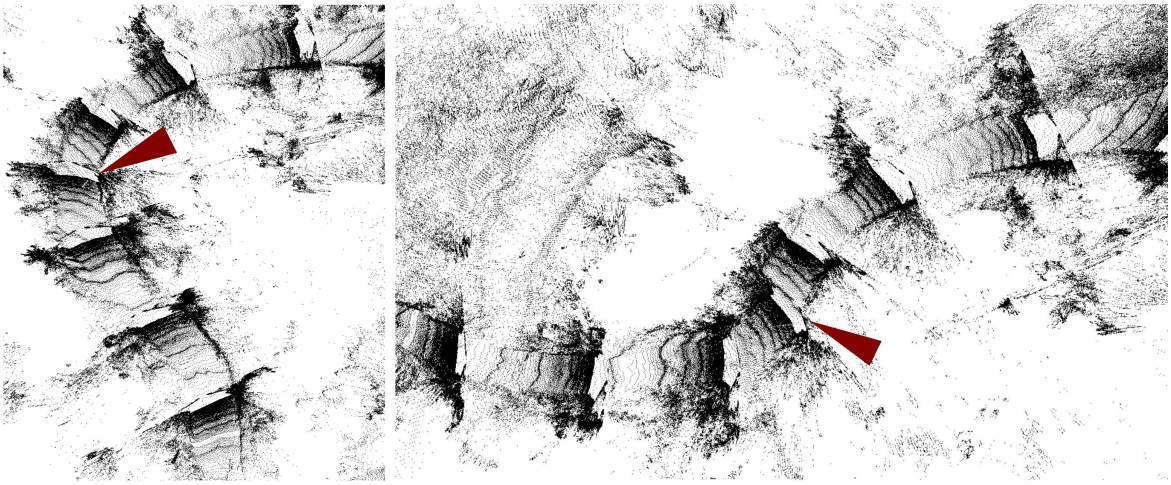


Abbildung 6.9: Vergleich zwischen ICP (rechts) und LUM (links) bei einer kleinen Schleife im Botanischen Garten. In der Kurve schlägt das ICP-Matching an der rot markierten Stelle fehl. Die Netzstruktur des LUM-Algorithmus hält die Scans in der richtigen Orientierung.

Des Weiteren haben wir eine Roboterfahrt auf dem Weg zum Botanischen Garten durchgeführt. Bei der in 6.10 abgebildeten Brücke haben wir Daten einer dreidimensionalen Roboterfahrt aufgenommen. Der Pfad begann auf der rechten Seite der Brücke, führte dann über sie hinüber, den Abhang von dem aus das Foto aufgenommen wurde hinunter, unter der Brücke durch und den Abhang hoch, zurück zum Startpunkt.

Abbildung 6.10: Die Brücke auf dem Weg zum Botanischen Garten.

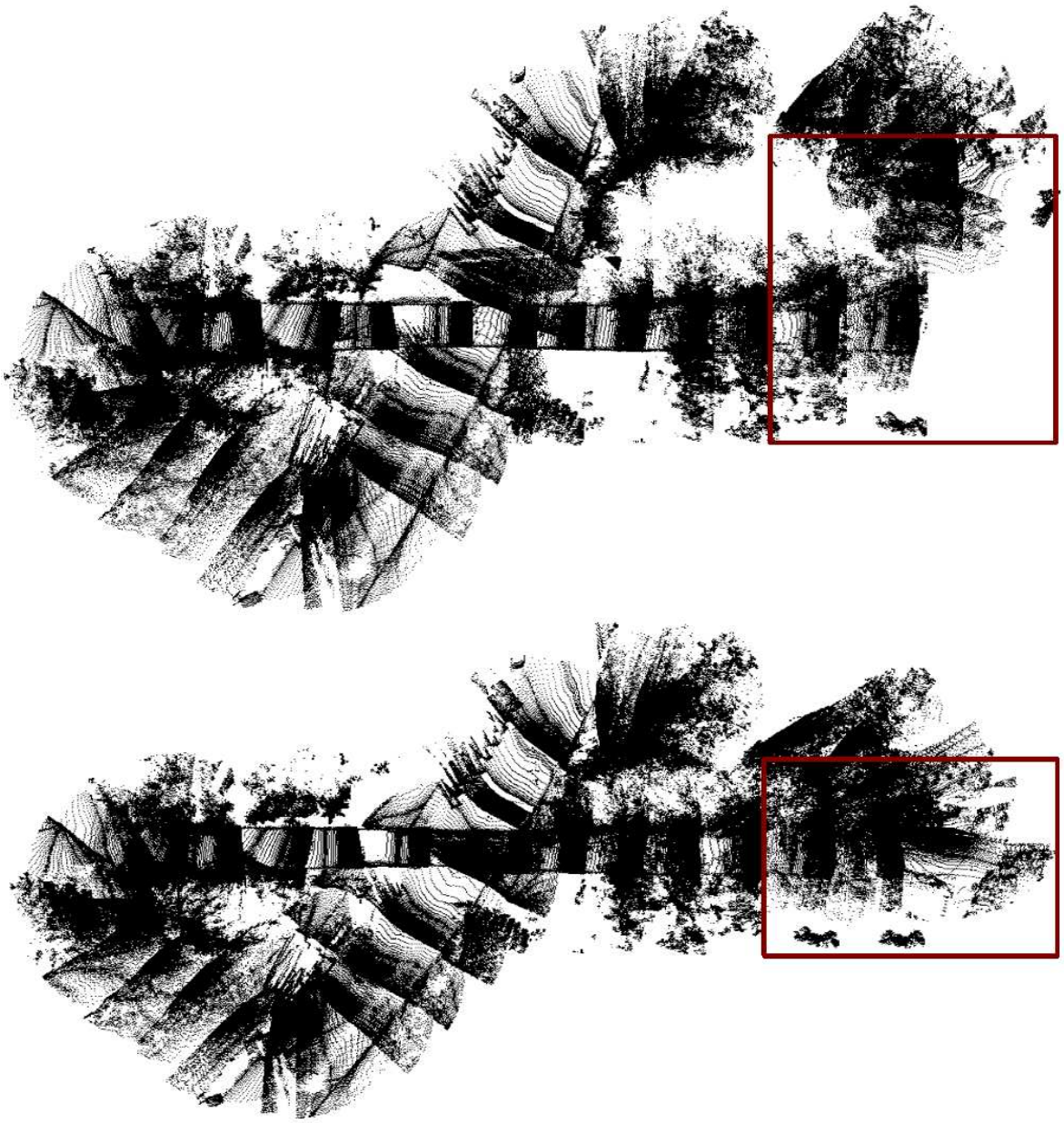


Abbildung 6.11: Blick auf die Roboterkarte aus der Vogelperspektive nach Scanmatching mit ICP (oben) und LUM (unten). Die Schnittstelle ist mit der roten Markierung versehen.

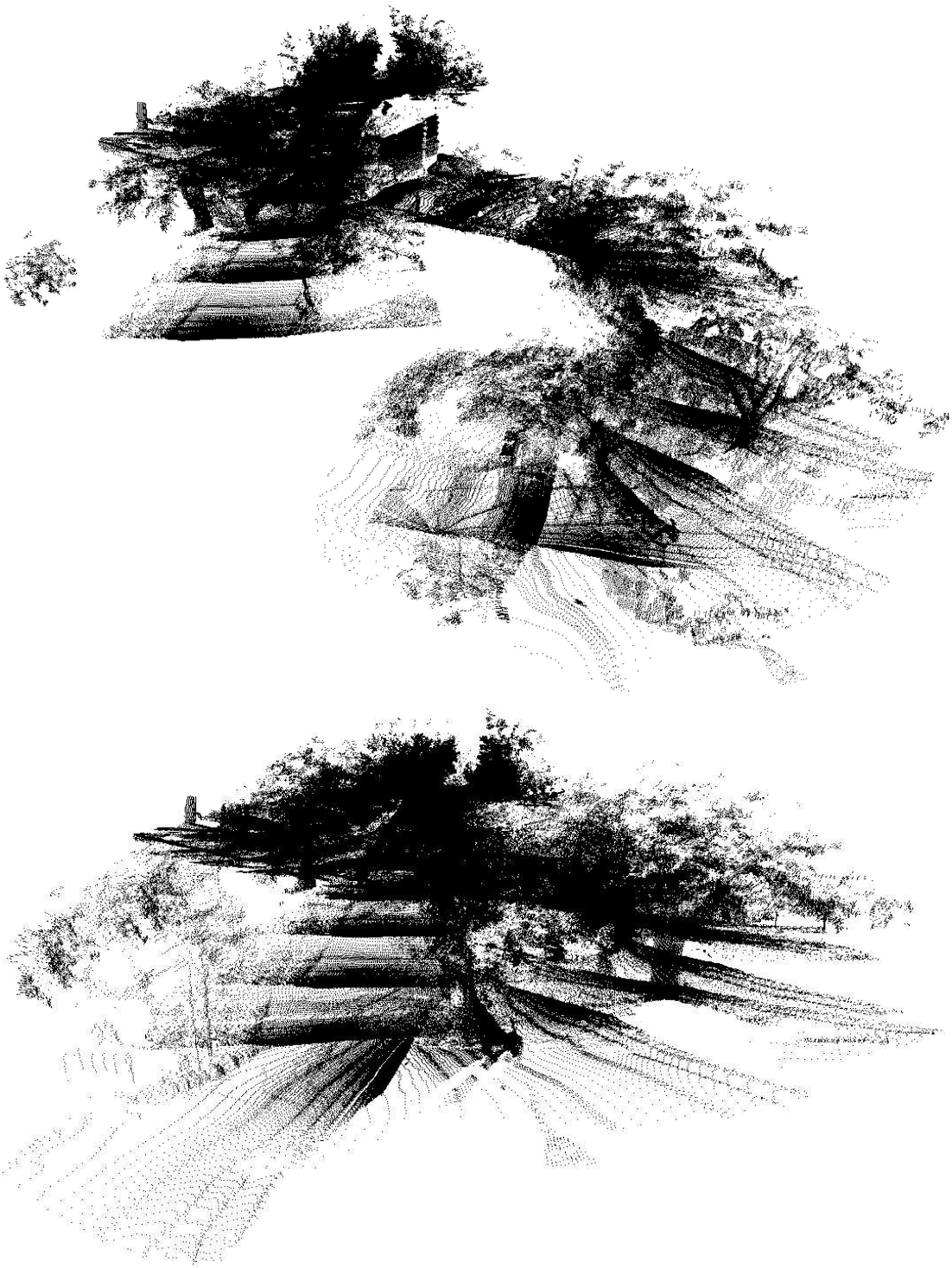


Abbildung 6.12: Ansicht der Schnittstelle zwischen erstem und letztem Scan nach Scanmatching mit ICP (oben) und LUM (unten) beim Brückendatensatz.

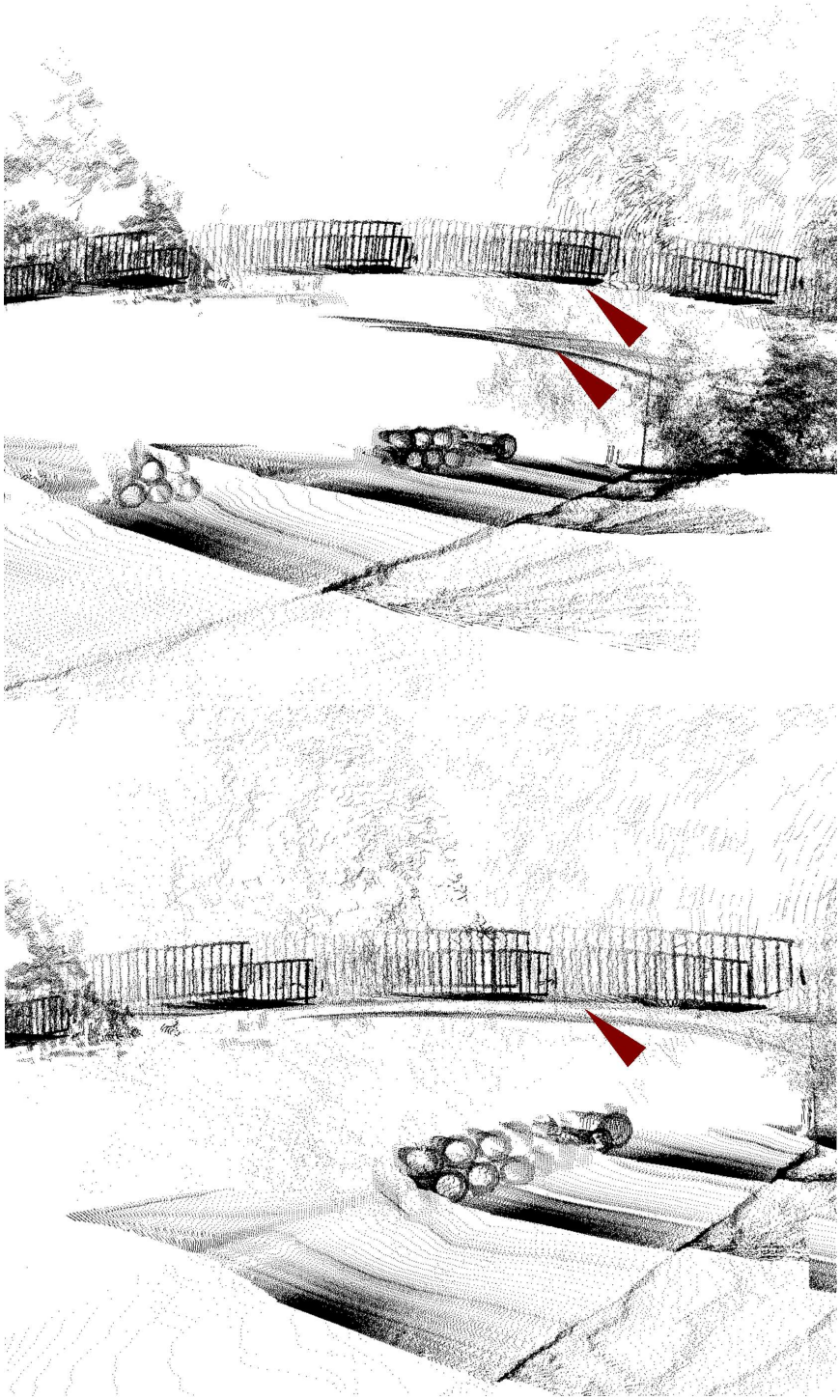


Abbildung 6.13: Seitenansicht der Brücke nach Scanmatching mit ICP (oben) und LUM (unten). Man erkennt beim ICP, dass die Brücke in zwei Ebenen aufgespalten ist, während die Teile beim LUM-Algorithmus näher aneinander liegen (rote Pfeile).

Tabelle 6.1: Laufzeit der Algorithmen mit 20 Iterationen bei unterschiedlicher Anzahl an Scans.

Anzahl der Scans	Zeit ICP in s	Zeit LUM in s	Zeit ICP-Metascan in s
5	16,045	29,205	21,608
10	28,925	56,233	42,474
15	46,766	84,798	70,817
20	70,844	115,849	109,056
25	82,681	142,331	137,792
30	96,559	169,218	173,953
35	110,848	199,042	215,086
40	125,502	225,465	265,465
45	139,306	231,190	319,150

Die während der Fahrt aufgenommenen Laserscans wurden per Hand angeordnet, da wegen der großen Höhenunterschiede die Odometriedaten als Anfangsschätzung nicht verwertbar waren. Anschließend wurden sie sowohl mit ICP als auch mit unserer Implementation verarbeitet. Beim ICP wurden die aufeinanderfolgenden Scans paarweise mit maximal 50 Iterationen gematcht. Für den Lu und Milios-Algorithmus wurde ein Netz erstellt, bei dem außer den direkt aufeinanderfolgenden Scans die Scans der Brücke mit denen der Straße unter der Brücke verbunden wurden.

Die Ergebnisse lassen sich mit den Abbildungen 6.11 – 6.13 demonstrieren. Abbildung 6.11 zeigt den Blick auf die komplette erstellte Karte aus der Vogelperspektive. In beiden Ansichten erkennt man, dass die Scans lokal gut zusammengefügt sind. Dies zeigt sich zum Beispiel dadurch, dass die Brücke gerade ist. Jedoch fehlt beim Ergebnis des ICP der Zusammenhang zwischen dem ersten und letzten Scan (rote Markierung), was auch dadurch bestätigt wird, dass zwischen diesen beiden Scans keine Punktpaare gefunden werden können. Dies ist beim Verfahren von Lu und Milios anders. Dort liegen der Anfang und das Ende deutlich näher beieinander. Auch hier existiert ein kleiner Fehler, der aber deutlich geringer ist als beim ICP. Unterhalb der Straße am Anfang des Pfades sieht man den Teil eines Baumes zweimal. Noch deutlicher sind die Unterschiede in Abbildung 6.12 sichtbar. Dort sieht man, dass beim ICP das Stück Weg vom Ende weit entfernt vom Anfang liegt, während beim LUM die Straße zusammenhängt.

Eine Betrachtung der Seitenansicht der Brücke 6.13 verdeutlicht den Vorteil der Netzstruktur. Das Bild des ICP erweckt den Eindruck es gäbe zwei Brücken (rote Pfeile), während bei der Implementation nach Lu und Milios von der Straße aus gemessenen Teile an den bereits vorhandenen Teil der Brücke angefügt werden.

6.2.3 Laufzeit

Tabelle 6.1 vergleicht die Laufzeit von unseres Algorithmus mit der von ICP und der von ICP mit Metascan-Matching bei einer einfachen Schleife. Bei letzterem Verfahren wird der kd-tree

nicht nur aus einem Referenzscan sondern aus allen bereits registrierten Scans gebildet. In den bereits vorgestellten Ergebnissen haben wir gezeigt, dass paarweiser ICP die Schwierigkeiten einer zyklischen Umgebungsexploration nicht bewältigen kann. Dies gilt insbesondere bei einer großen Anzahl zu registrierender Scans, da der Fehler mit jedem zusätzlichen Scan größer wird. An der Tabelle erkennt man, dass der ICP deutlich schneller ist als der LUM-Algorithmus und dieser Unterschied bei zunehmender Anzahl von Scans steigt. Unter Berücksichtigung des Ergebnisses ist dieser Unterschied aber als geringer zu bewerten.

Vergleicht man nun die Laufzeit mit der des Metascan-Matchings, erkennt man, dass unser Verfahren bereits bei 30 Laserscans schneller ist und sich der Vorteil noch vergrößert, da hier die Laufzeit in diesem Bereich annähernd linear ansteigt, während sie beim Meta-Scanmatching quadratisch wächst. Wenn man nun berücksichtigt, dass das Metascan-Matching auch nur begrenzt für Schleifen einsetzbar ist, da Posen bereits registrierter Scans nicht nachträglich verändert werden können und somit Inkonsistenzen nicht ausgeschlossen sind, wird deutlich, dass unser Verfahren auch in diesem Vergleich deutlich besser abschneidet.

6.2.4 Konvergenz

Am Beispiel des Datensatzes aus dem zweiten Stock des Universitätsgebäudes lässt sich das Verhalten des Algorithmus gut demonstrieren. Abbildung 6.5 zeigt eine Aufsicht auf den Flur. Die Folge von Bildern repräsentiert die Iterationen unseres global konsistenten Scanmatching-algorithmus. Mit zunehmenden Iterationen verbessert sich die Verteilung des Fehler optisch deutlich, bis letztendlich das Ende der Roboterfahrt auf den Anfang gezogen wird. Ähnliches beobachtet man auch für den Rotationsfehler in Abbildung 6.15. Dieser nimmt zwar in den ersten Iterationen zu. Danach nimmt der Fehler zunächst stark und dann geringer ab.

Nach 50 Iterationen liegen beide Fehler nahe Null. Diese und weitere Experimente zeigen, dass der Algorithmus nach kurzer Zeit zu einem guten Ergebnis konvergiert. Folglich lässt sich ähnlich dem ICP auch eine Abbruchbedingung einführen, durch die die Berechnung bei zu geringen Änderungen abgebrochen wird. Die Schwierigkeit liegt darin, einen geeigneten Schwellwert für einen solchen Abbruch zu finden. Hierbei ist zu berücksichtigen, dass in jedem Iterationsschritt jeder einzelne Scan angepasst wird. Der Abbruch eines Programms kann also dann erzwungen werden, wenn die Summe der Änderungen aller Scans ausreichend klein ist, oder aber wenn die maximale Änderung eines einzelnen Scans einen Schwellwert unterschreitet. Wir verzichten in unserer Implementation auf einen automatisierten Abbruch, da die Anzahl der Iterationen für alle Scans gleich ist und man somit manuell einstellen kann, wie genau die Karte werden soll.

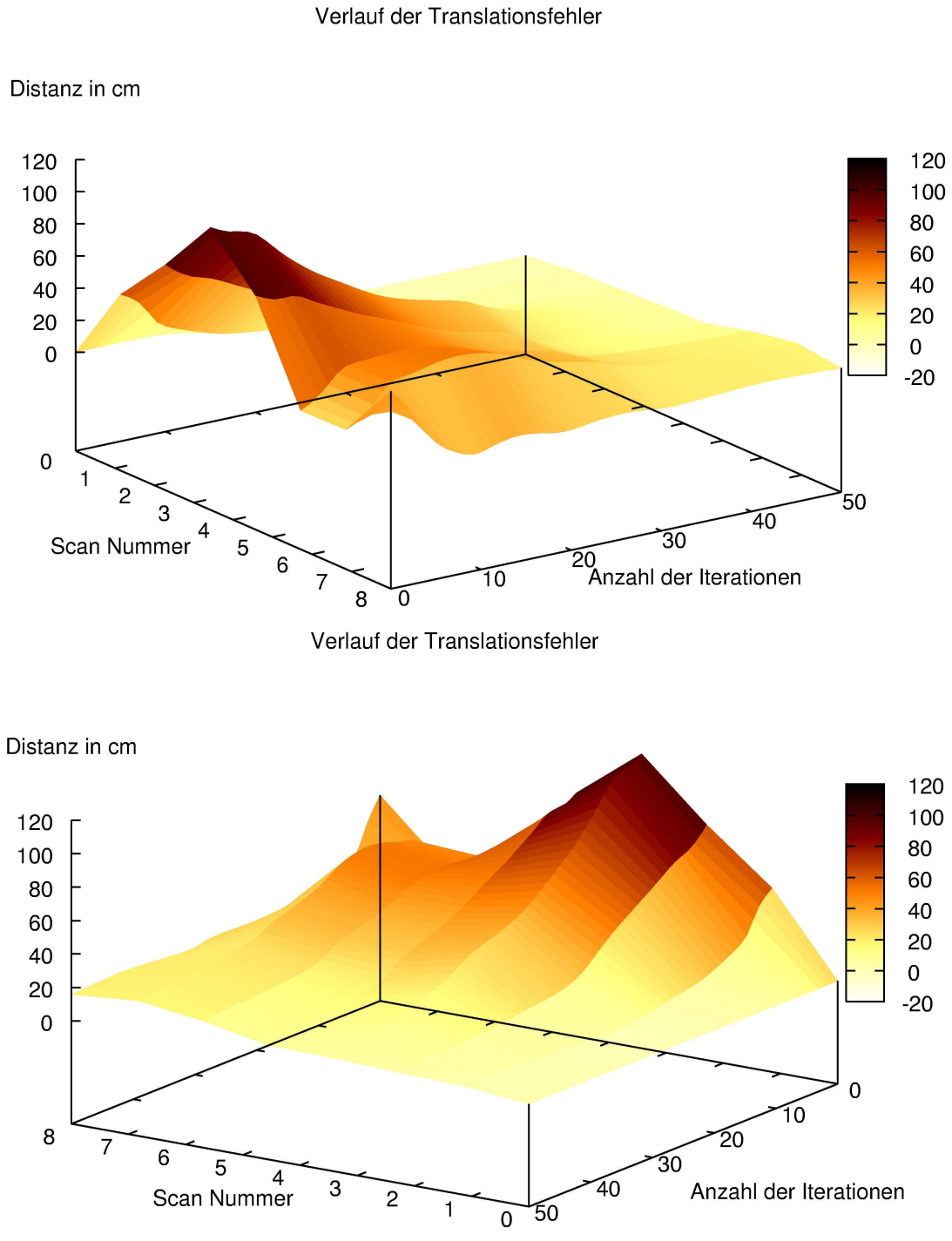


Abbildung 6.14: Fehlerverlauf der Translation bei 50 Iterationen aus zwei verschiedenen Ansichten beim LUM-Scanmatching des Datensatzes aus der Laborumgebung im 5. Stock des AVZ-Gebäudes der Universität Osnabrück.

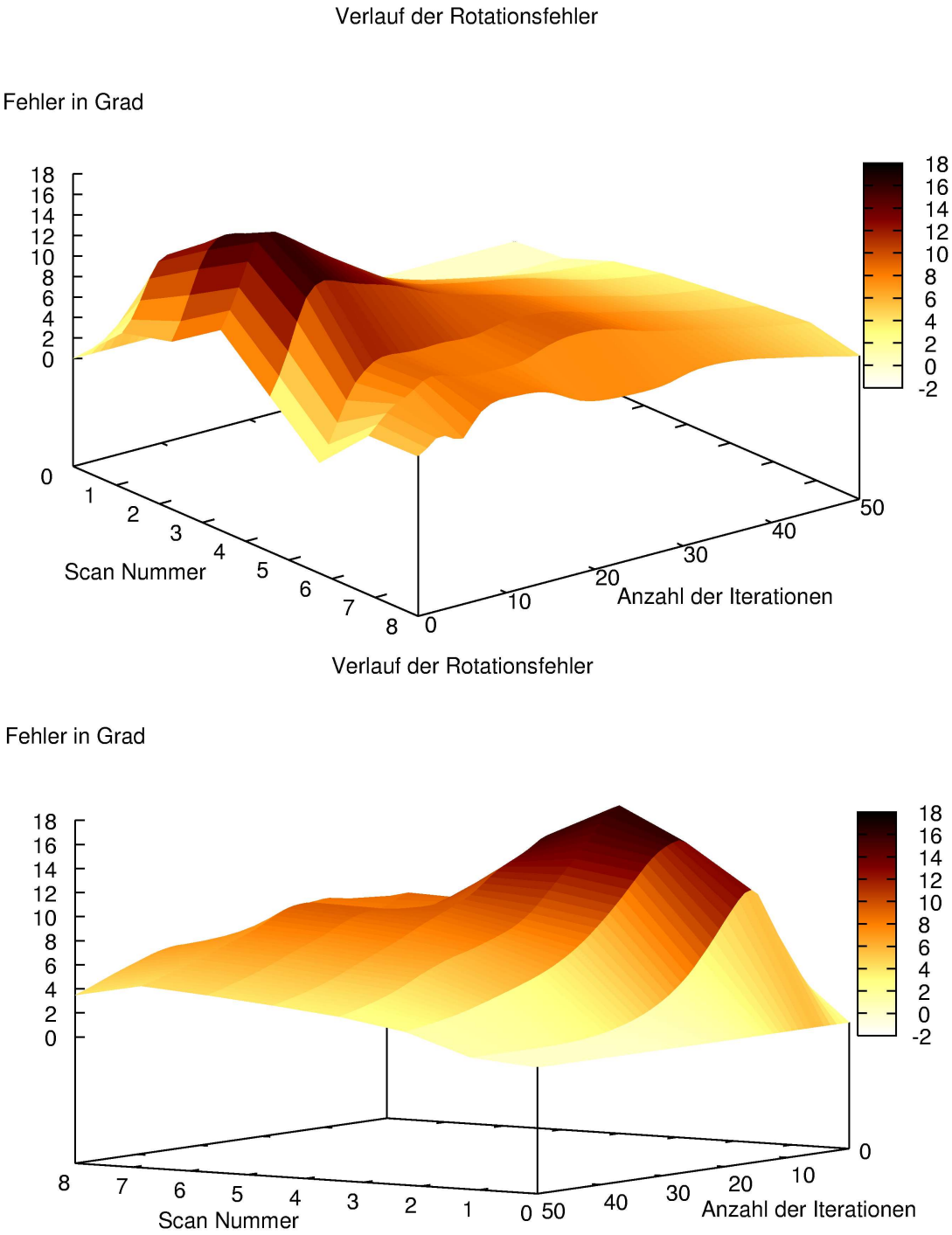


Abbildung 6.15: Fehlerverlauf der Rotation bei 50 Iterationen aus zwei verschiedenen Ansichten beim LUM-Scanmatching des Datensatzes aus der Laborumgebung im 5. Stock des AVZ-Gebäudes der Universität Osnabrück.

Kapitel 7

Schlussfolgerungen und Ausblick

Eins der wichtigsten Forschungsgebiete in der Robotik ist die Erstellung von Karten. Dies gilt sowohl für autonome Roboter, die eine Umgebung befahren sollen um dort Aufgaben zu erledigen als auch für manuell gesteuerte Roboter, die eingesetzt werden um unbekannte Umgebungen zu erkunden, was beispielsweise bei Rettungsrobotern relevant sein kann.

Dazu werden in der Forschung gewöhnlich Laserscanner eingesetzt, die Punktwolken aufzeichnen, aus denen sich eine Karte eines kleinen Teils der Umgebung erstellen lässt. Will man eine größere Umgebung darstellen, müssen diese lokalen Karten zu einer globalen Karte zusammengefügt werden. Gängige Verfahren minimieren die Distanz zwischen Punktpaaren, die aus zwei korrespondierenden Laserscans gesucht werden. Da sowohl die Laserscans selbst, als auch die Scanmatching-Verfahren fehlerbehaftet sind, führt ein sequentielles paarweises Scanmatching dazu, dass sich Fehler aufsummieren und so aus vielen lokalen Fehlern ein großer globaler Fehler entsteht. Dies führt zu Problemen, wenn die einzelnen Laserscans sich mit mehreren anderen Scans überlappen, insbesondere wenn die Scans in der Reihenfolge der Registrierung weit auseinander liegen.

Das in dieser Arbeit präsentierte Verfahren stellt eine Möglichkeit dar, um Laserscans global konsistent zu registrieren. Hierbei wird zunächst ein Netz aus Relationen zwischen den einzelnen Scans und dann ein lineares Gleichungssystem aus den Distanzwerten gebildet. Die Lösung des Gleichungssystems optimiert die Posen der Laserscans durch Minimierung der Distanzen. Da hierbei alle Posen gleichzeitig bearbeitet werden, wird ein Weiterreichen der Fehler von einem Scan auf die Folgenden verhindert.

Auf diese Weise produziert der Algorithmus bei guter Anfangsschätzung keine groben Fehler, wie es beim ICP vorkommt. Eine zusammenhängende Karte ist auch nach der Bearbeitung noch zusammenhängend. Im schlimmsten Fall führt das Programm zu keinen oder nur kleinen Änderungen. Probleme bestehen, wenn aufgrund der Anfangsschätzung keine oder nur sehr geringe Überlappungen zwischen den einzelnen Scans vorliegen. Dann wird die Verbindung mit einer sehr großen Unsicherheit belegt und die Änderungen bleiben sehr gering. Besonders zu beachten sind auch die Winkelfehler. Obwohl die Winkelberechnung in drei Dimensionen weit komplexer ist als in zwei Dimensionen, treten hier keine außergewöhnlichen Fehler auf.

Was die Laufzeit betrifft, besteht noch Verbesserungsbedarf. Im Vergleich zu sequentiellen Verfahren ist der Algorithmus inhärent langsamer. Zwar schneidet er besser ab als andere einfache globale Verfahren, wie zum Beispiel ICP-Metascan-Matching, benötigt aber dennoch viel Zeit, um zu ausreichenden Ergebnissen zu kommen. Hier bestehen durchaus noch Verbesserungsmöglichkeiten. So kann man die Dimension des Gleichungssystems verringern, indem man Verbindungen, deren Distanz ausreichend klein geworden ist, einfriert und nicht weiter verändert. Des Weiteren ist die Suche nach den Punktpaaren sehr aufwendig. Um diese zu beschleunigen lassen sich die in Kapitel 2.1.1 für den ICP vorgeschlagenen Methoden verwenden, wie Reduktion von Punktpaaren. Weitere Verbesserungsmöglichkeiten liegen im Aufstellen der Kovarianzmatrizen für die Verbindungen. So können die Matrizen durch vorherige Iterationen bestimmt werden, oder aber unterschiedlich gewichtet werden. So ist es möglich, die Verbindungen von expliziten Schleifen stärker zu bewerten, um die Änderungen in diesen Bereichen zu vergrößern.

Weitere Verbesserungsmöglichkeiten liegen in der Anwendung eines automatisierten Schleifenfindungsalgorithmus, der ein gutes Netz erstellt, indem genau die wichtigen Links gefunden werden. So sollten von mehreren aufeinanderfolgende Scans nicht unbedingt alle paarweise verbunden sein. Echte Schleifen sollten jedoch auf jeden Fall in dem Graphen wieder zu finden sein. Vorteilhaft wäre auch ein genaueres Verfahren für die Poseschätzung. Besonders bei Roboterfahrten in unebenen Umgebungen sind die Schätzungen durch Odometrie nicht ausreichend und erfolgreiches Matching erfordert eine Vorverarbeitung der Scans.

Literaturverzeichnis

[1] Sven Albrecht, Joachim Hertzberg, Kai Lingemann, Andreas Nüchter, Jochen Sprickerhof, and Stefan Stiene. Device Level Simulation of Kurt3D Rescue Robots. In *Third International Workshop on Synthetic Simulation and Robotics to Mitigate Earthquake Disaster (SRMED 2006)*, CDROM Proceedings, June 2006.

[2] Jon Louis Bentley. Multidimensional Binary Search Trees Used for Associative Searching. *Communications of the ACM*, 18(9):509–517, 1975.

[3] P. Besl and N. McKay. A method for Registration of 3–D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239 – 256, February 1992.

[4] I. N. Bronstein. *Taschenbuch der Mathematik*. Verlag Harri Deutsch, Frankfurt am Main, Thun, 3rd edition, 1997.

[5] Yang Chen and Gérard Medioni. Object Modelling by Registration of Multiple Range Images. *Image Vision Comput.*, 10(3):145–155, 1992.

[6] D. Chetverikov, D. Svirko, D. Stepanov, and Pavel Krsek. The Trimmed Iterative Closest Point Algorithm. *16th International Conference on Pattern Recognition (ICPR'02)*, 03:545–548, 2002.

[7] Ingemar J. Cox. Blanche – An Experiment in Guidance and Navigation of an Autonomous Robot Vehicle. *IEEE Transactions on Robotics and Automation*, 7(2), April 1991.

[8] Erik B. Dam, Martin Koch, and Martin Lillholm. Quaternions, Interpolation and Animation. Technical Report DIKU–TR–98/5, University of Copenhagen, July 1998.

[9] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Trans. Math. Softw.*, 3(3):209–226, 1977.

[10] Gamebots. `http://www.planetunreal.com/gamebots`.

[11] Natasha Gelfand, Leslie Ikemoto, Szymon Rusinkiewicz, and Marc Levoy. Geometrically Stable Sampling for the ICP Algorithm. In *Proc. International Conference on 3D Digital Imaging and Modeling*, Canada, October 2003.

[12] J.-S. Gutmann and K. Konolige. Incremental Mapping of Large Cyclic Environments. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, 2000.

[13] Dirk Hähnel, Wolfram Burgard, and Sebastian Thrun. Learning Compact 3D Models of Indoor and Outdoor Environments with a Mobile Robot. *Robotics and Autonomous Systems*, 44(2003):15–27, 2003.

[14] Sir William R. Hamilton. On new Species of Imaginary Quantities connected with a theory of Quaternions. In *Proceedings of the Royal Irish Academy*, November 1843.

[15] Kurt3d. `http://www.ais.fraunhofer.de/tp/kurt3d.html`, 2005.

[16] K. Lingemann. Schnelles Pose-Tracking auf Laserscan-Daten für autonome mobile Roboter. Master's thesis, Universität Bonn, July 2003.

[17] A. Lorusso, D. Eggert, and R. Fisher. A Comparison of Four Algorithms for Estimating 3-D Rigid Transformations. In *Proceedings of the British Machine Vision Conference (BMVC 95), pages 237 - 246*, Birmingham, England, September 1995.

[18] F. Lu and E. Milios. Globally Consistent Range Scan Alignment for Environment Mapping. *Autonomous Robots*, 4:333 – 349, April 1997.

[19] Paul Newman, Michael Bosse, and John Leonoard Seth Teller. Simultaneous Localization and Map Building in Large-Scale Cyclic Environments Using the Atlas Framework. *The International Journal of Robotics Research*, 23(12):1113–1139, December 2004.

[20] A. Nüchter, K. Lingemann, and J. Hertzberg. Kurt3D - A Mobile Robot for 3D Mapping of Environments. In *ELROB Technical Paper*, Univerität Osnabrück, January 2006.

[21] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. Heuristics-Based Laser Scan Matching for Outdoor 6D Slam. In *KI 2005: Advances in Artificial Intelligence. 28th Annual German Conference on AI, Proceedings Springer LNAI vol. 3698, pages 304 - 319*, Koblenz, Germany, September 2005.

[22] Kari Pulli. Multiview Registration for Large Data Sets. In *Second International Conference on 3-D Imaging and Modeling (3DIM 1999)*, Los Alamitos, CA, USA, 1999. IEEE Computer Society.

[23] T. Röfer. Building Consistent Laser Scan Maps. In *Proceedings of the 4th European Workshop on Advanced Mobile Robots (EUROBOT 01)*, volume 86, pages 83–90, 2001.

[24] P.J. Rousseeuw and B.C. van Zomeren. Unmasking Multivariate Outliers and Leverage Points. *Journal of the American Statistical Association*, 85:633–651, 1990.

[25] S. Rusinkiewicz and M. Levoy. Efficient Variants of the ICP Algorithm. In *Proceedings of the Third International Conference on 3D Digital Imaging and Modelling (3DIM '01)*, Quebec City, Canada, May 2001.

[26] Szymon Rusinkiewicz. 3D Scan Matching and Registration, 2005. ICCV 2005 Short Course.

[27] S.J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2nd edition, 2003.

[28] G. Shaffer, A. Stentz, W. L. Whittaker, and K. Fitzpatrick. Position Estimator for Underground Mine Equipment. *Journal of the IEEE Transactions on Industry Applications (TIA 92)*, 28(5):1131–1140, September 1992.

[29] SICK Laser Range Finder. `http://www.sick.de`.

[30] Ioannis Stamos and Marius Leordeanu. Automated Feature-Based Range Registration of Urban Scenes of Large Scale. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 555 – 561, June 2003.

[31] H. Surmann, K. Lingemann, A. Nüchter, and J. Hertzberg. 6D SLAM - Preliminary Report on Closing the Loop in Six Dimensions.

[32] H. Surmann, K. Lingemann, A. Nüchter, and J. Hertzberg. Aufbau eines 3D-Laserscanners für Autonome Mobile Roboter. GMD Report 126, GMD - Forschungszentrum Informationstechnik GmBH, Sankt Augustin, March 2001, March 2001.

[33] H. Surmann, A. Nüchter, and J. Hertzberg. An Autonomous Mobile Robot with a 3D Laser Range Finder for 3D Exploration and Digitalization of Indoor Environments. *Robotics and Autonomous Systems*, 45(3 – 4):181 – 198, Decembre 2003.

[34] Sebastian Thrun. Robotic Mapping: A Survey. *CMU-CS-02-111*, February 2002. School of Computer Science - Carnegie Mellon University. Pittsburgh, PA 15213.

[35] J. Wang. USARSim - A Game-based Simulation of the NIST Reference Arenas. `http://usl.sis.pitt.edu/ulab/usarsim_download_page.htm` and `http://sourceforge.net/projects/usarsim`, 2005 – 2006.

[36] Eric W. Weisstein. Rotation Formula. MathWorld – A Wolfram Web Resource, `http://mathworld.wolfram.com/RotationFormula.html`, 1999.

[37] G. Weiß, C. Wetzler, and E. von Puttkammer. Keeping Track of Position and Orientation of Moving Indoor Systems by Correlation of Range-Finder Scans. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 94)*, pages 595–601, Munich, Germany, 1994.

[38] R. Worst. KURT2: A Mobile Platform for Research in Robotics. In *Proceedings of the 2nd International Symposium on Autonomous Minirobots for Research and Edutainment (AMIRE '03), pages 3 – 12*, 2003.

[39] Michael Wünstel and Thomas Röfer. Feature Based Registration of Range Images in Domestic Environments. In *Computer Vision and Graphics, ICCVG 2004*, pages 648–654, September 2004.

Die einzelnen Kapitel wurden bearbeitet von:

Einleitung	Dorit Borrmann und Jan Elseberg
Scanmatching	Dorit Borrmann
3D-Transformationen	Jan Elseberg
Das Optimierungsproblem	Jan Elseberg
Ableitung der Posedifferenzen und Kovarianzen	Jan Elseberg
Der Algorithmus von Lu und Milios	Dorit Borrmann
Schlussfolgerungen und Ausblick	Dorit Borrmann

Erklärung

Hiermit versichern wir, dass wir die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht haben.

Osnabrück, 06. Oktober 2006