Technische Universität Ilmenau
Fakultät für Informatik und Automatisierung
Fachgebiet Neuroinformatik und Kognitive Robotik

Automatische Ermittlung der extrinsischen Parameter von Tiefensensoren unter Nutzung von ORB-SLAM und Kopplung von ORB-SLAM mit NDT-Mapping

Masterarbeit zur Erlangung des akademischen Grades
Master of Science

Tim van der Grinten

Betreuer: Dr.-Ing. Steffen Müller
Verantwortlicher Hochschullehrer:
Prof. Dr. H.-M. Groß, FG Neuroinformatik und Kognitive Robotik

Die Masterarbeit wurde am 28.07.2017 bei der Fakultät für Informatik und Automatisierung der Technischen Universität Ilmenau eingereicht.

Danksagung

Hiermit möchte ich mich bei allen Menschen bedanken, die mir während meiner Arbeit stets mit Rat und Tat zur Seite gestanden haben und ohne die diese Arbeit nicht möglich gewesen wäre.

Ein besonderer Dank gilt dabei meinem Betreuer Dr. Steffen Müller für die Bereitschaft, das Thema dieser Masterarbeit zu betreuen und zu unterstützen. An dieser Stelle möchte ich mich für die gute Zusammenarbeit sowie die Anregungen und konstruktiven Kritiken bedanken. Auch anderen Mitarbeitern des Fachgebietes *Neuroinformatik und Kognitive Robotik* sowie dessen Leiter Prof. Dr. H.-M. Groß, die mir mit wertvollen Tipps und Hilfestellungen sehr weitergeholfen haben, gilt mein Dank.

Weiterhin möchte ich meinen Eltern dafür danken, dass sie mir diese Ausbildung ermöglicht und mich während meines gesamten Studiums auf allen erdenklichen Wegen unterstützt haben. Vielen Dank, ihr seid die Besten!

Auch meinen Freunden und Kommilitonen danke ich für die Motivation und die Aufmunterungen während des gesamten Studiums und speziell der Zeit dieser Masterarbeit. Unsere gemeinsame Zeit in Ilmenau war unvergesslich und ich freue mich darauf, wenn wir uns in einigen Jahren alle zusammen wieder treffen, um über alte Zeiten zu plaudern.

Zu guter Letzt möchte ich noch besonders meiner Familie danken, die mich stets in meiner Arbeit bestärkt, mir Mut zugesprochen, den Rücken freigehalten oder einfach durch ein Lächeln die Motivation gegeben hat, diese Arbeit und damit meine Ausbildung erfolgreich abzuschließen.

Erklärung: „Hiermit versichere ich, dass ich diese Masterarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe. Alle von mir aus anderen Veröffentlichungen übernommenen Passagen sind als solche gekennzeichnet.“

Ilmenau, 28.07.2017 .....

Tim van der Grinten

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Mobile Roboterplattformen sind heutzutage in der Forschung sehr häufig anzutreffen. Ihre Navigation beruht dabei meist auf Karten, die mit den Daten entsprechender am Roboter befestigter Sensoren berechnet wurden. Mit „Simultaneous Localization and Mapping (SLAM)“-Algorithmen ist ein gleichzeitiges Mapping und die Lokalisation des Roboters in der so erhaltenen Karte möglich. Dies ist ein grundlegendes Verfahren fast aller mobilen Plattformen.

Mit einzelnen Sensoren (Laserscanner oder Kameras) ist der Kartenbau recht einfach realisierbar, sollen jedoch mehrere Sensoren zusammengefasst werden, setzt dies genaue Kenntnisse über die Anordnung der Sensoren auf der Roboterplattform voraus (s. Abb. 1.1). Eine ungenaue Kalibrierung z.B. der Kameras würde zu einem Versatz in den Aufnahmen führen, was wiederum zu ungenauen oder sogar fehlerhaften Karten führt (s. Abb. 1.2).

Eine Lösung für dieses Problem ist die Registrierung der Kameras, bei der die genauen Transformationen der Sensoren zueinander bestimmt werden.

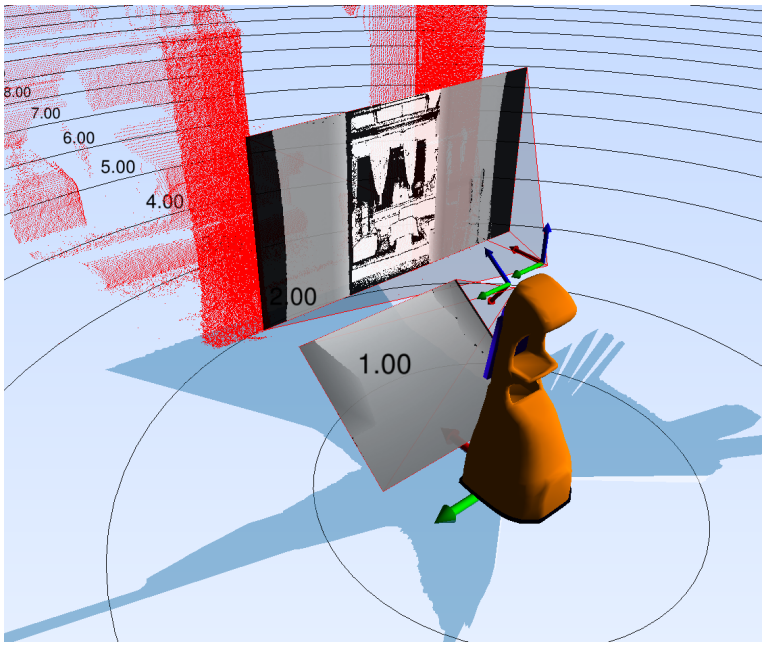


Abbildung 1.1: Anordnung mehrerer Sensoren auf einem mobilen System. *orange*: eingesetzter Roboter mit Basiskoordinatensystem, *dunkelblau*: Visualisierung der Aufnahme des Laserscanners, *grau*: aufgenommene Tiefenbilder der am Roboter befestigten Tiefenkameras mit entsprechenden Koordinatensystemen, *rot*: aus dem oberen Kamerabild rekonstruierte 3D-Punktwolke.

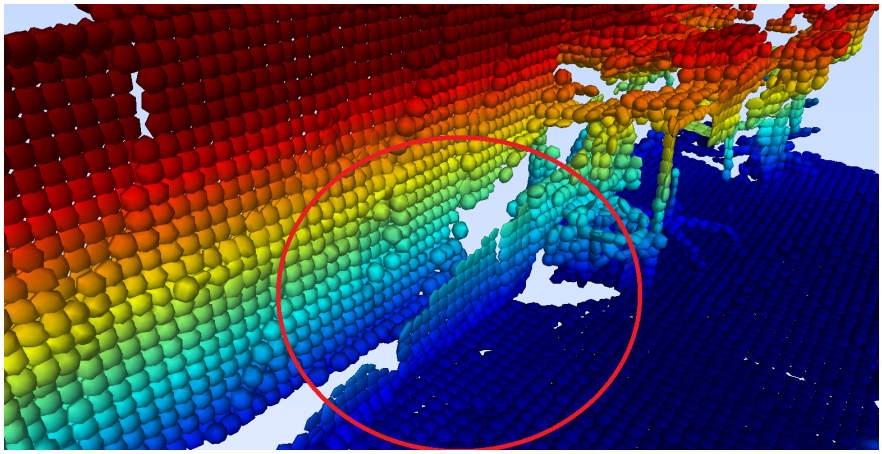


Abbildung 1.2: Fehlerhaft zusammengesetzte Karte. Zu sehen sind die Normalverteilungen der NDT-Zellen, welche entsprechend ihrer Höhe farblich kodiert sind. Deutlich zu erkennen ist im unteren, mittleren Bereich des Bildes der Versatz zwischen den rekonstruierten 3D-Daten aus den Aufnahmen zweier Kameras (*rot eingekreist*).

1.2 Ziel der Masterarbeit

Ziel dieser Masterarbeit ist die Konzeption und Implementierung eines Verfahrens, welches die unbekannten Posen von Tiefenkameras wie beispielsweise der *Microsoft Kinect v2* auf der Plattform berechnet und ausgibt. Dazu soll ein schon vorhandenes Mapping-Tool umgebaut und um weitere Komponenten erweitert werden.
Des Weiteren wird eine Literaturrecherche zum State-of-the-Art der Kameraregistrierung durchgeführt und vorhandene und im Internet frei verfügbare SLAM-Algorithmen werden in die Roboter-Middleware for Robotic Applications (MIRA) integriert. Darüber hinaus werden alternative Möglichkeiten diskutiert und eine Abschätzung der erreichbaren Genauigkeit und Eignung des entwickelten Verfahrens vorgenommen, zu deren Zweck geeignete Datensätze aufgenommen werden.

1.3 Bedeutung für das Fachgebiet

Das Fachgebiet *Neuroinformatik und Kognitive Robotik* (NIKR) der TU Ilmenau benutzt in ihren zahlreichen Projekten mobile Roboterplattformen der Firma [METRALABS], die mit verschiedensten Sensoren wie z.B. Laserscannern und *Microsoft Kinect v2*-Tiefenkameras ausgestattet sind. Das entwickelte Verfahren soll die Registrierung neu angebrachter Tiefenkameras automatisieren, sodass die Zuverlässigkeit der verwendeten Mapping-Verfahren gewährleistet wird, welche die Grundlage für weiterführende Navigations- und Detektionsalgorithmen bilden.

1.4 Kapitelübersicht

Die theoretischen Grundlagen, die zum Verständnis des in dieser Arbeit entworfenen Systems benötigt werden, werden in Kapitel 2 vorgestellt. Dabei handelt es sich vor allem um Verfahren aus der Bildverarbeitung und Optimierung.
In Kapitel 3 wird ein Überblick über den aktuellen Stand der Wissenschaft und Technik zum Thema Kameraregistrierung gegeben. Hierzu werden verschiedene Methoden vorgestellt und hinsichtlich des in dieser Arbeit vorgestellten Einsatzzwecks bewertet.
Die Vorstellung des entwickelten Systems folgt in Kapitel 4. Dabei werden seine grundlegenden Einzelkomponenten, ihre Funktionsweise und der Datenfluss zwischen ihnen erklärt.
Im Anschluss werden in Kapitel 5 die Durchführung und Ergebnisse experimenteller Untersuchungen des entworfenen Systems betrachtet. Hierbei wird seine Leistungsfähigkeit auf Basis mehrerer Evaluationsmethoden bewertet.
Abschließend wird in Kapitel 6 eine Zusammenfassung der gewonnen Erkenntnisse sowie ein Ausblick auf mögliche Verbesserungen gegeben.

Kapitel 2

Theoretische Grundlagen

2.1 SLAM

Damit sich ein Roboter orientieren kann, muss er sowohl seine Umgebung kennen als auch wissen, wo er sich in dieser befindet. Kennt der Roboter seine Position, kann er seine Karte mit den Daten seiner Sensoren erweitern. Kennt der Roboter seine Umgebung, kann er seine Position ebenfalls mithilfe seiner Sensoren bestimmen. Da jedoch anfangs weder seine Position noch seine Umgebung bekannt sind, muss beides zeitgleich geschätzt werden. Dies geschieht durch „Simultaneous Localization and Mapping (SLAM)“-Algorithmen. Für eine Übersicht verschiedener Spielarten und Implementierungen wird auf [Taketomi et al., 2017] verwiesen.
In dieser Arbeit werden SLAM-Algorithmen dazu verwendet, unabhängig von der Roboterodometrie Sensordaten zu fusionieren, um daraus dessen Trajektorie zu generieren.

2.2 ORB-Features

Um Einzelbilder zu einer größeren Informationsquelle zusammenfügen zu können, ist es notwendig, die Beziehungen zwischen den Bildern zu bestimmen. Dies geschieht zumeist über die Bildinhalte, wobei Korrespondenzen in zwei Bildern gesucht werden, um diese Bilder in räumliche Beziehung zueinander setzen zu können. Dazu müssen charakteristische Bildpunkte bestimmt werden, wozu sogenannte Features berechnet werden. Diese werden mithilfe eines Detektors gesucht und durch einen Deskriptor beschrieben. Dieser beinhaltet meist kodierte Informationen der lokalen Umgebung des gefundenen Bildmerkmals, auf deren Basis Features miteinander verglichen und die gesuchten Korrespondenzen bestimmt werden können.
Um robuste Features aus Bildern extrahieren zu können, haben [Rublee et al., 2011] ein neues Feature entwickelt, welches auf dem weit verbreiteten „Features from Accelerated Segment Test (FAST)“-Detektor und dem ebenfalls sehr häufig genutzten „Binary Robust Independent Elementary Features (BRIEF)“-Deskriptor beruht. Das sogenannte „Oriented FAST and Rotated BRIEF (ORB)“-Feature ist dementsprechend eine Kombination und vor allem eine Weiterentwicklung der beiden, dessen Hauptvorteil die Rotationsinvarianz des Deskriptors ist, sodass auch die Erkennung von übereinstimmenden Bildpunkten in Bildern von Kameras unterschiedlicher Ausrichtung kein Problem darstellt.
In dieser Arbeit werden ORB-Features vom *ORB-SLAM 2*-Algorithmus (s. Kap. 4.2.2) verwendet.

2.3 NDT-Karten

Wurde mithilfe von Sensoren die Umgebung des Roboters erfasst, müssen diese Informationen in einer Karte gespeichert werden, um später darauf wieder zugreifen und weitere Berechnungen wie z.B. Pfadplanung durchführen zu können. Hierbei gibt es verschiedene Arten von Karten (s. Abb. 2.1).

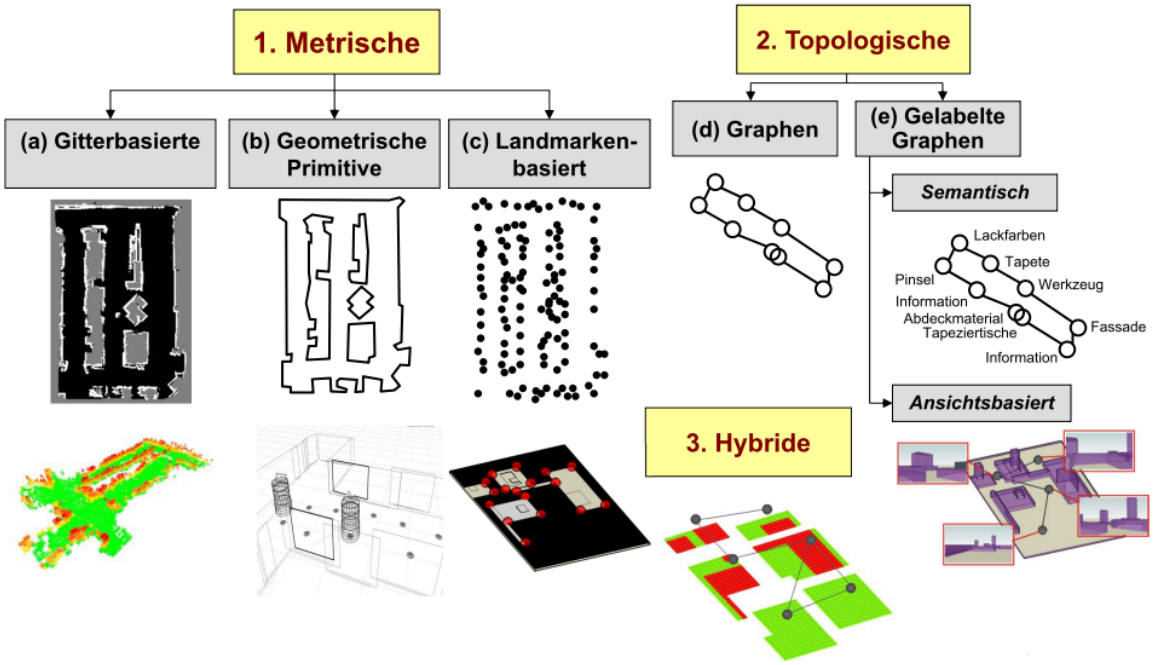


Abbildung 2.1: Übersicht über verschiedene Arten von Umgebungskarten (vgl. [Gross, 2015])

Im Fachgebiet *NIKR* werden überwiegend metrische, gitterbasierte Karten verwendet. Eine dieser Arten ist die „Normal Distribution Transform (NDT)“-Karte. Diese stellt den Raum gerastert als 3D-Verteilung, also Streuung der Punkte in den einzelnen Rasterzellen dar. Sie werden aus Punktewolken erstellt, wobei aufgrund der Raumrasterung jeder Punkt zu einer bestimmten Zelle gehört. Da in Karten von Gebäuden meist glatte Strukturen überwiegen, sehen die zugehörigen Verteilungen etwas plattgedrückt aus, sodass die ganze Karte wie eine „*Smarties*“-Karte aussieht (s. Abb. 2.2). Die Streuung der Punkte entlang der Normalenrichtung z.B. einer Wand kann dabei Aufschluss über das Rauschen des verwendeten Sensors geben. Der große Vorteil dieser Repräsentationsweise des Raumes ist der durch die Rasterung bedingte extrem niedrige Speicherplatzbedarf.

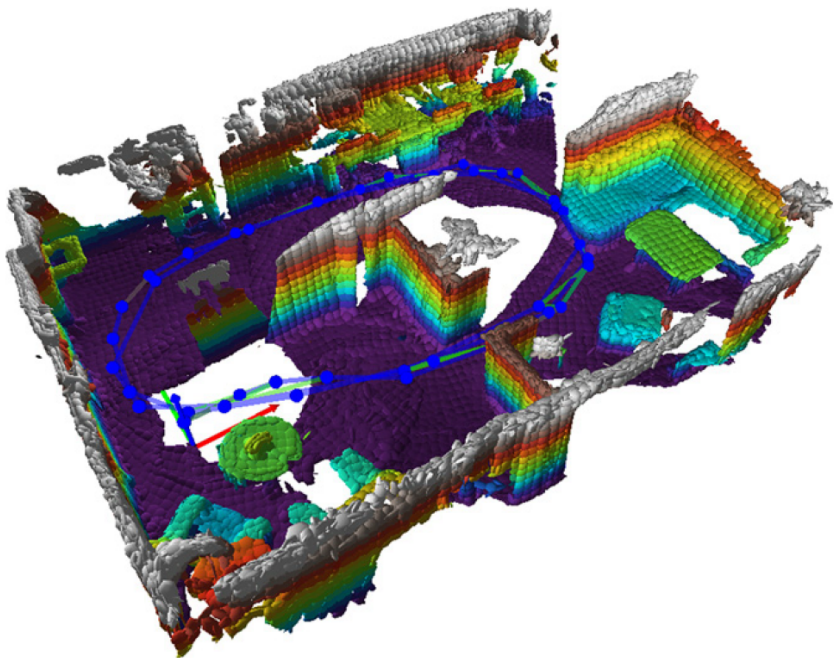


Abbildung 2.2: Beispielhafte NDT-Karte einer Wohnung. Zu sehen sind die Normalverteilungen der NDT-Zellen, welche entsprechend ihrer Höhe farblich kodiert sind. Des Weiteren sind der Koordinatenursprung und der Posengraph einer Robotertrajektorie (blau) sichtbar. (vgl. [Einhorn und Gross, 2015])

2.4 Geometrische Bedingungsgleichung

Die geometrische Bedingungsgleichung wird benötigt, um die unbekannte Transformation zwischen zwei Kameras zu bestimmen, wenn deren Trajektorien bekannt sind und sich die Transformation zwischen ihnen nicht ändert.

Dabei werden zwei verschiedene Kameras zu zwei unterschiedlichen Zeitschritten betrachtet (s. Abb. 2.3). Bekannt sind in diesem Fall die Pose der primären Kamera zum Zeitpunkt 0, P_0^0, die Pose derselben Kamera im Zeitschritt k, P_0^k und die Posen der sekundären Kamera zu beiden Zeitpunkten, P_i^0 und P_i^k. Ebenfalls bekannt, da aus ihnen berechenbar, sind die Bewegung der primären Kamera T_0^k und der sekundären Kamera T_i^k zwischen den beiden Zeitpunkten. Gesucht ist nun die fixe Transformation ΔT_i zwischen den beiden Kameras.

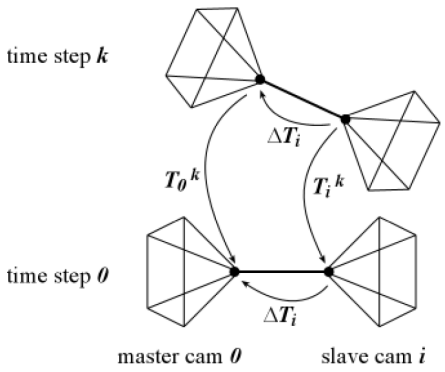


Abbildung 2.3: Verdeutlichung der geometrischen Bedingungsgleichung. Die Wege T_0^k und T_i^k der beiden Kameras sind bekannt, die Transformation ΔT_i zwischen den beiden Kameras ist gesucht, wird aber als konstant angenommen. (vgl. [Esquivel et al., 2007])

Es gibt mehrere Wege, um von der Pose der primären Kamera im Zeitschritt k zu der Pose derselben Kamera im Zeitschritt 0 zu gelangen. Zum einen ist die Transformation direkt durch T_0^k gegeben, zum anderen lässt sie sich auch über den Weg über die Posen der sekundären Kamera zu beiden Zeitschritten berechnen. Dieser kann mittels $\Delta T_i \cdot T_i^k \cdot \Delta T_i^{-1}$ beschrieben werden. Beide Wege erzeugen dabei dieselbe Transformation, sodass sie gleichgesetzt werden können:

$$T_0^k = \Delta T_i \cdot T_i^k \cdot \Delta T_i^{-1} \tag{2.1}$$

Gleichung 2.1 wird als geometrische Bedingungsgleichung bezeichnet.

2.5 Particle-Swarm-Optimization

Die von [Eberhart und Kennedy, 1995] entwickelte Particle-Swarm-Optimization (PSO) ist ein Verfahren der nicht-linearen Optimierung, welches sich am Schwarmverhalten von Tieren orientiert und auf diese Weise versucht, eine optimale Lösung für ein gegebenes Problem zu finden. Sie findet in dieser Arbeit zur Fehlerminimierung bei der Lösung der oben beschriebenen geometrischen Bedingungsgleichung Verwendung.

Ein *Schwarm* besteht dabei aus einzelnen Lösungskandidaten, genannt *Partikel*, welche sich im Lösungsraum anhand einfacher Regeln bewegen. Eine Position wird dabei mithilfe einer Kostenfunktion bewertet. Für die Bewegung eines *Partikels* spielen sowohl seine eigene bislang bestbewertete Position eine Rolle als auch die des gesamten *Schwarms*. Die einzelnen *Partikel* und damit der gesamte *Schwarm* werden dabei im Laufe der Zeit in Richtung der bestbewerteten Position gezogen, wobei sich diese im Verlauf auch durch Entdeckungen neuerer, besser bewerteter Positionen durch die *Partikel* ändern kann. Allerdings kann durch dieses Verfahren nicht gewährleistet werden, dass die tatsächliche optimale Lösung gefunden wird.

2.6 Messprinzipien der verwendeten Tiefenkameras

In dieser Arbeit wurden zwei verschiedene Tiefenkameras eingesetzt:

- *ASUS Xtion PRO LIVE*
- *Microsoft Kinect v2*

Die technischen Daten und Messprinzipien sind der Tabelle 2.1 zu entnehmen.

2.6.1 Strukturiertes Licht

Bei der *ASUS Xtion PRO LIVE* wird mit strukturiertem Licht gearbeitet. Dabei wird die Tiefe über Stereo-zu-Tiefe-Algorithmen berechnet. Es werden jedoch nicht zwei Kameras sondern ein Projektor und eine Kamera als Stereopaar verwendet, wodurch dieselben Algorithmen angewandt werden können. Der IR-Projektor des Systems sendet ein pseudozufälliges Punktmuster aus, welches von der IR-Kamera aufgefangen wird. Durch Kenntnis des Musters kann diese mithilfe der Epipolargeometrie Korrespondenzen zwischen ausgesandtem und aufgenommenem Muster detektieren und mithilfe der Disparität den Tiefenwert der Korrespondenz berechnen.

Tabelle 2.1: Technische Daten der verwendeten Tiefenkameras

Parameter	***ASUS Xtion PRO LIVE***	***Microsoft Kinect v2***
	RGB-Bild	
Auslösung	1280 x 1024	1920 x 1080
Frequenz	15 fps	30 fps
Blickwinkel (HxV)	58° x 45°	84° x 54°
	Tiefenbild	
Technik	strukturiertes Licht	Time-of-flight
Auflösung	640 x 480	512 x 424
Tiefenauflösung	11 bit	11 bit
Frequenz	30 fps	30 fps
Blickwinkel (HxV)	58° x 45°	71° x 60°
Reichweite	0.8 - 3.5 m	0.8 - 8.0 m

Da für eine Korrespondenz immer die Umgebung eines Pixels in die Berechnung mit einbezogen wird, findet eine große örtliche Tiefpassfilterung statt, wodurch viele Informationen verloren gehen. Daher spiegelt sich die Auflösung der Kamera nicht in der Qualität des Tiefenbildes wider.

2.6.2 Time-of-flight

Eine Time-of-flight-Kamera besitzt ebenfalls eine IR-Kamera, ein Projektor wird jedoch nicht benötigt. Dieser wird durch eine starke IR-Lichtquelle ersetzt, welche kurze Lichtimpulse aussendet. Die IR-Kamera misst die Laufzeit, die die IR-Lichtstrahlen für ihren Weg von der Quelle bis zum nächsten Hindernis und zurück benötigen und berechnet daraus dessen Distanz zur Kamera.
Hierbei kann für jedes Pixel unabhängig ein Tiefenwert berechnet werden, sodass die Auflösung die Qualität des Tiefenbildes maßgeblich beeinflusst. Allerdings wird von dieser Technik eine diffuse Reflektion der Hindernisse vorausgesetzt, weshalb es an z.B. Fenstern oder spiegelnden Oberflächen zu Problemen kommt. Des Weiteren beeinflusst die zeitliche Auflösung des Kamerasensors die Genauigkeit der Berechnung.

2.7 Registrierung von Farb- und Tiefenbildern

Da bei beiden verwendeten Kameratypen zwei unterschiedliche Kameras die Farb- und die Tiefendaten erfassen, müssen die Bilder zunächst zueinander ausgerichtet werden. Dabei ist zu beachten, dass die Bilder eine gewisse Strecke im Raum versetzt, durch die verschiedenen Kameras verschieden groß und durch unterschiedliche intrinsische Kameraparameter unterschiedlich verzerrt sind. Letzteres lässt sich jedoch durch Berechnung der Parameter mithilfe einer Kamerakalibrierung (s. [Zhang, 2000] und [Berger et al., 2011]) korrigieren, indem die Bilder anschließend mittels der bekannten Parameter entzerrt werden. Die Verschiebung der Kameras und damit auch der Bilder zueinander sowie die Anpassung der Größe werden durch die Treiber der Kameras über nicht näher beschriebene Algorithmen bewerkstelligt.
Im weiteren Verlauf der Arbeit wird davon ausgegangen, dass die Kameras korrekt kalibriert sind und die Entzerrung der Bilder sowie die Farb- und Tiefenbildausrichtung durch die Treibermodule in *MIRA* bereits geschehen ist.

Kapitel 3

Lösungsansätze für die Registrierung mehrerer Kameras an mobilen Plattformen

Es existieren verschiedene Methoden, die Kameraausrichtungen zueinander zu bestimmen. Diese setzen jedoch teilweise Bedingungen an die Kameraanordnung voraus. Unterschieden werden muss dabei vor allem zwischen Kameras mit überlappenden und Kameras mit nichtüberlappenden Sichtfeldern (s. Abb. 3.1).

3.1 Gemeinsames Sichtfeld

Existiert ein gemeinsames Sichtfeld der zueinander auszurichtenden Kameras, lassen sich die Transformationen direkt über Kalibrierobjekte bestimmen. [Pedersini et al., 1999] nutzten dazu eine weiße, ebene Platte, auf der sich in bekanntem Muster schwarze Punkte befinden (s. Abb. 3.2a). Dieses Kalibriermuster wurde mehrfach in verschiedenen Posen in das gemeinsame Sichtfeld eingebracht und von allen Kameras zeitsynchron aufgenommen (s. Abb. 3.2b). Anhand der bekannten Geometrie des Kalibriermusters konnte so für jede Kamera und jede Aufnahme die Transformation zwischen Kamera und Kalibriermuster bestimmt werden. Durch geometrische Beziehungen ließen sich daraus auch die Transformationen zwischen den Kameras berechnen.

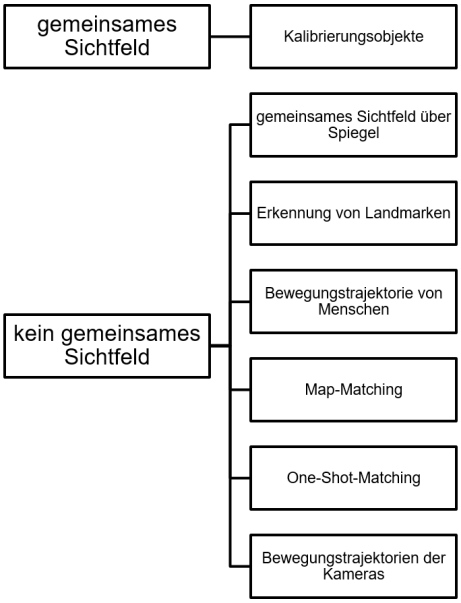
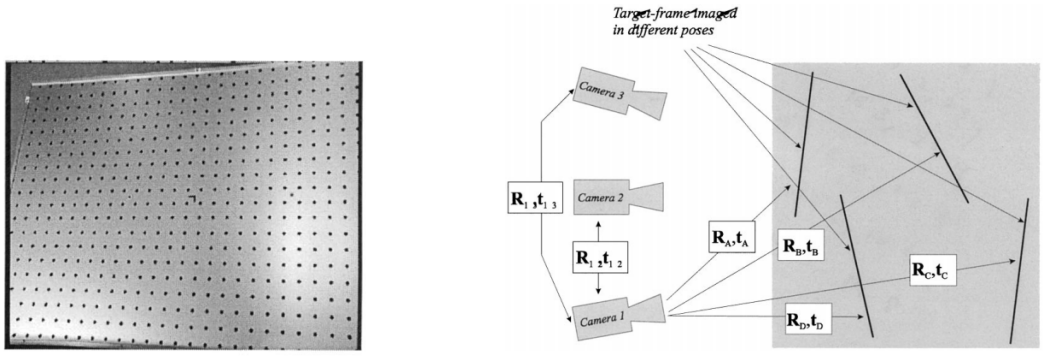


Abbildung 3.1: Übersicht über die State-of-the-Art-Methoden



(a) verwendetes Kalibriermuster

(b) Anordnung der Kameras und Kalibriermuster mit berechneten Transformationen

Abbildung 3.2: Übersicht über das Verfahren nach [Pedersini et al., 1999]

[Berger et al., 2011] erweiterten diesen Ansatz dahingehend, dass sie als Kalibriermuster ein Schachbrett verwendeten, welches anstelle der schwarzen Quadrate Alufolie enthält, sodass dort die IR-Strahlung von verwendeten Tiefenkameras nicht zu diesen reflektiert wird. Damit kann die IR-Kamera das Aluminium von den weißen, diffus reflektierenden Anteilen des Schachbretts unterscheiden (s. Abb. 3.3) und die vorgenannte Methode konnte auch bei IR- und Tiefenkameras angewendet werden.

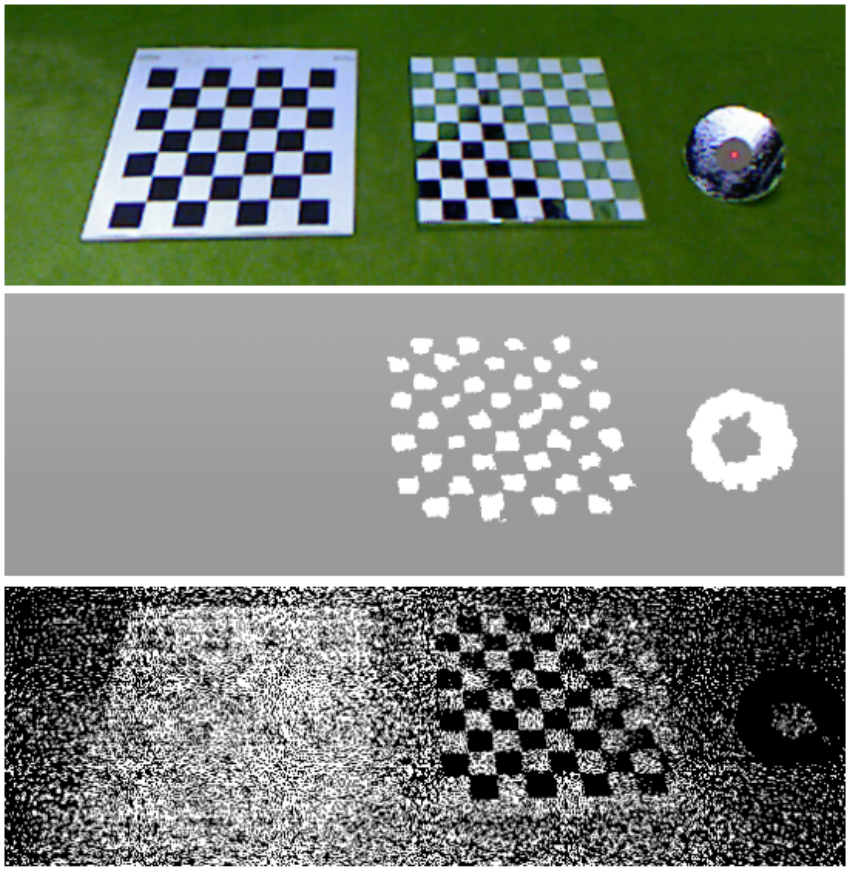


Abbildung 3.3: Verwendung von Alufolie als Hilfsmittel zur Kalibrierung von IR-Kameras. *oben*: Aufnahme mit RGB-Kamera, *mitte*: Aufnahme im Tiefenbild, *unten*: Aufnahme im IR-Bild, *links*: normales Schachbrettkalibriermuster, *mitte*: Schachbrettmuster mit Alufolie anstelle schwarzer Bereiche, *rechts*: Spiegelscheibe. (vgl. [Berger et al., 2011])

Da in vorliegendem Anwendungsfall ein gemeinsames Sichtfeld nicht vorausgesetzt werden kann, wurde dieser Ansatz nicht weiter verfolgt.

3.2 Kein gemeinsames Sichtfeld

Besteht kein gemeinsames Sichtfeld zwischen den zu kalibrierenden Kameras, gibt es u.a. nachfolgende Methoden, dieses Problem zu umgehen.

3.2.1 Nutzung von Spiegeln

[Kumar et al., 2008] setzten in ihrer Arbeit Spiegel ein, deren Posen in der Umgebung genau bekannt waren (s. Abb. 3.4). Dadurch stellten sie wieder ein gemeinsames Sichtfeld der Kameras her und konnten die Transformation zwischen ihnen mithilfe von Kalibrierobjekten bestimmen.

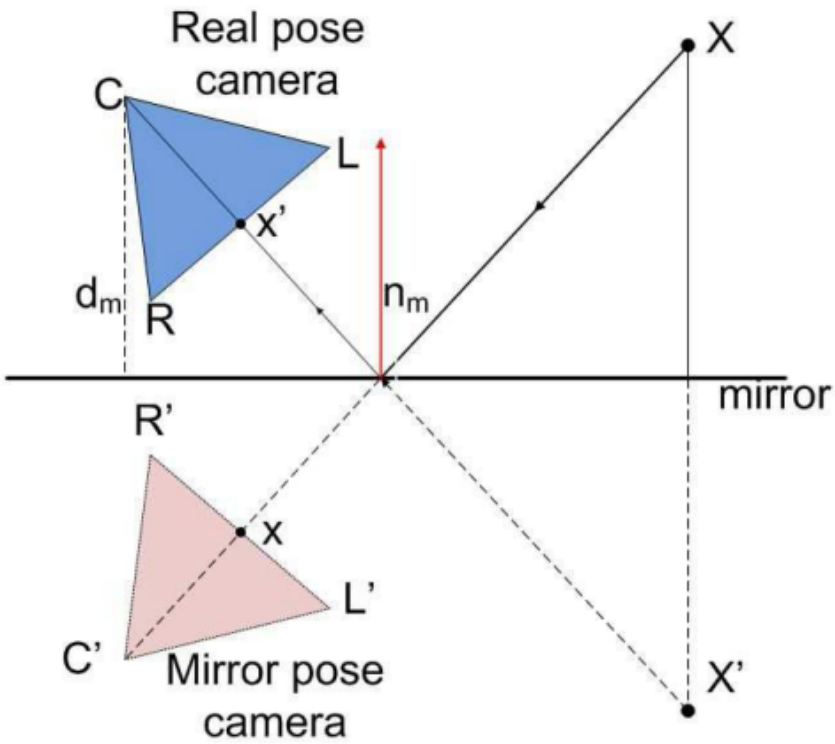
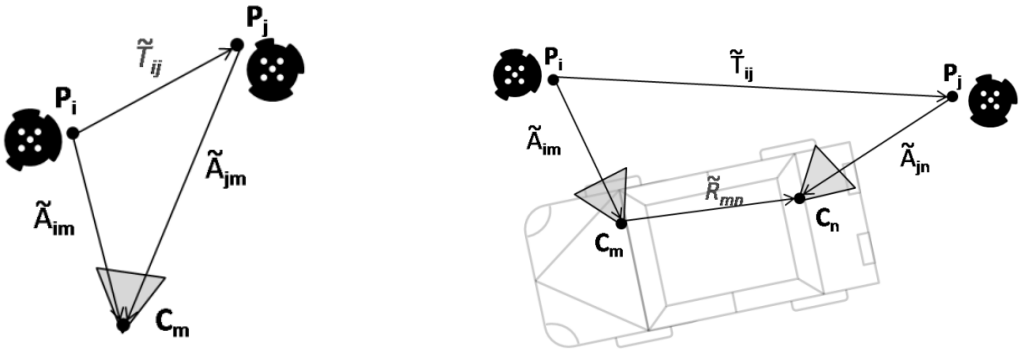


Abbildung 3.4: Darstellung der geometrischen Beziehungen zwischen Kamera und Objekt bei Nutzung eines Spiegels (vgl. [Kumar et al., 2008])

Dieses Verfahren setzt jedoch eine sehr genaue Kalibrierung der Spiegel voraus, was im Fachgebiet *NIKR* nicht gewährleistet werden kann. Somit wurde auch dieser Ansatz nicht weiter betrachtet.

3.2.2 Erkennung von Landmarken

[Pagel, 2009] und [Lamprecht et al., 2007] nutzten die Umgebung, indem sie eindeutig identifizierbare Landmarken zur Kalibrierung verwendeten. Dabei mussten die Posen dieser Landmarken bekannt sein oder zuerst durch eine einzelne Kamera bestimmt werden (s. Abb. 3.5a). Um schließlich mehrere Kameras auszurichten, wurden zeitsynchrone Aufnahmen mit allen Kameras gemacht, die darauf befindlichen Landmarken detektiert und deren Pose zur Kamera bestimmt (s. Abb. 3.5b). Mithilfe der bekannten Transformationen zwischen den Landmarken konnten so die unbekannten Transformationen der Kameras zueinander über geometrische Beziehungen berechnet werden.



(a) Bestimmung der Landmarkenpositionen mithilfe einer einzelnen Kamera

(b) Bestimmung der Kamerabeziehungen mithilfe von Landmarken

Abbildung 3.5: Übersicht des Verfahrens nach [Pagel, 2009]

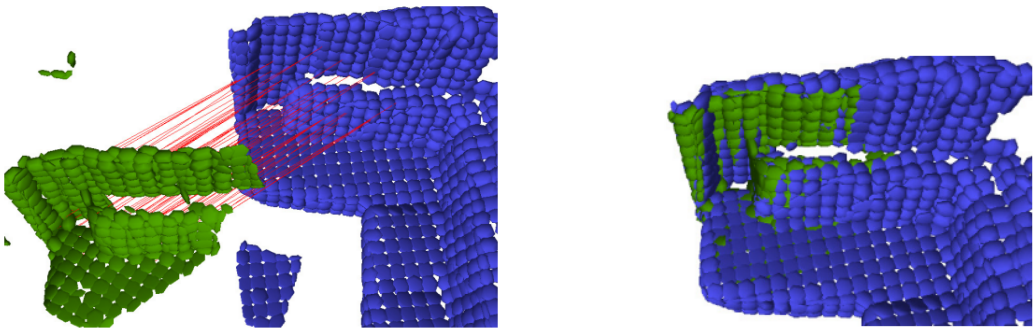
Der Vorteil dieses Verfahrens liegt darin, dass die erreichbare Genauigkeit nur von der Qualität der Detektion, demnach der Korrektheit der verwendeten Kameras abhängt. Es setzt allerdings voraus, dass die Marker manuell platziert werden. Da die zu entwickelnde Kameraregistrierung jedoch vollautomatisch und auch in unbekannter Umgebung funktionieren soll, wird dieses Verfahren nicht weiter berücksichtigt.

3.2.3 Bewegungstrajektorien von Menschen

Einen weiteren Ansatz stellten [Rahimi et al., 2004] vor. Sie nutzten die Trajektorien der in den Kameras detektierten Objekte (vorzugsweise Menschen), um Aussagen über die Posen der Kameras zu treffen. Während eine Person in einem Kamerabild zu sehen war, berechneten sie ein Bewegungsmodell, welches die Grundlage für die geschätzte Trajektorie der Person bildete, nachdem diese das Sichtfeld verlassen hatte. Durchlief die Person nun den Sichtbereich einer zweiten Kamera, konnte mithilfe der beiden Bewegungsmodelle die Transformation der Kameras zueinander bestimmt werden.
Dieses Verfahren benötigt sehr robuste Personendetektoren auf den Bildern jeder Kamera. Da dies jedoch auf den Robotern des Fachgebietes *NIKR* nicht auf allen Kameras durchgeführt werden kann und die Kameras auch teilweise so positioniert sind, dass sie ein sehr eingeschränktes Sichtfeld haben, ist dieser Ansatz nicht anwendbar. Des Weiteren ist die Annahme, dass die Person zwischen zwei Sichtfeldern ihr Bewegungsmodell nicht ändert, sehr fehleranfällig.

3.2.4 Map-Matching

Auch die aus den Sensorinformationen des Roboters erstellten Karten der Umgebung lassen sich zur Schätzung der Kameratransformationen nutzen. [Schmiedel et al., 2015] entwickelten die „Interest Point Descriptor for Robust NDT-Map Matching (IRON)“-Features, mit deren Hilfe die zuvor beschriebenen NDT-Maps zueinander ausgerichtet werden können. Dabei werden übereinstimmende Deskriptoren von Raumpunkten gesucht (s. Abb. 3.6a, *rote Linien*) und die Transformation der beiden Karten zueinander bestimmt (s. Abb. 3.6b). Diese Transformation ist gleichzeitig die relative Verschiebung der Kameras zueinander, da eine Karte ihren Ursprung in der jeweils zugehörigen Kamera hat.
Da eine Kamera sowohl eine aus einem Bild erstellte lokale oder eine aus mehreren Bildern fusionierte globale Karte nutzen kann, gibt es drei Kombinationsmöglichkeiten, die jeweils Vor- und Nachteile haben. Beim Vergleich zweier lokaler Karten müssen die Kameras ein überlappendes Blickfeld besitzen. Diese Bedingung entfällt mit der Nutzung mindestens einer globalen Karte, wobei hier die benötigt Rechenzeit steigt,



(a) Korrespondenzen (*rot*) zwischen zwei NDT-Maps

(b) Zusammengefügte NDT-Map

Abbildung 3.6: Nutzung von IRON-Features zum Matching von NDT-Maps. Die zwei verwendeten Karten sind hier farblich unterschieden worden. (vgl. [Schmiedel et al., 2015])

da die globale Karte zuvor erst erstellt werden muss. Werden zwei globale Karten genutzt, steigt die Genauigkeit des Verfahrens, da mehr Korrespondenzen zu Hilfe genommen werden können, jedoch steigt auch die benötigt Rechenzeit noch weiter. Des Weiteren muss zum Bau einer globalen Karte die Trajektorie der entsprechenden Kamera exakt bekannt sein, welche jedoch z.B. durch SLAM-Algorithmen berechnet werden kann. Dieses Verfahren wird daher als eine Lösungsmöglichkeit angesehen. Da jedoch keine funktionierende Implementierung der IRON-Features vorlag, konnte dieser Ansatz nicht weiter verfolgt werden.

3.2.5 One-Shot-Matching

Ebenfalls fertig gebaute Karten nutzten [Ataer-Cansizoglu et al., 2014]. Bei diesem Verfahren wurde mithilfe einer einzelnen Kamera und eines SLAM-Verfahrens eine globale Farbpunktwolke der Umgebung gebaut. Anschließend erfolgte eine zeitsynchrone Aufnahme aller Kameras, deren lokale Umgebungen danach in der globalen Karte gesucht und zugeordnet wurden (s. Abb. 3.7). Über die berechneten Posen der Kameras ließen sich dann auch die Kameratransformationen zueinander berechnen.

Abbildung 3.7: Übersicht des Verfahrens nach [Ataer-Cansizoglu et al., 2014]. *oben*: erstellte Karte des gesamten Raumes, *unten*: zeitsynchrone Aufnahmen der drei auszurichtenden Kameras, *mitte*: zugehörige berechnete Posen der Kameras im Raum

Dieses Verfahren ist ähnlich zu dem zuvor vorgestellten Verfahren mithilfe der IRON-Features mit dem Unterschied, dass hierbei eine Farbpunktwolke anstatt einer NDT-Map als Kartentyp verwendet wird. Da diese jedoch sehr speicherintensiv ist und die Genauigkeit der Berechnung bei Nutzung einer lokalen Umgebung stark von deren Gegebenheiten abhängt, wird auch dieses Verfahren nicht als optimal angesehen.

3.2.6 Bewegungstrajektorien der Kameras

Ein weiteres recherchiertes Verfahren wurde von [Esquivel et al., 2007] und [Pagel, 2009] entwickelt. Hierbei kommt die in Kap. 2.4 beschriebene geometrische Bedingungsgleichung (s. Formel 2.1) zum Einsatz. Wurden über z.B. SLAM-Verfahren die Trajektorien der Kameras über einen längeren Zeitraum bestimmt, kann mithilfe der geometrischen Bedingungsgleichung die Transformation der Kameras zueinander berechnet werden.

Dazu wird die Bedingungsgleichung mehrfach zwischen verschiedenen Zeitpunkten der Aufnahme aufgestellt und die freien Parameter durch Verfahren der nicht-linearen Optimierung gelöst. Um dabei alle sechs Freiheitsgrade der Transformation im 3D-Raum bestimmen zu können, muss das Multikamerasystem sowohl in jeder Dimension mindestens eine bestimmte Bewegung vollziehen als auch sich um jede Raumachse um mindestens einen gewissen Winkel drehen.

Eine Erweiterung für mobile Systeme, die nur innerhalb einer Ebene agieren und daher nicht alle geforderten Bedingungen des vorgenannten Verfahrens erfüllen können, stellten [Pagel und Willersinn, 2011] vor. Da sich z.B. ein Auto oder eine mobile Roboterplattform, wie jene im Fachgebiet *NIKR*, nur auf der Bodenebene bewegen kann, ist sowohl eine Änderung der Höhe der Kamera als auch die Rotation um zwei der drei Raumachsen nicht möglich, wodurch nicht alle Parameter der Kameratransformation berechenbar sind. Abhilfe wird in diesem Verfahren dadurch geschaffen, dass die beiden fehlenden Winkel und die Höhe der Kamera über eine Bodenplattenschätzung bestimmt werden. Sind diese drei Parameter für jede Kamera bekannt, können die Trajektorien auf die Bodenebene projiziert werden, wodurch die anschließende Berechnung in 2D erfolgen kann (s. Abb. 3.8). Hierbei können über die geometrische Bedingungsgleichung die übrigen drei Transformationsparameter bestimmt werden, wobei durch die Verlagerung in den 2D-Raum alle Bewegungsbedingungen für das grundlegende Verfahren auch durch ebenengebundene mobile Plattformen erfüllt werden können.

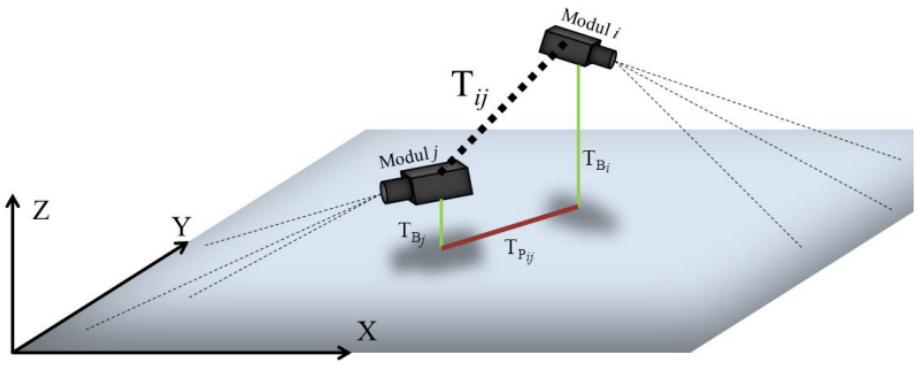


Abbildung 3.8: Erweiterung des Verfahrens nach [Pagel, 2009] durch [Pagel und Willersinn, 2011]. T_{B_i} und T_{B_j} werden durch die Bodenplattenschätzung berechnet, $T_{P_{ij}}$ ist weiterhin gesucht, T_{ij} kann daraus berechnet werden (vgl. [Pagel und Willersinn, 2011])

3.3 Eigener Ansatz

Da die im Fachgebiet *NIKR* verwendeten Roboter zu den ebenengebundenen mobilen Plattformen zählen, bildet das zuvor vorgestellte Verfahren nach [Pagel und Willersinn, 2011] die Grundlage für den in dieser Arbeit entwickelten Ansatz. Da jedoch davon ausgegangen werden muss, dass nicht jede Kamera den Boden sieht und damit eine Grundplattenschätzung ausführen kann, wird für die Berechnung der Höhe und zweier Raumwinkel eine Alternative benötigt.

Hier können die Trajektorien der Kameras verwendet werden, da diese sich nur in einer Ebene bewegt haben. Aufgrund der Tatsache, dass eine Trajektorie ihren Ursprung in ihrer ersten Pose hat, spiegelt die Lage dieser Ebene in den Kamerakoordinaten die Ausrichtung der zugehörigen Kamera wider. Sie kann berechnet werden durch Anwendung eines „Random Sample Consensus (RANSAC)“-Algorithmus auf den Positionen innerhalb der Trajektorie. Nachdem die Normale dieser Ebene bestimmt wurde, können die beiden gewünschten Raumwinkel der Kamera daraus berechnet werden.

Der Nachteil dieses Verfahrens ist, dass die Höhe der Kamera nicht bestimmt werden kann. Da dieser Wert jedoch relativ einfach manuell mithilfe eines Zollstocks gemessen werden kann, kann dies toleriert werden.

Kapitel 4

Umsetzung der Parameterbestimmung

Die Implementierung des entwickelten Systems erfolgte in *C++* als Module der im Fachgebiet *NIKR* genutzten Roboter-Middleware *MIRA*. Dazu wurde ein schon existierendes Mapping-Tool um neue Funktionalitäten und Plugin-Module erweitert. Dieses Tool basiert darauf, während einer Messfahrt Daten aufzunehmen und diese anschließend sequentiell zu verarbeiten. Da während des Ablaufs des entwickelten Verfahrens zum Teil schon Karten der Umgebung gebaut werden, können diese als Nebenprodukt mit ausgegeben werden.
MIRA beruht auf dem Konzept von Datenkanälen, sogenannten *Channels*, welche für den Austausch von Informationen zwischen den einzelnen Softwaremodulen zuständig sind. Dabei können die Komponenten Daten in *Slots* auf einen *Channel* schreiben, wodurch alle Module, welche diesen *Channel* beobachten, benachrichtigt werden und die Informationen lesen und verarbeiten können. Des Weiteren besteht die Möglichkeit, Daten von *Channels* aufzunehmen und in sogenannten *Tapes* abzuspeichern sowie sie später daraus wieder abzuspielen.
Darüber hinaus existiert in *MIRA* der sogenannte *TransformationTree*. Dieser enthält in einer Baumstruktur verschiedene Posen, hier genannt *Frames*. So können Transformationen zwischen verschiedenen Teilen des Roboters hierarchisch angeordnet werden, sodass von einem Bauteil nur die relative Position zu seinem Eltern-*Frame* bekannt sein muss. Soll nun die Transformation zu einem anderen Bauteil berechnet werden, kann entlang des Pfades im Baum zwischen den beiden Bauteilen die resultierende

Transformation aus den einzelnen auf dem Weg liegenden Transformationen zusammengerechnet werden. Jeder *Slot* auf einem *Channel* enthält neben seinen Nutzdaten und dem Zeitstempel auch die ID eines *Frames*, dem er zugeordnet sein soll. So können beispielsweise die Punkte einer mit einer Tiefenkamera aufgenommenen Punktewolke, deren Ursprung in der Kamera liegt, leicht in globale Koordinaten umgerechnet werden.
Ein weiteres Konzept von *MIRA* ist, dass alle Module in separaten Threads laufen, um parallel arbeiten zu können.
Für weitere Informationen zur Funktionsweise von *MIRA* wird auf [Einhorn et al., 2012] verwiesen.

4.1 Funktionsweise des Mapping-Tools

Das Mapping-Tool besteht aus mehreren Komponenten, wobei nachfolgend besonders auf den *SimpleMapper*, den *MappingProcess* und den *MappingPlayer* eingegangen wird. Darüberhinaus beinhaltet seine Implementierung die Klasse *MappingModule*, die als Basisklasse für alle hier vorgestellten Plugin-Module dient.
Der *SimpleMapper* ist die grafische Benutzeroberfläche des Mapping-Tools (s. Abb. 4.1). Über diese werden die Einstellungen wie Referenzpositionen, die Konfiguration des Mappings, die zu verwendenden *Channels* sowie die Genauigkeit der zu erstellenden Karten getätigt. Außerdem werden über die Oberfläche die Aufnahme der Daten gestartet und beendet, während der Messfahrt die Referenzpositionen aktiviert sowie anschließend der Fortschritt der Berechnung angezeigt und die Speicherung der fertigen Karten vollzogen.
Nachdem die Messfahrt durchgeführt und die Daten gespeichert worden sind, startet der *SimpleMapper* den *MappingProcess* in einem neuen Prozess. Dies dient dazu, die Berechnung der Karten mithilfe der aufgenommenen Daten vom noch laufenden Roboterframework zu separieren, um eine Datenkapselung gewährleisten zu können und zu verhindern, dass neue Sensordaten den Mapping-Vorgang stören. Der *MappingProcess* lädt und konfiguriert alle Mapping-Module entsprechend der zuvor im *SimpleMapper* eingestellten Konfiguration und bindet sie in sein Framework ein. Danach beginnt der

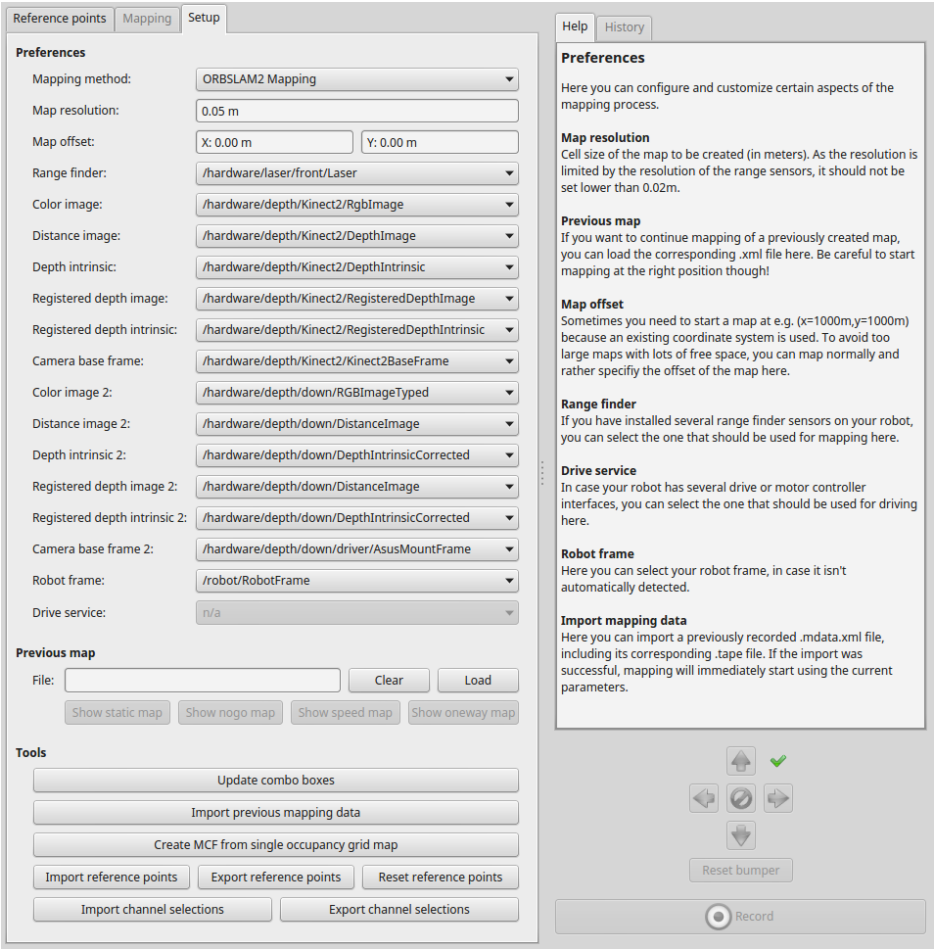


Abbildung 4.1: Einstellungsfenster des *SimpleMappers*. *links, von oben nach unten*: Auswahlbox, um den gewünschten Mapping-Prozess auszuwählen, Texteingabefelder, um die spatiale Auflösung der zu bauenden Karte einzustellen, Auswahlboxen, um die benötigten *Channels* auszuwählen, *links unten*: Schaltflächen, um die Inhalte der Auswahlboxen zu aktualisieren, zuvor gespeicherte Auswahlen oder Referenzpunkte zu laden oder die aktuellen zu speichern sowie zuvor aufgenommene *Tapes* zu öffnen und den Mapping-Prozess auf diesen Daten laufen zu lassen.

MappingPlayer, das während der Messfahrt aufgenommene *Tape* abzuspielen. Dabei stellen der *MappingProcess* und das *MappingModule* über Variablen sicher, dass der *MappingPlayer* solange mit dem Abspielen neuer Informationen wartet, bis der aktuelle Daten-*Slot* vollständig von den laufenden Modulen verarbeitet wurde und diese die Bereitschaft zum Empfangen neuer Daten wiederhergestellt haben. Dabei werden *Slots* mit selbem Zeitstempel als zusammengehörige Daten betrachtet und vom *MappingPlayer* zeitgleich an die Module weitergegeben.
Die Mapping-Konfiguration ermöglicht es, dass das aufgenommene *Tape* in mehreren Durchläufen, genannt *Passes*, abgespielt wird und die geladenen Module dabei in verschiedenen Modi laufen. Somit kann z.B. ein SLAM-Modul während des ersten *Passes* seine interne Datenbank füllen und die sich daraus ergebenen Kameraposen speichern. Im zweiten *Pass* müssen nun die zuvor gespeicherten Daten nur noch ausgegeben und nicht neu berechnet werden. Dies spart Rechenzeit und senkt die Datenmenge, die verarbeitet werden muss. Durch diese Multi-*Pass*-Architektur der Mapping-Konfiguration können Informationen ähnlich wie beim Multi-*Pass*-Encoding von Videodaten effizienter verarbeitet werden, da kausale Abhängigkeiten von zukünftigen Daten mitberücksichtigt werden können, die sonst aufgrund des sequentiellen Abspielens der *Tapes* nicht bekannt wären. Des Weiteren sind komplexere Abläufe des Mapping-Vorgangs umsetzbar (s. z.B. Abb. 4.4).

4.2 Verwendete Fremdbibliotheken

Um bereits etablierte Verfahren nutzen zu können, wurden zwei Fremdbibliotheken in *MIRA* integriert, die nachfolgend vorgestellt werden.

4.2.1 GMapping

[Grisetti et al., 2005] entwickelten ein SLAM-Verfahren, welches die Daten der Odometrie und eines 2D-Laserscanners verarbeitet und daraus mithilfe einer speziellen Variante eines *Partikel*-Filters eine 2D-Belegtheits-Karte des Raumes erstellt. Dieses Verfahren nannten sie *GMapping* und stellten den Quellcode frei zur Verfügung.
In dieser Arbeit wird *GMapping* dazu verwendet, die Odometrie des Roboters zu korrigieren, da diese als Referenztrajektorie genutzt wird und somit hinreichend genau sein sollte. Sie ergibt sich nach Verarbeitung aller Daten aus der Trajektorie des besten *Partikels*.
Ein Modul für die Nutzung von *GMapping* in *MIRA*, genannt *GMappingModule*, existierte bereits am Fachgebiet *NIKR*, sodass es ohne größeren Aufwand genutzt werden konnte. Lediglich die Funktionalität zur Ausgabe der korrigierten Odometrie im zweiten *Pass* musste nachgerüstet werden.

4.2.2 ORB-SLAM 2

Wie der Name schon sagt, ist das von [Mur-Artal und Tardos, 2017] entwickelte *ORB-SLAM 2* ein visuelles SLAM-Verfahren, welches ORB-Features verwendet und mittlerweile in Version 2 existiert.
Es detektiert dabei die ORB-Features im monokularen Farbbild der verwendeten Kamera und verwendet als Distanz zwischen Feature und Kamera den zugehörigen Wert aus dem Tiefenbild. So kann die räumliche Position des Features relativ zur Kamera durch ein einzelnes Bild bestimmt werden, wodurch eine lokale Szene unabhängig von Vorwissen vollständig erstellt werden kann. Diese wird anschließend mit der internen Datenbank verglichen. Werden Übereinstimmungen gefunden, wird die neue Szene entsprechend der schon vorhandenen Daten ausgerichtet und zu diesen hinzugefügt.

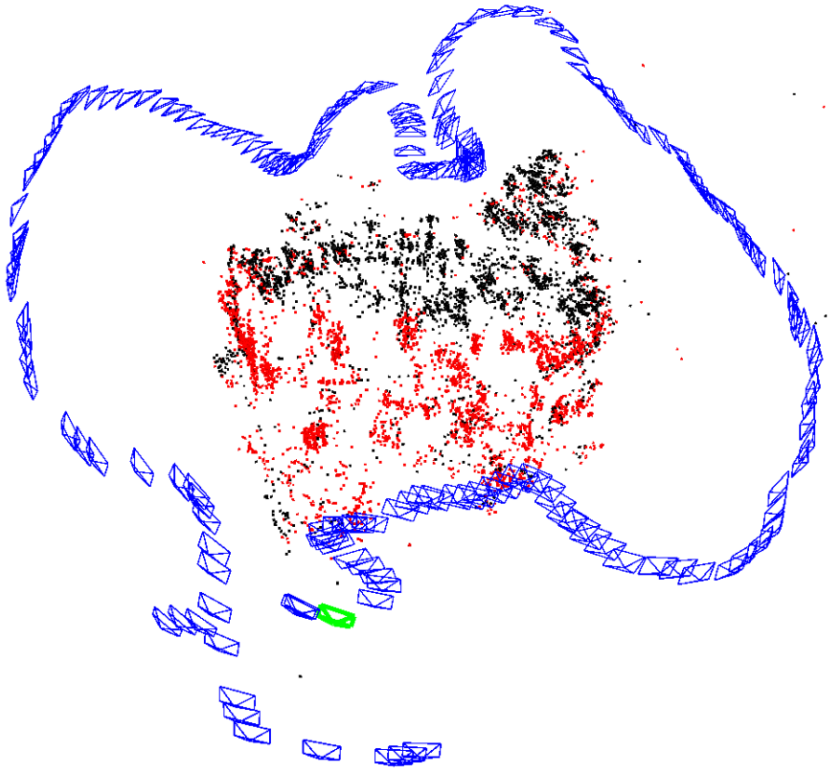


Abbildung 4.2: Visualisierung der Informationen aus der *ORB-SLAM 2*-Datenbank. *blau*: *Keyframes*, *grün*: aktueller *Frame*, *rot*: Raumpunkte der Datenbank, die im aktuellen *Frame* zu sehen sind, *schwarz*: Raumpunkte der Datenbank, die im aktuellen *Frame* nicht zu sehen sind. (vgl. [Mur-Artal et al., 2015])

So entsteht durch das Einfügen aller Informationen eine globale Karte (s. Abb. 4.2). Die dabei berechnete Kamerapose, der sogenannte *Frame*, wird ebenfalls gespeichert. Redundanzen innerhalb der Datenbank werden dabei soweit eliminiert, dass am Schluss von allen *Frames* nur noch die sogenannten *Keyframes* vorhanden sind, welche den Großteil der Informationen beinhalten.

Ein weiterer Vorteil dieses Verfahrens gegenüber anderen SLAM-Algorithmen ist, dass *ORB-SLAM 2* eine Kombination aus lokalem und globalem Bündelausgleich nutzt, um die erstellten Karten zu optimieren, wodurch das Verfahren sehr speicher- und recheneffizient ist und trotzdem eine hohe Genauigkeit liefert. Des Weiteren teilt es das Tracking, das lokale Mapping und den Schleifenschluss in drei einzelne Threads

auf, wodurch die Implementierung mehrere Rechenkerne benutzen kann und somit die drei Arbeitsschritte parallel laufen können, was die Verarbeitungsgeschwindigkeit verbessert und das Verfahren unter bestimmten Umständen echtzeitfähig macht.
In dieser Arbeit wird *ORB-SLAM 2* zur Bestimmung der Bewegungstrajektorien der am Roboter befestigten Tiefenkameras verwendet.
Zu diesem Zweck wurde ein Wrapper implementiert, der die Funktionalität der *ORB-SLAM 2*-Bibliothek als *MIRA*-Modul bereitstellt.

4.3 Ablauf des Algorithmus zur extrinsischen Kamerakalibrierung

In den Abb. 4.3 und 4.4 sind der Ablauf sowie der Datenfluss des entwickelten Systems zu sehen. Mit den auf dem mobilen Roboter angebrachten Sensoren werden während einer Messfahrt Aufnahmen gemacht sowie Referenzpositionen aktiviert (s. Abb. 4.5) und die Daten in einem *Tape* gespeichert. Anschließend können im ersten *Pass* parallel mithilfe der *GMapping*- und *ORB-SLAM 2*-Algorithmen die Roboter- und Kameratrajektorien berechnet werden, welche danach im zweiten *Pass* bei der Berechnung der Kamerapose Verwendung finden. Mit deren Hilfe, der Robotertrajektorie und den Tiefenbildern kann zuletzt im dritten *Pass* eine NDT-Karte der Umgebung erstellt werden.

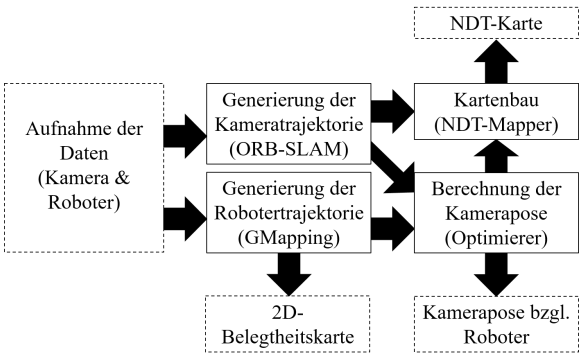


Abbildung 4.3: Ablaufplan des Gesamtsystems. *in Klammern*: Modul, in welchem die Berechnung stattfindet.

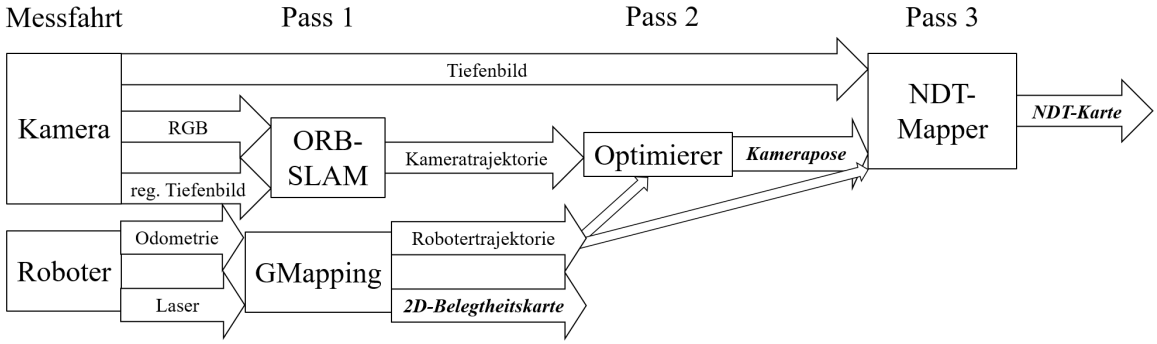


Abbildung 4.4: Flussdiagramm der Implementierung. ***kursiv***: Ausgabedaten, die weitere Verwendung beim Betrieb der Roboter finden.

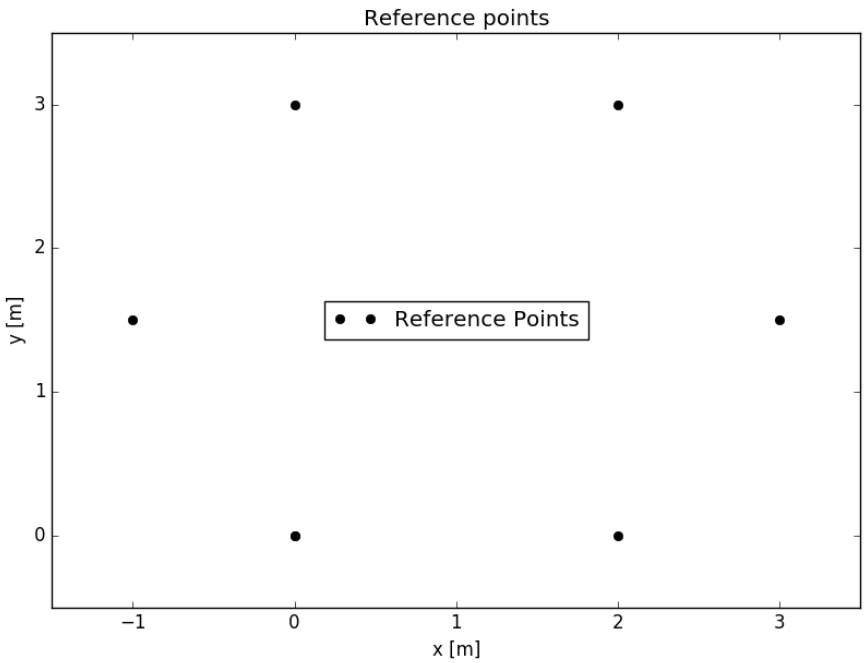


Abbildung 4.5: Anordnung der Referenzpositionen im Raum.

4.4 Funktionsweise der Implementierung

Im folgenden Abschnitt wird die Funktionsweise der einzelnen Komponenten nochmals genauer vorgestellt. Dabei wird auf die genutzten Eingangsdaten, die implementierten Algorithmen sowie die resultierenden Ausgabewerte eingegangen.

Tabelle 4.1: Anforderungen an die Messfahrt

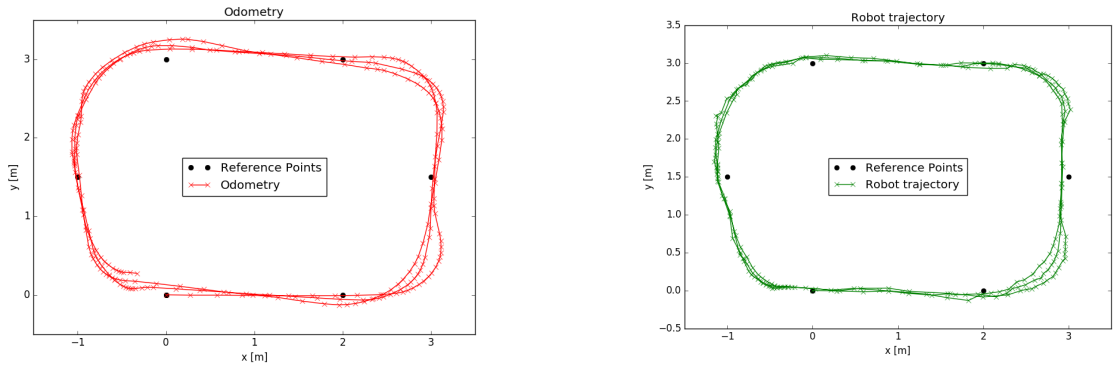
strukturierte Umgebung	In einer unstrukturierten Umgebung kann der *ORB-SLAM 2*-Algorithmus keine Features im Kamerabild detektieren und somit keine Trajektorie berechnen.
Kreisschlüsse	Nach Möglichkeit sollte die gesamte Strecke mehrmals abgefahren werden, um über Kreisschlüsse den Drift des *ORB-SLAM 2*-Algorithmus zu erkennen und seine Genauigkeit erhöhen zu können.
langsame Kurvenfahrten	Bei schnellen Drehungen des Roboters liefern die Kameras u.U. durch die Bewegungsunschärfe Bilder, auf denen weniger oder sogar gar keine Features detektiert werden können.
großer Bewegungsradius	Wie bereits in Kap. 3.2.6 erwähnt, muss sich der Roboter in jeder Dimension der Ebene ausreichend bewegt haben und darüber hinaus genügend rotiert sein, um eine sichere Bestimmung der freien Parameter der Transformation gewährleisten zu können.
Blendungsfreiheit der Sensoren	Da die Tiefenkameras mit IR-Licht arbeiten, sollte direkte Sonneneinstrahlung vermieden werden, da diese die von den Kameras ausgesandte IR-Strahlung überdecken und undetektierbar machen könnte.

4.4.1 Aufnahme der Daten

Vor der Messfahrt müssen in der grafischen Oberfläche des *SimpleMappers* alle notwendigen Einstellungen getroffen werden. Hierbei sind besonders die aufzunehmenden und später zu verarbeitenden *Channels* (s. Abb. 4.1) sowie die gewünschte spatiale Auflösung der zu erstellenden Karte wichtig.
An die Messfahrt werden mehrere Anforderungen gestellt, um eine zuverlässige Verarbeitung der Daten gewährleisten zu können. Diese sind in Tabelle 4.1 vermerkt.
Nachdem der Raum vollständig befahren wurde, kann die Aufnahme über die entsprechende Schaltfläche des *SimpleMappers* wieder beendet werden. Daraufhin werden die Daten in einem *Tape* gespeichert und der sich anschließende Mapping-Vorgang wird ausgelöst.

4.4.2 Generierung der Robotertrajektorie

Die Generierung der Robotertrajektorie geschieht durch das oben beschriebene *GMappingModule*. Dieses liest die Roboterodometrie (s. Abb4.6a) und die Informationen des auf dem Roboter befestigten 2D-Laserscanners und speist sie in den *GMapping*-Algorithmus ein. Nachdem alle Daten verarbeitet worden sind, können die erstellte Karte und die Trajektorie des besten *Partikels* als korrigierte Odometrie und damit Robotertrajektorie ausgegeben werden (s. Abb. 4.6b).



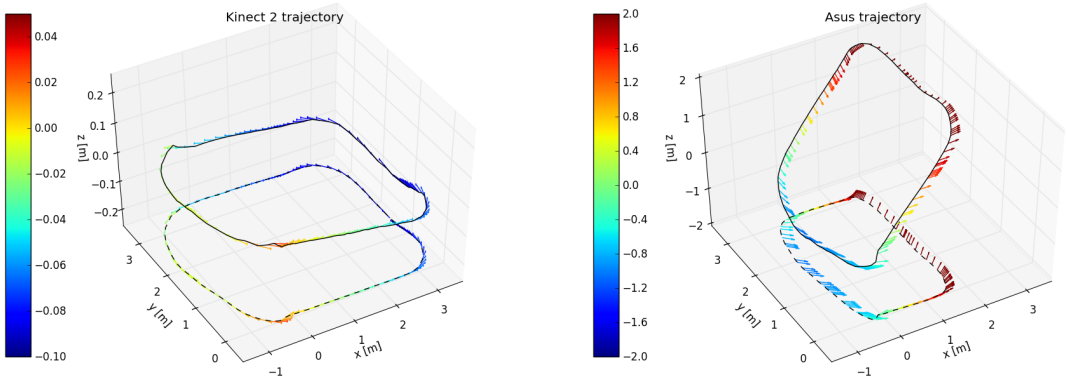
(a) Aufgenommene Odometrie des Roboters (unkorrigiert)

(b) Durch den *GMapping*-Algorithmus korrigierte Robotertrajektorie

Abbildung 4.6: Aufgenommene Odometrie und korrigierte Trajektorie des Roboters

4.4.3 Generierung der Kameratrajektorie

Parallel zur Berechnung der Referenztrajektorie kann mithilfe des oben beschriebenen *ORBSLAM2Modules* die Trajektorie der verwendeten Tiefenkamera bestimmt werden. Es empfängt die Daten des RGB- und Tiefensensors sowie deren intrinsische Parameter und gibt sie an den *ORB-SLAM 2*-Algorithmus weiter. Nachdem dieser alle Informationen verarbeitet hat, können die gespeicherten *Keyframes* als Kameratrajektorie angesehen werden, da sie mithilfe der Kreisschlüsse korrigiert wurden. Die *Keyframes* werden gespeichert und im zweiten *Pass* ausgegeben (s. Abb. 4.7).



(a) Durch den *ORB-SLAM 2*-Algorithmus berechnete Trajektorie der *Microsoft Kinect v2*.

(b) Durch den *ORB-SLAM 2*-Algorithmus berechnete Trajektorie der *ASUS Xtion PRO LIVE*.

Abbildung 4.7: Berechnete Kameratrajektorien. Gezeigt werden die relativen Posen, d.h. Ursprung der Koordinatensysteme sind die jeweils ersten Posen der Trajektorie, die Farbe codiert dabei die z-Koordinate. *gestrichelt*: die zur besseren Visualisierung auf die Grundfläche projizierte Trajektorie.

4.4.4 Berechnung der Kamerapose

Die Berechnung der Kamerapose geschieht durch das *TrajectoryAlignerModule*, welches den Hauptbestandteil des entwickelten Verfahrens darstellt. Es empfängt die korrigierte Roboterodometrie und die Trajektorie einer Kamera.
Nachdem beide Trajektorien vollständig einglesen wurden, beginnt die Initialisierung der Berechnung. Dazu wird zunächst der existierende *TransformationTree* um einen neuen Zweig erweitert. Dieser beginnt beim Roboter-*Frame* und beinhaltet drei aneinanderhängende *Frames*. Als erstes ist dies der *ZFrame*, welcher die vorgegebene Höhe der Kamera enthält. An ihn wird ein weiterer *Frame*, der *MotionFrame*, gehängt. Dieser enthält die 2D-Korrektur, die mithilfe der geometrischen Bedingungsgleichung (s. Formel 2.1) aus den Trajektorien bestimmt wird. Der letzte *Frame* ist der *RotationFrame*, der die beiden verbleibenden Raumwinkel, den Roll- und den Nickwinkel, enthält, die über die Ebenenschätzung der Trajektorie bestimmt werden (s. Abb. 4.8). Dieser ist nach abgeschlossener Berechnung als neuer Befestigungs-*Frame* der Kamera anzusehen. Aus diesem Grund werden bei der Initialisierung alle Kind-*Frames* des alten Befestigungs-*Frames* von diesem gelöst und an den neuen angehängt. Der *MotionFrame* und der *RotationFrame* werden mit Einheitstransformationen initialisiert.
Um zeitsynchrone Daten zu erhalten, wird die Roboterodometrie, die mit höherer Frequenz als die Kameratrajektorie aufgenommen wurde, linear interpoliert und es werden daraus Datenpunkte mit den Zeitstempeln der Kameraposen bestimmt und gespeichert. Anschließend werden die Kameraposen, die noch in Tiefenbildkoordinaten vorliegen, auf Weltkoordinaten umgerechnet.

Schätzung der Grundebene anhand der Trajektorie

Als nächstes folgt eine Schätzung der Trajektorienebene. Dazu werden die Positionen der Kamera entlang dieser Trajektorie als Punktwolke betrachtet und auf dieser ein RANSAC-Algorithmus durchgeführt. Er berechnet die Normale der Ebene, aus der dann sowohl der Roll- als auch der Nickwinkel der Kamera bestimmt werden können (s. Abb. 4.8).

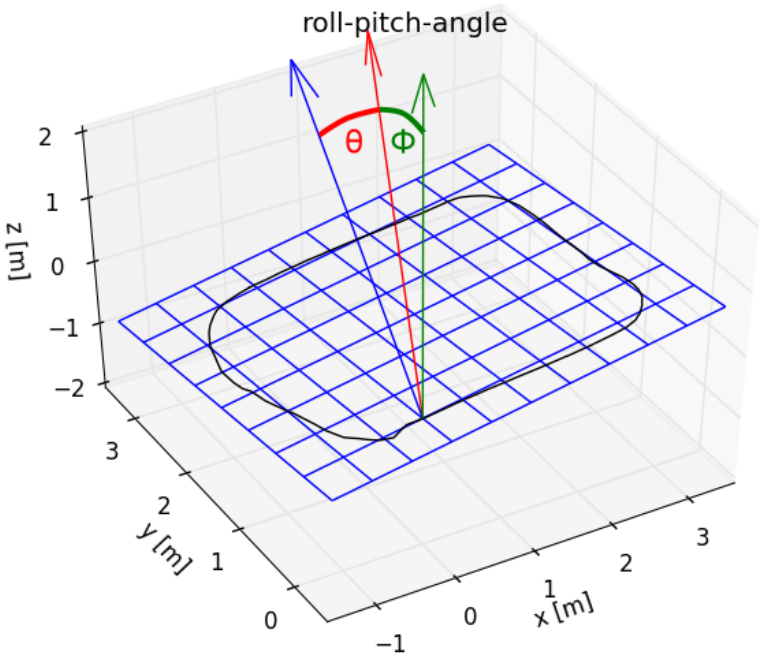
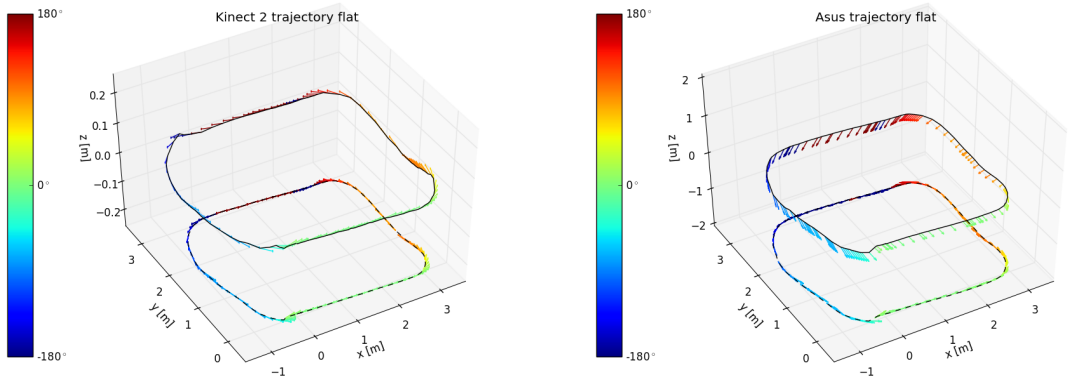


Abbildung 4.8: Berechnung des Roll- und Nickwinkels aus der Kameratrajektorie. *schwarz*: Kameratrajektorie, *blau*: aus der Trajektorie berechnete Ebene und auf ihr stehende Normale, *rot*: aus der Normale berechneter Nickwinkel und um diesen Winkel gedrehter Vektor, dieser besitzt eine x-Komponente von 0, *grün*: aus diesem Vektor berechneter Rollwinkel und um ihn gedrehter Vektor, dieser ist die Normale der Bodenplatte.

Diese werden als zweiter Teil der Transformation als *RotationFrame* ausgegeben. Des Weiteren werden die Posen der Trajektorie mithilfe der Transformation ausgerichtet, sodass die vorher berechnete Ebene nun parallel zur tatsächlichen Bodenebene liegt (s. Abb. 4.9).

Projektion der Trajektorie auf die Grundebene

Anschließend können die Posen einzeln auf die Grundebene projiziert werden (s. Abb. 4.9). Dadurch befinden sie sich wie die korrigierte Roboterodometrie im 2D-Raum der Bodenebene, wodurch die Ausrichtung zueinander einfacher zu berechnen ist.



(a) Zur Grundfläche ausgerichtete Trajektorie der *Microsoft Kinect v2* und ihre Projektion auf die Grundebene

(b) Zur Grundfläche ausgerichtete Trajektorie der *ASUS Xtion PRO LIVE* und ihre Projektion auf die Grundebene

Abbildung 4.9: Ausgerichtete und projizierte Kameratrajektorien. Gezeigt werden die relativen Posen, d.h. Ursprung der Koordinatensysteme sind die jeweils ersten Posen der Trajektorie, die Farbe codiert dabei den Gierwinkel.

Ausrichtung der Kamera zum Roboter

Um die letzten drei freien Parameter der Kamerapose zu bestimmen, wird versucht, die geometrische Bedingungsgleichung (s. Formel 2.1) durch Zuhilfenahme einer PSO zu lösen. Der Suchraum ist dabei dreidimensional, wobei die erste Dimension die x-Verschiebung widerspiegelt, die zweite Dimension die y-Verschiebung und die dritte Dimension den Gierwinkel der Kamera.

Die Kostenfunktion der PSO ist als Summe der euklidischen Distanzen zwischen den Transformationen der beiden Seiten der geometrischen Bedingungsgleichung (s. Formel 2.1) über alle in den Trajektorien enthaltenen Zeitschritten definiert, wobei die aktuelle Position des *Partikels* im Suchraum als ΔT_i interpretiert wird. Weiterhin wird die korrigierte Robotertrajektorie als Referenzkamera mit den k Transformationen T_0^k und die Kameratrajektorie als sekundäre Kamera mit den k Transformationen T_i^k verwendet. Der resultierende Unterschied des Drehwinkels findet bei der Bewertung keine Beachtung.

Die x- und y-Verschiebungen werden um 0 herum initialisiert, da nicht bekannt ist, wo sich die Kamera am Roboter befindet. Um den Drehwinkel initialisieren zu können, wird dieser zu Beginn aus den vorliegenden Daten geschätzt. Dies geschieht, indem die größte Ausdehnung der Kameratrajektorie durch Berechnung der Abstände aller Positionen zueinander bestimmt und die beiden mit der größten Distanz ausgewählt werden. Durch den Winkel zwischen deren Strecke und der der korrespondierenden Positionen der Robotertrajektorie kann der initiale Drehwinkel α für die PSO berechnet werden (s. Abb. 4.10, *blau*).

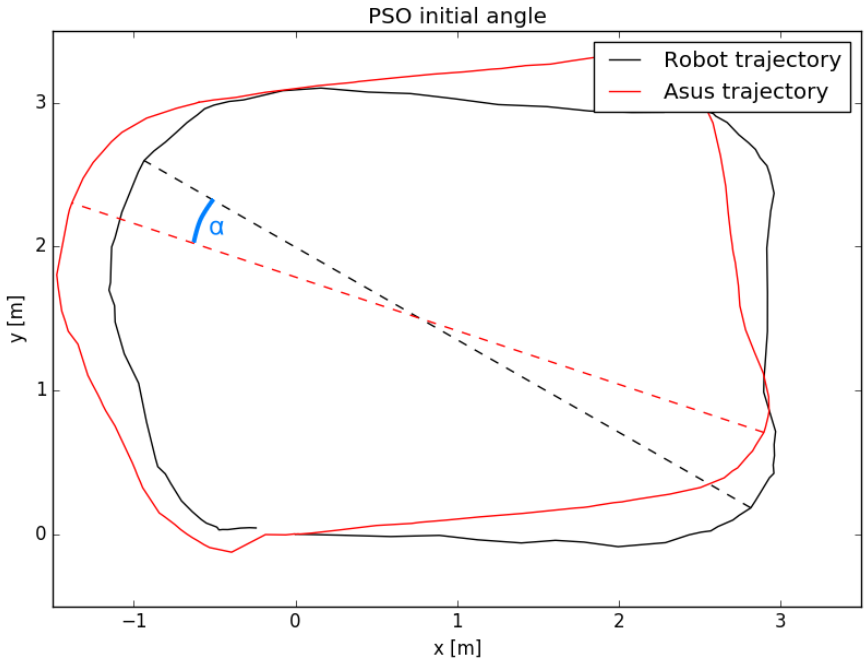
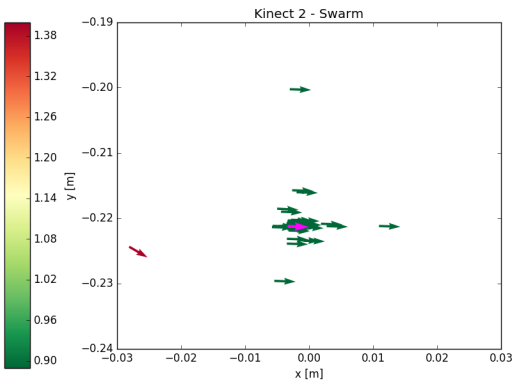
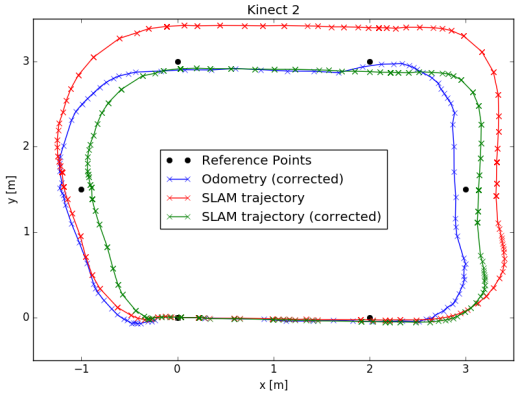


Abbildung 4.10: Berechnung des initialen Drehwinkels für die PSO, *gestrichelt*: Strecke der weitesten Ausdehnung der Trajektorien, *blau*: initialer Drehwinkel α.

Nachdem die PSO durchgelaufen ist, kann die Position des besten *Partikels* im Suchraum als gesuchte 2D-Transformation interpretiert werden (s. Abb. 4.11a, *pink*). Sie wird als letzter Teil der Transformation als *MotionFrame* ausgegeben.



(a) *Partikel*schwarm der PSO. Die *Partikel* sind farblich nach ihrer Bewertung codiert, geringer ist dabei besser, *pink*: bester *Partikel*.



(b) Vergleich der aufgenommenen und korrigierten Trajektorie der *Microsoft Kinect v2* mit der Robotertrajektorie und den Referenzpositionen

Abbildung 4.11: Ergebnisse der PSO.

Ausgabe der gefundenen Transformation

Die Kamera ist nun im *TransformationTree* als Kombination der beiden gefundenen Teiltransformationen mit der vorher eingestellten Höhe relativ zum Roboter ausgerichtet, sodass sie ohne weiteres Zutun beim anschließenden Kartenbau mit dem Roboter „mitfährt“ und dem Mapping-Modul bekannt ist.

4.4.5 Kartenbau

Das Erstellen einer NDT-Karte übernimmt das bereits existierende *NDTMappingModule*. Es liest die korrigierte Odometrie des Roboters sowie die intrinsischen Parameter und die Tiefenbilder einer oder mehrerer Kameras ein und erstellt daraus iterativ eine NDT-Karte (s. Abb. 4.12). Diese wird als Ergebnis der Berechnung ausgegeben.

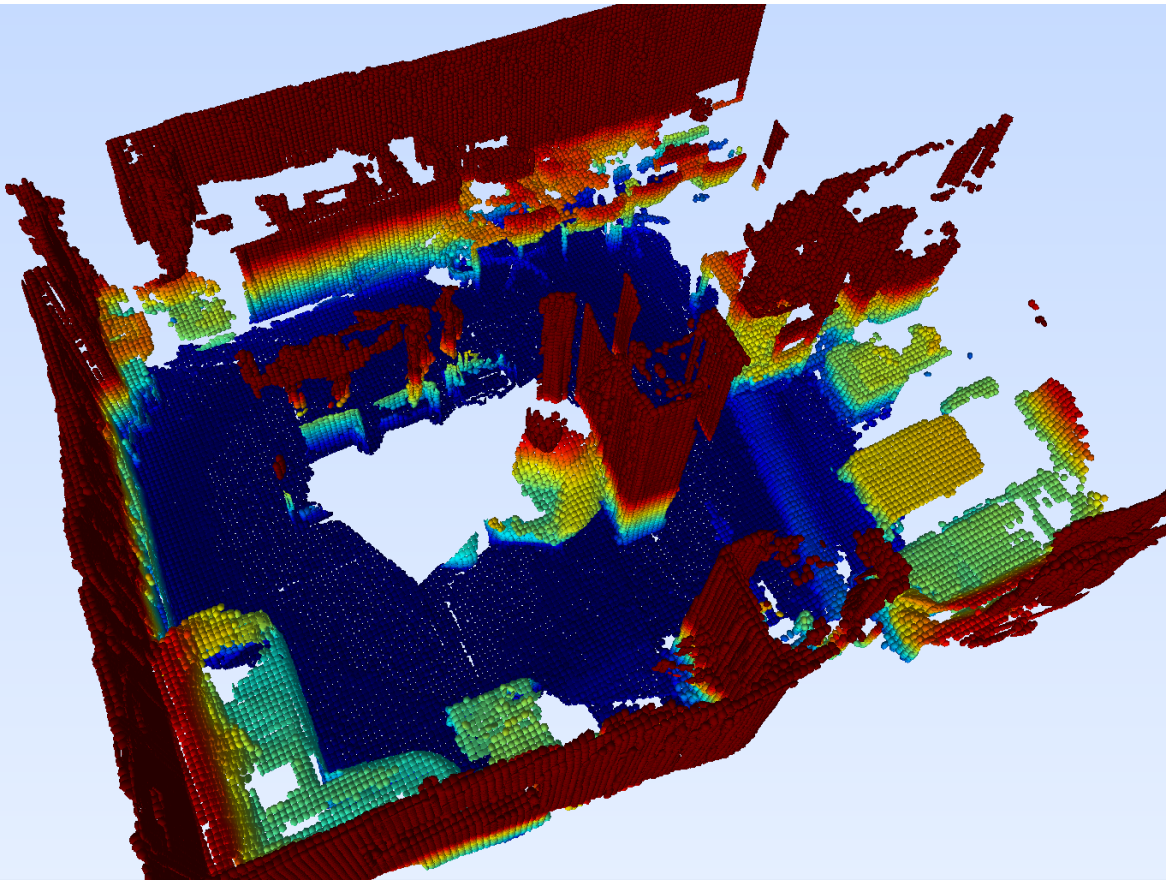


Abbildung 4.12: Erstellte NDT-Karte der Umgebung. Zu sehen sind die Normalverteilungen der NDT-Zellen, welche entsprechend ihrer Höhe farblich kodiert sind.

Kapitel 5

Experimentelle Untersuchungen

5.1 Testkonfiguration

Die in Abb. 5.1 dargestellte Konfiguration des entwickelten Algorithmus wurde bei den durchgeführten Testläufen angewandt. Sie verwendet die Daten zweier am Roboter angebrachter Tiefenkameras, einer *Microsoft Kinect v2* und einer *ASUS Xtion PRO LIVE*. Für jede Kamera wird dabei unabhängig voneinander zuerst die Trajektorie und daraus die Pose bestimmt. Darüber hinaus wird sowohl für jede Kamera einzeln als auch für beide Kameras gemeinsam jeweils eine NDT-Karte erstellt. Diese können miteinander verglichen werden, um daraus Schlüsse über die Korrektheit des Verfahrens zu ziehen. Dabei wird der prozentuale Anteil der Überdeckung der Karten berechnet, da er Aufschluss darüber geben kann, ob die Einzelkarten einen Versatz besitzen. Je größer die Überdeckung ist, desto mehr gemeinsame Zellen besitzen die Karten, was einen geringen Versatz und damit eine korrekte Ausrichtung der Kameras bedeutet. Da die Kameras nicht dieselben Bereiche des Raumes sehen, wird die Überdeckung nie vollständig sein. Als Referenzwert wird hier der prozentuale Anteil verwendet, der mithilfe vorher manuell ausgemessener und getesteter Kameraposen berechnet wurde.

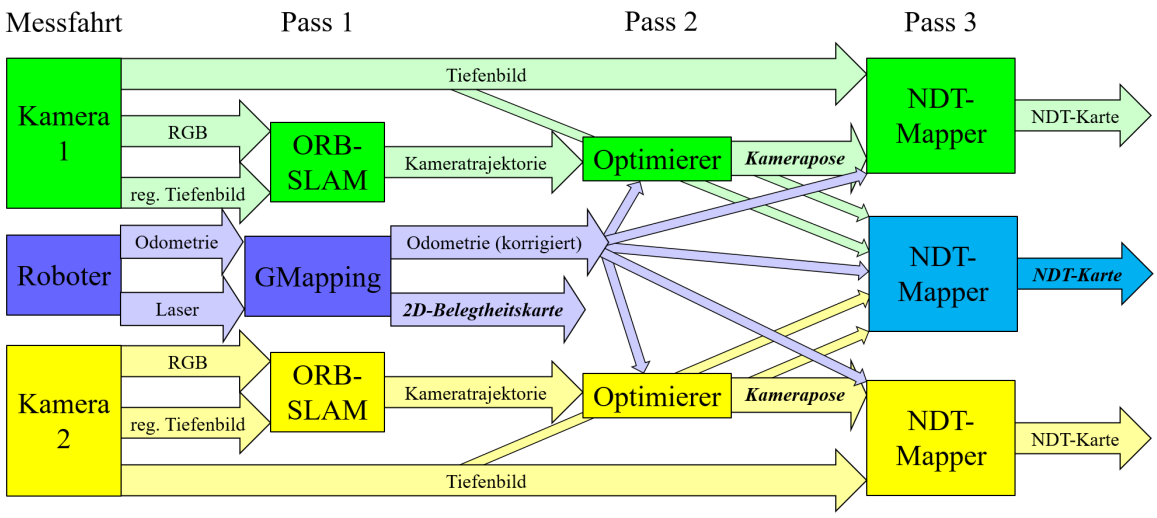


Abbildung 5.1: Flussdiagramm der Testkonfiguration unter Nutzung von 2 Kameras. ***kursiv***: Ausgabedaten, die weitere Verwendung beim Betrieb der Roboter finden

5.2 Zusätzlich verwendete Module

Während der Testphase kamen mehrere zusätzliche Mapping-Module zum Einsatz, welche der Evaluation oder der Vereinfachung der Entwicklung dienen.

5.2.1 Speichern der Referenzpositionen

Zum Zweck der Umwandlung der vom *SimpleMapper* ausgegebenen Mitteilungen über die Aktivierung oder Eliminierung einer Referenzposition (s. Kap. 5.3) zu einer Trajektorie wurde das *ReferencePointsToPosesModule* entwickelt. Dieses liest die vom *SimpleMapper* ausgesandten Daten ein, erstellt die daraus resultierende Trajektorie und gibt sie aus.

5.2.2 Laden zuvor berechneter Trajektorien & Posen

Nachdem mithilfe des *GMappingModules* oder des *ORBSLAM2Modules* eine Trajektorie berechnet wurde, wird sie in einer Datei gespeichert. Um die Entwicklung der nachfolgenden Module des Algorithmus zu beschleunigen, soll die zuvor gespeicherte

Trajektorie geladen und ausgegeben werden, sodass die Rechenzeit der zuvor genannten Module eingespart wird. Hierzu wurde das *TrajectoryPublisherModule* entwickelt. Gleiches gilt für die mithilfe des *TrajectoryAlignerModules* berechneten Posen. Hierzu wurde das *PosePublisherModule* implementiert.

5.2.3 Evaluierung der erstellten Karten

Zum Zweck der Evaluierung der gebauten Karten wurde das *NDTMapBenchmarker-Module* entwickelt. Dieses nimmt die beiden NDT-Karten, die mithilfe jeder Kamera einzeln erstellt wurden, und diejenige, welche mit beiden Kameras gemeinsam gebaut wurde. Es zählt in allen Karten die Zellen, in denen mindestens eine frei konfigurierbare Anzahl Punkte vorhanden ist, und berechnet aus diesen Werten die prozentuale Überdeckung der NDT-Karten. Dieser Wert wird als Ergebnis der Evaluation ausgegeben.

5.3 Ablauf der Testfahrt

Um später Aussagen über die Genauigkeit des Verfahrens treffen zu können, müssen zuvor vom Roboter während der Messfahrt anzufahrenden Referenzpositionen im *SimpleMapper* eingestellt werden (s. Abb. 5.2).
Wurde die Aufnahme gestartet, müssen diese Referenzpositionen aktiviert werden, wenn sich der Roboter an der entsprechenden Stelle befindet (s. Abb. 5.3). Sie werden in der vorliegenden Arbeit durch zuvor ausgemessene Markierungen auf dem Boden gekennzeichnet.
Da diese Schritte nur zur Evaluation notwendig sind, entfallen sie im späteren Produktiveinsatz.

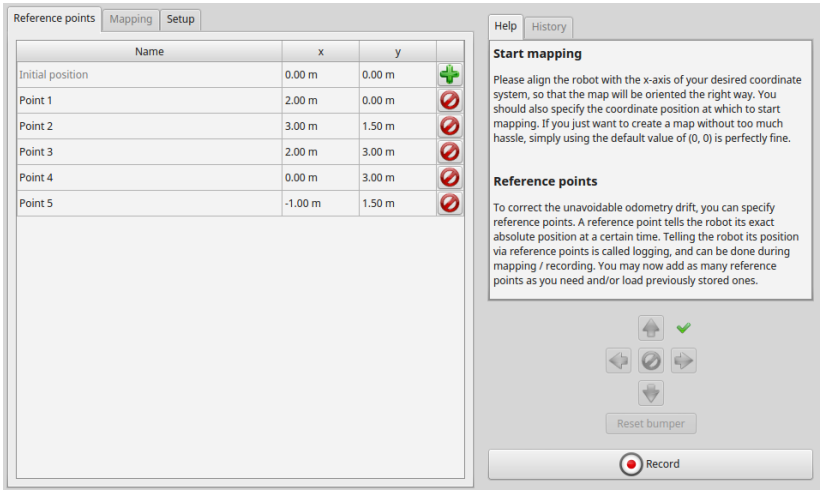


Abbildung 5.2: Eingabe der Referenzpunkte im *SimpleMapper*. Durch Betätigung der Schaltfläche mit dem grünen *Plus* können neue Referenzpunkte hinzugefügt und danach positioniert werden, durch Betätigung der roten Schaltfläche werden bereits angelegte Referenzpunkte eliminiert. Durch Auswahl der Schaltfläche *Record* (*unten rechts*) wird die Aufnahme gestartet.

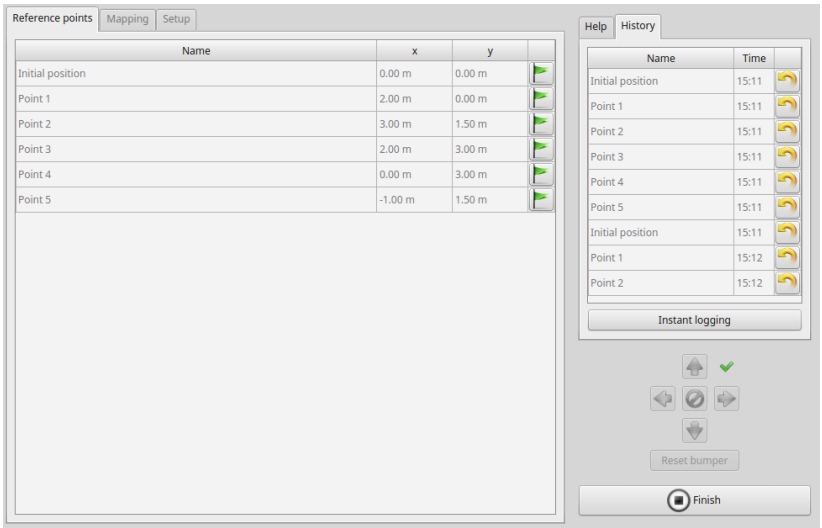


Abbildung 5.3: Aktivierung der Referenzpunkte im *SimpleMapper*. Durch Betätigung der Schaltflächen mit der grünen Fahne wird der entsprechende Referenzpunkt aktiviert und zur Liste *rechts* hinzugefügt, durch Betätigung der dort befindlichen gelben Schaltflächen werden bereits aktivierte Referenzpunkte wieder gelöscht. Mit Auswahl der Schaltfläche *Finish* (*unten rechts*) wird die Aufnahme beendet.

5.4 Ergebnisse

Es wurden während der Testphase mehrere Testfahrten durchgeführt, von denen jedoch nur eine einzige akzeptable Ergebnisse lieferte. Die anderen Testfahrten trugen jedoch dazu bei, Probleme zu erkennen und die in Tab. 4.1 genannten Anforderungen an die Messfahrt zu spezifizieren. Somit steht zur Auswertung eine Aufnahme zur Verfügung.
Bei der verwendeten Testfahrt ist der Roboter in einer definierten Testumgebung dreimal eine Kreisstrecke durch den Raum gefahren worden. Auf diesen Daten wurde die Referenzberechnung einmalig und die gesamte Testkonfiguration zehn mal laufen gelassen.

5.4.1 Fehler der Trajektorien

Zum einen wird die Qualität der Trajektorien einer einzelnen Berechnung bewertet. Dazu wurde als Fehler zwischen zwei Trajektorien der mittlere euklidische Abstand zwischen zeitsynchronen Positionen bestimmt. Zur Zeitsynchronisation wurde eine Trajektorie auf die Zeitstempel der jeweils anderen interpoliert. Damit dies bei Verwendung der *ORB-SLAM 2*-Trajektorien keine Probleme bereitete, wurden jeweils nur die berechneten Werte der ersten Kreisrunde verwendet. Verglichen wurden dabei folgende Trajektorien-Paare:

- Robotertrajektorie - Referenzpositionen
- Robotertrajektorie - Kameratrajektorie *Microsoft Kinect v2*
- Robotertrajektorie - Kameratrajektorie *ASUS Xtion PRO LIVE*
- Kameratrajektorie *Microsoft Kinect v2* - Referenzpositionen
- Kameratrajektorie *ASUS Xtion PRO LIVE* - Referenzpositionen

Die Ergebnisse sind in Abb. 5.4 zu sehen.

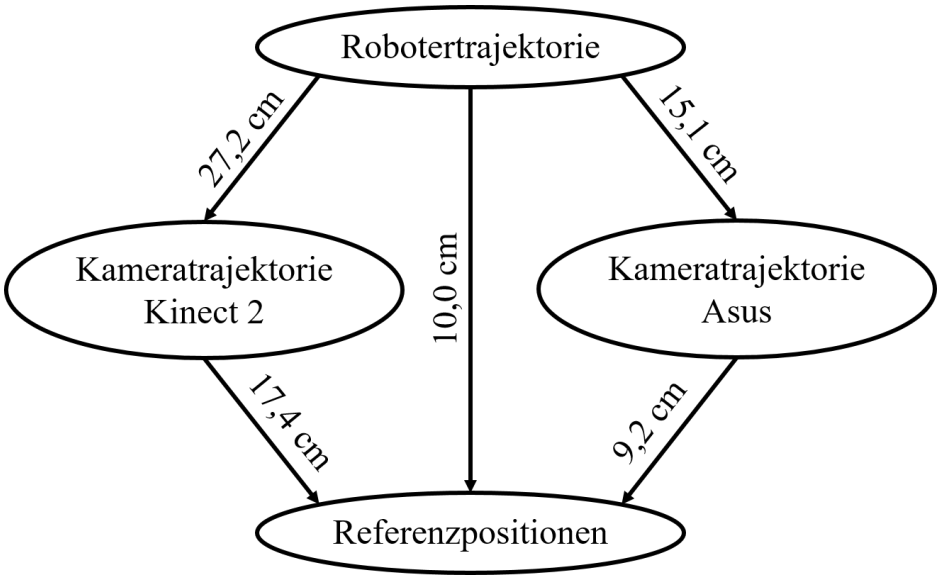


Abbildung 5.4: Mittlerer euklidischer Fehler der Trajektorien. Dabei wurde die Trajektorie, von der der Pfeil weg zeigt, auf die Zeitstempel der anderen Trajektorie interpoliert und darauf der Fehler berechnet.

5.4.2 Fehler der Posen & Überdeckung der NDT-Karten

Zum anderen können die berechneten Kameraposen mit den zuvor manuell eingestellten Posen verglichen werden (s. Tab. 5.1). Hierbei sollten die berechneten Posen nahe an der ausgemessenen Referenz liegen.
Des Weiteren werden in der Tabelle die berechneten Kartenüberdeckungen gegenübergestellt. Je größer der Wert ist, desto besser stimmen die Karten überein und desto genauer sind die Kameras zueinander positioniert.

Tabelle 5.1: Ergebnisse der Berechnung

Pose *Microsoft Kinect v2*			
	Referenz	Ergebnisse der Berechnung	
		Mittelwert	Standardabweichung
x	3,0 cm	2,3 cm	4,1 cm
y	-10,0 cm	-24,6 cm	2,8 cm
z	122,5 cm	122,5 cm	0,0 cm
Gier	0,0 °	-1,3 °	1,0 °
Nick	0,0 °	-1,1 °	0,0 °
Roll	0,0 °	0,6 °	0,1 °
Pose *ASUS Xtion PRO LIVE*			
	Referenz	Ergebnisse der Berechnung	
		Mittelwert	Standardabweichung
x	8,0 cm	7,7 cm	4,1 cm
y	5,0 cm	-1,4 cm	2,6 cm
z	115,0 cm	115,5 cm	0,0 cm
Gier	0,0 °	-0,2 °	1,0 °
Nick	55,0 °	55,5 °	0,7 °
Roll	0,0 °	1,6 °	1,2 °
Überdeckung NDT-Karten			
	Referenz	Ergebnisse der Berechnung	
		Mittelwert	Standardabweichung
	10,1 %	7,7 %	0,6 %

5.5 Bewertung der Ergebnisse

5.5.1 Fehler der Trajektorien

Wie in den Abb. 5.4 und 5.5 zu sehen ist, weichen die drei Trajektorien sehr von den Referenzpositionen ab, die *Microsoft Kinect v2* dabei am weitesten. Dies kann mehrere Ursachen haben. Zum einen sind für die Parameter der *GMapping*- und *ORB-SLAM 2*-Algorithmen noch nicht die optimalen Einstellungen gefunden worden, sodass sich daraus eine ungenaue Trajektorie ergibt. Zum anderen kann auch die Umgebung einen Einfluss auf die berechnete Trajektorie haben. Unter Umständen sind hier noch nicht alle Anforderungen an die Messfahrt (s. Tab. 4.1) zu Genüge erfüllt. Durch weitere Testfahrten, auch in anderen Umgebungen, können die Parameter optimiert und die Fehler weiter minimiert werden.

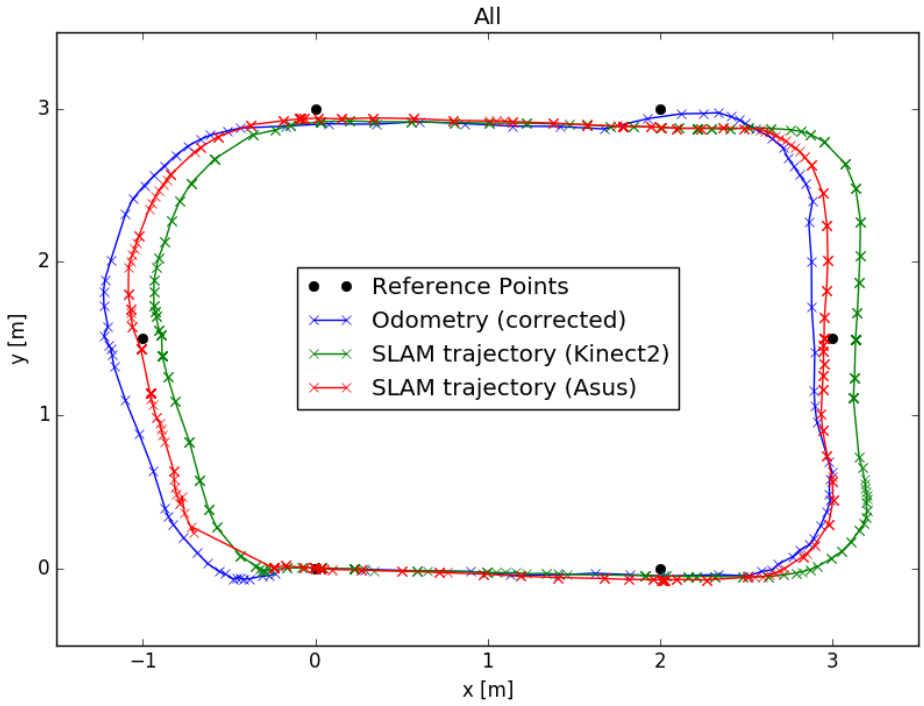
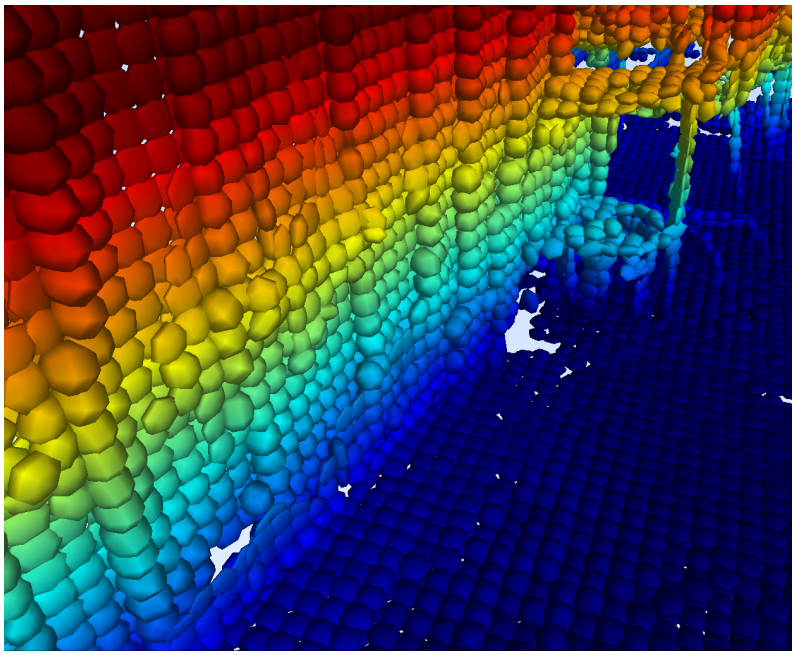


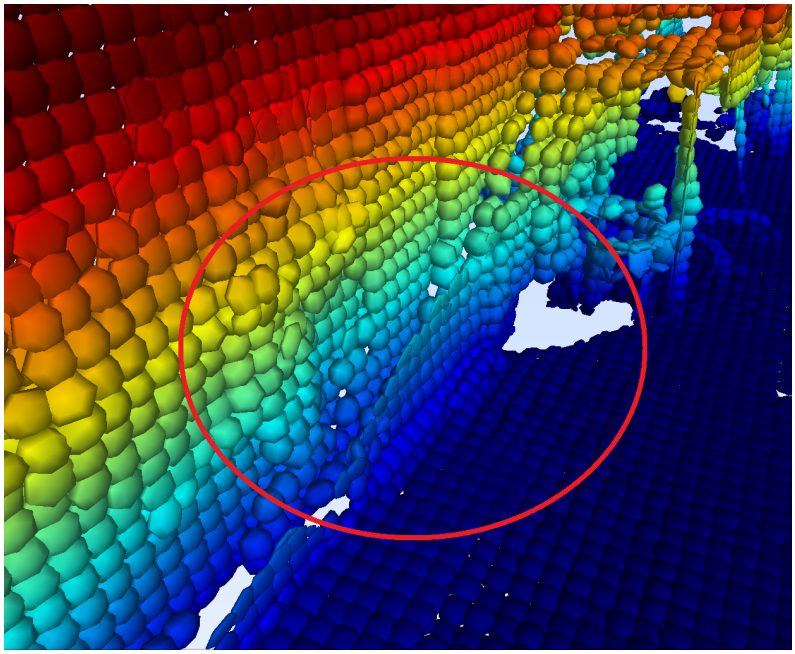
Abbildung 5.5: Vergleich der korrigierten Trajektorien der Kameras mit der Robotertrajektorie und den Referenzpositionen.

5.5.2 Fehler der Posen & Überdeckung der NDT-Karten

Wie in Tab. 5.1 zu sehen ist, weichen die berechneten Kameraposen erheblich von den Referenzposen ab. Die geringere Überdeckung deutet darauf hin, dass die Abweichungen nicht positiv zu bewerten sind. Jedoch kann festgestellt werden, dass die Überdeckung noch immer in ausreichendem Maße existiert und die Kamerawinkel der Referenz schon sehr nahe kommen, sodass davon ausgegangen werden kann, dass der Optimierungsprozess grundsätzlich korrekt funktioniert und die Ergebnisse nur aufgrund der zuvor genannten Probleme beim Erstellen der Trajektorien unzureichend sind.



(a) Mit voreingestellten Parametern gebaute NDT-Karte. Es ist kein Versatz in den Daten zu erkennen.



(b) Mit berechneten Parametern gebaute NDT-Karte. Im mittleren Bereich des Bildes ist ein deutlicher Versatz in den Daten zu erkennen (*rot eingekreist*).

Abbildung 5.6: Gebaute NDT-Karten. Zu sehen sind die Normalverteilungen der NDT-Zellen, welche entsprechend ihrer Höhe farblich kodiert sind.

Kapitel 6

Zusammenfassung und Ausblick

6.1 Zusammenfassung der Ergebnisse

Das Ziel des in dieser Arbeit erstellten Systems war es, mithilfe der während einer Messfahrt aufgenommenen Daten die Posen von auf dem Roboter montierten Kameras zu bestimmen. Dazu wurden die Trajektorien des Roboters und der Kameras durch SLAM-Algorithmen anhand der Laser-, Bild- und Tiefendaten erstellt. Diese wurden auf die Grundebene projiziert und zueinander ausgerichtet, wobei die freien Parameter der Kameraposen berechnet wurden. Des Weiteren werden mithilfe dieser Posen 2D-Belegtheits- und NDT-Karten der Umgebung gebaut, die im Produktiveinsatz genutzt werden können.

Im Rahmen der Analyse zeigte sich, dass die Kameras ungenau ausgerichtet werden und die daraus resultierende NDT-Karte dadurch fehlerhaft ist, da die SLAM-Algorithmen noch nicht optimal parametrisiert sind. Grundsätzlich lässt sich jedoch festhalten, dass der Optimierungsprozess zur Ausrichtung der Trajektorien zueinander funktioniert.

6.2 Ausblick

Wie in Kap. 5 angesprochen, ist das System nicht vollständig fehlerfrei. Hauptproblem sind die suboptimalen Parameter der *GMapping*- und *ORB-SLAM 2*-Algorithmen. Diese sollten in weiteren Tests untersucht und optimiert werden.
Hat der Versatz der einzelnen NDT-Karten ihre spatiale Auflösungsgrenze unterschritten, wird eine weitere Möglichkeit zur Evaluation benötigt. Hier kann die Verteilung der in den einzelnen Zellen liegenden Punkte Aufschluss geben. Sind die „*Smarties*“ eher rund als flach, d.h. ist die Streuung entlang der berechneten Zellennormale groß, deutet dies auf einen existierenden Versatz der einzelnen Karten hin. Hierzu sollte eine Implementierung erfolgen, welche dies überprüft.
Eine weitere Möglichkeit, die Ergebnisse zu verbessern, ist, die direkte Transformation zwischen den beiden Kameras zu bestimmen. Hier kann als Kostenfunktion des Optimierungsprozesses die Überdeckung der Karten als primärer und die zuvor beschriebene Streuung innerhalb der Zellen als sekundärer Faktor herangezogen werden. Dies würde jedoch zu einem sehr hohen Rechenaufwand führen, da hierzu für jedes *Partikel* der PSO in jedem Schritt eine komplette NDT-Karte mithilfe beider Kameras erstellt werden müsste.
Um die Automatisierung des Systems zu vervollständigen, sollte die Höhe der Kamera über dem Boden vom System selbstständig berechnet werden. Dies kann durch eine Bodenplattenschätzung geschehen. Weil jedoch nicht jede Kamera zwangsläufig den Boden sieht, sollte auswählbar sein, ob dieser Schritt automatisch durchgeführt werden oder die manuelle Eingabe verwendet werden soll.
Da mit dem implementierten System nur Kameras ausgerichtet werden können, die sowohl Farb- als auch Tiefendaten ausgeben, können *Depth-only*-Kameras nicht berücksichtigt werden. Hier kann jedoch der Ansatz des *One-Shot*-Matchings (s. Kap. 3.2.5) Abhilfe schaffen.

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

BRIEF Binary Robust Independent Elementary Features

FAST Features from Accelerated Segment Test

IRON Interest Point Descriptor for Robust NDT-Map Matching

MIRA Middleware for Robotic Applications

NDT Normal Distribution Transform

NIKR *Neuroinformatik und Kognitive Robotik*

ORB Oriented FAST and Rotated BRIEF

PSO Particle-Swarm-Optimization

RANSAC Random Sample Consensus

SLAM Simultaneous Localization and Mapping

Glossar

Channel	Datenkanal in *MIRA*
Disparität	Versatz von korrespondierenden Punkten in verschiedenen Bildern derselben Szene
Epipolargeometrie	stellt die geometrischen Beziehungen von korrespondierenden Bildpunkten auf verschiedenen Kamerabildern derselben Szene dar
extrinsische Parameter	äußere Parameter
Feature	charakteristisches Merkmal innerhalb eines Bildes, z.B. Ecken
intrinsische Parameter	innere Parameter
Mapping	Kartenbau
Matching	Suche nach Übereinstimmungen und Ausrichtung zueinander
Middleware	Software als Zwischenschicht zur Entkopplung von verschiedenen Softwarekomponenten
Pass	Durchlauf eines *MIRA-Tapes*
Pose	Kombination aus Position und Ausrichtung

Simultaneous Localization and Mapping	gleichzeitiges Erstellen einer Karte der Umgebung und Lokalisation in dieser
Slot	Datenpaket auf einem *MIRA-Channel*
Tape	Datei, in welcher Daten-*Slots* aus *MIRA-Channels* gespeichert sind
Tracking	Detektion und Verfolgung eines Objektes
Trajektorie	Bewegungsspur im Raum

Literaturverzeichnis

[Ataer-Cansizoglu et al., 2014] Ataer-Cansizoglu, Esra, Y. Taguchi, S. Ramalingam und Y. Miki (2014). *Calibration of non-overlapping cameras using an external SLAM system*. In: *International Conference on 3D Vision*, Bd. 1, S. 509–516. IEEE.

[Berger et al., 2011] Berger, Kai, K. Ruhl, Y. Schroeder, C. Bruemmer, A. Scholz und M. A. Magnor (2011). *Markerless motion capture using multiple color-depth sensors*. In: *Vision, Modeling and Visualization Workshop*, S. 317–324.

[Eberhart und Kennedy, 1995] Eberhart, R. und J. Kennedy (1995). *A new optimizer using particle swarm theory*. In: *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, S. 39–43.

[Einhorn und Gross, 2015] Einhorn, Erik und H.-M. Gross (2015). *Generic NDT mapping in dynamic environments and its application for lifelong SLAM*. Robotics and Autonomous Systems, 69:28–39.

[Einhorn et al., 2012] Einhorn, Erik, T. Langner, R. Stricker, C. Martin und H.-M. Gross (2012). *MIRA - middleware for robotic applications*. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, S. 2591–2598.

[Esquivel et al., 2007] Esquivel, Sandro, F. Woelk und R. Koch (2007). *Calibration of a multi-camera rig from non-overlapping views*. In: *Joint Pattern Recognition Symposium*, S. 82–91. Springer.

[Grisetti et al., 2005] Grisetti, Giorgio, C. Stachniss und W. Burgard (2005). *Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by*

Adaptive Proposals and Selective Resampling. In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, S. 2432–2437.

[Gross, 2015] Gross, H. M. (2015). *Vorlesungsskript Kognitive Systeme und Robotik (SS 2015)*, Book Section 2, S. 19–23. FG NIKR, TU Ilmenau.

[Kumar et al., 2008] Kumar, Ram Krishan, A. Ilie, J.-M. Frahm und M. Pollefeys (2008). *Simple calibration of non-overlapping cameras with a mirror*. In: *IEEE Conference on Computer Vision and Pattern Recognition*, S. 1–7. IEEE.

[Lamprecht et al., 2007] Lamprecht, Bernhard, S. Rass, S. Fuchs und K. Kyamakya (2007). *Extrinsic camera calibration for an on-board two-camera system without overlapping field of view*. In: *IEEE Intelligent Transportation Systems Conference*, S. 265–270. IEEE.

[MetraLabs] MetraLabs. *MetraLabs GmbH | Neue Technologien und Systeme*. `http://www.metralabs.com/`. Eingesehen am: 07.07.2017.

[Mur-Artal et al., 2015] Mur-Artal, Raul, J. M. M. Montiel und J. D. Tardos (2015). *ORB-SLAM: A Versatile and Accurate Monocular SLAM System*. IEEE Transactions on Robotics, 31(5):1147–1163.

[Mur-Artal und Tardos, 2017] Mur-Artal, Raul und J. D. Tardos (2017). *ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras*. IEEE Transactions on Robotics, PP(99):1–8.

[Pagel, 2009] Pagel, Frank (2009). *Extrinsische Kalibrierung einer Multikameraanordnung mit nichtüberlappenden Sichtfeldern auf einer mobilen Plattform*. In: *Workshop Fahrerassistenzsysteme*.

[Pagel und Willersinn, 2011] Pagel, Frank und D. Willersinn (2011). *Kontinuierliche Kalibrierung und Fusion mobiler Kamerasysteme mit nicht überlappenden Sichten in Fahrzeugen*. In: *Workshop Fahrerassistenzsysteme*.

[Pedersini et al., 1999] Pedersini, F., A. Sarti und S. Tubaro (1999). *Accurate and simple geometric calibration of multi-camera systems*. Signal Processing, 77(3):309–334.

[Rahimi et al., 2004] Rahimi, Ali, B. Dunagan und T. Darrell (2004). *Simultaneous calibration and tracking with a network of non-overlapping sensors*. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Bd. 1, S. I–I. IEEE.

[Rublee et al., 2011] Rublee, Ethan, V. Rabaud, K. Konolige und G. Bradski (2011). *ORB: An efficient alternative to SIFT or SURF*. IEEE International Conference on Computer Vision, S. 2564–2571.

[Schmiedel et al., 2015] Schmiedel, Thomas, E. Einhorn und H.-M. Gross (2015). *IRON: A fast interest point descriptor for robust NDT-map matching and its application to robot localization*. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, S. 3144–3151.

[Taketomi et al., 2017] Taketomi, Takafumi, H. Uchiyama und S. Ikeda (2017). *Visual SLAM algorithms: a survey from 2010 to 2016*. IPSJ Transactions on Computer Vision and Applications, 9(1):16.

[Zhang, 2000] Zhang, Zhengyou (2000). *A flexible new technique for camera calibration*. IEEE Transactions on pattern analysis and machine intelligence, 22(11):1330–1334.