UNIVERSITÄT KOBLENZ · LANDAU

Fachbereich 4: Informatik

Institut für Informatik

3D-Scan-Registrierung auf Basis der Helixtranslation

Diplomarbeit

zur Erlangung des Grades eines/r

Diplom-Informatikers / Diplom-Informatikerin
im Studiengang Computervisualistik

vorgelegt von

Peter Schneider

Betreuer:	Prof. Dr.-Ing. Dietrich Paulus, Institut für Computervisualistik, Fachbereich Informatik
Erstgutachter:	Prof. Dr.-Ing. Dietrich Paulus, Institut für Computervisualistik, Fachbereich Informatik
Zweitgutachter:	Dr. Andreas Nüchter, Universität Osnabrück

Koblenz, im September 2008

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Die Richtlinien der Arbeitsgruppe für Studien- und Diplomarbeiten habe ich gelesen und anerkannt, insbesondere die Regelung des Nutzungsrechts

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden. ja ☐ nein ☐

Der Veröffentlichung dieser Arbeit im Internet stimme ich zu. ja ☐ nein ☐

Koblenz, den ..... Unterschrift

Zusammenfassung

Das Problem des Schleifenschlusses ist ein bekanntes Hindernis bei Simultanious Localising and Mapping Verfahren, um zusammenhängende Karten zu erstellen. Es beschreibt den Fehler, der entsteht, wenn man beim Registrieren von neuen Daten wieder in Bereiche der Karte gelangt, in denen vorher bereits Daten registriert wurden. Dies geschieht zum Beispiel durch das Abfahren eines Kreises oder einer Schleife. Hier kann es, auf Grund von aufsummierten minimalen Fehlern, dazu kommen, dass ein korrektes Matchen der neuen Daten auf alle bereits vorhandenen Daten nicht mehr möglich ist. In solchen Fällen muss eine globale Korrektur angewendet werden, die den Schleifenschlussfehler behebt. Dies soll in dieser Arbeit für das Registrieren von dreidimensionalen Scans geschehen. Die Korrektur muss hierbei nicht nur die Verschiebung in den drei Dimensionen beachten, sondern auch die Rotation um diese drei Achsen. Sie muss also in sechs Freiheitsgraden geschehen. Ein bekanntes Korrekturverfahren hierfür ist der Iterative Closest Point Algorithmus. Basierend auf diesem soll hier eine neuartig erweiterte Korrekturberechnung eingeführt werden, die einerseits zur lokalen Korrektur eingesetzt werden kann, aber eben auch global den Fehler des Schleifenschlusses behebt. Die Ergebnisse dieses Verfahrens werden abschließend mit dem Referenzverfahren für sechdimensionales SLAM nach Nüchter et al, weleches in [NBE+08] beschrieben ist, verglichen und bewertet.

Inhaltsverzeichnis

Verzeichnis der Bilder

Verzeichnis der Tabellen

Kapitel 1

Einleitung

Wie kann man es autonom fahrenden Robotern ermöglichen, in unserer komplexen Welt einen für sie befahrbaren Weg von Punkt A zu Punkt B zu finden? Grundsätzlich verwenden Roboter zur Orientierung in ihrer Umgebung Karten, die ihr Umfeld auf geeignete Art und Weise darstellen. Hauptsächlich werden hierbei zweidimensionale Karten verwendet. Eine solche zweidimensionale Karten bildet die Umgebung anhand einer horizontal durch den Raum verlaufenden Schnittebene ab. Ein Beispiel für eine solche Karte ist auf Bild 1.1 links zu sehen. Diese wurde komplett selbstständig vom Roboter während der Durchquerung der Arena erstellt und dabei auch zur Navigation verwendet. Hierfür wurden verschiedene Simultanious-Localising-and-Mapping (kurz SLAM) Algorithmen verwendet, die in [Wir07] beschrieben sind. In einfachen Umgebungen lassen sich solche Karten auch problemlos verwenden. Während der Generierung dieser zweidimensionalen Karte wurde ebenfalls eine dreidimensionale Karte erstellt. Diese ist in Bild 1.1 auf der rechten Seite zu sehen und zeigt erste Unterschiede und Probleme. Dort ist die Komplexität der Arena schon ein wenig besser zu erkennen. Sie besteht unter anderem aus Schrägen und Stufenfeldern. Somit wird ein Problem der 2D-Karten deutlich, denn was passiert, wenn Hindernisse auftauchen, die nicht in der Schnittebene der Karte liegen, oder wenn die Umgebung verschiedenen Höhenlevel hat, die befahren werden können. Hier stößt man an die Grenzen zweidimensionaler Karten. Das Problem der eingeschränkten Hinderniserkennung ist in Bild 1.2 genauer zu erkennen. Dort sind aus der gleichen Position die Kamerasicht, die

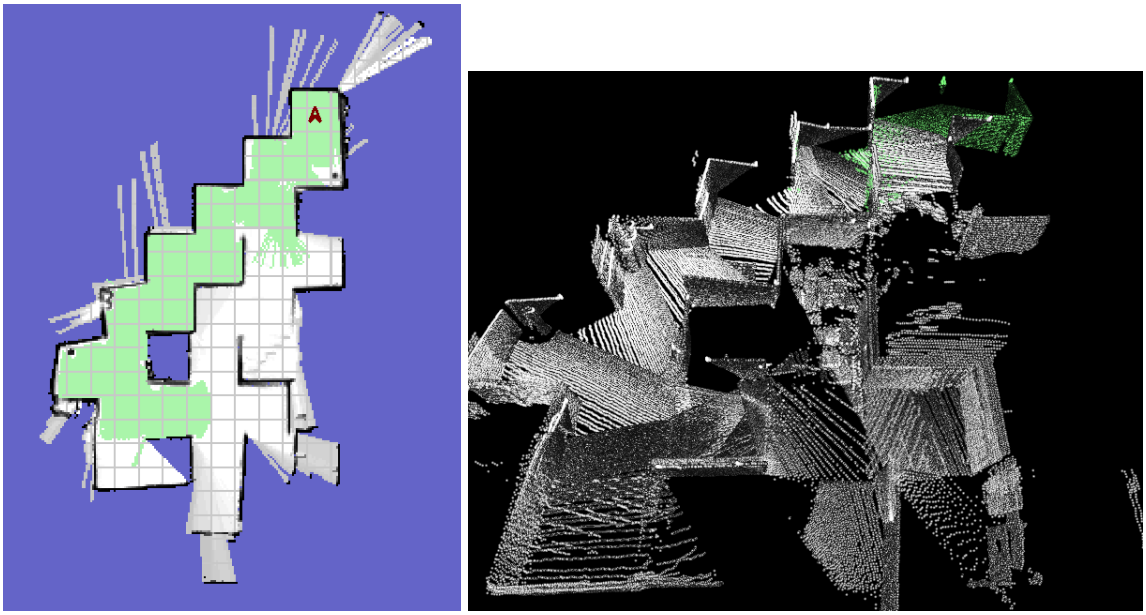

Bild 1.1: Beispiele für 2D-Karten (links) und äquivalente 3D-Karten (rechts)

zweidimensionale Kartierung sowie ein dreidimensionaler Scan zu sehen. Auf dieser 3D-Ansicht ist deutlich das Hindernis zu erkennen, wohingegen in der 2D-Karte das Areal an gleicher Position als frei markiert wurde.

Somit könnten in komplexen Umgebungen dreidimensionale Karten eine komplette Hindernisvermeidung ermöglichen, da bei ihrer Generierung die komplette Umgebung z. B. mit Hilfe eines 3D-Laserscanners abgetastet wird. Als Beispiel für einen solch großen 3D-Scan soll eine dreidimensionale Karte der Universität Osnabrück dienen, von der ein Ausschnitt Bild 1.3 zeigt. Die Scans, die zur Erstellung einer Karte dieser Dimension nötig sind, haben natürlich nur eine gewisse Reichweite und es können Verschattungen entstehen. Um also eine größere Umgebung vollständig zu kartieren, müssen in gewissen Abständen einzelne lokale 3D-Scans geschossen werden. Diese müssen dann zu einer solchen globalen Aufnahme zusammengefügt werden. Dieses sogenannte Registrieren lässt sich mit Hilfe des Iterative Closest Point (ICP) Algorithmus automatisieren. Der ICP-Algorithmus wird im einfachen Fall nur zur lokalen Korrektur angewendet, das heißt, zur Registrierung eines Scans auf den direkt vorhergehenden Scan. Da der ICP-Algorithmus

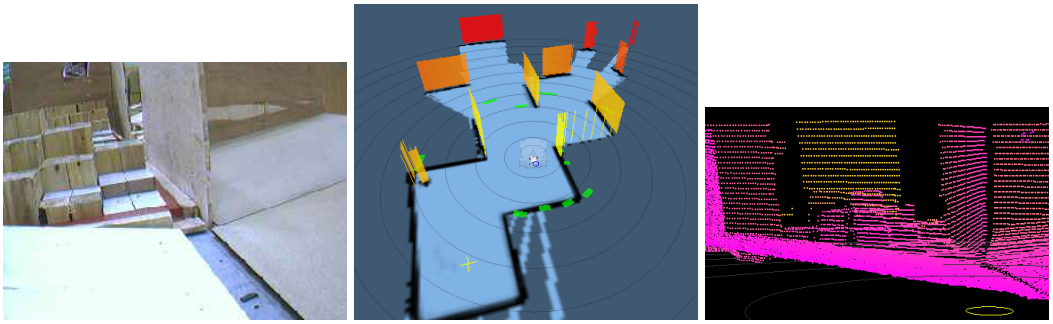

Bild 1.2: Vergleich der Kartierungsarten, links Kamerasicht, Mitte 2D-Karte, Rechts 3D-Sicht

nie absolut korrekt arbeitet, können sich hierbei allerdings Fehler aufsummieren. Dies kann dann dazu führen, dass, z. B. im Falle eines Schleifenschlusses, die beiden Enden der Schleife nicht mehr aufeinander passen. Ein Beispiel für einen solchen Fehler ist in Bild 1.4 zu sehen. Solch fehlerbehaftete Karten sind für eine automatische Pfadplanung des Roboters nicht geeignet. An genau diesen Stellen könnte man keine korrekte Aussage mehr über die Befahrbarkeit des Areals treffen. An dieser Stelle setzt die globale Korrektur ein. Sie minimiert den Abstand zwischen den beiden Enden der Schleife und betrachtet dabei alle Scans, die zwischen Startscan und Endscan der Schleife liegen. Das Standardverfahren hierfür wird von F. Lu und E. Milios in [LM97] beschrieben und ihm Rahmen der Arbeit von Nüchter et al [NBE+08] für den sechdimensionalen Fall erweitert.

Die hier vorgelegte Arbeit basiert auf dem von Nüchter entwickelten und implementierten Framework, welches zusammen mit dem Standardverfahren im nächsten Kapitel kurz beschrieben wird. Darauf folgt die Erläuterung des Algorithmus von Pottmann, welcher in [PH03] und [PLH02] beschrieben wird, sowie die im Rahmen dieser Arbeit entwickelte Erweiterung dieser Korrekturberechnung. Diese Erweiterung führt in Pottmanns Berechnung eine global optimierende Korrektur der Scanserie ein. Anschließend werden diese neuen Versionen mit der bereits im Framework integrierten in Bezug auf Qualität der Ergebnisse und der benötigten Laufzeit verglichen. Abschließend folgen eine Zusammenfassung sowie ein Ausblick auf weitere Beschleunigungsmöglichkeiten dieser Art von Fehlerminimierung.

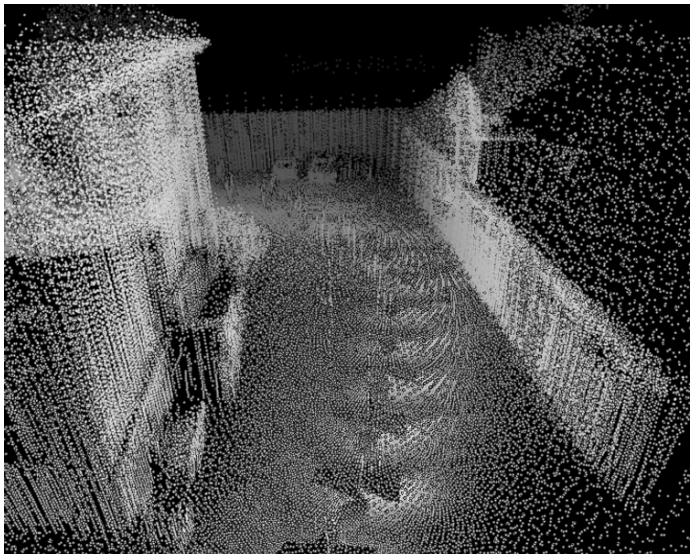

Bild 1.3: Dreidimensionaler Scan eines Außenareals

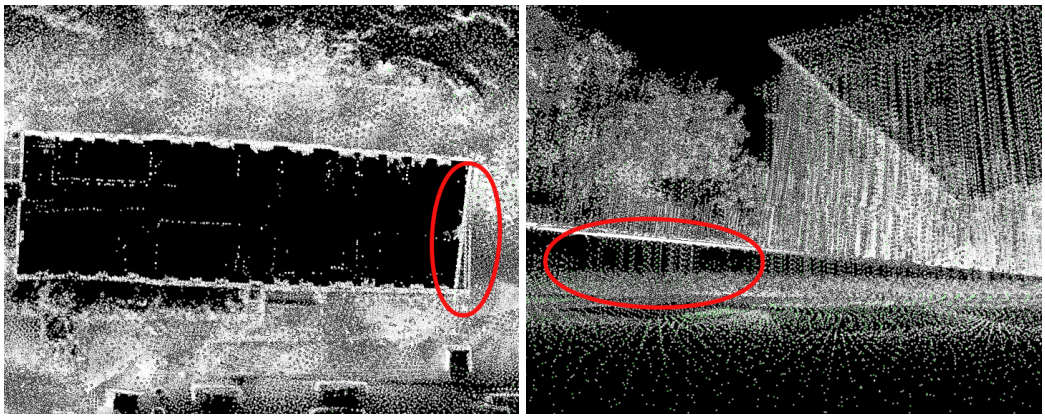

Bild 1.4: Fehler beim Schließen der Schleife, links 2D-Projektion, rechts gleiche Stelle in 3D-Ansicht (Fehler markiert)

Kapitel 2

Stand der Wissenschaft

Dieses Kapitel soll einen kurzen Überblick über die Grundlagen des Iterative-Closest-Point-Algorithmus geben. Außerdem wird das Standardverfahren von Lu und Milios zur Lösung des globalen Korrekturproblems in Grundzügen erläutert. Beispiele werden Anhand der Implementation von Nüchter gezeigt.

2.1 Der lokale ICP-Algorithmus

Beim Erstellen konsistent zusammenhängender 3D-Karten besteht das erste Problem darin, jeweils einen Scan an den Vorgängerscan beziehungsweise an die bisher erstellte 3D-Karte anzufügen. Eine grobe Positionierung des neu zu registrierenden Scans kann dabei an Hand der Odometriedaten des Roboters erfolgen. Diese sind aber oftmals fehlerhaft, beispielsweise auf Grund durchdrehender Räder. Auch ist es dem Roboter nicht möglich, diese Daten in allen sechs möglichen Freiheitsgraden zu erheben. So sind aus den Odometriedaten meist nur die Verschiebungen des Roboters entlang der X- und Z-Achse relativ zur Blickrichtung seiner Ausgangsposition sowie die Rotation um die der Ausgangsposition entsprechenden Hochachse des Roboters zu ermitteln. Eventuell aufgetretene Neigungen des Roboters sowie Höhenunterschiede der Aufnahmepositionen durch möglicherweise zurückgelegte Steigungen, sind in diesen Daten selten vorhanden. Die dadurch auftretende falsche Positionierung versucht der lokale ICP-Algorithmus zu korrigieren. Hierbei

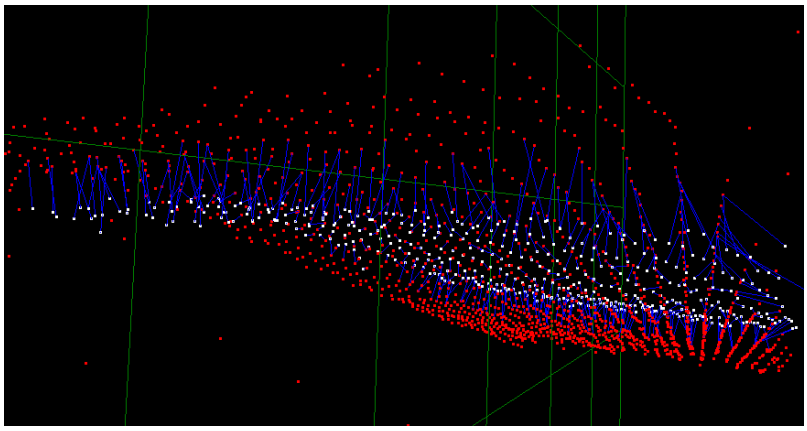

Bild 2.1: Korrespondenzen (blaue Linien) zwischen Punkten zweier Scans (Rot und Weiß)(Quelle: [Sch07])

wird beim Registrieren zweier Scans für jeden Punkt im einen Scan ein korrespondierender Punkt im anderen Scan gesucht. In Bild 2.1 sind solche Punktkorrespondenzen zweier Scans eingezeichnet worden. Diese werden dort durch die blauen Linien zwischen den weißen Punkten des einen und den jeweils dazu korrespondierenden roten Punkten des anderen Scans angezeigt. Der Abstand zwischen solchen zwei Korrespondenzpunkten darf dabei allerdings einen vorher festzulegenden Grenzwert nicht überschreiten, da sonst falsche Beziehungen gebildet werden. Diese würden dann zu einer Korrekturverschiebung in die falsche Richtungen führen. Hat man die Korrespondenzen gebildet, wird davon ausgegangen, dass diese Punktpaare eigentlich an gleicher Position sein müssten. Das korrekte Registrieren gestaltet sich so zu einer Minimierung der folgenden Summe

$$\min \sum_i \boldsymbol{m}_i - \boldsymbol{d}_i \tag{2.1}$$

Es muss also der Abstand aller gefundenen Punktkorresponenzen zwischen den Punkten $\boldsymbol{m}_i$ der Modellmenge M und den Punkten $\boldsymbol{d}_i$ der Datenmenge D auf ein Minimum gebracht werden. Die Minimierungsfunktion dazu lautet

$$\mathrm{E}(\boldsymbol{R}, \boldsymbol{t}) = \sum_i \|\boldsymbol{m}_i - (\boldsymbol{R}\boldsymbol{d}_i + \boldsymbol{t})\|^2 \tag{2.2}$$

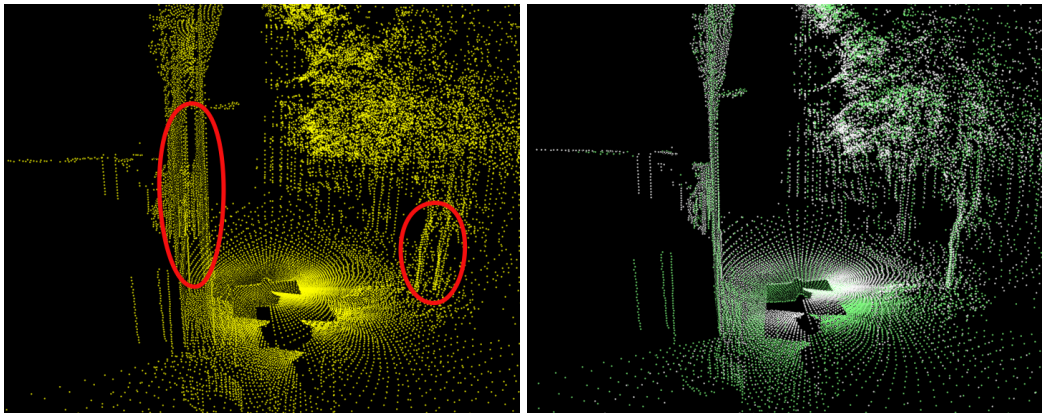

Bild 2.2: Lokale Korrektur zweier Scans mit ICP, links vorher (Fehler markiert), rechts nachher

Da bei dieser Minimierungsfunktion sowohl die Rotation $\boldsymbol{R}$ als auch die Translation $\boldsymbol{t}$ betrachtet wird, lassen sich dadurch Fehler in allen sechs Freiheitsgraden korrigieren. Das Ergebnis einer solchen lokalen Korrektur ist in Bild 2.2 zu sehen. Dieses Verfahren wurde von Arun und Huang in [AHB87] beschrieben. Verschiedene Lösungsmöglichkeiten für diese Funktion, unter anderem mit Hilfe der Singulärwertzerlegung oder Quaternionen, werden von Nüchter in [NÖ06] gezeigt. In Kapitel 3.1 wird ein weiteres lokales Lösungsverfahren erläutert, das Pottmann in [PH03] beschreibt und welches von ihm dort zur Registrierung von 3D-Punktewolken auf CAD-Modelle verwendet wird.

2.2 Die globale Korrektur nach Lu & Milios

Die globale Korrektur wurde erstmal von Lu und Milios in [LM97] für den zweidimensionalen Fall beschrieben. Anfang dieses Jahres wurde dieser Algorithmus von Nüchter et al in [NBE+08] für die Anwendung auf den dreidimensionalen Fall erweitert, wobei er dort globale Fehler in allen sechs Freiheitsgraden korrigieren kann.

2.2.1 Das Grundprinzip

Um eine globale Korrektur durchzuführen werden bei diesem Verfahren die Posen benachbarter Scans zueinander in Beziehung gesetzt. Dies geschieht, nachdem man die Scans zunächst einmal lokal zueinander korrigiert hat. Aus den dann lokal zueinander passenden Scans lässt sich die relative Posenverschiebung eines Scans zu seinem Vorgängerscan ermitteln. All diese Posenbeziehungen werden dann gleichzeitig betrachtet und optimiert. Wenn also die Stelle des Schleifenschlusses eine gewisse Fehlerdistanz zwischen Schleifenbeginn und Schleifenende aufweist, so wird dieser Fehler dort korrigiert und anteilig an die dazwischen liegenden Posen weitergegeben. Dadurch werden auch diese um einen ihrer Position entsprechenden Wert korrigiert. Das Ergebnis ist ein Netz aus Posenbeziehungen, bei dem jede Beziehung jeweils ihren im Netzwerk optimalen Abstand aufweist. Da dies eben auch für die Beziehung des Scans am Schleifenende zum Scan am Schleifenanfang zutrifft, ist für die zugehörige Schleife der entsprechende Fehler am Schleifenschluss korrigiert worden.

2.2.2 Der Ablauf des Algorithmus

Betrachtet man den Ablauf dieses Verfahrens genauer, so kann man es in sieben, sich teilweise wiederholende, Schritte aufteilen.

1. Lokale Korrektur der einzelnen Scans
2. Erstellen des Nachbarschaftsgraphen
3. Punktkorrespondenzen ermitteln
4. Matrixgenerierung
5. Gleichungssystem füllen und lösen
6. Scans verschieben
7. Fehlerkontrolle am Schleifenschluss

Hierbei werden in der Regel die Schritte 2 bis 6 solange wiederholt, bis der Abstand der beiden Scans, die die Schleife schließen, unter einem festzulegenden Toleranzwert liegt. Im Folgenden werden, bis auf den ersten, alle Schritte kurz erläutert. Da dieser schon in Kapitel 2.1 beschrieben wurde, sei hier nur soviel dazu gesagt: Die lokale Korrektur zu Beginn ist theoretisch nicht nötig, erleichtert aber das Detektieren des Schleifenschlusses und die Generierung des Graphen. Wenn auf diesen Punkt verzichtet wird, so werden zur Generierung des Graphen in Schritt 2 nur die Odometriedaten verwendet. Da diese stark fehlerbehaftet sein können, enthält der dadurch erstellte Graph möglicherweise falsche Einträge, die zu einem komplett inkorrektem Ergebnis führen. Im Folgenden werden die einzelnen Teilschritte nun etwas näher betrachtet. Sie werden, abgesehen von der eigentlichen Korrekturberechnung in den Schritten 4 und 5, auch für das Helixverfahren verwendet.

2.2.3 Der Graph und die Korrespondenzen

Das Erstellen eines Nachbarschaftsgraphen beginnt, sobald bei der lokalen Korrektur eine Schleife erkannt wurde. Hierfür werden nun alle Scans, die zwischen Schleifenstart und -ende liegen, berücksichtigt. Ein Beispiel für einen solchen Graphen im zweidimensionalen Fall, ist in Bild 2.3 zu sehen. Diese bilden nun die Knoten des Graphen. Jetzt wird zunächst zwischen den direkt aufeinander folgenden Scans eine Kante gezogen, da diese auf jeden Fall benachbart sind. In Bild 2.3 sind dies die durchgezogenen Linien zwischen den Knotenpunkten. Innerhalb des Graphen existiert also auf jeden Fall zwischen Anfang und Ende eine durchgehende Verbindung, die über alle Scans oder Knoten geht. Nun werden noch Kanten zwischen den Scanknoten gezogen, bei denen sich ein gewisser Mindestanteil der jeweiligen Punktewolken der Scans überlappt. Diese gelten nun ebenfalls als benachbart. Im Bild 2.3 sind dies die gestrichelten Linien. Nun werden für alle Nachbarschaftsbeziehungen die jeweiligen Punktkorrespondenzen gebildet. Da jeder Scan mindestens einen benachbarten Scan besitzt, können auch für jeden Scan Punktkorrespondenzen erstellt werden. Diese Korrespondenzen werden nun in den folgenden Schritten betrachtet.

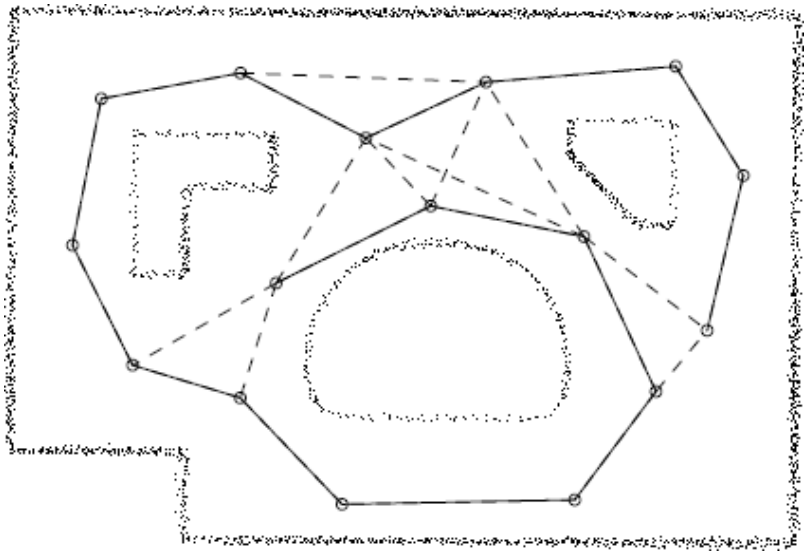

Bild 2.3: Beispiel für einen Graphen im 2D-Fall (Quelle: [LM97])

2.2.4 Die Matrizen und die Gleichung

Im Folgenden soll nur ein grober Überblick über die Herleitung beziehungsweise die Belegung der dort verwendeten einzelnen Matrizen gegeben werden. Dies soll später den Unterschied im Aufwand der Verfahren deutlich machen. Eine komplette Herleitung ist in [NBE+08] zu finde. Wie bereits erwähnt, arbeitet dieses Verfahren mit den Posen der einzelnen Scans sowie deren Abständen zueinander. Der Abstand $\boldsymbol{D}_{jk}$ zweier Scans oder Knoten $\boldsymbol{K}_j$ und $\boldsymbol{K}_k$ ergibt sich aus $\boldsymbol{D}_{jk} = \boldsymbol{K}_j - \boldsymbol{K}_k$. Eine Beobachtung für diesen Abstand wird durch $\bar{\boldsymbol{D}}_{jk} = \boldsymbol{D}_{jk} + \Delta\boldsymbol{D}_{jk}$ modelliert. $\Delta\boldsymbol{D}_{jk}$ ist hierbei eine Gaussverteilung mit Kovarianzmatrix $\boldsymbol{C}_{jk}$. Um nun eine optimale Positionierung der Scans zu ermitteln, wird versucht, eine maximale Ähnlichkeit zwischen $\boldsymbol{D}_{jk}$ und $\bar{\boldsymbol{D}}_{jk}$ zu erreichen. Dies geschieht über die folgende Mahalanobis-Distanz:

$$\boldsymbol{W} = \sum_{(j,k)} (\boldsymbol{D}_{jk} - \bar{\boldsymbol{D}}_{jk})^{\mathrm{T}} {\boldsymbol{C}_{jk}}^{-1} (\boldsymbol{D}_{jk} - \bar{\boldsymbol{D}}_{jk}) \tag{2.3}$$

Dies kann so umgeformt werden, dass sich daraus das Gleichungssystem $\boldsymbol{G}\boldsymbol{x} = \boldsymbol{b}$ ergibt, wobei $\boldsymbol{G}$ eine $6N \times 6N$ Matrix ist. Die entsprechenden Belegungsvorschriften für $\boldsymbol{G}$ und $\boldsymbol{b}$

lauten dann wie folgt:

$$\begin{aligned} \boldsymbol{G}_{j,j} &= \sum_{k=0}^{N} \boldsymbol{C}_{j,k}{}^{-1} \\ \boldsymbol{G}_{j,k} &= \boldsymbol{C}_{j,k}{}^{-1} \text{ mit } (i \neq j) \\ \boldsymbol{b}_j &= \sum_{\substack{k=0 \\ k \neq j}} \boldsymbol{C}_{j,k}{}^{-1} \bar{\boldsymbol{D}}_{j,k} \end{aligned}$$

Die hierfür benötigten Kovarianzmatrizen $\boldsymbol{C}_{j,k}$ und Vektoren $\bar{\boldsymbol{D}}_{j,k}$ ergeben sich aus:

$$\begin{aligned} \boldsymbol{C}_{j,k} &= s^2(\boldsymbol{M}^{\mathrm{T}}\boldsymbol{M}) \\ \boldsymbol{D}_{j,k} &= (\boldsymbol{M}^{\mathrm{T}}\boldsymbol{M})^{-1}\boldsymbol{M}^{\mathrm{T}}\boldsymbol{z} \end{aligned}$$

Hierbei ist die 3×6 Matrix $\boldsymbol{M}$ eine über alle, der jeweils korrespondierenden Punkte, aufsummierte Matrix. Der sechselementige Vektor $\boldsymbol{z}$ verknüpft die Position des jeweiligen Scans mit dem Abstand der jeweils korrespondierenden Punkte. Der Skalar s beinhaltet eine Abschätzung über die in $\boldsymbol{z}$ enthaltenen Fehler. Für deren jeweilige Erzeugung wird ebenfalls die dazu gehörende aufsummierte Matrix $\boldsymbol{M}$ sowie der jeweilige Vektor $\boldsymbol{D}$ benötigt. Das eben dadurch entstandene Gleichungssystem $\boldsymbol{G}\boldsymbol{x} = \boldsymbol{b}$ lässt sich dann mittels der Cholesky-Zerlegung lösen.

2.2.5 Die Verschiebung und die Kontrolle

Nach dem Lösen der Gleichung sind im Vektor $\boldsymbol{x}$ alle Verschiebungsdaten für die einzelnen Scans enthalten. Diese werden nun auf die zugehörigen Einzelscans angewendet. Als Ergebnis der Schritte 2 bis 6 schieben sich nun alle Scans, die zwischen Schleifenanfang und Schleifenende liegen, entsprechend der Korrektur des Schleifenendes, in eine passendere Position. Iteriert man nun diese Schritte, so verschieben sie sich immer weiter Richtung korrektem Schleifenschluss und konvergieren schließlich. Um die Iteration passend abzubrechen, wird hierfür der Abstandsfehler zwischen Schleifenbeginn und Schleifenende bei jedem Schritt ermittelt. Wenn dieser unter einem festzulegenden Grenzwert liegt kann das Iterieren des Algorithmus abgebrochen werden. Das Ergebnis des Verfahrens, einen korrekten Schleifenschluss, kann man in Bild 2.4 sehen. An der Stelle, an der bei

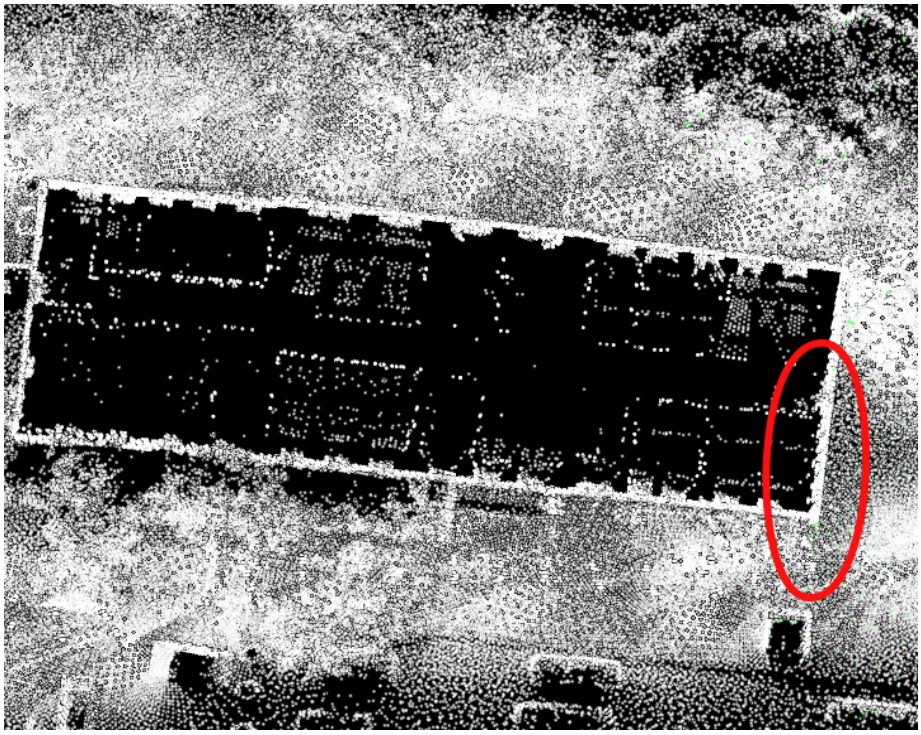

Bild 2.4: Korrekt mit Lu & Milios geschlossene Schleife, markierte Stelle war ohne Korrektur fehlerhaft

Bild 1.4 eine Doppelwand zu erkennen war, ist nun nur noch eine korrekte einzelne Wand zu sehen. Auch der in Bild 1.4 zu erkennende Höhenunterschied zwischen den beiden Enden wurde behoben. Zudem wurden nicht nur die Enden der Schleife angepasst, auch die dazwischenliegenden Scans wurden durch die neuen Positionsinformationen, die durch den Schleifenschluss hinzugefügt wurden, anders ausgerichtet. Somit liefert das Verfahren nach Lu & Milios ein global optimiertes Ergebnis der Positionen der einzelnen Scans.

Kapitel 3

Die Helixtransformation

In diesem Kapitel soll nun die Helixkorrekturberechnung als Lösungsweg für den ICP-Algorithmus beschrieben werden. Diese Korrekturvariante kann bei der lokalen Korrektur als Alternative für die Lösung mittels Singulärwertzerlegung, Quaternionen oder den anderen bekannten Verfahren angesehen werden. Sie ist aber auch für die globale Korrektur anwendbar. Dabei geht sie, im Vergleich zum Verfahren nach Lu & Milios, nicht den Umweg über die Beziehungen der einzelnen Posen der Scans, sondern arbeitet nur mit den Punktkorrespondenzen. Aber dazu folgt später in diesem Kapitel mehr. Um die Idee der Helixtransformation zu erklären wird hier mit der Beschreibung für den lokalen Fall begonnen, wie er in [PH03] eingeführt wird. Dieser wurde dort zur Registrierung von Punktewolken auf CAD-Modelle angewendet. Im Anschluss wird dann die globale Lösungsvariante nach Pottmann [PLH02] beschrieben, wobei auch auf den dort gemachten globalen Fehler eingegangen wird. Abschließend folgt die neue und erweiterte globale Korrekturberechnung, die diesen Fehler behebt.

3.1 Lokale Korrektur mit Helixtransformation

Zu Beginn dieses Abschnittes soll hier zunächst die Helixtransformation an sich beschrieben werden, da diese Transformationsvariante nicht sehr bekannt ist. Anschließend wird dann ihre Anwendung bei der lokalen Korrektur erläutert.

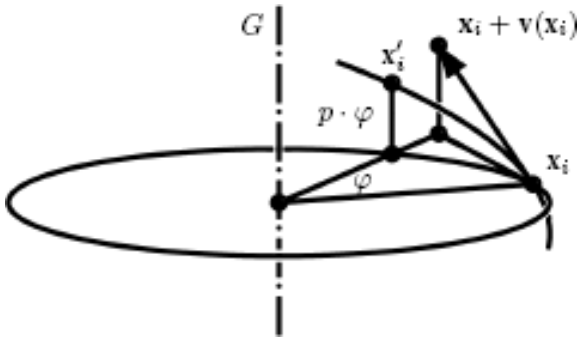

Bild 3.1: Prinzip der Helixtransformation eines Punktes (Quelle: [PH03])

3.1.1 Die Helixtransformation

Das grundsätzliche Transformationsprinzip der Helixtransformation ist in Bild 3.1 gut zu erkennen. Dort wird der Punkt $\boldsymbol{x}_i$ auf Position $\boldsymbol{x}_i'$ verschoben. Hierbei wird gleichzeitig eine Rotation um die Achse $\boldsymbol{G}$ und eine Translation parallel zu dieser Achse auf den Punkt $\boldsymbol{x}_i$ angewendet. Das Ergebnis dieser Kombination ist eine Bewegung, die der Form einer Helixspirale gleicht. In dem Sonderfall der reinen Translation wird hierbei einfach die Rotationskomponente weggelassen, bei reiner Rotation eben die Translationskomponente. Bei einer Rotation des Objektes um seine eigene Achse ist zudem der Radius der Rotation um die Achse $\boldsymbol{G}$ gleich Null. Doch woher bekommt man eine mathematische Beschreibung für diese Transformation? Hierfür wird eine nach der Zeit t parametrisierte Bewegung eines starren zu einem ruhenden System betrachtet. Ein Punkt $\boldsymbol{x}$ aus dem starren System lässt sich dann im ruhenden System als $\boldsymbol{x}_0(t)$ wie folgt darstellen:

$$\boldsymbol{x}_0(t) = \boldsymbol{u}(t) + \boldsymbol{A}(t) \cdot \boldsymbol{x} \tag{3.1}$$

Der Vector $\boldsymbol{u}$ stellt hierbei die Position des Ursprungs des starren System dar, die Matrix $\boldsymbol{A}$ ist eine orthogonale Matrix mit $\det(\boldsymbol{A}) = 1$ und stellt somit die Rotation des starren zum ruhenden System da. Mittels einer Differenzierung dieser Gleichung nach der Zeit t erhält man den Geschwindigkeitsvektor $\boldsymbol{v}(\boldsymbol{x})$ für den Punkt $\boldsymbol{x}$. Dieser ist in Bild 3.1 durch $\boldsymbol{x}_i + \boldsymbol{v}(\boldsymbol{x}_i)$ eingezeichnet. Ein solcher Geschwindigkeitsvektor setzt sich wie folgt zusammen:

$$\boldsymbol{v}(\boldsymbol{x}) = \bar{\boldsymbol{c}} + \boldsymbol{c} \times \boldsymbol{x} \tag{3.2}$$

Die direkte Anwendung dieses Geschwindigkeitsvektors auf den Punkt ist allerdings nicht ratsam, da dieser eine affine und keine euklidische Transformation eines Systems beschreibt. Das transformierte System würde so also verzerrt werden. In Bild 3.1 soll also die Position $\boldsymbol{x}_i'$ und nicht $\boldsymbol{x}_i + \boldsymbol{v}(\boldsymbol{x}_i)$ erreicht werden. Um aus diesem Vektor eine solche euklidische Transformation zu erzeugen sind einige Umrechnungen nötig. Hierfür macht man sich Plücker-Koordinaten und Quaternionen zu nutze. Die für die durch den Geschwindigkeitsvektor $\boldsymbol{v}$ eines Systems beziehungsweise dessen einzelnen Komponenten $(\boldsymbol{c}, \bar{\boldsymbol{c}})$ beschriebene Achse $\boldsymbol{G}$ der Helixtransformation hat die Plücker-Koordinaten $(\boldsymbol{g}, \bar{\boldsymbol{g}})$. Diese Plücker-Koordinaten kann man aus dem Geschwindigkeitsvektor wie folgt berechnen:

$$\boldsymbol{g} = \frac{\boldsymbol{c}}{\|\boldsymbol{c}\|}, \qquad \bar{\boldsymbol{g}} = \frac{\bar{\boldsymbol{c}} - p\boldsymbol{c}}{\|\boldsymbol{c}\|}, \qquad p = \frac{\boldsymbol{c}^{\mathrm{T}} \cdot \bar{\boldsymbol{c}}}{\boldsymbol{c}^2} \tag{3.3}$$

Der Skalar p ist hierbei ein Schraubparameter, der über das Produkt aus $p \cdot \varphi$ die Stärke der Helixschraubung beschreibt (siehe Bild 3.1). Den Drehwinkel φ für die durchzuführende Rotation erhalten wir aus

$$\varphi = \arctan \|\boldsymbol{c}\| \tag{3.4}$$

Die euklidsche Transformation mit Hilfe der Plücker-Koordinaten lautet dann:

$$\boldsymbol{x}_0 = \boldsymbol{R}(\boldsymbol{x} - \boldsymbol{p}) + (p \cdot \varphi)\boldsymbol{g} + \boldsymbol{p} \tag{3.5}$$

Der Vektor $\boldsymbol{p}$ ist hierbei ein beliebiger Punkt, der auf Drehachse $\boldsymbol{G}$ liegt, den man zum Beispiel aus $\boldsymbol{p} = \boldsymbol{g} \times \bar{\boldsymbol{g}}$ erhalten kann. Nun ist nur noch die Rotationsmatrix $\boldsymbol{R}$ mit Hilfe der Quaternionen zu belegenden:

$$\boldsymbol{R} = \begin{pmatrix} \boldsymbol{q}_0^2 + \boldsymbol{q}_1^2 - \boldsymbol{q}_2^2 - \boldsymbol{q}_3^2 & 2(\boldsymbol{q}_1\boldsymbol{q}_2 - \boldsymbol{q}_0\boldsymbol{q}_3) & 2(\boldsymbol{q}_1\boldsymbol{q}_3 + \boldsymbol{q}_0\boldsymbol{q}_2) \\ 2(\boldsymbol{q}_1\boldsymbol{q}_2 + \boldsymbol{q}_0\boldsymbol{q}_3) & \boldsymbol{q}_0^2 - \boldsymbol{q}_1^2 + \boldsymbol{q}_2^2 - \boldsymbol{q}_3^2 & 2(\boldsymbol{q}_2\boldsymbol{q}_3 - \boldsymbol{q}_0\boldsymbol{q}_1) \\ 2(\boldsymbol{q}_1\boldsymbol{q}_3 - \boldsymbol{q}_0\boldsymbol{q}_2) & 2(\boldsymbol{q}_2\boldsymbol{q}_3 + \boldsymbol{q}_0\boldsymbol{q}_1) & \boldsymbol{q}_0^2 - \boldsymbol{q}_1^2 - \boldsymbol{q}_2^2 + \boldsymbol{q}_3^2 \end{pmatrix} \tag{3.6}$$

wobei $\boldsymbol{q}_0 \ldots \boldsymbol{q}_3$ die einzelnen Elemente des Quaternion

$$\boldsymbol{q} = \cos\frac{\varphi}{2} + \boldsymbol{g}\sin\frac{\varphi}{2} \tag{3.7}$$

sind.

Sobald nun also der Geschwindigkeitsvektor v bzw. seine Komponenten $\boldsymbol{c}$ und $\bar{\boldsymbol{c}}$ für eine Helixtransformation bekannt sind kann die euklidische Transformation korrekt ausgeführt

werden. Für die Implementation der Transformation mittels einer üblichen 4×4 Transformationsmatrix ist noch eine kleine Umformung der Gleichung 3.5 nötig. Die übliche Transformationsmatrix hat die Form:

$$\boldsymbol{T} = \begin{pmatrix} & \boldsymbol{R} & & \boldsymbol{t} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Bei dieser Belegung ist die Matrix $\boldsymbol{R}$ die Rotationsmatrix aus Belegungsvorschrift 3.6. Den dreielementigen Verschiebungsvektor $\boldsymbol{t}$ erhält man dann, indem die Gleichung 3.5 ohne den Term $\boldsymbol{Rx}$ verwendet wird, also:

$$\boldsymbol{t} = -\boldsymbol{Rp} + (p \cdot \varphi)\boldsymbol{g} + \boldsymbol{p}$$

Die zu Beginn dieses Abschnittes angesprochenen zwei Sonderfälle bei fehlender Rotation bzw. fehlender Translation können noch einzeln betrachtet werden. Sie sind anhand der Komponenten des Geschwindigkeitsverktors $\boldsymbol{c}, \bar{\boldsymbol{c}}$ zu erkennen. Ist bei diesen der Vektor $\boldsymbol{c} = 0$ so ist keine Rotation vorhanden und man kann eine reine Translation mit $\boldsymbol{t} = \bar{\boldsymbol{c}}$ anwenden. Im anderen Fall, der reinen Rotation, ist $\boldsymbol{c} \cdot \bar{\boldsymbol{c}} = 0$ und man kann eine Transformationsmatrix mit $\boldsymbol{t} = 0$ und einer wie in Gleichung 3.6 zu belegenden Rotationsmatrix $\boldsymbol{R}$ erzeugen. Da diese Sonderfälle allerdings in der Praxis kaum auftreten und die bei ihnen mögliche vereinfachte Berechnung der Transformationsmatrix nur minimale Geschwindigkeitsvorteile liefert, wurde die Unterscheidung dieser bei der Implementation nicht berücksichtigt.

3.1.2 Die lokale Korrektur

Im vorherigen Abschnitt wurde die Anwendung der Helixtransformation beschrieben. Diese wurde durch den Geschwindigkeitsvektor $\boldsymbol{v}$ bzw. seinen Komponenten $\boldsymbol{c}, \bar{\boldsymbol{c}}$ beschrieben. Doch wie erhält man den korrekten Geschwindigkeitsvektor für die auszuführende Korrekturverschiebung? Dies soll nun hier zunächst für den lokalen Fall beschrieben werden, bevor es in den folgenden Abschnitten für die globale Korrektur erläutert wird. Zunächst folgt an dieser Stelle ein kurzer Überblick über die für die Anwendung des Verfahrens nötigen Voraussetzungen:

- Die gegenseitige Orientierung der zu registrierenden Scans muss näherungsweise bekannt sein. Da eine grobe Distanzmessung der Scanpositionen über die Odometriedaten des Roboters gegeben ist, ist diese Voraussetzung erfüllt.
- Es müssen Punktkorrespondenzen zwischen den zu registrierenden Scans vorhanden sein. Dies kann auf gleiche Weise wie bei den anderen lokalen Korrekturverfahren geschehen. Da die Orientierung der Scans grob bekannt ist sollten zwischen diesen eben auch Korrespondenzen gezogen werden können.
- Es können nur kleine Verlagerungen durchgeführt werden da der Geschwindigkeitsvektor nur die über die Zeit differenzierte momentane Bewegung des Scans darstellt. Da die grobe Orientierung der Scans bekannt ist sollten nur kleine Korrekturen nötig sein. Ist dies nicht der Fall kann die Korrektur iteriert werden um auch größere Verschiebungen zu ermöglichen.

Da diese Voraussetzungen gegeben sind, sollte dieses Verfahren für die Registrierung zweier 3D-Laserscans anwendbar sein. Doch wie ist der Geschwindigkeitsvektor $\boldsymbol{v}$ hierfür zu errechnen? Hier hilft ein Vergleich mit der Minimierungsfunktion der anderen lokalen Korrekturverfahren (vergleiche 2.1):

$$\mathrm{E}(\boldsymbol{R},\boldsymbol{t}) = \sum_i \|\boldsymbol{m}_i - (\boldsymbol{R}\boldsymbol{d}_i + \boldsymbol{t})\|^2$$

Doch statt den Abstand zwischen den Punkten der beiden Scans direkt über die Rotationsmatrix $\boldsymbol{R}$ und dem Verschiebungsvektor $\boldsymbol{t}$ zu minimieren soll dies nun über den Geschwindigkeitsvektor $\boldsymbol{v}$ beziehungsweise seinen Komponenten $\boldsymbol{c}, \bar{\boldsymbol{c}}$ erreicht werden. Die Abstandsfunktion zweier korrespondierender Punkt lautet hier also:

$$\mathrm{A}_{loc}(\boldsymbol{m}_i,\boldsymbol{d}_i) = (\boldsymbol{m}_i - (\boldsymbol{d}_i + \boldsymbol{c}\times\boldsymbol{d}_i + \bar{\boldsymbol{c}}))^2 \tag{3.8}$$

Daraus folgt folgende Minimierungsfunktion für die gesamte Betrachtung der beiden korrespondierenden Scans:

$$\begin{aligned}\mathrm{E}(\boldsymbol{c},\bar{\boldsymbol{c}}) &= \sum_i \mathrm{A}_{loc}(\boldsymbol{m}_i,\boldsymbol{d}_i) \\ &= \sum_i \|\boldsymbol{m}_i - (\boldsymbol{d}_i + \bar{\boldsymbol{c}} + \boldsymbol{c}\times\boldsymbol{d}_i)\|^2 \end{aligned} \tag{3.9}$$

Es soll also der Abstand zwischen der Summe des jeweiligen Punktes aus dem zu registrierenden Datenscan und des auf diesen angewendeten Geschwindigkeitsvektors auf den dazu korrespondierenden Punkt aus der Modellmenge minimiert werden. Dieser Abstand ist eben abhängig von den Komponenten $\boldsymbol{c}, \bar{\boldsymbol{c}}$ des Geschwindigkeitsvektors. Es seien nun:

$$\boldsymbol{a}_i := \boldsymbol{d}_i - \boldsymbol{m_i}, \quad \boldsymbol{c}_{loc} := \begin{pmatrix} \boldsymbol{c} \\ \bar{\boldsymbol{c}} \end{pmatrix}, \quad \boldsymbol{B}(\boldsymbol{d}_i) := \begin{pmatrix} 0 & \boldsymbol{d}_{i,z} & -\boldsymbol{d}_{i,y} & 1 & 0 & 0 \\ -\boldsymbol{d}_{i,z} & 0 & \boldsymbol{d}_{i,x} & 0 & 1 & 0 \\ \boldsymbol{d}_{i,y} & -\boldsymbol{d}_{i,x} & 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.10}$$

Mit diesen Definitionen gilt dann:

$$\boldsymbol{c} \times \boldsymbol{d}_i + \bar{\boldsymbol{c}} = \boldsymbol{B}(\boldsymbol{d}_i)\boldsymbol{c}_{loc}$$

Um nun ein Minimum der Funktion 3.9 zu erreichen ist folgende Gleichung für $\boldsymbol{c}_{loc}$ zu lösen:

$$\sum_i \boldsymbol{B}(\boldsymbol{d}_i)^{\mathrm{T}} \boldsymbol{B}(\boldsymbol{d}_i)\boldsymbol{c}_{loc} = \sum_i \boldsymbol{B}(\boldsymbol{d}_i)^{\mathrm{T}} \boldsymbol{a}_i$$

Diese Gleichung lässt sich zu

$$\boldsymbol{B}_{ges}\boldsymbol{c}_{loc} = \boldsymbol{a}_{ges} \tag{3.11}$$

zusammenfassen. Nach Lösung dieser Gleichung enthält $\boldsymbol{c}_{loc}$ die Komponenten für den Geschwindigkeitsvektor der Korrektur mittels der Helixtransformation. Dieser wird nun auf den Datenscan angewendet. Diese Korrekturberechnung und Verschiebung kann nun iterativ angewendet werden bis der Abstand zwischen den beiden Scans unter einem akzeptablen Minimum liegt. Das Ergebnis dieser lokalen Korrektur ist in Bild 3.2 zu sehen. Dieses Ergebnis ist vergleichbar mit Bild 2.2, welches mit den Standartverfahren erzielt wurde.

3.2 Lokal optimale globale Korrektur

Diese Art der Korrekturberechnung über die Ermittlung eines Geschwindigkeitsvektors $\boldsymbol{v}$ und dessen Berechnung mittels einer Matrix soll nun auch für den globalen Fall angewendet werden. Pottmann hat hierfür in [PLH02] bereits einen Ansatz vorgestellt. Auf diesen soll hier nun zunächst eingegangen werden. Allerdings arbeitet dieser nicht global optimal,

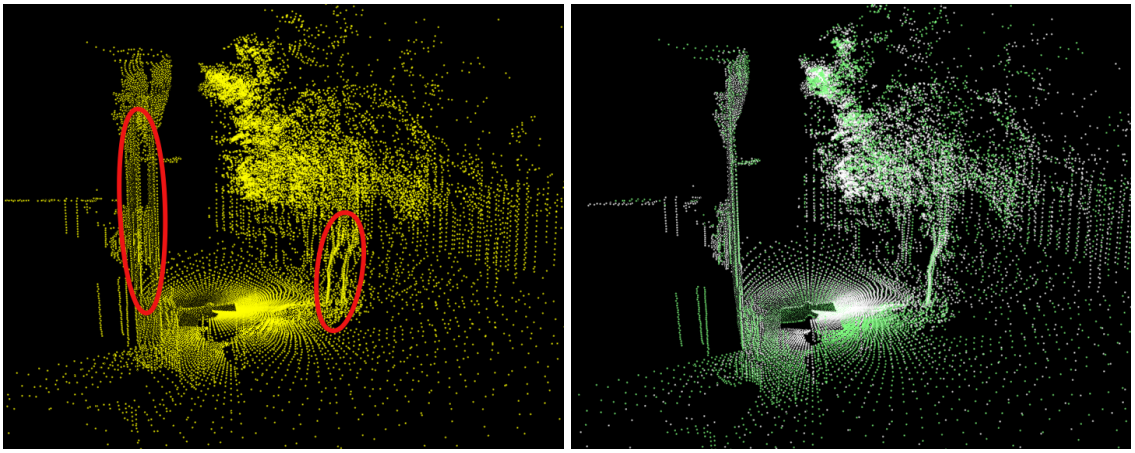

Bild 3.2: Lokale Korrektur mittels Helixtransformation, Links vorher (Fehler markiert), Rechts nachher

wie in den Ergebnissen des Verfahrens zu sehen ist. Da aber die in Kapitel 3.3 vorgestellte neue Version auf den Ansätzen dieser Lösung beruht, soll sie hier dennoch genau erläutert werden.

3.2.1 Die Korrekturberechnung

Um eine globale Korrektur zu ermöglichen werden jetzt nicht mehr nur zwei Scans pro Korrekturschritt betrachtet sondern alle Scans einer Schleife gleichzeitig. Dies ermöglicht es, einen Scan nicht nur an einen benachbarten Scan anzupassen sondern an alle benachbarten Scans. Hierfür werden zunächst die gleichen drei Anfangsschritte ausgeführt, die auch beim Algorithmus nach Lu & Milios angewendet werden (siehe Kapitel 2.2.2). Es werden also auch hier zunächst alle Scans lokal korrigiert, dann ein Nachbarschaftsgraph erstellt, und schließlich für alle dort benachbarten Scans die jeweiligen Punktkorrespondenzen zwischen ihnen erzeugt. Doch anstatt beim darauf folgenden Schritt, der Matrixbelegung, nun den Umweg über die Posenbeziehungen der einzelnen Scans zu gehen, und diese dann zu optimieren, wird hier nun weiterhin nur mit den Punktkorrespondenzen und deren Abständen gearbeitet. Es gilt jetzt im Vergleich zu der lokalen Korrektur der

Grundsatz, dass der erste Scan einer Schleife den Ursprung des Koordinatensystems der gesamten zu korrigierenden Schleife bildet. Dieser erste Scan bleibt also immer ruhend, alle anderen Scans werden relativ zu diesem verschoben. Dies hat bei der Aufsummierung der Matrix Auswirkungen, auf die an der dortigen Stelle noch eingegangen werden soll. Für die Aufstellung der Abstandsfunktion bei der gleichzeitigen Betrachtung aller Scans muss also auch die möglicherweise vorhandene Verschiebung des Modellscans mit einbezogen werden. Da dieser bei der lokalen Korrektur ruhend war, konnte man dies dort vernachlässigen (siehe 3.8). Hier werden aber jetzt, bis auf den ersten, alle Scans gleichzeitig betrachtet und verschoben. Der Modellscan ist also nur innerhalb einer Korrespondenzbeziehung tatsächlich ein Modellscan und kann innerhalb einer anderen Beziehung Datenscan sein und daher auch einer Verschiebung unterliegen. Im Vergleich zur lokalen Abstandsfunktion $\mathrm{A}_{loc}(\boldsymbol{m}_i, \boldsymbol{d}_i)$ lautet diese nun für die globale Sicht:

$$\mathrm{A}_{glob}(\boldsymbol{m}_i, \boldsymbol{d}_i) = (\boldsymbol{m}_i - (\boldsymbol{d}_i - (\boldsymbol{c}_j \times \boldsymbol{d}_i + \bar{\boldsymbol{c}}_j) + (\boldsymbol{c}_k \times \boldsymbol{d}_i + \bar{\boldsymbol{c}}_k)))^2 \tag{3.12}$$

Hierbei sind $\boldsymbol{c}_j, \bar{\boldsymbol{c}}_j$ die Komponenten des zum Modellscan gehörenden Geschwindigkeitsvektors und $\boldsymbol{c}_k, \bar{\boldsymbol{c}}_k$ die des zum Datenscan gehörenden Geschwindigkeitsvektors. In [PLH02] wird nun an dieser Stelle für die globale Korrektur folgende Minimierungsfunktion aufgestellt:

$$\mathrm{E}(\boldsymbol{m}_i, \boldsymbol{d}_i) = \sum_{(j,k)} \sum_{i} \mathrm{A}_{glob}(\boldsymbol{m}_i, \boldsymbol{d}_i)$$

Aus dieser Funktion folgert Pottmann direkt und in Analogie zu lokalen Formel folgende Zusammenfassung:

$$\mathrm{E}(\boldsymbol{m}_i, \boldsymbol{d}_i) = {\boldsymbol{c}_{ges}}^{\mathrm{T}} \boldsymbol{B}_{ges} \boldsymbol{c}_{ges} + 2{\boldsymbol{a}_{ges}}^{\mathrm{T}} \boldsymbol{c}_{ges} + \sum_{(j,k)} \sum_{i} (\boldsymbol{m}_i - \boldsymbol{d}_i)^2 \tag{3.13}$$

Hierbei wird über alle benachbarten Scans j, k und deren dort vorhandenen Korrespondenzen i aufsummiert. Außerdem gelten für diese Gleichung sowie für daraus hergeleitete Gleichung $\boldsymbol{B}_{ges}\boldsymbol{c}_{ges} = \boldsymbol{a}_{ges}$ zur Lösung nach $\boldsymbol{c}_{ges}$ folgende Belegungsvorschriften der ent-

haltenen Matrizen und Vektoren:

$$\boldsymbol{B}_{ges} = \begin{pmatrix} \boldsymbol{B}_2 & 0 & 0 & \dots \\ 0 & \boldsymbol{B}_3 & & \\ 0 & & \ddots & \\ \vdots & & & \boldsymbol{B}_N \end{pmatrix} \qquad \boldsymbol{c}_{ges} = \begin{pmatrix} \boldsymbol{c}_2 \\ \bar{\boldsymbol{c}}_2 \\ \boldsymbol{c}_3 \\ \bar{\boldsymbol{c}}_3 \\ \vdots \\ \boldsymbol{c}_N \\ \bar{\boldsymbol{c}}_N \end{pmatrix} \qquad \boldsymbol{a}_{ges} = \begin{pmatrix} \boldsymbol{a}_2 \\ \boldsymbol{a}_3 \\ \vdots \\ \boldsymbol{a}_N \end{pmatrix} \tag{3.14}$$

Der Index $(2, 3, \dots, N)$ der jeweiligen Subelemente geht dabei über die in der Schleife vorhandenen Scans. Wie oben bereits erwähnt wird hier der erste Scan nicht beachtet, da dieser der Ursprungsscan ist und daher nicht verschoben wird. Die Matrix $\boldsymbol{B}_{ges}$ ist daher eine $6(N-1) \times 6(N-1)$ große Matrix, die Vektoren jeweils $6(N-1)$-elementige Spaltenvektoren. $\boldsymbol{B}_{ges}$ ist außerdem nur auf der Blockdiagonalen mit Submatrizen belegt, die folgendermaßen aufgestellt werden:

$$\boldsymbol{B}_j = \sum_{(j,k)} \sum_i \boldsymbol{B}(\boldsymbol{m}_i)^{\mathrm{T}} \boldsymbol{B}(\boldsymbol{m}_i) + \sum_{(k,j)} \sum_i \boldsymbol{B}(\boldsymbol{d}_i)^{\mathrm{T}} \boldsymbol{B}(\boldsymbol{d}_i) \tag{3.15}$$

mit $2 \leq j \leq N$ und $1 \leq k \leq N$. Dies bedeutet, dass die Submatrix, die zu einem Scan gehört, über alle Nachbarschaften, zu der der Scan gehört, und dort dann über alle Punkte, die der Scan zu dieser Nachbarschaft beziehungsweise Korrespondenz beiträgt, aufsummiert wird. Ähnlich werden die Subvektoren des Vektors $\boldsymbol{a}_{ges}$ belegt:

$$\boldsymbol{a}_j = \sum_{(j,k)} \sum_i \begin{pmatrix} \boldsymbol{m}_i \times (\boldsymbol{m}_i - \boldsymbol{d}_i) \\ \boldsymbol{m}_i - \boldsymbol{d}_i \end{pmatrix} + \sum_{(k,j)} \sum_i \begin{pmatrix} \boldsymbol{d}_i \times (\boldsymbol{d}_i - \boldsymbol{m}_i) \\ \boldsymbol{d}_i - \boldsymbol{m}_i \end{pmatrix} \tag{3.16}$$

Das so belegte Gleichungssystem $\boldsymbol{B}_{ges}\boldsymbol{c}_{ges} = \boldsymbol{a}_{ges}$ lässt sich dann für $\boldsymbol{c}_{ges}$ mit Hilfe der QR-Zerlegung auflösen. Die Komponenten $\boldsymbol{c}_j, \bar{\boldsymbol{c}}_j$ werden dann für die Helixtransformation und somit für die Korrekturverschiebung der jeweils zu den Komponenten gehörenden Scans verwendet. Danach erfolgt dann wie bei dem Verfahren nach Lu & Milios die Fehlerkontrolle des Abstandes beim Schleifenschluss sowie die Wiederholung des gesamten Ablaufes ab Schritt 2, der Graphgenerierung. Dies geschieht solange bis der Abstand einen akzeptablen Wert unterschreitet. Das Ergebnis des Algorithmus ist in Bild 3.3 zu sehen.

Bild 3.3: Ergebnis des lokal optimalen Verfahrens

Dort ist an der rechten unteren Ecke kein Fehler in Form einer doppelten Wand (vergleiche Bild 1.4) zu sehen. Auch der Höhenunterschied der beiden Enden ist nicht mehr vorhanden.

3.2.2 Warum nur lokale Optimierung?

Das Ergebnis des von Pottmann beschriebenen Verfahrens ist eins, das den globalen Fehler an der Stelle des Schleifenschluss korrigiert. Wenn man dieses Ergebnis nun aber mit dem unkorrigierten Bild (Bild 3.4) und dem mit dem Verfahren nach Lu & Milios (Bild 3.5) vergleicht, ist ein deutlicher Unterschied zu erkennen. Es wurden bei diesen Bilder der lokal optimierte globale 3D-Scan, der rot gefärbt ist, im ersten Bild über den unkorrigierten 3D-Scan gelegt, im zweiten über den mit Lu & Milios korrigierten 3D-Scan. An den Stellen, an den der jeweilige Vergleichsscan in weiß zu sehen ist, erkennt man die Unterschiede. So ist beim Vergleich mit der unkorrigierten Karte nur der Fehler am Treffpunkt der beiden Schleifenende korrigiert worden. Dies ist an der rechten unteren Ecke der aufgenommenen Schleife sichtbar. Dort schimmert noch die weiße Doppelwand, die durch den Fehler beim Schleifenschluss entstanden ist, hindurch (siehe Markierung in Bild 3.4). Beim Vergleich der lokal optimalen Korrektur mit der Lu & Milios Korrektur (Bild 3.5) fällt hingegen auf,

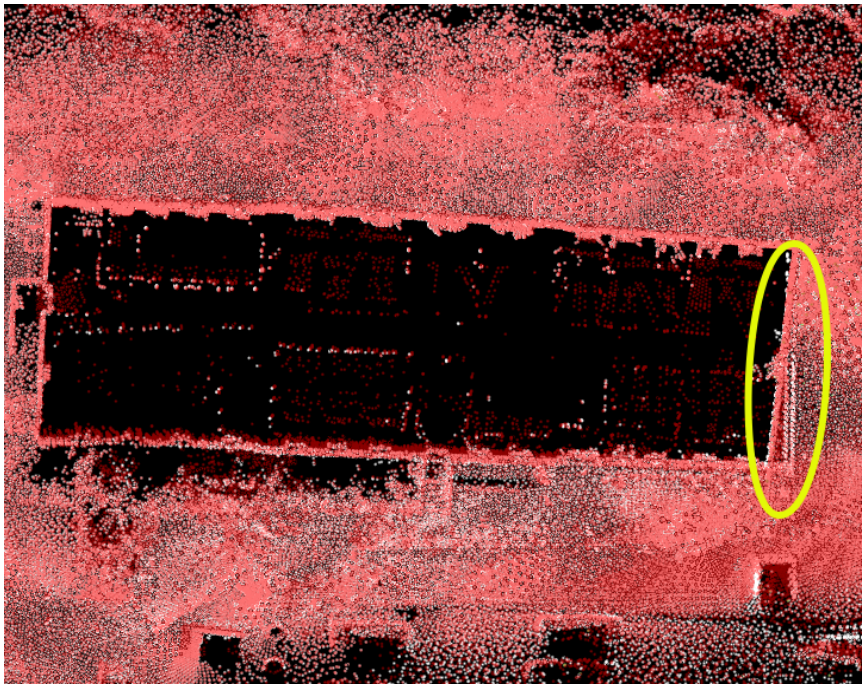

Bild 3.4: Vergleich lokal optimale globale Helixkorrektur (Rot) mit unkorrigierter Karte (Weiß)

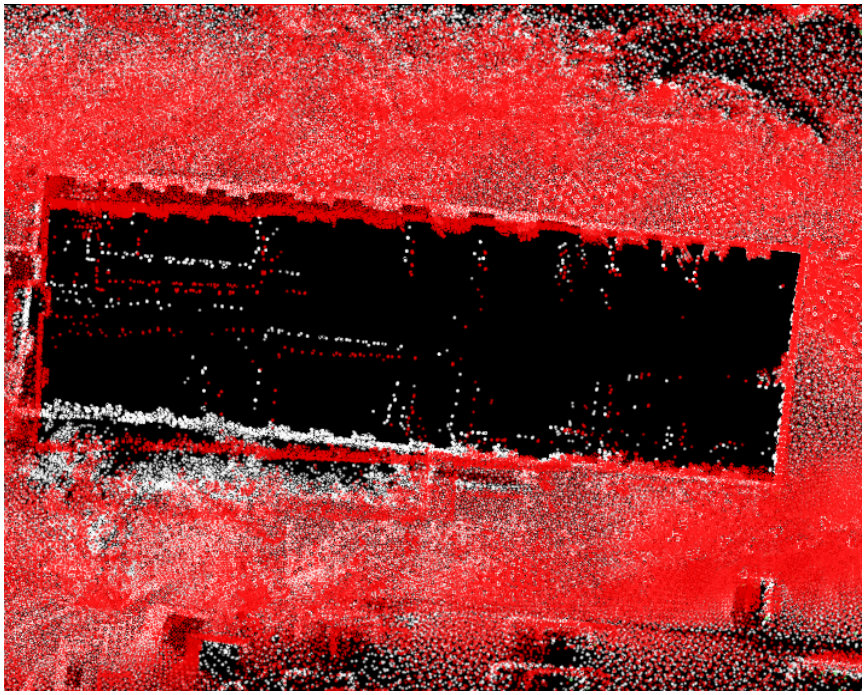

Bild 3.5: Vergleich lokal optimale globale Helixkorrektur (Rot) mit Lu & Milios Korrektur (Weiß)

das zwar die Stelle des Schleifenschlusses nahezu identisch ist, sich allerdings bei Lu & Milios auch alle Scans, die zwischen Schleifenanfang und Schleifenende liegen, verschoben haben. Diese ist an den im Vergleich zu den roten Punkten teilweise gänzlich anders positionierten weißen Punkt zu deutlich erkennen. Grund für diese Unterschiede ist die fehlende Übernahme der durch den Schleifenschluss neu gewonnenen Lagepositionen an die Scans innerhalb der Schleife. Bei der Korrektur nach Lu & Milios wird die Korrektur am Schleifenschluss anteilig an die Scans innerhalb der Schleife weitergegeben. Somit erhalten auch diese die durch das Zusammentreffen der Schleifenenden hinzugefügte neue Lageinformation. Bei der lokal optimalen globalen Helixkorrektur fehlt diese Informationsweitergabe. Es werden dort nur lokal, also innerhalb eines gewissen Bereiches, in dem sich mehrere Scans überlappen, alle Lageinformationen dieser Scans verwendet, um ihre Position zu korrigieren. Ein Weiterreichen dieser Positionsinformation findet aber nicht statt. Dies ist zwar eine Verbesserung im Vergleich zur reinen lokalen Helixkorrektur, diese verwendet nur die Positionsinformationen von zwei Scans, liefert aber somit kein global optimiertes Ergebnis.

3.3 Global optimale Korrektur

Genau dieses global optimierte Ergebnis was mit der bisherigen globalen Helixkorrektur nach Pottmann nicht zu erzielen war, versucht nun der erweiterte Ansatz zu erreichen. Er wird nun im folgenden Abschnitt beschrieben. Auch hier sind die ersten drei Schritte der Verfahrens gleich denen von Lu & Milios und der lokal optimalen globalen Helixkorrektur. Die Scans werden also auch hier zunächst lokal Korrigiert, aus ihnen dann ein Nachbarschaftsgraph erstellt und dort aus den benachbarten Scans die jeweiligen Punktkorrespondenzen ermittelt. Um nun aber bei der Berechnung der Geschwindigkeitsvektoren zu Korrektur auch die Verschiebung andere Scans innerhalb der Schleife zu beachten sind Matrixeinträge außerhalb der Blockdiagonalen der Matrix nötig. Es bedarf also einiger Umformungen und Erweiterungen der Belegungsvorschrift der Matrix. Diese werden im folgenden Abschnitt erläutert.

3.3.1 Umformung für globale Anwendung

Für die weitere Betrachtung werden aus Gründen der Übersichtlichkeit zunächst die Verschiebungskomponenten $(\bar{\boldsymbol{c}}_j + \boldsymbol{c}_j \times \boldsymbol{m}_i)$ und $(\bar{\boldsymbol{c}}_k + \boldsymbol{c}_k \times \boldsymbol{m}_k)$ wieder zum jeweilige Geschwindigkeitsvektor $\boldsymbol{v}_j$ und $\boldsymbol{v}_k$ zusammengefasst und somit Gleichung 3.12 aus der lokal optimierenden globalen Korrektur vereinfacht:

$$\begin{aligned} \mathrm{A}_{glob}(\boldsymbol{m}_i, \boldsymbol{d}_i) &= (\boldsymbol{m}_i - \boldsymbol{d}_i + (\bar{\boldsymbol{c}}_j + \boldsymbol{c}_j \times \boldsymbol{m}_i) - (\bar{\boldsymbol{c}}_k + \boldsymbol{c}_k \times \boldsymbol{m}_i))^2 \\ &:= ((\boldsymbol{m}_i - \boldsymbol{d}_i) + (\boldsymbol{v}_j - \boldsymbol{v}_k))^2 \\ &= (\boldsymbol{m}_i - \boldsymbol{d}_i)^2 + (\boldsymbol{v}_j - \boldsymbol{v}_k)^2 + 2\left((\boldsymbol{m}_i - \boldsymbol{d}_i) \cdot \boldsymbol{v}_j + (\boldsymbol{d}_i - \boldsymbol{m}_i) \cdot \boldsymbol{v}_k\right) \end{aligned}$$

Genaugenommen müsste im Folgenden auch noch über alle Paare (j, k) summiert werden. Dies wird der Einfachheit wegen hier nur implizit gemacht.

$$\begin{aligned} F &= \sum_i \mathrm{A}_{glob}(\boldsymbol{m}_i, \boldsymbol{d}_i) \\ &= \sum_i (\boldsymbol{v}_j - \boldsymbol{v}_k)^2 + \sum_i 2((\boldsymbol{m}_i - \boldsymbol{d}_i) \cdot \boldsymbol{v}_j + (\boldsymbol{d}_i - \boldsymbol{m}_i) \cdot \boldsymbol{v}_k) + \sum_i (\boldsymbol{m}_i - \boldsymbol{d}_i)^2 \end{aligned}$$

Diese Funktion vergleichen wir nun mit der Fehlerfunktion 3.13 der lokal optimalen globalen Helixkorrektur:

$$\mathrm{F} = \boldsymbol{c}_{ges}{}^{\mathrm{T}} \boldsymbol{B}_{ges} \boldsymbol{c}_{ges} + 2 \cdot \boldsymbol{a}_{ges}{}^{\mathrm{T}} \boldsymbol{c}_{ges} + \sum_i (\boldsymbol{m}_i - \boldsymbol{d}_i)^2$$

und stellen folgenden Koeffizientenvergleich auf:

$$\sum_i (\boldsymbol{v}_j - \boldsymbol{v}_k)^2 = \boldsymbol{c}_{ges}{}^{\mathrm{T}} \boldsymbol{B} \boldsymbol{c}_{ges} \tag{3.17}$$

$$\sum_i 2((\boldsymbol{m}_i - \boldsymbol{d}_i)\boldsymbol{v}_j + (\boldsymbol{d}_i - \boldsymbol{m}_i)\boldsymbol{v}_k) = 2\boldsymbol{a}_{ges}{}^{\mathrm{T}} \boldsymbol{c}_{ges} \tag{3.18}$$

Aus dem Koeffizientenvergleich 3.18 wird dann der Vektor $\boldsymbol{a}_{ges}$ wie folgt hergeleitet:

Der $6(N-1) \times 1$ Spaltenvektor $\boldsymbol{a}_{ges}$ setzt sich aus $N-1$ Subvektoren $\boldsymbol{a}_j$ nach dem Belegungsmuster $\boldsymbol{a}_{ges} = (\boldsymbol{a}_2{}^{\mathrm{T}}, \boldsymbol{a}_3{}^{\mathrm{T}}, \ldots, \boldsymbol{a}_N{}^{\mathrm{T}})^{\mathrm{T}}$ zusammen. Der Index j bezeichnet hierbei

den dazugehörigen Scan. Die Herleitung der Subvektoren $\boldsymbol{a}_j$ ergibt sich wie folgt:

$$\begin{aligned}
2\boldsymbol{a}_j\boldsymbol{c}_{loc_i} &= 2\sum_i((\boldsymbol{m}_i-\boldsymbol{d}_i)\boldsymbol{v}_j+(\boldsymbol{d}_i-\boldsymbol{m}_i)\boldsymbol{v}_k)\\
&= 2\sum_i\begin{pmatrix}\boldsymbol{m}_i-\boldsymbol{d}_i\\ \boldsymbol{d}_i-\boldsymbol{m}_i\end{pmatrix}\cdot\begin{pmatrix}\boldsymbol{v}_j\\ \boldsymbol{v}_k\end{pmatrix}\\
&= 2\sum_i\begin{pmatrix}\boldsymbol{m}_i-\boldsymbol{d}_i\\ \boldsymbol{d}_i-\boldsymbol{m}_i\end{pmatrix}\cdot\begin{pmatrix}\bar{\boldsymbol{c}}_j\\ \bar{\boldsymbol{c}}_k\end{pmatrix}+\begin{pmatrix}\boldsymbol{m}_i-\boldsymbol{d}_i\\ \boldsymbol{d}_i-\boldsymbol{m}_i\end{pmatrix}\cdot\begin{pmatrix}\boldsymbol{c}_j\times\boldsymbol{m}_i\\ \boldsymbol{c}_k\times\boldsymbol{m}_i\end{pmatrix}
\end{aligned}$$

durch die Äquivalenz des Spatproduktes:

$$\begin{aligned}
&= 2\sum_i\begin{pmatrix}\boldsymbol{m}_i-\boldsymbol{d}_i\\ \boldsymbol{d}_i-\boldsymbol{m}_i\end{pmatrix}\cdot\begin{pmatrix}\bar{\boldsymbol{c}}_j\\ \bar{\boldsymbol{c}}_k\end{pmatrix}+\begin{pmatrix}\boldsymbol{m}_i\times(\boldsymbol{m}_i-\boldsymbol{d}_i)\\ \boldsymbol{m}_i\times(\boldsymbol{d}_i-\boldsymbol{m}_i)\end{pmatrix}\cdot\begin{pmatrix}\boldsymbol{c}_j\\ \boldsymbol{c}_k\end{pmatrix}\\
&= 2\sum_i\begin{pmatrix}\boldsymbol{m}_i\times(\boldsymbol{m}_i-\boldsymbol{d}_i)\\ \boldsymbol{m}_i-\boldsymbol{d}_i\\ \boldsymbol{m}_i\times(\boldsymbol{d}_i-\boldsymbol{m}_i)\\ \boldsymbol{d}_i-\boldsymbol{m}_i\end{pmatrix}\cdot\begin{pmatrix}\boldsymbol{c}_j\\ \bar{\boldsymbol{c}}_j\\ \boldsymbol{c}_k\\ \bar{\boldsymbol{c}}_k\end{pmatrix}
\end{aligned}$$

Somit ist $\boldsymbol{a}_{ges}$ durch seine Subvektoren $\boldsymbol{a}_j$ mit den jeweiligen Punkten $\boldsymbol{m}_{i,j}$ und $\boldsymbol{d}_{i,j}$ (der zusätzliche Index j bezeichnet den dazugehörigen Scan j) gegeben durch:

$$\boldsymbol{a}_j=\sum_{(j,k)}\sum_i\begin{pmatrix}\boldsymbol{m}_{i,j}\times(\boldsymbol{m}_{i,j}-\boldsymbol{d}_{i,k})\\ (\boldsymbol{m}_{i,j}-\boldsymbol{d}_{i,k})\end{pmatrix}+\sum_{(k,j)}\sum_i\begin{pmatrix}\boldsymbol{m}_{i,k}\times(\boldsymbol{d}_{i,j}-\boldsymbol{m}_{i,k})\\ (\boldsymbol{d}_{i,j}-\boldsymbol{m}_{i,k})\end{pmatrix}\tag{3.19}$$

Verständlicher ausgedrückt: Bei jeder Verbindung 2er Scans j,k addiere man bei $\boldsymbol{a}_j$ die vordere Summe und bei $\boldsymbol{a}_k$ die hintere Summe. Dies ist genau das gleiche Verfahren das auch bei der lokal optimalen Helixkorrektur angewendet wird.

Aus dem anderen Koeffizientenvergleich 3.17 entsteht nun die erweiterte Matrix $\boldsymbol{B}_{ges}$. Diese hat die Form:

$$\boldsymbol{B}_{ges}=\begin{pmatrix}
\boldsymbol{B}_{2,2} & \boldsymbol{B}_{2,\bar{2}} & \boldsymbol{B}_{2,3} & \boldsymbol{B}_{2,\bar{3}} & \dots & \boldsymbol{B}_{2,\bar{N}}\\
\boldsymbol{B}_{\bar{2},2} & \boldsymbol{B}_{\bar{2},\bar{2}} & & & & \\
\boldsymbol{B}_{3,2} & & \boldsymbol{B}_{3,3} & & & \\
\vdots & & & \boldsymbol{B}_{\bar{3},\bar{3}} & & \\
 & & & & \ddots & \\
\boldsymbol{B}_{\bar{N},2} & & & & & \boldsymbol{B}_{\bar{N},\bar{N}}
\end{pmatrix}.$$

Jede Submatrix ist eine 3×3 Matrix. Diese werden im folgenden durch Quotientenvergleich aufgestellt.

Der Einfachheit halber betrachten wir nun für die Herleitung der Matrix $\boldsymbol{B}_{ges}$ nur die Verbindung zweier Scans j, k. $\boldsymbol{B}_{ges}$ ist nun also eine 12×12 Matrix, Vektor $\boldsymbol{c}_{ges}$ ein 12-elementiger Spaltenvektor.

$$\begin{aligned} {\boldsymbol{c}_{ges}}^{\mathrm{T}} \cdot \boldsymbol{B}_{ges} \cdot \boldsymbol{c}_{ges} &= \sum_i (\boldsymbol{v}_j - \boldsymbol{v}_k)^2 \\ &= \sum_i \boldsymbol{v}_j^2 + \boldsymbol{v}_k^2 - 2\boldsymbol{v}_j \cdot \boldsymbol{v}_k \end{aligned}$$

Der Term $\boldsymbol{v}_j^2 + \boldsymbol{v}_k^2$ wird die Blockdiagonale der Matrix $\boldsymbol{B}_{ges}$ bilden, während $-2\boldsymbol{v}_j \cdot \boldsymbol{v}_k$ den Rest der Matrix füllt. Betrachten wir zunächst die Blockdiagonale und somit $\boldsymbol{v}_j^2 + \boldsymbol{v}_k^2$.

$$\begin{aligned} \boldsymbol{v}_j^2 + \boldsymbol{v}_k^2 = \bar{\boldsymbol{c}}_j^2 + (\boldsymbol{c}_j \times \boldsymbol{m}_i)^2 + \bar{\boldsymbol{c}}_k^2 + (\boldsymbol{c}_k \times \boldsymbol{m}_i)^2 \\ + 2\bar{\boldsymbol{c}}_j \cdot (\boldsymbol{c}_j \times \boldsymbol{m}_i) + 2\bar{\boldsymbol{c}}_k \cdot (\boldsymbol{c}_k \times \boldsymbol{m}_i) \end{aligned}$$

mit Hilfe der Lagrange Identität:

$$\begin{aligned} = \bar{\boldsymbol{c}}_j^2 + \boldsymbol{m}_i^2 \cdot \boldsymbol{c}_j^2 - (\boldsymbol{c}_j \cdot \boldsymbol{m}_i)^2 + \bar{\boldsymbol{c}}_k^2 + \boldsymbol{m}_i^2 \cdot \boldsymbol{c}_k^2 - (\boldsymbol{c}_k \cdot \boldsymbol{m}_i)^2 \\ + 2\bar{\boldsymbol{c}}_j \cdot (\boldsymbol{c}_j \times \boldsymbol{m}_i) + 2\bar{\boldsymbol{c}}_k \cdot (\boldsymbol{c}_k \times \boldsymbol{m}_i) \end{aligned}$$

Die Submatrix $\boldsymbol{B}_{\bar{j},\bar{j}}$ ist damit einfach durch:

$$\boldsymbol{B}_{\bar{j},\bar{j}} = \sum_{\substack{(j,k) \\ (k,j)}} \sum_i I_3$$

gegeben.

Die Submatrix $\boldsymbol{B}_{j,j}$ hat ihren Ursprung in $\boldsymbol{m}_i^2 \cdot \boldsymbol{c}_j^2 - (\boldsymbol{c}_j \cdot \boldsymbol{m}_i)^2$ und $\boldsymbol{m}_i^2 \cdot \boldsymbol{c}_k^2 - (\boldsymbol{c}_k \cdot \boldsymbol{m}_i)^2$:

$$\boldsymbol{B}_{j,j} = \sum_{\substack{(j,k)\\(k,j)}} \sum_i \begin{pmatrix} \boldsymbol{m}_i^2 - \boldsymbol{m}_{x,i}^2 & -\boldsymbol{m}_{x,i}\boldsymbol{m}_{y,i} & -\boldsymbol{m}_{x,i}\boldsymbol{m}_{z,i} \\ -\boldsymbol{m}_{x,i}\boldsymbol{m}_{y,i} & \boldsymbol{m}_i^2 - \boldsymbol{m}_{y,i}^2 & -\boldsymbol{m}_{y,i}\boldsymbol{m}_{z,i} \\ -\boldsymbol{m}_{x,i}\boldsymbol{m}_{z,i} & -\boldsymbol{m}_{y,i}\boldsymbol{m}_{z,i} & \boldsymbol{m}_i^2 - \boldsymbol{m}_{z,i}^2 \end{pmatrix}.$$

Nun müssen noch die Ausdrücke $2\bar{\boldsymbol{c}}_j \cdot (\boldsymbol{c}_j \times \boldsymbol{m}_i)$ und $2\bar{\boldsymbol{c}}_k \cdot (\boldsymbol{c}_k \times \boldsymbol{m}_i)$ verwertet werden. Diese werden von den Submatrizen $\boldsymbol{B}_{\bar{j},j}$ bzw. $\boldsymbol{B}_{j,\bar{j}}$ gebildet:

$$\boldsymbol{B}_{j,\bar{j}} = -\boldsymbol{B}_{\bar{j},j} = \sum_{\substack{(j,k)\\(k,j)}} \sum_i \begin{pmatrix} 0 & -\boldsymbol{m}_{z,i} & \boldsymbol{m}_{y,i} \\ \boldsymbol{m}_{z,i} & 0 & -\boldsymbol{m}_{x,i} \\ -\boldsymbol{m}_{y,i} & \boldsymbol{m}_{x,i} & 0 \end{pmatrix}.$$

Die restlichen Einträge von $\boldsymbol{B}_{ges}$ werden durch $-2\boldsymbol{v}_j \cdot \boldsymbol{v}_k$ abgedeckt:

$$\begin{aligned} -2\boldsymbol{v}_j \cdot \boldsymbol{v}_k = -2(&\bar{\boldsymbol{c}}_j \cdot \bar{\boldsymbol{c}}_k \\ &+ (\boldsymbol{c}_j \times \boldsymbol{m}_i) \cdot (\boldsymbol{c}_k \times \boldsymbol{m}_i) \\ &+ \bar{\boldsymbol{c}}_j \cdot (\boldsymbol{c}_k \times \boldsymbol{m}_i) + \bar{\boldsymbol{c}}_k \cdot (\boldsymbol{c}_j \times \boldsymbol{m}_i)) \end{aligned}$$

mit Hilfe der Lagrange Identität:

$$\begin{aligned} = -2(&\bar{\boldsymbol{c}}_j \cdot \bar{\boldsymbol{c}}_k \\ &+ (\boldsymbol{c}_j \cdot \boldsymbol{c}_k) \cdot \boldsymbol{m}_i^2 - (\boldsymbol{c}_j \cdot \boldsymbol{m}_i) \cdot (\boldsymbol{c}_k \cdot \boldsymbol{m}_i) \\ &+ \bar{\boldsymbol{c}}_j \cdot (\boldsymbol{c}_k \times \boldsymbol{m}_i) + \bar{\boldsymbol{c}}_k \cdot (\boldsymbol{c}_j \times \boldsymbol{m}_i)) \end{aligned}$$

Somit ist nun $\boldsymbol{B}_{\bar{j},\bar{k}}$ mit $j \neq k$ durch:

$$\boldsymbol{B}_{\bar{j},\bar{k}} = \boldsymbol{B}_{\bar{k},\bar{j}} = -\sum_{(j,k)} \sum_i \boldsymbol{I}_3$$

gegeben. Ebenso ergibt sich die Submatrix $\boldsymbol{B}_{j,k}$ mit $j \neq k$:

$$\boldsymbol{B}_{j,k} = -\sum_{(j,k)} \sum_i \begin{pmatrix} \boldsymbol{m}_i^2 - \boldsymbol{m}_{x,i}^2 & -\boldsymbol{m}_{x,i}\boldsymbol{m}_{y,i} & -\boldsymbol{m}_{x,i}\boldsymbol{m}_{z,i} \\ -\boldsymbol{m}_{x,i}\boldsymbol{m}_{y,i} & \boldsymbol{m}_i^2 - \boldsymbol{m}_{y,i}^2 & -\boldsymbol{m}_{y,i}\boldsymbol{m}_{z,i} \\ -\boldsymbol{m}_{x,i}\boldsymbol{m}_{z,i} & -\boldsymbol{m}_{y,i}\boldsymbol{m}_{z,i} & \boldsymbol{m}_i^2 - \boldsymbol{m}_{z,i}^2 \end{pmatrix}.$$

Wie vorher ergeben sich auch $\boldsymbol{B}_{\bar{j},k}$ sowie $\boldsymbol{B}_{j,\bar{k}}$:

$$\boldsymbol{B}_{j,\bar{k}} = \boldsymbol{B}_{\bar{j},k} = \sum_{(j,k)} \sum_{i} \begin{pmatrix} 0 & \boldsymbol{m}_{z,i} & -\boldsymbol{m}_{y,i} \\ -\boldsymbol{m}_{z,i} & 0 & \boldsymbol{m}_{x,i} \\ \boldsymbol{m}_{y,i} & -\boldsymbol{m}_{x,i} & 0 \end{pmatrix}.$$

3.3.2 Zusammenfassung der Umformungen

Zur besseren Übersicht folgen hier nun nochmal die erweiterten Belegungsvorschriften der Subvektoren $\boldsymbol{a}_j$ des Vektors $\boldsymbol{a}_{ges}$ sowie die Belegung der Matrix $\boldsymbol{B}_{ges}$:

Der $6(N-1) \times 1$ Vektor $\boldsymbol{a}_{ges} = (\boldsymbol{a}_2{}^{\mathrm{T}}, \boldsymbol{a}_3{}^{\mathrm{T}}, \ldots, \boldsymbol{a}_N{}^{\mathrm{T}})^{\mathrm{T}}$ besteht aus den 6×1 Vektoren $\boldsymbol{a}_j$, welche folgende Form haben:

$$\boldsymbol{a}_j = \sum_{(j,k)} \sum_{i} \begin{pmatrix} \boldsymbol{m}_{i,j} \times (\boldsymbol{m}_{i,j} - \boldsymbol{d}_{i,k}) \\ (\boldsymbol{m}_{i,j} - \boldsymbol{d}_{i,k}) \end{pmatrix} + \sum_{(k,j)} \sum_{i} \begin{pmatrix} \boldsymbol{m}_{i,k} \times (\boldsymbol{d}_{i,j} - \boldsymbol{m}_{i,k}) \\ (\boldsymbol{d}_{i,j} - \boldsymbol{m}_{i,k}) \end{pmatrix}$$

Die $6(N-1) \times 6(N-1)$ Matrix $\boldsymbol{B}_{ges}$ wird nun kompakter mit 6×6 Submatrizen $\boldsymbol{B}_{j,k}$ dargestellt:

$$\boldsymbol{B}_{ges} = \begin{pmatrix} \boldsymbol{B}_{2,2} & \boldsymbol{B}_{2,3} & \ldots & \boldsymbol{B}_{2,N} \\ \boldsymbol{B}_{3,2} & \boldsymbol{B}_{3,3} & & \\ \vdots & & \ddots & \vdots \\ \boldsymbol{B}_{N,2} & & \ldots & \boldsymbol{B}_{N,N} \end{pmatrix}.$$

Auf der Blockdiagonalen von $\boldsymbol{B}_{ges}$ lautet dann die die Belegungsvorschrift für die einzelnen Submatrizen $\boldsymbol{B}_{j,j}$:

$$\boldsymbol{B}_{j,j} = \sum_{\substack{(j,k) \\ (k,j)}} \sum_{i} \begin{pmatrix} \boldsymbol{m}_i^2 - \boldsymbol{m}_{x,i}^2 & -\boldsymbol{m}_{x,i}\boldsymbol{m}_{y,i} & -\boldsymbol{m}_{x,i}\boldsymbol{m}_{z,i} & 0 & -\boldsymbol{m}_{z,i} & \boldsymbol{m}_{y,i} \\ -\boldsymbol{m}_{x,i}\boldsymbol{m}_{y,i} & \boldsymbol{m}_i^2 - \boldsymbol{m}_{y,i}^2 & -\boldsymbol{m}_{y,i}\boldsymbol{m}_{z,i} & \boldsymbol{m}_{z,i} & 0 & -\boldsymbol{m}_{x,i} \\ -\boldsymbol{m}_{x,i}\boldsymbol{m}_{z,i} & -\boldsymbol{m}_{y,i}\boldsymbol{m}_{z,i} & \boldsymbol{m}_i^2 - \boldsymbol{m}_{z,i}^2 & -\boldsymbol{m}_{y,i} & \boldsymbol{m}_{x,i} & 0 \\ 0 & \boldsymbol{m}_{z,i} & -\boldsymbol{m}_{y,i} & 1 & 0 & 0 \\ -\boldsymbol{m}_{z,i} & 0 & \boldsymbol{m}_{x,i} & 0 & 1 & 0 \\ \boldsymbol{m}_{y,i} & -\boldsymbol{m}_{x,i} & 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.20}$$

Außerhalb der Blockdiagonalen lautet sie für die Submatrizen $\boldsymbol{B}_{j,k}$:

$$\boldsymbol{B}_{j,k} = \sum_{(j,k)} \sum_{i} \begin{pmatrix} \boldsymbol{m}_{x,i}^2 - \boldsymbol{m}_i^2 & \boldsymbol{m}_{x,i}\boldsymbol{m}_{y,i} & \boldsymbol{m}_{x,i}\boldsymbol{m}_{z,i} & 0 & \boldsymbol{m}_{z,i} & -\boldsymbol{m}_{y,i} \\ \boldsymbol{m}_{x,i}\boldsymbol{m}_{y,i} & \boldsymbol{m}_{y,i}^2 - \boldsymbol{m}_i^2 & \boldsymbol{m}_{y,i}\boldsymbol{m}_{z,i} & -\boldsymbol{m}_{z,i} & 0 & \boldsymbol{m}_{x,i} \\ \boldsymbol{m}_{x,i}\boldsymbol{m}_{z,i} & \boldsymbol{m}_{y,i}\boldsymbol{m}_{z,i} & \boldsymbol{m}_{z,i}^2 - \boldsymbol{m}_i^2 & \boldsymbol{m}_{y,i} & -\boldsymbol{m}_{x,i} & 0 \\ 0 & -\boldsymbol{m}_{z,i} & \boldsymbol{m}_{y,i} & -1 & 0 & 0 \\ \boldsymbol{m}_{z,i} & 0 & -\boldsymbol{m}_{x,i} & 0 & -1 & 0 \\ -\boldsymbol{m}_{y,i} & \boldsymbol{m}_{x,i} & 0 & 0 & 0 & -1 \end{pmatrix} \tag{3.21}$$

Für alle Belegungsvorschriften gelten die Summenindizes $2 \leq j \leq N$, $1 \leq k \leq N$ und i über jeweils alle gefundenen Punktkorrespondenzen der jeweils benachbarten Scans j und k. Das Bild 3.6 verdeutlicht die Belegung der Matrix $\boldsymbol{B}_{ges}$. Sie ist außerhalb der Blockdiagonalen nur an den Stellen mit Werten ungleich Null belegt, an denen im Graphen Scans benachbart sind. Dies sind also in einer normalen Schleife wie sie bislang als Beispiel betrachtet wurde, jeweils die Blöcke direkt oberhalb und unterhalb der Blockdiagonalen. Dort wird eben die normale Folge der Scans n auf $n + 1$ betrachtet. Weit außerhalb der Diagonalen sind dann eventuelle Schleifenschlüsse eingetragen worden, im Bild 3.6 sind dies die Punkte im Eck rechts oben sowie links unten. Im Vergleich zur lokal optimierenden Helixkorrektur, bei der die Matrix nur auf der Blockdiagonalen belegt war, sorgen genau die Einträge außerhalb der Diagonalen dafür, dass Korrekturen der Scans an andere Scans innerhalb der Schleife weitergegeben werden.

3.3.3 Ergebnis der Umformungen

Betrachten wir nun hier die Ergebnisse dieser neuen Korrektur. In Bild 3.7 ist das Ergebnis des Verfahrens an der Stelle des Schleifenschlusses zu sehen. Wie bei der lokal optimierenden globalen Helixkorrektur (vergleiche Bild 3.3) ist die Doppelwand sowie der Höhenunterschied am Schleifenschluss nicht mehr zu sehen. Die beiden Enden der Schleife wurden also zusammengeführt. Schaut man genauer hin, so ist hier nun im linken Bild, der Ansicht von oben, in der rechten unteren Ecke der Bereich innerhalb der eingezeichneten Markierung sehr viel genauer korrigiert worden als an gleicher Stelle bei Bild 3.3. Doch die beabsichtigte Weitergabe der Posekorrekturen im Bereich des Schleifenschlus-

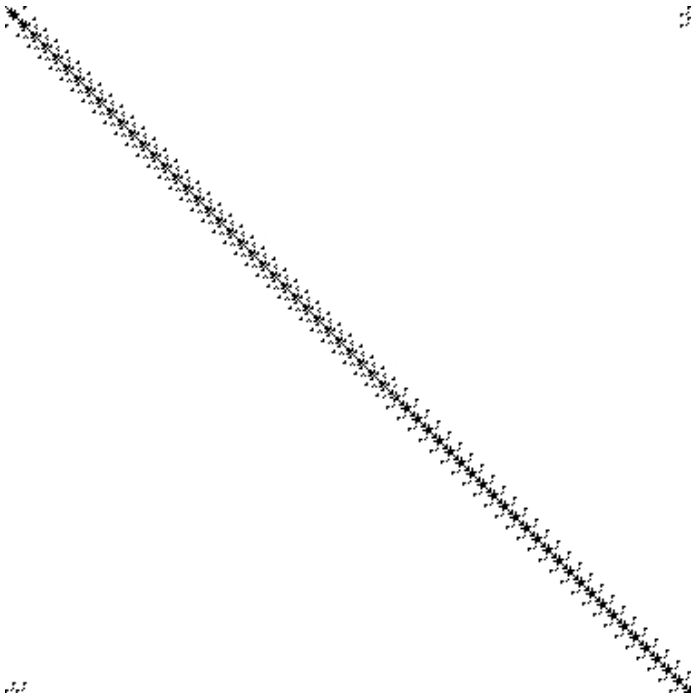

Bild 3.6: Belegungsskizze der Matrix $\boldsymbol{B}_{ges}$

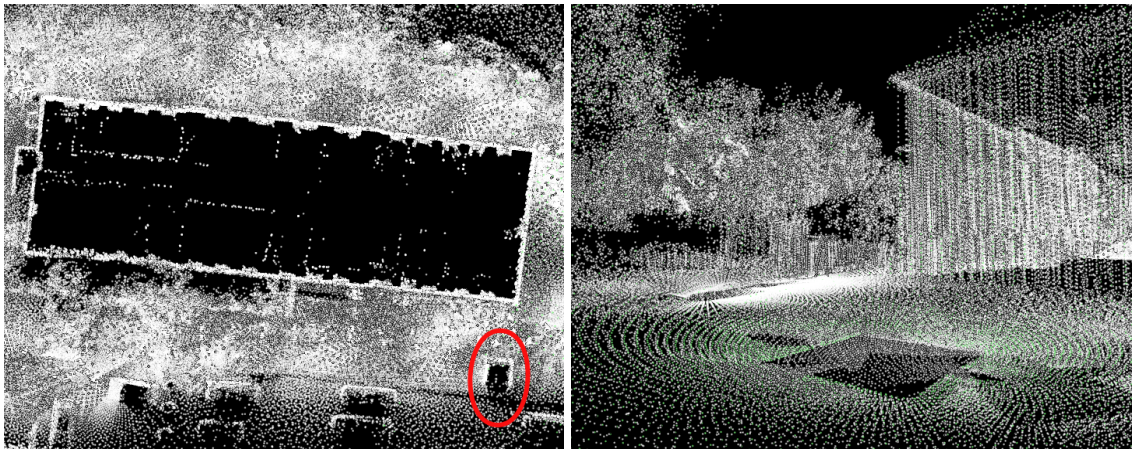

Bild 3.7: Schleifenschluss mittels global optimierender Helixkorrektur, markierter Bereich verdeutlicht bessere Korrektur

ses an die Scans innerhalb der Schleife ist hier noch nicht zu sehen. Dies ist erst beim direkten Vergleich mittels Überlagerung der Scans zu erkennen. Hierfür wurde in Bild 3.8 der mittels dem neuen Verfahren korrigierte globale Scan über den lokal optimierten globalen Scan gelegt. In Bild 3.9 wurde er zudem noch über den mit dem nach Lu & Milios korrigierten Scan gelegt. Beim dem Vergleich mit dem alten Verfahren, das eben nicht die Posekorrekturen des Schleifenschlusses weiter gibt, ist deutlich zu sehen, dass der erweiterte Algorithmus dies nun leistet. Hier werden jetzt nämlich auch die Scans innerhalb der Schleife, also die, die auf dem Bild auf der linken Seite liegen, mit verschoben. Der neue, in weiß gefärbte, Scan ist dort deutlich nach oben verschoben worden. Der Vergleich des Scans des neuen Verfahrens mit dem des Verfahrens nach Lu & Milios zeigt zu dem, dass dort im Prinzip keine Unterschiede zu erkennen sind. Es werden also bei der neuen global optimierenden Helixkorrektur im Ergebnis die Posen genau so an die dazwischen liegenden Scans weitergereicht wie dies auch bei dem Standardverfahren der Fall ist.

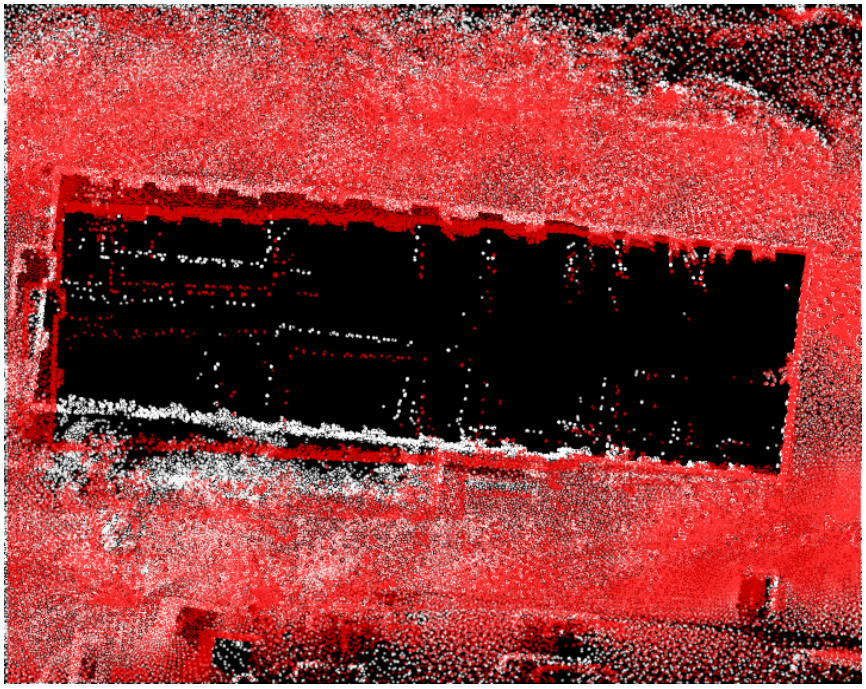

Bild 3.8: Vergleich der neuen Helixkorrektur (Weiß) mit lokal optimaler globaler Karte (Weiß)

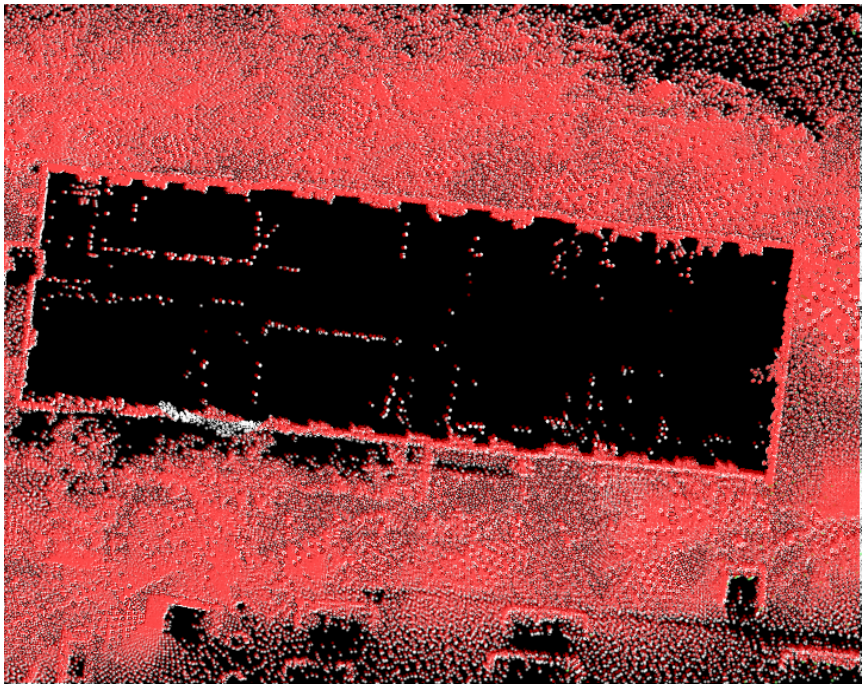

Bild 3.9: Vergleich der neuen Helixkorrektur (Weiß) mit Lu & Milios Korrektur (Rot)

Kapitel 4

Ergebnis und Leistungsvergleich

Am Ende des vorherigen Kapitels wurde gezeigt, dass die erweiterte Korrekturberechnung funktioniert und auch global optimierte Ergebnisse liefert. Doch ist diese Variante nur eine weitere Möglichkeit, das Problem des Schleifenschlusses zu lösen oder liefert es sogar bessere Ergebnisse? Dies soll in diesem Kapitel geklärt werden. Hierfür wird zum Einen ein qualitativer Vergleich erstellt, zum Anderen ein Laufzeitvergleich betrachtet. In den folgenden Abschnitten dieses Kapitels sollen die beiden Versuchsvarianten erläutert werden und im Anschluss an die jeweilige Beschreibung die Ergebnisse der jeweiligen Versuche betrachtet werden.

4.1 Der qualitative Vergleich

Eine Aussage über die Qualität einer Korrektur zu geben ist schwierig. Das erste Problem hierbei ist der reine Vergleich. Wie man in Bild 3.9 sehen kann, reicht ein einfaches Überlagern der zu vergleichenden Scans nur zu einer groben Einschätzung. Durch die Menge an Punkten gehen allerdings Details und kleinste Abweichungen verloren. Ebenso ist es schwierig, eine Aussage über „besser“und „schlechter“zu treffen wenn man nicht genau weiß, wo die korrekte Position ist. Wie diese Probleme hier gelöst wurden, soll in der Versuchsbeschreibung nun zuerst erläutert werden, bevor dann auf die Ergebnisse des qualitativen Vergleichs eingegangen wird.

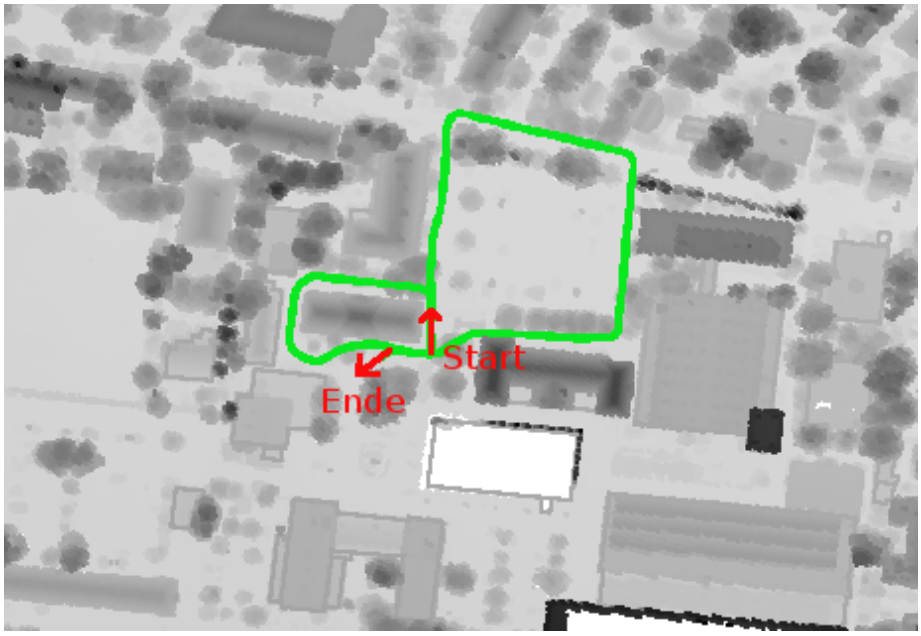

Bild 4.1: Luftbildlaserscan des abgefahrenen Areals, Roboterpfad eingezeichnet

4.1.1 Die Messung der Qualität

Für eine möglichst umfassende Qualitätsanalyse wurde eine größeres Areal der Universität Osnabrück abgefahren und dabei in gleichbleibenden Abständen lokale Scans geschossen. Der dabei zurückgelegte Weg bildete in etwa die Form einer Acht und enthält somit zwei Schleifen. Außerdem ist innerhalb des Weges ein gewisser Höhenunterschied zu überwinden. Der Pfad des Roboters aus der Luft ist auf Bild 4.1 zu sehen. Der bei dem zurückgelegten Pfad entstandene Laserscan ist in Bild 4.2 zu sehen. In seinem Zentrum, der Kreuzung der Acht sind deutlich mehrere Schleifenschlussfehler in Form von Doppelwänden zu sehen. In der 3D-Ansicht an der Kreuzung der Acht, Bild 4.3, ist außerdem deutlicher ein Höhenversatz erkennbar.

Es gilt nun zunächst einen Referenzkarte festzulegen. An diesem Richtwert sollen sich die korrigierten Karten der beiden Verfahren messen. Sie sollen mit ihrem Ergebnis möglichst nah an dieser als korrekt angenommene Referenzkarte liegen. Doch woher bekommt man eine solche Karte? Die Daten des Katasteramtes liefern meist nur unvollständige Karten,

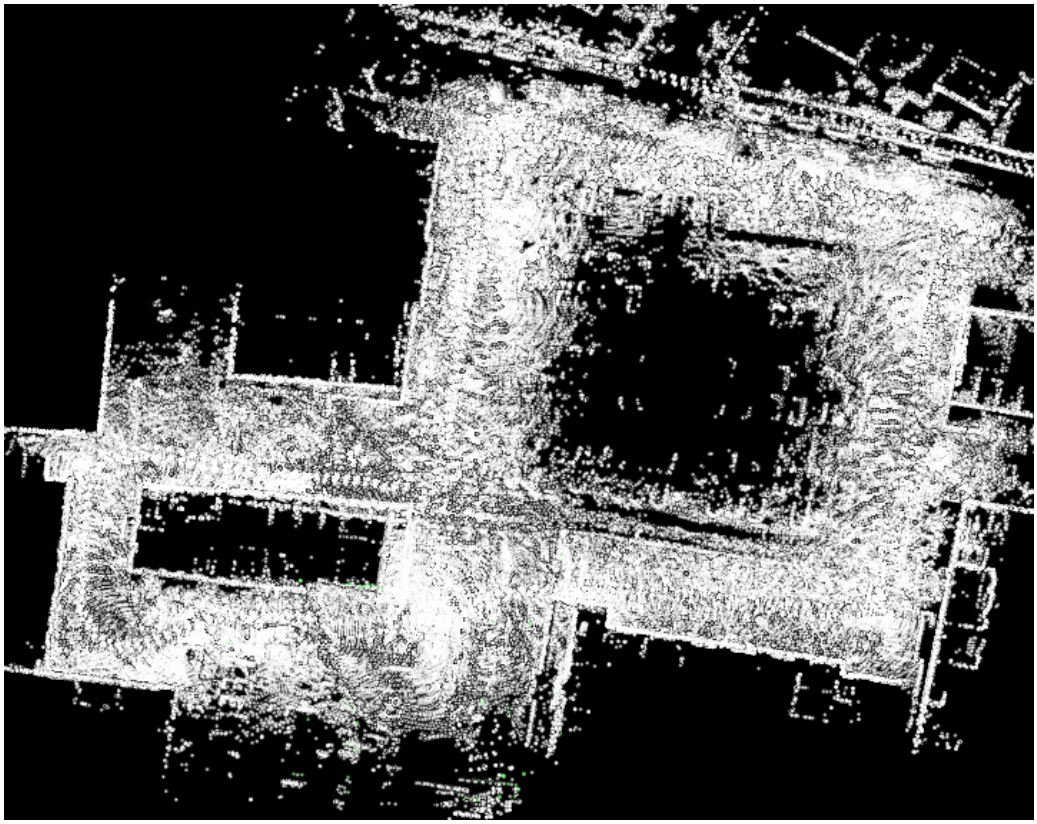

Bild 4.2: Unkorrigierter globaler Scan aus dem Pfad, der in Bild 4.1 eingezeichnet ist

es sind nicht für alle Positionen vollständige dreidimensionale Punkte vorhanden. Daher lässt sich aus diesen keine als global korrekt zu bezeichnende Karte erstellen. Auch eine per Hand oder GPS erstellte Karte weißt mehr oder weniger starke Fehler auf womit man wieder vor dem ureigentlichen SLAM-Problem steht. Doch Abhilfe für ein solches noch relativ begrenztes Areal schafft der Schritt in die Luft. Um eine nutzbare Referenzkarte zu bekommen wurde mit einem Flugzeug ein Laserscan aus der Luft geschossen. Da dieser erheblich mehr Reichweite und eine sehr viel größere Abdeckung aufweist, hat hier ein einziger Scan genügt, um das Testareal abzudecken. Daher ist für das Generieren der Referenzkarte kein Zusammensetzen einzelner Scans nötig und es kann sich somit kein Fehler einschleichen. Es ist ein lokaler Scan der groß genug ist um unseren globalen Scan abzudecken und somit auch nicht die üblichen globalen Fehler enthält. Um die Referenz-

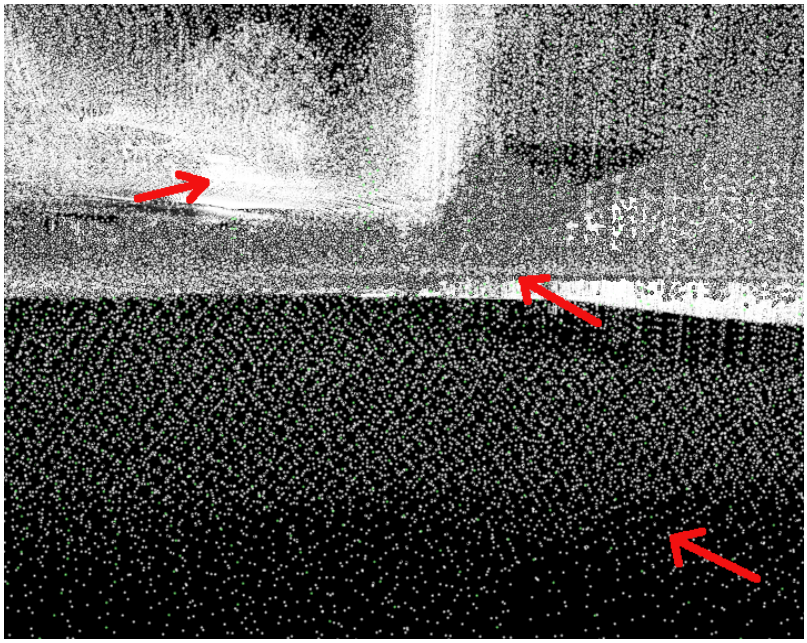

Bild 4.3: 3D-Ansicht aus dem Zentrum der Acht (markiert sind die verschiedenen falschen Böden)

posedaten der einzelnen lokalen Scans, die während der Testserie geschossen wurden, zu erlangen werden diese mittels lokalem ICP auf den Referenzluftscan registriert. Sie gelten nun als Referenzposen der jeweiligen lokalen Scans und sollten möglichst genau durch die Korrekturverfahren erreicht werden.

Dadurch ist nun auch schon festgelegt, wie die Scans miteinander verglichen werden sollen. Um eine Aussage über die Qualität des Ergebnisses treffen zu können wurde auf die Betrachtung eines gesamten Scans mit all seinen Punkten verzichtet. Statt dessen soll hier nur die Trajektorie einer gesamten Scanserie mit denen anderer Serien verglichen werden. Bei den Trajektorien handelt es sich um die Betrachtung der Roboterpose bei der Bewegung des Roboters durch den Raum. Diese ist für jeden lokalen Scan einer Scanserie bekannt und kann genau wie die Scanpunkte eines lokalen Scans innerhalb einer Aufnahmenserie verschoben und somit korrigiert werden. Wird nun die Roboterpose eines lokalen Scans mit der Roboterpose des Vorgängerscans beziehungsweise des Nachfolgerscans

verbunden so erhält man eine Annäherung an den Pfad des Roboters durch die gesamte Aufnahme. Um diese deutlicher anzeigen zu können soll hier für die Darstellung des Pfades eine Dreitafelprojektion verwendet werden. Das heißt, dass der dreidimensionale Pfad durch den Raum auf die XY-Ebene, die XZ-Ebene sowie die ZY-Ebene projiziert wird. Die Koordinatenachsen repräsentieren dabei die Achsen des Roboterkoordinatensystems am Startpunkt der Scanserie. Bei diesem Koordinatensystem handelt es sich um ein Linke-Hand-System mit Blickrichtung entlang der Z-Achse. Auf der Y-Achse wird somit der Höhenunterschied des Geländes eingetragen. Durch diese Art der Betrachtung lassen sich die Pfade der unterschiedlichen Korrekturen leichter Vergleichen. Weil es in manchen Fällen hilft ist zusätzlich zu diesen drei Projektionen noch ein dreidimensionaler Plot des Pfades dazu gestellt. Ein Beispiel für eine solche Projektion ist in Bild 4.4 zu sehen. Dort ist bei der Projektion der unkorrigierten Trajektorie auf die XZ-Ebene, sozusagen der Betrachtung von Oben, gar kein so großer Fehler zu erkennen. Wenn man aber die beiden anderen Projektionen, die die Höhenänderung (Y-Achse im Ursprungssystem) zur Verschiebung auf der X-Achse bzw. Y-Achse betrachtet und vergleicht diese mit dem tatsächlich zurückgelegten Pfad, in grün eingezeichnet, fallen einige Fehler auf. So müsste bei der XY-Verschiebung im Bereich des Nullpunktes der X-Achse ein Schnittpunkt der Trajektorie sein da an dieser Stelle der Kreuzungspunkt der gefahrenen Acht liegt und man sich daher auf einem gemeinsamen Höhenlevel befinden sollte. Auch bei der ZY-Verschiebung fehlt dieser gemeinsame Schnittpunkt. Hier liegt der Punkt, der das Ende der Trajektorie zeigt, in etwa bei den Koordinaten $(0, 400)$ und befindet sich somit weit oberhalb der Position, die an gleicher Stelle bereits vorher aufgenommen wurde.

4.1.2 Ergebnisse des qualitativen Vergleichs

Betrachten wir nun eine solche Trajektorienprojektion für das Korrekturverfahren nach Lu & Milios sowie für die global optimale Helixkorrektur. Hierfür werden zum Einen die korrigierten Trajektorien jeweils mit der Referenztrajektorie verglichen, zum Anderen die beiden Trajektorien direkt miteinander verglichen. Bei den Vergleichen der Korrekturtrajektorien mit der Referenztrajektorie (Bild 4.5 und Bild 4.6) fällt sofort auf, dass sich dort an den Positionen die weit vom Ursprung und somit dem Zentrum der Acht entfernt sind die Höhe stark verändert hat. Sogar im Vergleich zu unkorrigierten Trajektorie ist es

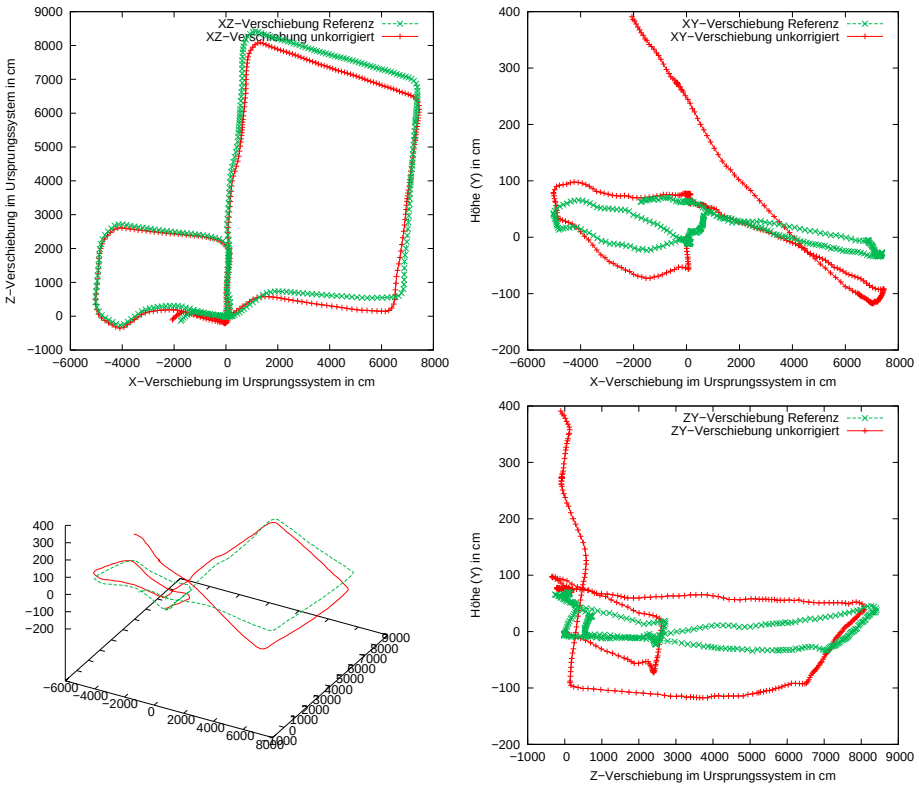

Bild 4.4: Vergleich der Referenztrajektorie mit der unkorrigierten Trajektorie

dort eher schlechter als besser geworden. Grund dafür ist der teilweise sehr große Fehler, mit dem die unkorrigierte Trajektorie im Bereich des Zentrums der Acht behaftet war. In genau diesem Bereich des Schleifenschlusses, in den Koordinatensystemen der Projektion liegt er in deren jeweiligen Ursprüngen, soll aber der Fehler minimiert werden. Und dies gelingt auch. Die korrigierten Pfade treffen sich immer wieder im Zentrum wohingegen die unkorrigierte Version gerade dort enorm von ihren vorherigen Positionen abweicht. Diese Abweichung wird bei der Korrektur auf die bisherige Wegstrecke verteilt. Da im Vergleich zum Zentrum der Acht in ihren Außenbereichen die Unsicherheit über die korrekte Lage am größten ist wird dort der Großteil des Fehlers vom Schleifenschluss aus hin verlagert. Da also vor der Korrektur vor allem die Höhe an der Stelle des Schleifenschlusses inkorrekt war ist nun genau dieser Höhenfehler nach außen gewandert und verursacht

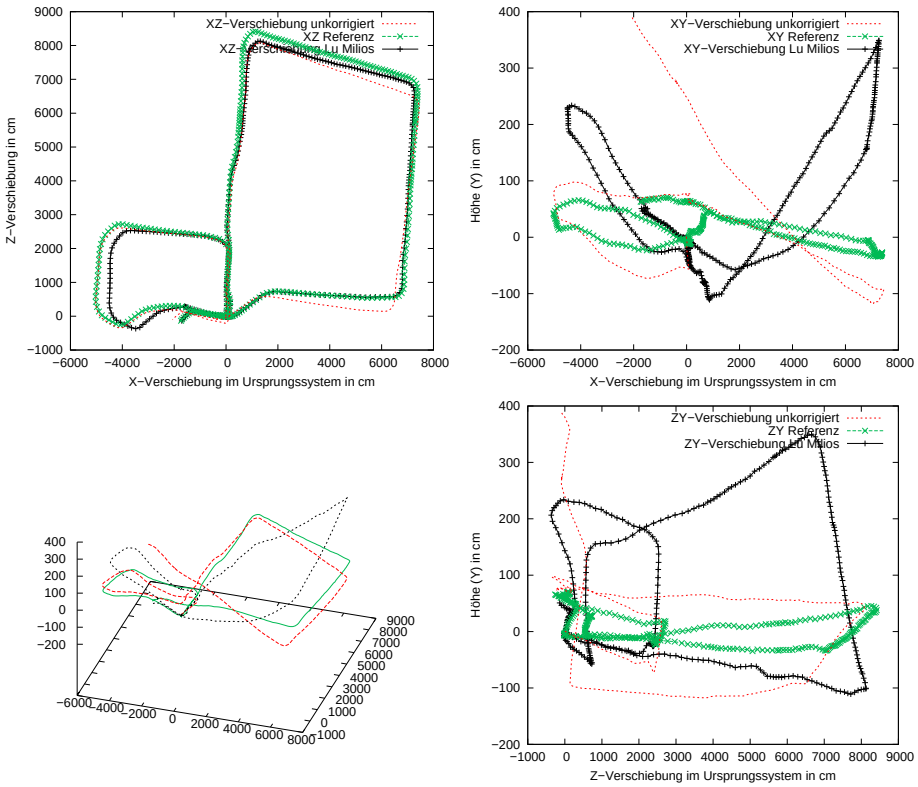

Bild 4.5: Vergleich Lu & Milios Trajektorie mit Referenz

nun dort die deutlichen Abweichungen. Der Höhenfehler am Schleifenschluss ist nicht mehr vorhanden und verursacht dort nun keine doppelten Wände oder doppelte Böden. Durch das korrekte Zusammentreffen der korrigierten Trajektorien im Zentrum ist daher nun genau die konsistente Karte erzeugt worden, die gefordert war. Sie trifft dort sogar ziemlich genau mit den Referenzposen zusammen.

Wenn man den Vergleich zwischen Helix-Trajektorie und Lu & Milios-Trajektorie betrachtet (Bild 4.7), so ist dort kaum eine Abweichung zwischen den beiden Verfahren zu erkennen. Man kann durchaus sagen, dass das qualitative Ergebnis beider Korrekturen gleichwertig ist. Beide verteilen den Fehler auf gleich Art und Weise in den Bereichen, deren Posedaten weniger gesichert sind und reduzieren ihn dadurch gleich gut im Bereich des Schleifenschlusses. Das Ergebnis beider Korrekturen sind gleichwertige Karten (sie-

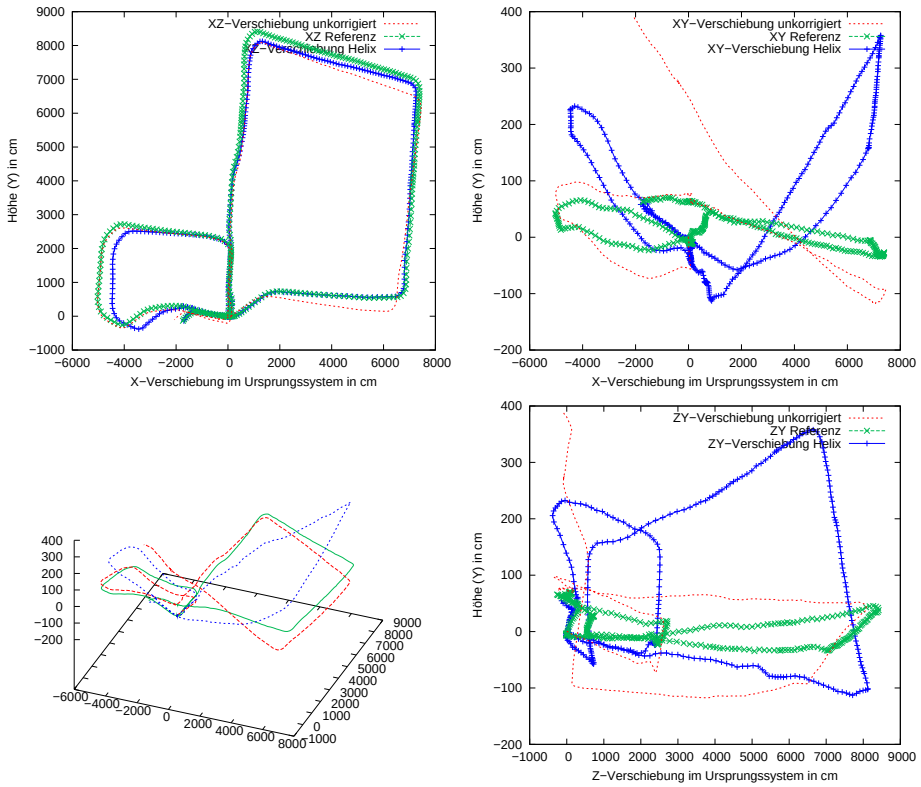

Bild 4.6: Vergleich Helix Trajektorie mit Referenz

he Bild 4.8), die im Bereich des Schleifenschlusses keine Fehler aufzeigen. Sie sind zwar nicht überall korrekt aber durchgehend konsistent. doch wo liegt nun der Vorteil des Helixverfahrens. Gibt es überhaupt einen, oder ist es nur eine weitere Möglichkeit für globales sechsdimensionales SLAM?

4.2 Der Geschwindigkeitsvergleich

Im vorhergehenden Abschnitt wurde gezeigt, dass das neue Helixverfahren qualitativ gleichwertige Ergebnisse zu der Korrektur von Lu & Milios liefert. Es stellt sich also die Frage nach dem Unterschied zwischen beiden Algorithmen. Dieser ist möglicherweise in der Ge-

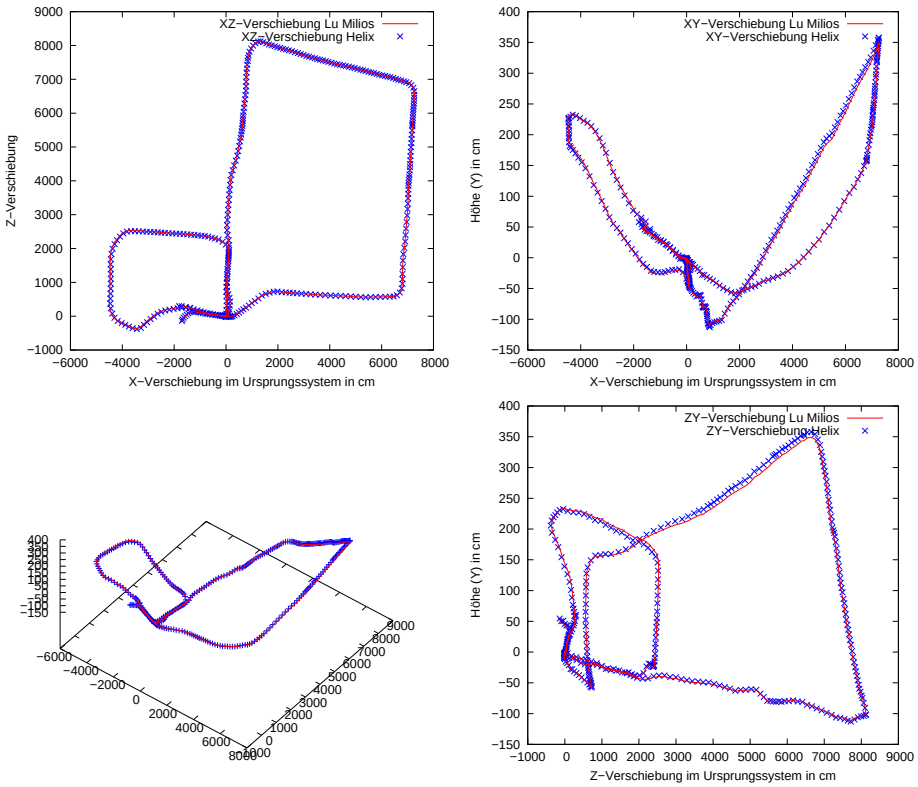

Bild 4.7: Vergleich Helix Trajektorie mit Lu & Milios Trajektorie

schwindigkeit zu finden. Dabei soll zum Einen die benötigte Laufzeit bei gleicher Anzahl Iterationen gemessen werden, zum Anderen soll verglichen werden, wie viele Iterationen jeweils benötigt werden, bis das Verfahren konvergiert. Beides soll im Folgenden zunächst separat betrachtet werden.

4.2.1 Der Laufzeitvergleich

Um einen korrekten Vergleich zu gewährleisten wurden die Laufzeitmessung immer mit immer gleichen Parametern durchgeführt. So wurden für die lokale Korrektur immer 50 Iterationen des ICP-Algorithmus mit Quaternionenkorrektur gewählt, bei der globalen Korrektur wurden immer 25 Iterationen ausgeführt. Außerdem wurden die einzelnen Ver-

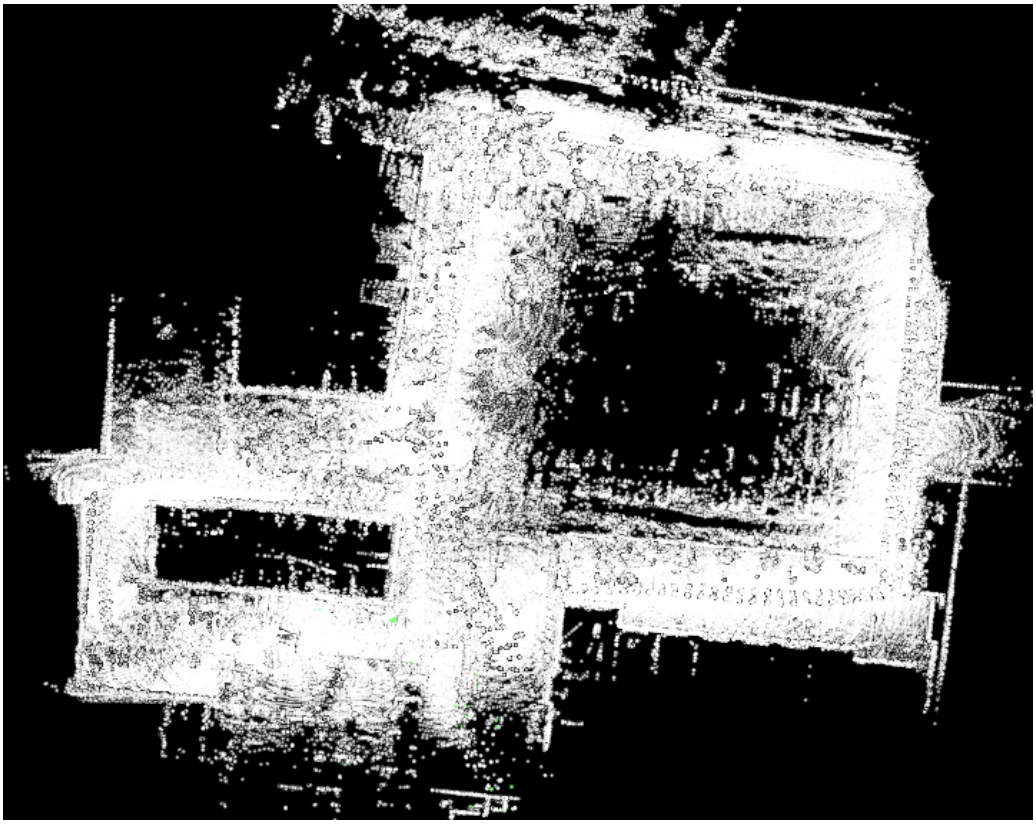

Bild 4.8: Helixkorrektur von Bild 4.2

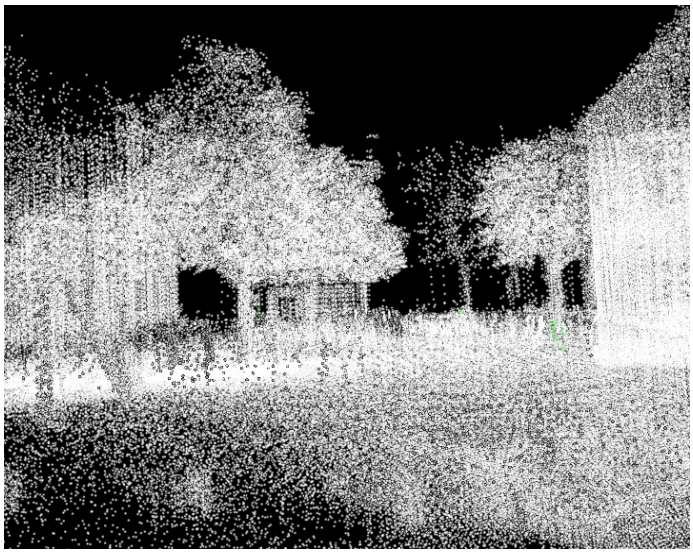

Bild 4.9: 3D-Ansicht nach Korrektur von Bild 4.3

gleiche sowohl für die singlethreaded Version als auch für die multithreaded Version aufgestellt. Dies soll auch eine Laufzeitmessung ermöglichen, die unabhängig von eventuell anderer oder besserer Parallelisierung ist. Die singlethreaded Version wurde dabei mit dem GNU-C-Compiler Version 4.2.3 erstellt, für die multithreaded Variante wurde der Intel-C-Compiler und die OpenMP-Multithreading-Bibliothek verwendet. Diese erzeugt nicht schon von Prozessstart an mehrere Threads sondern startet diese erst wenn vorher markierte aufwändige Programmpassagen wie z. B. Schleifen abgearbeitet werden. Da die Matrixbelegung innerhalb einer recht aufwändigen Schleife vonstatten geht und den Großteil der Laufzeit verschlingt, bietet sich diese Bibliothek zur Implementierung des Multithreading hier an. Des weiteren wurden unterschiedlich große Scanserien betrachtet die auch verschiedenartig komplexe Verknüpfungen aufweisen. Dies soll die Abhängigkeit von Größe und Belegung der zur Korrektur benötigten Matrix zeigen. So wurde für den einfachen Fall die Scanserie mit nur einer einfachen Schleife und insgesamt 65 einzelnen lokalen Scans verwendet, die auch in den Beispielen von Kapitel 3 zu sehen ist (Beispielsweise in Bild 3.7). Als Beispiel für eine kompliziertere Scanserie wurde der aus 363 lokalen Scans bestehender globale Scan aus Kapitel 4.1 verwendet, dessen Verlauf eine Acht bildet, wodurch bei der Korrekturmatrix an mehr als nur den üblichen Stellen, wie an Schleifenbeginn und Schleifenende, Einträge vorhanden sind. Zudem wurde noch unterschieden zwischen der Laufzeit des gesamten Programms und der Laufzeit die nur der globale Korrekturalgorithmus benötigt. Während die Programmlaufzeit auch noch das Einlesen der Scans und die lokale Korrektur umfasst misst die Algorithmuslaufzeit nur die Zeit, die für die globale Korrektur mit all ihren Schritten benötigt wurde. Es ist klar, dass die Korrektur einer Scanserie mit vielen lokalen Scans auch länger für das Einlesen und den lokalen ICP benötigt als bei einer Serie mit nur recht wenigen Scans. Die Betrachtung der Algorithmuslaufzeit ist daher die aussagekräftigere. Unter diesen Voraussetzungen wurden Tabelle 4.1 und Tabelle 4.2 erstellt. Diese zeigen für den Singlethreaded-Fall, dass der Helixalgorithmus bei gleicher Anzahl Iterationen etwa 14% schneller ist als der Algorithmus nach Lu & Milios. Auch die Verbesserung der Programmlaufzeit nähert sich diesem Wert an, je mehr Scans die Serie umfasst. Dies liegt an dem gestiegenen Anteil der Zeit die für die globale Korrektur benötigt wird im Vergleich zu der Zeit, die für die lokalen Korrekturen und zum Einlesen der Scans benötigt wird. Die Ursache für den geringeren Zeitaufwand der Helixkorrektur ist in der Belegung der Matrix $\boldsymbol{B}_{ges}$ zu finden. Während

# Scans	Was wurde betrachtet?	Lu & Milios	Helix	Verbesserung
65	Programm	40 s	37,5 s	6,2%
	Algorithmus	17,5 s	14,9 s	14,8%
336	Programm	431 s	382 s	11,3%
	Algorithmus	354 s	305 s	13,8%

Tabelle 4.1: Singlethreaded Laufzeitvergleich

# Scans	Was wurde betrachtet?	Lu & Milios	Helix	Verbesserung
65	Programm	38,8 s	34,0 s	12%
	Algorithmus	12,7 s	9,0 s	29%
336	Programm	359 s	263 s	26,7%
	Algorithmus	270 s	174 s	35,5%

Tabelle 4.2: Multithreaded Laufzeitvergleich

diese durch eine einfache Summe über alle Punkte und benachbarte Scans aufgestellt wird, müssen bei Lu & Milios zu Belegung der Hauptmatrix noch Submatrizen invertiert werden und eine Fehlerabschätzung geleistet werden (siehe Kapitel 2.2.4). Der Grund für die nochmalige Verbesserung durch Multithreading circa 30% Laufzeitersparnis ist in einer anderen Parallelisierung der Helixkorrektur zu suchen. diese ist so implementiert, dass nicht nur das Aufsummieren der Submatrizen parallel geschieht sondern auch das darauf folgende Belegen der Hauptmatrix. Dieses muss zwar gegen zeitgleichen Zugriff durch einen Mutex geschützt werden, sorgt dann aber für eine erneute Beschleunigung.

4.2.2 Der Konvergenzvergleich

Im nun folgenden Abschnitt soll verglichen werden wie schnell der Algorithmus nach Lu & Milios und der globale Helixalgorithmus konvergieren. Falls sich einer schneller seinem Konvergenzwert nähert als der andere, bedeutet dies, das er weniger Iterationen benötigt, um diesen zu erreichen. Wie in Kapitel 4.1.2 gezeigt, konvergieren beide Verfahren gegen die selben Werte. Somit würde ein schneller konvergierender Algorithmus das gleiche Ergebnis bei weniger Iterationen und somit eventuell auch weniger Laufzeit bedeuten.

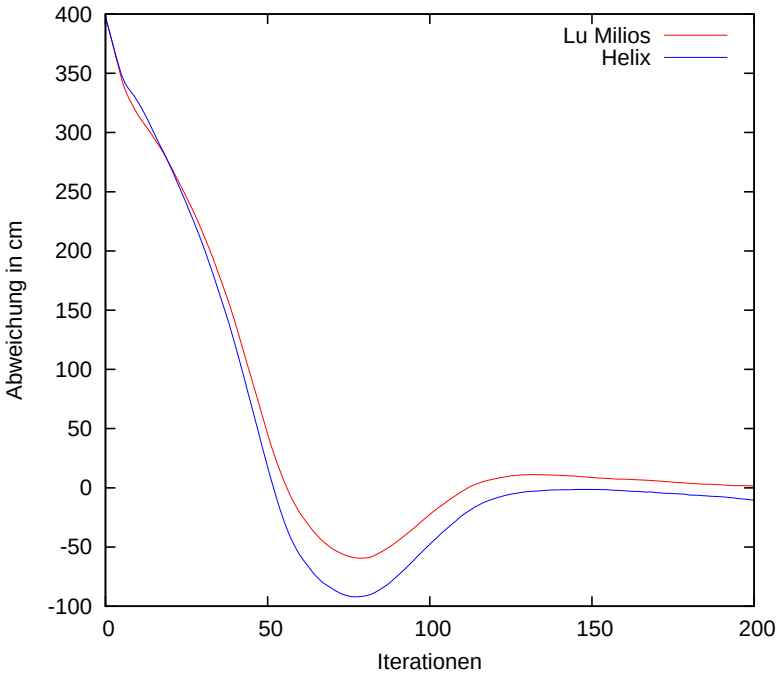

Bild 4.10: Konvergenzgraph des 60. Scans

Um dies zu ermitteln werden im Folgenden Graphen aufgestellt, die für den jeweils gleichen Punkt der Trajektorie den Abstand zum Konvergenzwert pro Iteration einmal für die Helixkorrektur sowie für die Lu & Milios-Korrektur aufzeigen. Als Referenz für den Konvergenzwert wurde der Roboterpositionsabstand zum Ursprung korrigiert durch das Verfahren nach Lu & Milios gewählt. Die Y-Achse zeigt also die Abweichung von diesem Wert auf. Die Graphen des Konvergenzvergleiches wurden wieder mit der aus 363 einzelnen Scans bestehenden Scanserie aus Kapitel 4.1, die eine Acht bildet, erstellt. Zur Auswahl der lokalen Scans ist zu sagen, dass zwei aus den Randbereichen der Acht stammen, nämlich Scan Nummer 60 und 180, und zwei aus dem Zentrum der Acht, Nummer 130 und 362, wo der Schleifenschluss stattfindet. Für jede dieser Scans wurde jeweils die Lu & Milios Korrektur als auch die Helixkorrektur ermittelt. Die sich daraus ergebenden Graphen sind in Bild 4.10, Bild 4.11, Bild 4.12 und Bild 4.13 für die einzelnen Scans zu sehen. Dort fällt der Unterschied beim Erreichen des Konvergenzwertes teilweise sehr

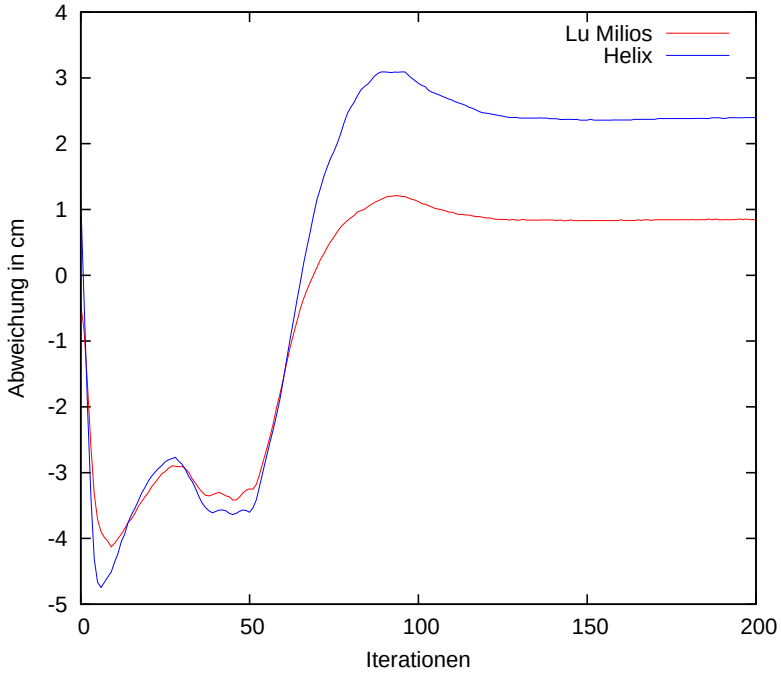

Bild 4.11: Konvergenzgraph des 130. Scans

deutlich aus. In Bild 4.10 erreicht das Verfahren nach Lu & Milios seinen Konvergenzwert bereits 20 Iterationen vor der Helixkorrektur, in Bild 4.12 sogar 25 Iterationen früher. Hierzu muss aber gesagt werden, dass diese beiden Messungen zu den außen liegenden Scans gehören. Diese spielen daher bei der Registrierung des Schleifenschlusses keine so große Rolle. Die Konvergenzkurve der Scans aus dem Zentrum der Acht, Bild 4.11 und Bild 4.13, zeigen da schon ein anderes Bild. Hier konvergieren beide Verfahren zur annähend gleichen Iteration. Auch erreichen alle Scans, die außerhalb des Zentrums liegen, ihren jeweiligen Konvergenzwert bevor dieser am Schleifenschluss erreicht wird. Der teilweise deutliche Iterationsunterschied der Verfahren bei äußeren Scans hat daher keinen Auswirkung auf das Ergebnis.

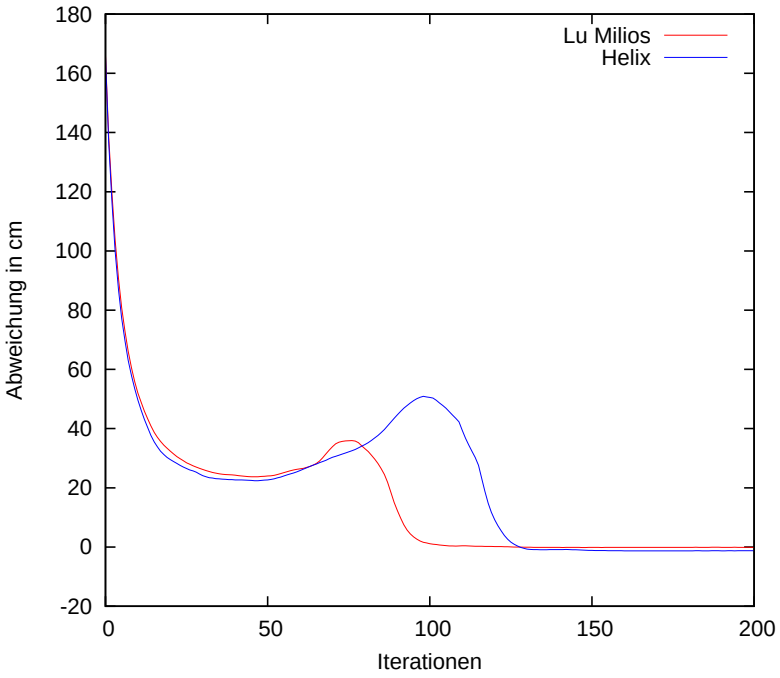

Bild 4.12: Konvergenzgraph des 180. Scans

4.3 Ergebnisse

Durch diese durchgeführten Experimente hat sich gezeigt, dass sich beide Verfahren in Puncto Ergebnisqualität in nichts nachstehe. Beide liefern zwar keine korrekte, aber global konsistente Karten. Auch erreichen beide den Konvergenzwert an den wichtigen Stellen zeitgleich. Dadurch kann das Iterieren des Algorithmus bei gleicher Anzahl Iterationen abgebrochen werden. Der Laufzeitvorteil der Helixkorrektur, bei der auf Grund der einfacheren Matrixbelegung weniger Zeit pro Iteration vergeht, schlägt damit voll zu Gunsten der Helixkorrektur. Um es in nackten Zahlen auszudrücken: Bei großen Scans ist das gesamte Korrekturprogramm mittels des Helixverfahrens in der singlethreaded Version 14% schneller. Bei Verwendung der derzeitigen multithreaded Implementierung ist es sogar 32% schneller als die Korrektur nach Lu & Milios. Die Qualität der Karte bleibt dabei gleich. Bei kleineren Scans schrumpft der Geschwindigkeitszuwachs ein wenig, da dort

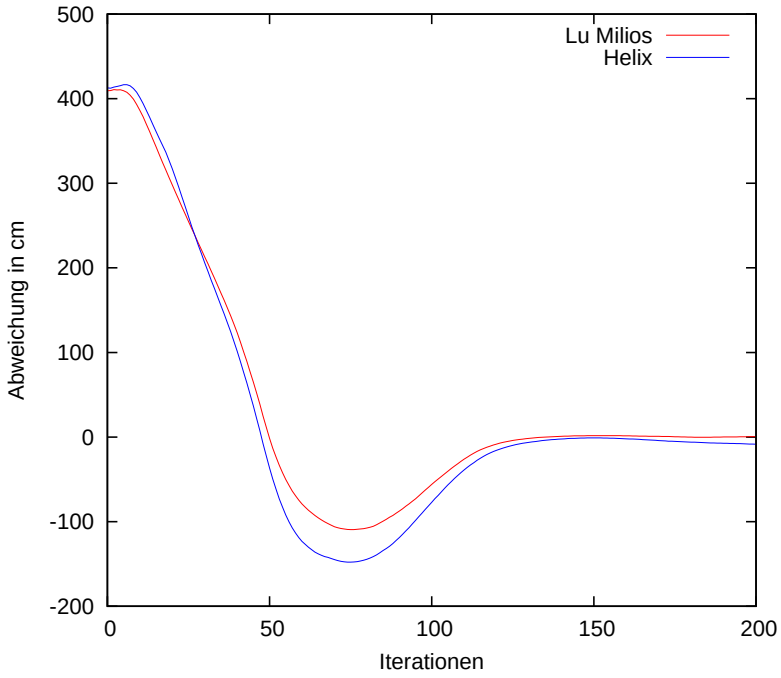

Bild 4.13: Konvergenzgraph des 362. Scans

der Anteil der globalen Korrektur geringer ausfällt, ist aber dennoch vorhanden. Allerdings ist die Ergebniskarte bei keinem der Verfahren korrekt, sie erfüllt nur das geforderte Kriterium der Konsistenz.

Kapitel 5

Fazit

Das Helixkorrekturverfahren liefert eine weitere Möglichkeit zur Lösung von sechsdimensionalem Simultanious Localising and Mapping. Es kann dabei in der erweiterten Version, der global optimierenden, auch zur globalen Korrektur, dem Beheben des Schleifenschlussfehlers, verwendet werden. Die Ergebniskarten sind dabei im Vergleich zu den Karten des Referenzverfahrens nach Lu & Milios qualitativ gleichwertig. Der Vorteil dieses neuen Verfahrens ist dabei die kürzere Laufzeit pro Iteration. Dadurch wird die benötigte Zeit für die globale Korrektur um 14% verringert. Dennoch wird für das Verfahren noch viel zu viel Zeit benötigt, um es in Echtzeit auf einem autonom fahrenden Roboter einzusetzen. Selbst wenn dort die globale Korrektur erst bei tatsächlich gefahrenen Schleifen starten und auf den vorher bereits lokal korrigierten Scans arbeiten würde, wäre die Weiterfahrt des Roboters für einige Zeit unterbrochen. Es müsste zunächst auf die korrigierten Kartendaten gewartet werden um auf diesen dann den weiteren Pfad zu planen. Bei der momentan benötigten Zeit ist diese globale Korrektur also nur für die nachträgliche 3D-Kartenerstellung nutzbar.

Ein weiteres Problem bei der Anwendung dieses Korrekturverfahrens auf autonom aufgenommene 3D-Scans zeigt Bild 5.1. Auf Grund der Bedingungen, die für das Aufstellen des Nachbarschaftsgraphen der Scans gestellt wurden, müssen zwei aufeinander folgende Scans einer Scanserie immer direkt benachbart sein, das heißt, es müssen Korrespondenzen zwischen ihnen vorhanden sein. Dies muss bei der Aufnahme der Scans beachtet

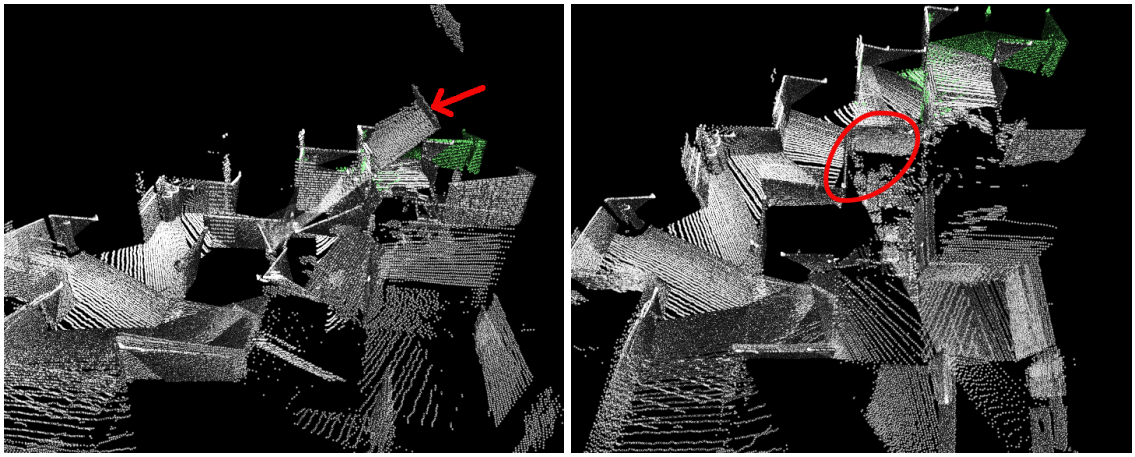

Bild 5.1: Autonom aufgenommene Scanserie (Links: unbearbeitet, fehlerhafter Scan markiert; Rechts: bearbeitet, Schleifenschluss markiert)

werden. Im autonomen Fall ist dies nicht unbedingt trivial zu lösen und benötigt eine weitere Betrachtung. Denn genau die Missachtung dieser Bedingung ist der Grund, weswegen im linken Bild die Initialposen einiger Scans soweit abweichen, dass eine Korrektur nicht mehr möglich ist. Erst durch eine manuelle Reorganisation der lokalen Scans konnte diese Bedingung erfüllt werden. So wurde damit im rechten Bild auch tatsächlich eine Schleife geschlossen und eine globale Korrektur eingeleitet. Dieses Problem trifft allerdings sowohl auf die Helixkorrektur als auch auf die Korrektur nach Lu & Milios zu.

5.1 Ausblick

Eine weitere Beschleunigung des Verfahrens ist möglicherweise durch Auslagerung der Matrixbelegung und -berechnung auf die GPU zu erreichen. Da hier bei der Helixkorrektur nur einfache Summierung der jeweils korrespondierenden Punkte benötigt wird, sollte dies durchaus leicht auf der GPU zu implementieren sein. Durch die dort vorhandenen parallelen Rechenpiplines könnte dann aufgrund gleichzeitiger Berechnung eben ein großer Geschwindigkeitszuwachs erzielt werden. Außerdem würde damit die CPU entlastet die in dieser Zeit dann für andere Aufgaben wie zum Beispiel der Punktkorrespondenzbildung

zur Verfügung stünde.

Es bleibt abschließend zu sagen, dass in diesem Forschungsbereich sicher noch einige weitere Entwicklungen in Zukunft zu erwarten sind, die für weiteren Geschwindigkeitszuwachs sorgen werden. Ob das Verfahren aber jemals in Echtzeit einsetzbar sein wird, bleibt abzuwarten.

Literaturverzeichnis

[AHB87] K. S. Arun, Thomas S. Huang, and S. D. Blostein. Least-squares fitting of two 3-D point sets. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 9(5):698–700, 1987.

[LM97] F. Lu and E. Milios. Globally consistent range scan alignment for enviroment mapping. *Autonomous Robots*, (4):333–349, 1997.

[NÖ6] Andreas Nüchter. *Semantische dreidimensionale Karten für autonome mobile Roboter*. PhD thesis, Rheinische Friedrich-Wilhelms-Universität Bonn, 2006.

[NBE+08] Andreas Nüchter, Dorit Borrmann, Jan Elseberg, Kai Lingemann, and Joachim Hertzberg. Globally consistent 3d mapping with scan matching. *Journal of Robotics and Autonomous Systems*, 56(2):130–142, 2 2008.

[PH03] Helmut Pottmann and Michael Hofer. Orientierung von laserscanner-punktwolken. *Vermessung & Geoinformation*, (91):297–306, 2003.

[PLH02] Helmut Pottmann, Stefan Leopoldseder, and Michael Hofer. Simultaneous registration of multiple views of a 3d object. *ISPRS Archives*, 34(3A):265–270, 2002.

[Sch07] Peter Schneider. Implementierung von 6d slam auf basis eines schnellen icp-algorithmus. Master's thesis, Universität Koblenz-Landau, www.uni-koblenz.de, 2007.

[Wir07] Stephan Wirth. Autonome gründliche exploration unbekannter innenräume mit dem mobilen roboter robbie. Master's thesis, Universität Koblenz Landau, 2007.