Online-6D-SLAM für RGB-D-Sensoren

6D Visual SLAM for RGB-D Sensors

Felix Endres, Jürgen Hess, Nikolas Engelhard, Jürgen Sturm, Wolfram Burgard

Zur Automatisierung komplexer Manipulationsaufgaben in dynamischen oder unbekannten Umgebungen benötigt die Steuerungssoftware eines autonomen Roboters eine Repräsentation des Arbeitsbereiches, mit der die Kollisionsfreiheit bei der Durchführung der Aufgabe gewährleistet werden kann. Dieser Beitrag beschreibt ein neues System zur Erstellung von 3D-Umgebungsrepräsentationen aus den RGB-D-Daten neuartiger Kameras, wie der Microsoft Kinect. Durch die Unabhängigkeit von weiterer Sensorik ist der Ansatz insbesondere zur Ergänzung von rein bildbasierten Regelungssystemen geeignet.

This paper presents an approach to 6-DOF simultaneous localization and mapping (SLAM) particularly suited for collision avoidance in visually guided robotic manipulation tasks in dynamic or unknown environments. We exploit the properties of novel RGB-D sensors such as the Microsoft Kinect to build highly accurate voxel maps.

Schlagwörter: Visual SLAM, RGB-D-Sensoren

Keywords: Visual SLAM, RGB-D Sensors

1 Einleitung

In der Robotik ist eine aktuelle und genaue Repräsentation der Umgebung eine wichtige Grundlage in vielen Aufgabenbereichen, die autonomes Verhalten erfordern. Für Bewegungsplanung die über den Sichtbereich des Sensors hinausgeht, zum Beispiel bei der autonomen Navigation mobiler Roboter, ist eine Karte der Umgebung unerlässlich, um kollisionsfreie Pfade planen zu können. Dasselbe gilt für Manipulationsroboter, die genau über die Position von Objekten und Hindernissen in ihrem Arbeitsbereich informiert sein müssen. Abbildung 1 zeigt eine solche Situation, in der relevante Bereiche der Arbeitsumgebung außerhalb der aktuellen Sensorwahrnehmung sind. Ohne eine Karte der Umgebung ist es hier nicht möglich Kollisionen des Roboters und der transportierten Ladung mit der Umgebung auszuschließen.

Bild 1: Nach dem Anfahren des Ziels ist ein großer Teil der Umgebung nicht mehr im Sichtbereich des auf dem Greifer plazierten Sensors. Zur weiteren Gewährleistung einer kollisionsfreien Trajektorie benötigt der Roboter daher eine dreidimensionale Belegungskarte des Arbeitsbereichs. Das im vorliegenden Beitrag beschriebene System erlaubt es eine solche Karte simultan zur Ansteuerung des Ziels zu erstellen.

Methoden zur Erstellung einer solchen Umgebungsrepräsentation bzw. Karte stehen in der mobilen Robotik vor der Herausforderung, diese aus Sensordaten zu berechnen, ohne dass die Posen des Sensors über die Zeit hinweg bekannt sind. Die Karte der Umgebung muss also erstellt werden, während der Sensor gleich-

Bild 2a: Farb- und Tiefenbilder der Sequenz „FR1 Desk2“

Bild 2b: Resultierende 3D-Belegungskarte für „FR1 Desk2“

zeitig anhand der Karte lokalisiert wird. Dieses Vorgehen nennt sich Simultaneous Localization and Mapping (SLAM) [28]. Lösungsansätze für das SLAM-Problem sind stark abhängig von den verfügbaren Sensoren und von der Art der erstellten Karte.

Für mobile Plattformen im industriellen Einsatz werden zumeist Laserscanner eingesetzt, die in hoher Frequenz die Entfernungen zu den nächstgelegenen Hindernissen in einem Sichtfeld von typischerweise 180° bis 270° in einer Ebene messen. Vorteile sind eine hohe Präzision, hohe Verlässlichkeit und Unabhängigkeit von Lichtverhältnissen. Für fahrbare Plattformen mit solchen Sensoren nutzen State-of-the-art-SLAM-Algorithmen die Odometrieinformationen aus dem Fahrwerk als Ausgangspunkt für die Registrierung der einzelnen Sensormessungen. Aus der Registrierung ergibt sich wiederum eine Korrektur der Trajektorie des Roboters. Mit der optimierten Trajektorie und den zugehörigen Sensormessungen wird eine geeignete Repräsentation der Karte, wie z.B. eine probabilistische Belegungskarte in 2D, berechnet.

Für komplexe Bewegungsplanung bei Manipulationsrobotern mit vielen Freiheitsgraden reicht eine zweidimensionale Belegungskarte als Umgebungsrepräsentation üblicherweise nicht aus. Je nach Einsatzgebiet können hier Erweiterungen wie Elevation Maps oder Multilevel Surface Maps ausreichen. Im Allgemeinen wird aber eine dreidimensionale Karte benötigt, um beispielsweise Kollisionen mit dreidimensionalen Hindernissen vermeiden zu können.

Neuartige RGB-D-Sensoren, wie z.B. die Microsoft Kinect, liefern gleichzeitig Farb- und Tiefenbilder in VGA-Auflösung bei 30 Hz. Dadurch stehen der Automatisierungstechnik nun hochfrequente, dichte räumliche Entfernungsmessungen (sogenannte Punktwolken) gepaart mit visuellen Informationen zu Verfügung, wie sie mit bisherigen Sensoren nur unter hohem Kosten- und Rechenaufwand erzeugt werden konnten. Durch die geringen Kosten (100€ – 200€) ist zu erwarten, dass durch den Einsatz dieser Sensoren viele neue Anwendungen erschlossen werden. Insbesondere Low-Cost-Anwendungen die sich auf Kameradaten stützen, können durch die Verfügbarkeit dichter 3D-Information profitieren.

Das in diesem Beitrag vorgestellte SLAM-System berechnet aus solchen RGB-D-Sensordaten in Echtzeit hochakkurate Karten ohne Beschränkung der Freiheitsgrade für die Sensortrajektorie und ohne Abhängigkeit von weiterer Sensorik. Sie ist damit hervorragend für die Kombination mit visuellen Verfahren wie Visual Servoing geeignet. Im folgenden Abschnitt werden vorangegangene Arbeiten zur Schätzung der Sensortrajektorie sowie der Aufbau der Umgebungsrepräsentation im Detail besprochen. Darauf folgt eine Analyse des vorgestellten Systems in Bezug auf Genauigkeit und Laufzeit anhand öffentlich verfügbarer Datensätze.

2 Verwandte Arbeiten

Das Problem gleichzeitiger Lokalisierung und Kartierung ist schon lange Gegenstand der Forschung in der Robotik [27, 20, 9, 5, 14, 8, 21]. Um dreidimensionale Karten der Umgebung zu erstellen, werden dabei zumeist Laser-Scanner oder Time-of-Flight-Kameras mit dichten Tiefeninformationen verwendet. Die meisten aktuellen Systeme verwenden den Iterative-Closest-Point-(ICP-)Algorithmus oder eine seiner Varianten [2, 23, 24], um über die Registrierung der Punktwolken geometrische Relationen zwischen den Observationen zu finden. Diese Relationen werden dann genutzt, um eine Maximum-Likelihood-Karte zu erstellen. Fioraio und Konolige [6] verwenden in einem kürzlich vorgestellten Ansatz das ICP-Verfahren zum Registrieren von Punktwolken einer Kinect, jedoch ohne die Möglichkeiten der Farbkamera auszunutzen.

Im Gegensatz dazu basiert Visual SLAM [4, 15, 25] – oft auch als *structure and motion estimation* [13, 19] bezeichnet – meist darauf, über die vektorielle Merkmalsbeschreibung einzelner Bildpunkte Assoziationen zwischen verschiedenen Observationen herzustellen. Während bei den Varianten von ICP die Punktassoziationen über paarweise Distanzen berechnet und in jeder Iteration erneuert werden, sind Assoziationen im Visual SLAM anhand der paarweisen Distanz von Merkmalsvektoren gegeben. Diese können mit dem gegen Ausreißer robusten RANSAC-(Random-Sample-Consensus-)Algorithmus [7]) effizient auf Fehlassoziationen überprüft werden.

Um visuelle Merkmale zur Assoziation von Bildpunkten zu nutzen, wird generell ein dreistufiger Prozess durchlaufen. Der „Detektor“ durchsucht das Bild nach Interessenpunkten, deren Lage und Aussehen möglichst eindeutig beschrieben werden kann. Dafür ist es vorteilhaft, wenn die Merkmalsbeschreibung möglichst invariant gegenüber Veränderungen der Kameraposition und -lage ist. Eine solche Invarianz lässt sich z.B. dadurch erreichen, dass für jeden Interessenpunkt eine normierte Pose bestimmt wird, die die Bildposition, -orientierung und -skalierung möglichst eindeutig fixiert. Zwei geeignete und in Anwendungen wie Visual SLAM und Visual Servoing bewährte Verfahren zur Merkmalsgewinnung sind SIFT [18] und SURF [1]. In beiden Fällen werden Beschreibungen in Form eines hochdimensionalen Merkmalsvektors extrahiert. Da die Varianz der Beschreibungen im Merkmalsraum nicht einheitlich ist, ist der Abstand zwischen den Merkmalsvektoren kein aussagekräftiges Maß für deren Assoziation. Um die Merkmale zweier Bilder zu assoziieren, werden daher jeweils die zwei räumlich nächsten Nachbarn eines Merkmals gesucht und die Eindeutigkeit der Assoziation anhand des Verhältnisses der Abstände evaluiert [18].

Der von Henry et al. [10] vorgestellte Ansatz nutzt wie der hier vorgestellte auch visuelle Merkmale und GICP um RGB-D-Punktwolken zu registrieren. GICP wird von Henry et al. dabei für jede Transformation benutzt. Im Gegensatz zur von Henry et al. erstellten Repräsentation durch Surfels (kleine Oberflächenelemente) wird im hier vorgestellten System eine volumetrische Belegungskarte [30] erstellt, wie sie für die Pfadplanung und Navigation mobiler Roboter verwendet werden kann [12]. Da weder die Software noch die Daten der Auswertungen in [10] öffentlich verfügbar sind, ist ein direkter Vergleich der rekonstruierten Trajektorien nicht möglich, es ist jedoch eine ähnliche Genauigkeit zu erwarten.

Im Gegensatz dazu sind sowohl die hier vorgestellte Software[1] als auch die Auswertungsmethoden und -daten [26] öffentlich verfügbar, damit die Ergebnisse reproduziert werden und als Vergleichsbasis für neue Methoden dienen können. Die Software und Dokumentation stehen als Open Source unter der GPL für Erweiterungen zur Verfügung.

3 RGB-D-SLAM

Vollständige SLAM-Systeme bestehen in der Regel aus drei Hauptkomponenten: Das so genannte *Frontend* interpretiert die Sensordaten und liefert die geometrischen Verknüpfungen zwischen einzelnen Messungen. Das *Backend* vereint diese geometrischen Relationen und berechnet daraus die Trajektorie des Sensors (oder der Sensoren) mit der maximalen Wahrscheinlichkeit. Eine weitere Komponente berechnet aus den Rohdaten

[1] http://ros.org/wiki/rgbdslam

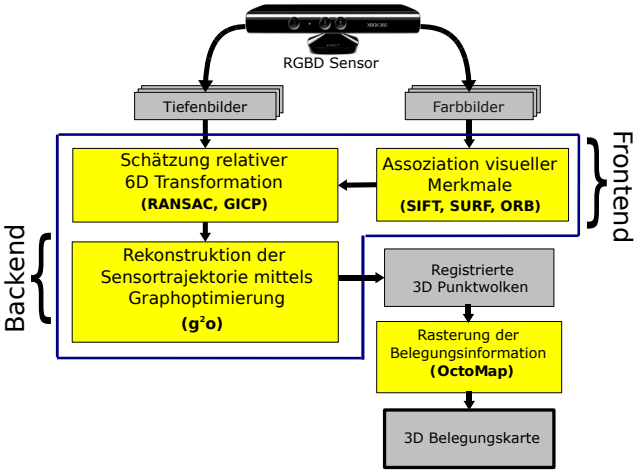

Bild 3: Überblick: Die Transformationen zwischen den Sensorposen einzelner RGB-D-Bilder werden über Merkmalsassoziationen oder Generalized Iterative Closes Point geschätzt. Aus den einzelnen Schätzungen wird durch Graphoptimierung die Trajektorie mit dem geringsten quadratischen Fehler berechnet. Diese ist die Grundlage zum Aufbau einer probabilistischen 3D-Belegungskarte aus den Sensordaten.

und der Trajektorie eine effizient nutzbare Kartenrepräsentation (*Post-Processing*).

3.1 Sensorik

RGB-D-Sensoren wie z.B. die Microsoft Kinect und die Asus Xtion Pro liefern hochauflösende, Farb- und Tiefenbilder mit einer Frequenz von 30 Hz. Diese Kameras projizieren ein Muster im Infrarotbereich in den Sichtbereich einer leicht versetzten Infrarotkamera. Analog zur Triangulierung bei Stereobildern kann so die Entfernung zur Projektionsfläche berechnet werden. Im Gegensatz zum Stereosehen mit zwei Farbkameras hängt die Tiefenwahrnehmung damit nicht von der Texturiertheit der Szene ab und liefert für nahezu jeden Pixel der Infrarotkamera einen Tiefenwert.

Die Tiefenwerte werden dann auf das Bild einer Farbkamera abgebildet. Abbildung 2a zeigt Beispielhaft vier Farbbilder (oben) sowie die zugehörigen Tiefenbilder (unten). Da die Messung von Tiefenwerten abhängig von der Sichtbarkeit des projizierten Infrarotmusters ist, ist die Anwendung dieser Art von Sensoren vornehmlich für Innenräume geeignet.

3.2 SLAM-Backend

Reines Tracking schätzt für jeden Zeitpunkt die Transformation zur vorherigen Pose. Diese Transformationen ergeben eine lineare Kette. Werden die vorhergehenden Schätzungen akkumuliert, ergibt sich die aktuelle globale Positionsschätzung eines Zeitpunkts. Dieses Vorgehen erweist sich allerdings als problematisch, da sich die Fehler der einzelnen Schätzungen aggregieren und dadurch die Genauigkeit der Lokalisierung stetig abnimmt.

SLAM-Systeme versuchen daher, neben dem reinen Tracking sogenannte Loop Closures zu finden. Loop Closures sind Schätzungen der relativen Bewegung zwischen nicht aufeinander folgenden Messpunkten. Die lineare Kette der Transformationen wird durch diese Querverbindungen zu einer Graphstruktur, deren Knoten und Kanten die Sensorposen und deren relative Transformation repräsentieren. Kanten zwischen zeitlich versetzten Knoten bilden einen Kurzschluss in der Akkumulation des Fehlers und verhindern somit die stetige Degeneration der Positionsschätzung.

Durch die Repräsentation der Bewegungsinformation als Graph wird die Trajektorie nicht mehr durch Aufsummierung der bisherigen Transformationen gebildet. Stattdessen wird die Trajektorie $\mathbf{x}^*$ über die Minimierung einer quadratischen Fehlerfunktion $\mathbf{F}(\mathbf{x})$ berechnet:

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}} \mathbf{F}(\mathbf{x}). \tag{1}$$

$$\mathbf{F}(\mathbf{x}) = \sum_{\langle i,j\rangle \in \mathcal{C}} \underbrace{\mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})^\top \mathbf{\Omega}_{ij} \mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})}_{\mathbf{F}_{ij}} \tag{2}$$

Der Vektor $\mathbf{x} = (\mathbf{x}_1^\top, \ldots, \mathbf{x}_n^\top)^\top$ enthält die Posen $\mathbf{x}_i$. $\mathbf{z}_{ij}$ und $\mathbf{\Omega}_{ij}$ repräsentieren die vom Frontend geschätzte Transformation zwischen den Posen $\mathbf{x}_i$ und $\mathbf{x}_j$ und die der Transformation zugehörige Informationsmatrix. Die Fehlerfunktion $\mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})$ bemisst, wie gut die zu errechnenden Posen $\mathbf{x}_i$ und $\mathbf{x}_j$ auf die gemessene relative Transformation $\mathbf{z}_{ij}$ passen. Entspricht die Transformation zwischen $\mathbf{x}_i$ und $\mathbf{x}_j$ genau der Schätzung $\mathbf{z}_{ij}$, ist der Fehler $\mathbf{0}$.

Da das Problem nicht linear ist, werden häufig iterative Verfahren für die Minimierung der obigen Fehlerfunktion angewendet. Für das vorgestellte System wird die Open-Source-Graphoptimierungsbibliothek $\mathbf{g^2o}$ verwendet [16]. Die Laufzeit von $\mathbf{g^2o}$ hängt stark vom Verfahren zur Matrixinvertierung ab. Die für $\mathbf{g^2o}$ in Abschnitt 4 angegebenen Laufzeiten basieren auf der Cholesky-Zerlegung, die sich in unseren Experimenten als die schnellste Methode erwies.

3.3 SLAM-Frontend

Das Frontend eines SLAM-Systems berechnet geometrische Relationen aus den Sensordaten. Bei Sensoren, die nicht direkt die Bewegung messen, sondern die Umgebung, muss die Bewegung aus dem Unterschied der Sensormessungen rekonstruiert werden. Hierzu werden, je nach Art der Sensordaten, verschiedene Methoden verwendet. Für rein räumliche Messungen in 2.5D, wie die von Laserscannern, benutzen state-of-the-art-SLAM-Systeme meist Varianten des Iterative-Closest-Point-(ICP-)Algorithmus um die Bewegungsschätzung von einer ersten Hypothese ausgehend zu verbessern. Für die dichten 3D-Punktwolken von RGB-D-Sensoren liefert Generalized ICP [24] aufgrund der Point-to-Plane

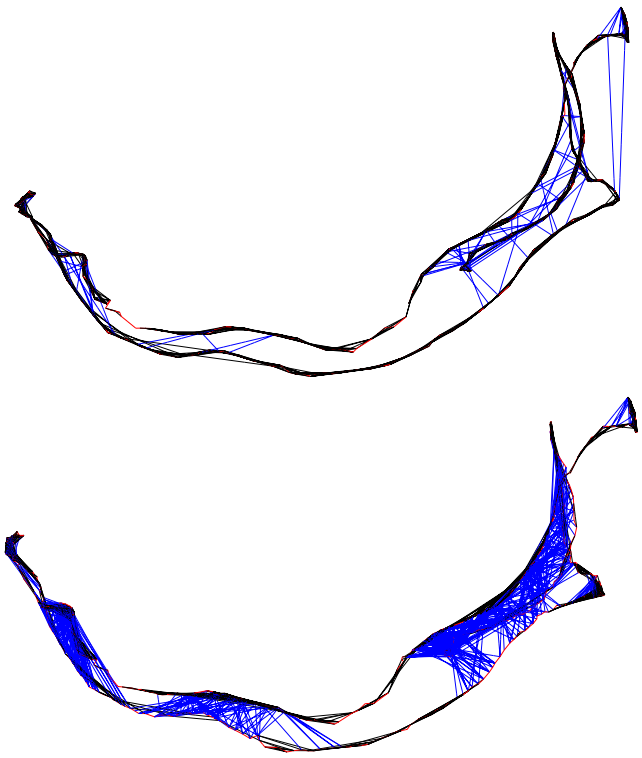

Bild 4: Posengraphen für die Sequenz „FR1 Desk2“. Blaue Kanten repräsentieren Loop Closures. Oben: Suche nach Transformation zu sequenziellen Vorgängern. Unten: Suche nach Transformation zu Nachbarn im Graph. Das Ausnutzen bestehender Loop Closures bewirkt eine deutlich stärkere Vernetzung des Graphen und somit auch eine Verminderung der Posenunsicherheit.

Metrik zur Assoziation von Punkten die besten Ergebnisse. Durch das hohe Datenaufkommen und den höherdimensionalen Suchraum (Bewegungen im Raum statt in der Ebene) ist diese Methode jedoch nicht ohne Einschränkungen übertragbar. Insbesondere kann aus dem Ergebnis von ICP nicht ohne Weiteres abgelesen werden, ob die beiden Punktwolken tatsächlich miteinander korrespondieren. So können allgegenwärtige Strukturen wie z.B. Boden und Wand zur Konvergenz von Punktwolken unterschiedlicher Orte führen. ICP ist daher sowohl von einer geometrisch guten Anfangshypothese abhängig als auch von dem Wissen, ob die Punktwolken tatsächlich korrespondieren. Im Gegensatz zu Laserscans bieten RGB-D-Sensoren zusätzlich zur räumlichen Information jedoch auch photometrische Informationen. Diese erlauben es, Punktwolken zu registrieren, indem einzelne Punkte über visuelle Merkmalsbeschreibungen assoziiert werden. Können jeweils drei oder mehr Punkte in beiden zu registrierenden Punktwolken identifiziert werden, kann eine Bewegung errechnet werden, die die Punkte mit dem kleinsten quadratischen Fehler aufeinander führt. Im vorgestellten System werden SIFT [18] und SURF [1] verwendet. Experimente mit den kürzlich vorgestellten ORB [22] ergaben schnellere Laufzeiten, aber stark erhöhte Fehler in der Trajektorie. Für SURF wird die Implementierung von

OpenCV [3] verwendet, für SIFT die GPU-basierte Implementierung SIFTGPU [29].

Unabhängig von der gewählten Beschreibungsmethode treten beim Vergleich visueller Merkmale unvermeidlich auch Fehlassoziationen auf. Daher wird für die Berechnung der Sensorbewegung der RANSAC-Algorithmus verwendet, ein iteratives Verfahren, welches robust gegenüber statistischen Ausreißern ist. Dabei werden in jeder Iteration drei zufällige Punktpaare aus der Menge der assoziierten Punkte gewählt. Da die euklidischen Abstände zwischen den jeweiligen Merkmalen eines RGB-D-Bilds in 3D berechnet werden können, kann schnell verifiziert werden, ob die paarweisen Distanzen der drei Punkte übereinstimmen. Nach erfolgreicher Verifikation wird eine erste Schätzung der Kameratransformation anhand dieser Korrespondenzen berechnet. Die Schätzung wird verwendet um die assoziierten Merkmalspositionen in ein gemeinsames Koordinatensystem zu bringen. Die Marginalwahrscheinlichkeit $p(\mathbf{y}_i, \mathbf{y}_j)$, dass zwei Positionsmessungen $\mathbf{y}_i, \mathbf{y}_j \in \mathbb{R}^3$ durch dasselbe Merkmal verursacht werden, kann dann durch Integrieren über dessen Position $\mathbf{y}^*$ berechnet werden. Unter der Annahme normalverteilter Messungenauigkeit mit den jeweiligen Kovarianzmatrizen $\mathbf{\Sigma}_i, \mathbf{\Sigma}_j$ ergibt sich

$$p(\mathbf{y}_i, \mathbf{y}_j) = \int \mathcal{N}(\mathbf{y}^* \mid \mathbf{y}_i, \mathbf{\Sigma}_i)\mathcal{N}(\mathbf{y}^* \mid \mathbf{y}_j, \mathbf{\Sigma}_j)d\mathbf{y}^* \quad (3)$$

$$= \mathcal{N}(\mathbf{y}_i - \mathbf{y}_j \mid \mathbf{0}, (\mathbf{\Sigma}_i + \mathbf{\Sigma}_j)). \quad (4)$$

Zur Bewertung der einzelnen Datenassoziation berechnen wir deshalb die dementsprechende Mahalanobisdistanz

$$d(\mathbf{y}_i, \mathbf{y}_j) = \sqrt{(\mathbf{y}_i - \mathbf{y}_j)^T(\mathbf{\Sigma}_i + \mathbf{\Sigma}_j)^{-1}(\mathbf{y}_i - \mathbf{y}_j)}. \quad (5)$$

Eine verbesserte Transformation wird dann anhand der mittels Schwellwert als korrekt bestimmten Assoziationen berechnet. Die so berechnete Transformation wird an das Backend weitergegeben, wobei die Sicherheit der Transformation mit der Anzahl an korrekten Assoziationen gewichtet wird.

Die Laufzeiten für die Suche nach möglichen Merkmalskorrespondenzen im Merkmalsraum zweier Bilder sind zu hoch, um die Merkmale einer neuen Messung online mit allen vorherigen zu vergleichen. Daher bedarf es einer effizienten Strategie zur Auswahl der Vergleichskandidaten. Henry et al. [10] vergleichen neue Aufnahmen deshalb nur zu einer kleinen Auswahl sogenannter „Keyframes“, die rekursiv dadurch definiert sind, dass sie über die Merkmalsassoziation nicht mehr zum vorherigen Keyframe lokalisiert werden können. Bei sich bewegenden Sensoren steigt die Anzahl der Vergleiche dadurch nicht mehr proportional zur Laufzeit, sondern abhängig von der Bewegung des Kamerasichtfelds. Diese Methode stößt bei Sequenzen mit viel Bewegung trotzdem schnell an ihre Grenzen und wurde daher durch eine auf Sampling basierende Methode ersetzt. Die hohe Datenrate der Sensoren gewährleistet eine hohe Übereinstimmung der aufgenommenen Szene, wodurch die Transformationsschätzung im Regelfall zu einer variierenden Anzahl vorheriger Aufnahmen möglich ist. Mehr paarweise Schätzungen erhöhen dabei die Robustheit gegenüber Fehlern in den einzelnen Transformationsschätzungen, insbesondere bei Bildern schlechter Qualität (z.B. durch Beleuchtungseffekte oder Bewegungsunschärfe) oder zeitlichem Versatz von Tiefen- und Farbbild. Wie im Abschnitt 3.2 erläutert, ist zur Minimierung des sich akkumulierenden Fehlers insbesondere interessant, Transformationen zu möglichst frühen Zeitpunkten zu finden. Durch die stetige Degeneration der Positionsschätzung können solche Loop Closures allerdings nicht anhand dieser Schätzung vorhergesagt werden. Daher werden zusätzlich zu den unmittelbaren Vorgängern n zufällig ausgewählte Kandidaten aus den Keyframes zum Vergleich herangezogen. Um den korrigierenden Effekt gefundener Loop Closures weitgehend auszunutzen, werden die unmittelbaren Vorgänger nicht nur auf der zeitlichen Sequenz ermittelt, sondern zusätzlich auf der Graphdistanz. Dazu wird auf dem Posengraph ein minimaler Spannbaum T mit begrenzter Tiefe berechnet, aus dem eine vorgegebene Anzahl k zu vergleichender Vorgänger gezogen wird. Abbildung 4 unten zeigt den mit $n = 5$, $k = 5$ und $T = 3$ ermittelten Posengraph für die Sequenz „FR1 Desk2“. Rote Kanten bezeichnen Transformationsschätzungen zwischen direkt aufeinanderfolgende Posen. Kanten zwischen Posen, die zeitlich länger als eine Sekunde auseinander liegen, sind blau gefärbt. Für den Posengraph in Abbildung 4 oben wurden die fünf direkten Vorgänger, sowie wieder $n = 5$ zufällig gezogene Vorgänger als Kandidaten für eine Transformationsschätzung benutzt. Die dichtere Netzstruktur blauer Kanten in Abbildung 4 unten bezeugt die Fähigkeit des vorgeschlagenen Algorithmus, das Wissen über gefundene Loop Closures im weiteren Verlauf auszunutzen.

3.4 Repräsentation der Karte

Die Berechnung einer konsistenten Trajektorie erlaubt die Registrierung der an den einzelnen Zeitpunkten aufgenommenen Punktwolken. Für Anwendungen wie die Kollisionsvermeidung ist die Integration aller Punkte in ein gemeinsames Koordinatensystem allerdings nur beschränkt nützlich, da z.B. die Information über die Belegung des Raums und insbesondere auch über die Belegungsfreiheit nur implizit über die Sensorposen aller Punkte verfügbar ist. Daher muss eine Repräsentation gewählt werden, die diese Informationen effizient zur Verfügung stellt. Hochaufgelöste probabilistische Rasterkarten wie sie für zweidimensionales SLAM benutzt werden können aufgrund der Speicherbeschränkungen moderner Rechner nicht direkt ins Dreidimensionale übertragen werden. Eine speichereffiziente Datenstruktur für dreidimensionale Rasterkarten sind die OctoMaps [12]. OctoMaps unterteilen den Raum rekursiv in

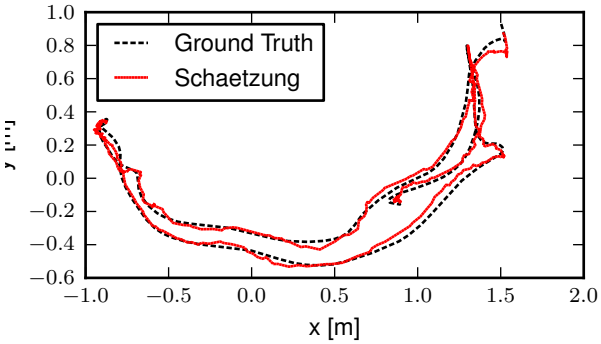

Bild 5: Die Rekonstruktion der Sensortrajektorie von Sequenz „FR1 Desk2“ im Vergleich zur Ground Truth des Bewegungserfassungssystems.

Würfel (Voxel), wodurch eine Baumstruktur entsteht, deren Blätter die Belegungswahrscheinlichkeit des Volumens enthalten. Da die weitere Unterteilung eines Voxels nur notwendig ist, wenn die Belegungswahrscheinlichkeit innerhalb eines Voxels variiert, können große Teile des unbekannten und des belegungsfreien Raums zusammengefasst werden. Die Integration der Messpunkte in den Voxeln reduziert die Redundanz der Daten und macht die Umgebungsrepräsentation robust gegen einzelne fehlerhaft rekonstruierte Sensorposen. Dazu wird die Belegungswahrscheinlichkeit über ihren Logit (oft auch log-odds genannt) gespeichert und kann bei einer neuen Messung z mittels

$$\text{Logit}(p(v_i)) := \text{Logit}(p(v \mid z)) + \text{Logit}(p(v_i)) \qquad (6)$$

aktualisiert werden. $p(v_i)$ steht hier für die Wahrscheinlichkeit, dass Voxel i belegt ist, $p(v \mid z)$ für die Belegungswahrscheinlichkeit eines Voxels gegeben die Messung z. Wichtig für die Zusammenfassung von freien Voxeln bei gleicher Belegungswahrscheinlichkeit und die Aktualisierungsgeschwindigkeit bei dynamischen Änderungen in der Umgebung ist es, die Belegungswahrscheinlichkeiten mit Schwellwerten für minimale und maximale Wahrscheinlichkeit zu begrenzen. Abbildung 2b zeigt eine hochauflösende OctoMap einer Büroumgebung, die mit dem beschriebenen System erstellt wurde. Zur besseren Anschauung ist hier die Farbinformation für Voxel mit Belegtheitswahrscheinlichkeit über 0,5 dargestellt. Die in den Voxeln enthaltenen Informationen kann je nach Anwendung angepasst werden.

4 Experimente

4.1 Daten und Fehlermaß

Zur experimentellen Auswertung des vorgestellten Systems wird der öffentlich verfügbare RGB-D-Benchmark-Datensatz der TU München verwendet [26]. Dieser besteht aus 27 Testsequenzen, die unterschiedliche Szenarien abdecken. Jede Sequenz beinhaltet hochgenaue Positionsmessungen eines externen Bewegungserfassungssystems, welche im Folgenden als Grundlage zur Ermittlung des Positionierungsfehlers benutzt werden. Die Auswertung erfolgt auf den neun „Freiburg 1“-Sequenzen, in denen die Kamera frei von Hand in einer typischen Büroumgebung bewegt wird (siehe auch Tabelle 1). Anschließend wird die Diskrepanz zwischen geschätzter Trajektorie $\mathbf{x}_{1:n}$ und wahrer Trajektorie $\mathbf{x}^*_{1:n}$ berechnet und der mittlerer Fehler bestimmt. Zur Berechnung dieses Fehlers sind grundsätzlich verschiedene Fehlermaße möglich [26]. Da in unserem Anwendungsfall eine möglichst akkurate dreidimensionale Karte der Umgebung erzeugt werden soll, ist für uns insbesondere der absolute Trajektorienfehler relevant. Hierzu werden zunächst die beiden Trajektorien durch eine Starrkörpertransformation $S \in \text{SE}(3)$ ausgerichtet, da der Startpunkt beider Trajektorien beliebig gewählt sein kann. Für die Sensorpose zum Zeitpunkt i ergibt sich der translatorische Fehler als

$$e_i := \left\| translation\left((\mathbf{x}^*_i)^{-1} S \mathbf{x}_i\right) \right\|. \qquad (7)$$

Anschließend kann aus diesen Einzelfehlern der Root-Mean-Squared-Error (RMSE) über die gesamte Trajektorie bestimmt werden:

$$\text{RMSE}(e_{1:n}) := \left(\frac{1}{n} \sum_{i=1}^{n} e_i^2 \right)^{1/2}. \qquad (8)$$

Die zur Ausrichtung verwendete Starrkörpertransformation wird so gewählt, dass der absolute Trajektorienfehler $\text{RMSE}(e_{1:n})$ minimal wird, d.h.

$$S := \underset{S}{\text{argmin}}\, \text{RMSE}(e_{1:n}), \qquad (9)$$

was sich mit der Methode von Horn [11] in geschlossener Form bestimmen lässt. Abbildung 5 zeigt die so an der Ground Truth ausgerichtete Schätzung der Trajektorie der Sequenz „FR1 Desk2“. Alternativ zum absoluten Trajektorienfehler kann auch der relative Posenfehler auf einer geeigneten Menge an Vergleichspaaren evaluiert werden [17]. Dies ist insbesondere für die Bestimmung der Abweichung (pro Frame) interessant, kann aber auch zur Bestimmung des globalen Positionsfehlers verwendet werden, indem z.B. alle verfügbaren Posenpaare betrachtet werden.

Neben dem Fehler in der Trajektorienschätzung wird außerdem die Laufzeit des Systems gemessen. Die experimentelle Evaluierung für die einzelnen Sequenzen wird in Abschnitt 4.2 vorgestellt. Da die Transformationsschätzung des Frontends stark von den benutzen Merkmalsdetektoren und -beschreibungen abhängt, werden die Ergebnisse danach in Kapitel 4.3 in Abhängigkeit des gewählten Merkmalstyp betrachtet.

4.2 Genauigkeit in der Rekonstruktion der Sensortrajektorie

Für eine detaillierte Analyse des vorgestellten Systems werden für jede der neun „Freiburg 1“-(FR1-)Sequenzen

Tabelle 1: Der Fehler für die „FR1“ Datensätze beträgt durchschnittlich 5,1 cm bei einer durchschnittlichen Bearbeitungsdauer 0,39 s pro RGB-D-Aufnahme.

Sequenz	Länge der Trajektorie	∅ Rotations-geschw.	∅ Transl.-Geschw.	Bilder	Gesamt-Laufzeit	**g²o** Laufzeit	Transl. RMSE	OctoMap Größe
FR1 360	5,82 m	41,60 deg/s	0,21 m/s	745	217 s	0,34 s	0,079 m	22 MB
FR1 Desk	9,26 m	23,33 deg/s	0,41 m/s	575	220 s	0,40 s	0,023 m	6,0 MB
FR1 Desk2	10,16 m	29,31 deg/s	0,43 m/s	614	220 s	0,31 s	0,043 m	8,1 MB
FR1 Floor	12,57 m	15,07 deg/s	0,26 m/s	1214	482 s	1,31 s	0,022 m	4,2 MB
FR1 Plant	14,80 m	27,89 deg/s	0,37 m/s	1112	442 s	0,74 s	0,091 m	22 MB
FR1 Room	15,99 m	29,88 deg/s	0,33 m/s	1332	526 s	0,80 s	0,084 m	21 MB
FR1 RPY	1,66 m	50,15 deg/s	0,06 m/s	687	343 s	1,82 s	0,026 m	11 MB
FR1 Teddy	15,71 m	21,32 deg/s	0,32 m/s	1395	540 s	0,78 s	0,076 m	25 MB
FR1 XYZ	7,11 m	8,92 deg/s	0,24 m/s	788	379 s	11,5 s	0,014 m	3,6 MB

die Abweichung zwischen der geschätzten Trajektorie und der mit dem Bewegungserfassungssystem ermittelten Trajektorie berechnet. Dabei werden für jede Sequenz alle RGB-D-Bilder genutzt, um den jeweiligen Posengraphen zu erstellen. Dieser wird abschließend iterativ optimiert, bis das Konvergenzkriterium erfüllt ist.

Um die Ergebnisse einfacher und einheitlicher zwischen verschiedenen Methoden vergleichen zu können, wird für jede Aufnahme der Sequenzen eine Positionsschätzung angegeben. Hierdurch wird die Genauigkeit der Rekonstruktion unabhängig von der Leistungsfähigkeit der verwendeten Hardware. Wird keine verlässliche Schätzung der Transformation gefunden, wie z.B. durch die variierende Bewegungsunschärfe bei starken Beschleunigungen hervorgerufen, wird die Position gleich der vorhergehenden Position gesetzt und, assoziiert mit einer hohen Unsicherheit, an das SLAM-Backend gegeben. Experimente mit einem konstanten Bewegungsmodell lieferten hier keine robusten Ergebnisse, da durch Ungenauigkeiten in der Bewegungsschätzung vor und nach Schätzungsausfällen keine verlässliche Vorhersage getroffen werden kann. Beim Einsatz auf einem autonomen Roboter ist es jedoch sinnvoller, die Trajektorie (und damit die Karte) in separate Segmente zu unterteilen und bis zu einer Verschmelzung über Loop Closures nur das aktuelle Segment zu nutzen.

Die Auswertung ist in Tabelle 1 zusammengefasst. Die Transformationen wurden hier mit der RANSAC-Methode basierend auf den Assoziationen von SIFT-Merkmalen berechnet. Die Laufzeit wird von der Suche nach Merkmalsassoziationen dominiert. Bei den auf der CPU berechneten SURF ist auch die Detektion und Extraktion der Merkmale relevant. Durch die Nebenläufigkeit dieser Berechnungen zur Assoziationssuche fällt der Geschwindigkeitsunterschied aber nur geringfügig ins Gewicht. Um das Verhältnis zwischen Laufzeit und Qualität der Schätzungen stabil zu halten, wurde daher die Anzahl der Merkmale für die Assoziationssuche auf maximal 1000 beschränkt. Für SURF wurde zusätzlich der Schwellwert für den Detektor während der Laufzeit adaptiert, um eine Extraktion von minimal 600 bis maximal 1000 Merkmalen pro Frame zu erreichen. Eine Erhöhung dieser Grenzwerte auf 1500 und 2000 verringert den durchschnittlichen RMSE um ca. 5 mm, die durchschnittliche Laufzeit pro Bild erhöht sich dadurch jedoch im Gegenzug um ca 0,2 s pro Bild.

Die Optimierung wurde pro Sequenz nach dem Aufbau des Posengraphen ausgeführt. Die hohe Laufzeit des Graph-Optimierers für die Sequenz „FR1 XYZ“ erklärt sich durch die geringen Änderungen im Sichtbereich, wodurch der vom Backend zu optimierende Graph sehr viele Kanten enthält. Ist diese Eigenschaft in der Anwendung vorhersehbar, z.B. bei beschränktem Arbeitsbereich, kann die Anzahl der Loop-Closure-Kandidaten und die Wahl des Optimierungs-Backend entsprechend angepasst werden um die Laufzeit des SLAM-Backends zu minimieren.

4.3 Vergleich der Methoden zur Transformationsschätzung

Die Qualität der Trajektorienrekonstruktion hängt in hohem Maße von der benutzen Methode zur Schätzung der Transformation ab. Tabelle 2 listet den Durchschnitt und die Standardabweichung des ermittelten Fehlers für die „FR1“-Sequenzen, gruppiert nach Schätzungsmethode. Die Experimente mit GICP ergaben für viele Szenarien keine robusten Schätzungen innerhalb akzeptabler Laufzeiten. Für die Werte in Tabelle 2 wurde GICP zum Schätzen von Transformation zu den jeweils vorherigen fünf Punktwolken ermittelt. Die Punktwolken wurden hierbei durch Sampling von ca. 300 000 auf 10 000 bzw. 1 000 Punkte geschrumpft und die Anfangshypothese auf die vorherige Position gesetzt. Henry et. al. [10] nutzen jede merkmalsbasierte Transformationsschätzung als Anfangshypothese für GICP. In Experimenten mit dem vorgestellten System ergaben sich dadurch jedoch keine signifikanten Verbesserungen in der Trajektorienrekonstruktion. Die angegebenen Laufzeiten wurden auf einer Quad-Core-CPU mit 8 GB Arbeitsspeicher gemessen. Durch Parallelisierung der Verarbeitung mit CPU-Threads wird dabei ein Beschleunigungsfaktor von 2,5 bis 3 erreicht. Die Optimierung wurde jeweils einmal nach dem Aufbau des

Tabelle 2: Auswertung der Schätzgenauigkeit und Laufzeit, gruppiert nach Methode (siehe Abschnitt 3.3)

Methode	RMSE	∅ Laufzeit pro Bild
GICP (10000 Pkt.)	0,204 m ± 0,136 m	0,66 s
GICP (1000 Pkt.)	0,293 m ± 0,169 m	0,13 s
SIFT	0,051 m ± 0,031 m	0,39 s
SURF	0,058 m ± 0,035 m	0,45 s

Posengraphen ausgeführt und in der Berechnung der durchschnittlichen Laufzeit pro Bild einbezogen.

4.4 Datenvolumen der Kartenrepräsentation

Um die Daten bestmöglich für die Navigation nutzen zu können fusionieren wir die Punktwolken in OctoMaps wie in Abschnitt 3.4 beschrieben. Neben der expliziten Verfügbarkeit der Frei- und Belegtheitswahrscheinlichkeiten wird das Datenvolumen dabei stark reduziert. Die Punktwolke einer RGB-D-Aufnahme (unter Ausschluss von Messungen ohne Tiefenwert) belegt durchschnittlich 3,6 Megabyte Speicher. Kumulierte Umgebungsrepräsentationen aus Punktwolken benötigen für die in diesem Beitrag ausgewerteten Sequenzen somit zwischen zwei und fünf Gigabyte. Der Speicherbedarf entsprechender OctoMaps mit einer Auflösung von 2 cm ist in Tabelle 1 angegeben und liegt bei maximal 25 Megabyte. Abbildung 2b zeigt eine solche OctoMap. Je nach Anwendung können die OctoMaps durch Weglassen der Farbinformation und Beschränkung auf binäre Werte für „frei“ und „belegt“ weiter reduziert werden. Für die obigen OctoMaps ergeben sich bei gleichbleibender Auflösung so Größen zwischen 97 KB und 528 KB.

5 Zusammenfassung und Ausblick

Durch neuartige RGB-D-Sensoren eröffnen sich der mobilen Robotik neue Möglichkeiten. Das in diesem Artikel vorgestellte SLAM-System nutzt die Eigenschaften dieser Sensorik für die effiziente Erstellung von 3D-Karten, ohne Einschränkungen für die Bewegung des Sensors. Die Unabhängigkeit von weiterer Sensorik und die kostengünstige Verfügbarkeit macht RGB-D-SLAM zu einer geeigneten Erweiterung für eine Vielzahl von bestehenden Systemen. Insbesondere im Kontext bildbasierter Steuerung und Regelung erweitert der vorgestellte Ansatz bestehende Anwendungen mit geringem Hardwareaufwand um eine konsistente Karte der Umgebung, die z.B zur Kollisionserkennung und/oder -vermeidung eingesetzt werden kann. Dadurch können z.B. bei Seitwärtsbewegungen auch Kollisionen mit Hindernissen vermieden werden, die nicht mehr im aktuellen Sichtfeld sind. Experimente mit öffentlich verfügbaren Datensätzen zeigen, dass die Genauigkeit der rekonstruierten Sensortrajektorie im Bereich weniger Zentimeter liegt, bei Laufzeiten, die dem Onlinebetrieb genügen. Das vorgestellte System selbst sowie die Auswertungsmethoden und -daten sind als Open Source frei zugänglich und dokumentiert. Damit wird eine Grundlage für den Vergleich nachfolgender Ansätze anderer Forschungseinrichtungen geboten.

Weiterführende Aufgabenstellungen sind die Integration des Systems für die Kollisionsvermeidung in Visual-Servoing-Systeme sowie die Kalibrierung verschiedener Sensoren eines Systems über deren jeweilige Bewegungschätzungen.

Literatur

[1] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (SURF). *Comput. Vis. Image Underst.*, 110:346–359, 2008.

[2] P. J. Besl and H. D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 14(2):239–256, 1992.

[3] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, 2008.

[4] A.J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2003.

[5] F. Dellaert. Square Root SAM. In *Proc. of Robotics: Science and Systems (RSS)*, pages 177–184, Cambridge, MA, USA, 2005.

[6] N. Fioraio and K. Konolige. Realtime visual and point cloud slam. In *Proc. of the RGB-D Workshop on Advanced Reasoning with Depth Cameras at Robotics: Science and Systems Conf. (RSS)*, Los Angeles, USA, 2011.

[7] M.A. Fischler and R.C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[8] U. Frese, P. Larsson, and T. Duckett. A multilevel relaxation algorithm for simultaneous localisation and mapping. *IEEE Trans. on Robotics*, 21(2):1–12, 2005.

[9] G. Grisetti, S. Grzonka, C. Stachniss, P. Pfaff, and W. Burgard. Efficient estimation of accurate maximum likelihood maps in 3d. In *Proc. of the Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

[10] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In *Proc. of the Intl. Symp. on Experimental Robotics (ISER)*, Delhi, India, 2010.

[11] B.K.P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987.

[12] A. Hornung, K.M. Wurm, and M. Bennewitz. Humanoid robot localization in complex indoor environments. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, 2010.

[13] H. Jin, P. Favaro, and S. Soatto. Real-time 3-d motion and structure of point-features: A front-end for vision-based control and interaction. In *Proc. of the IEEE Intl. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2000.

[14] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Trans. on Robotics*, 24(6):1365–1378, 2008.

[15] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. IEEE and ACM*

Intl. Symp. on Mixed and Augmented Reality (ISMAR), Nara, Japan, 2007.

[16] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Shanghai, China, 2011.

[17] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner. On measuring the accuracy of SLAM algorithms. *Autonomous Robots*, 27:387–407, 2009.

[18] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[19] D. Nister. Preemptive RANSAC for live structure and motion estimation. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2003.

[20] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6D SLAM with approximate data association. In *Proc. of the 12th Intl. Conference on Advanced Robotics (ICAR)*, pages 242–249, 2005.

[21] E. Olson, J. Leonard, and S. Teller. Fast iterative optimization of pose graphs with poor initial estimates. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 2262–2269, 2006.

[22] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: an efficient alternative to SIFT or SURF. 13, 2011.

[23] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Proc. of the Intl. Conf. on 3-D Digital Imaging and Modeling*, Quebec, Canada, 2001.

[24] A. Segal, D. Haehnel, and S. Thrun. Generalized-ICP. In *Proceedings of Robotics: Science and Systems*, Seattle, USA, 2009.

[25] H. Strasdat, J. M. M. Montiel, and A. Davison. Scale drift-aware large scale monocular slam. In *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain, 2010.

[26] J. Sturm, S. Magnenat, N. Engelhard, F. Pomerleau, F. Colas, W. Burgard, D. Cremers, and R. Siegwart. Towards a benchmark for RGB-D SLAM evaluation. In *Proc. of the RGB-D Workshop on Advanced Reasoning with Depth Cameras at Robotics: Science and Systems Conf. (RSS)*, Los Angeles, USA, 2011.

[27] S. Thrun. Robotic mapping: A survey. In *Exploring Artificial Intelligence in the New Millennium*. 2003.

[28] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[29] Changchang Wu. SiftGPU: A GPU implementation of scale invariant feature transform (SIFT). `http://cs.unc.edu/~ccwu/siftgpu`, 2007.

[30] K.M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In *Proc. of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, Anchorage, USA, 2010.

Manuskripteingang: 20. Dezember 2011.

M.Sc. Felix Endres promoviert am Lehrstuhl Autonome Intelligente Systeme von Prof. Wolfram Burgard an der Albert-Ludwigs-Universität Freiburg. Sein besonderes Interesse gilt der dreidimensionalen Wahrnehmung mobiler Roboter im Rahmen von Manipulationsaufgaben.

Adresse: Universität Freiburg, Institut für Informatik, Autonome Intelligente Systeme, Georges-Koehler-Allee 79, D-79110 Freiburg, E-Mail: endres@informatik.uni-freiburg.de

M.Sc. Jürgen Hess promoviert am Lehrstuhl Autonome Intelligente Systeme an der Albert-Ludwigs-Universität Freiburg. Sein Forschunginteresse gilt der Lösung mobiler Manipulationsaufgaben für Sevicerobotер mittels Methoden des Reinforcement Lernens.

Adresse: Universität Freiburg, Institut für Informatik, Autonome Intelligente Systeme, Georges-Koehler-Allee 79, D-79110 Freiburg, E-Mail: hess@informatik.uni-freiburg.de

B.Sc. Nikolas Engelhard studiert an der Albert-Ludwigs-Universität in Freiburg im Masterstudiengang Informatik. Sein Forschungsschwerpunkt liegt in der kamerabasierten Umgebungsmodellierung.

Adresse: Universität Freiburg, Institut für Informatik, Autonome Intelligente Systeme, Georges-Koehler-Allee 79, D-79110 Freiburg, E-Mail:engelhar@informatik.uni-freiburg.de

Dr. Jürgen Sturm ist Wissenschaftlicher Assistent am Lehrstuhl für Bildverarbeitung und Mustererkennung von Prof. Daniel Cremers an der TU München. Sein Forschungsschwerpunkt liegt im Bereich kamerabasierte Lokalisation und Rekonstruktion für autonome Flugsysteme. Zuvor hat er 2011 am Lehrstuhl Autonome Intelligente Systeme von Prof. Wolfram Burgard im Bereich mobile Manipulation promoviert. In seiner Doktorarbeit hat er eine Reihe neuer Methoden entwickelt, die es Robotern ermöglichen, kinematische Modelle der Objekte in ihrer Umgebung zu lernen und somit flexibler eingesetzt werden können.

Adresse: Technische Universität München, Fakultät für Informatik, Boltzmannstraße 3, 85748 Garching.
E-Mail: juergen.sturm@in.tum.de

Prof. Dr. Wolfram Burgard leitet den Lehrstuhl Autonome Intelligente Systeme am Institut für Informatik der Albert-Ludwigs-Universität Freiburg. Seine Interessensgebiet umfasst autonome mobile Roboter und künstlichen Intelligenz.
Seine Forschungsschwerpunkte liegen in der Entwicklung robuster und anpassungsfähiger Techniken für die Zustandsschätzung und die Steurung mobiler Roboter. In diesem Bereich entwickelte er mit seiner Arbeitsgruppe eine Serie innovativer probabilistischer Techniken u.a. zur Lokalisierung, Kartographierung, SLAM, Exploration und Pfadplanung.

Adresse: Universität Freiburg, Institut für Informatik, Autonome Intelligente Systeme, Georges-Koehler-Allee 79, D-79110 Freiburg, E-Mail: burgard@informatik.uni-freiburg.de