



UNIVERSITÄT
BAYREUTH

Inkrementelle Rekonstruktion von planaren Volumenmodellen mit handgehaltenen Tiefenkameras

Von der Universität Bayreuth
zur Erlangung des Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat)
genehmigte Abhandlung

von
Maximilian Sand
aus Weiden i.d. Oberpfalz

1. Gutachter: Prof. Dr. Dominik Henrich
2. Gutachter: Prof. Dr. Michael Guthe
3. Gutachter: Prof. Dr. Hans Hagen

Tag der Einreichung: 26.07.2018
Tag des Kolloquiums: 06.02.2019

Danksagung

Auf das Gelingen dieser Arbeit hatten einige Personen einen nicht unwesentlichen Einfluss, denen ich an dieser Stelle von ganzen Herzen danken will. Allen voran meinem Doktorvater Prof. Dr. Dominik Henrich für die Möglichkeit der Promotion an seinem Lehrstuhl sowie die vielen konstruktiven Diskussionen, die einem den Blick auf das "große Ganze" nicht vergessen ließen und oft Quelle von Inspirationen für neue Ideen waren.

Ebenfalls zu großem Dank verpflichtet bin ich meinen Kollegen am Lehrstuhl, insbesondere Johannes Baumgartl, Michael Gradmann, Christian Groth, Eric Orendt, Dominik Riedelbauch, Michael Riedl, Dorian Rohner, Michael Spangenberg, Tobias Werner und Kim Wölfel, die durch viele aufschlussreiche Gespräche zur Weiterverfolgung oder Eliminierung eines Konzeptes beitrugen.

Ein großer Dank geht zudem auch an die Studenten, deren Arbeiten ich während meiner Zeit als wissenschaftlicher Mitarbeiter betreuen durfte. Besonders hervorzuheben sind dabei Johannes Hartwig, Johannes Jakob und Lukas Schwarz. Nicht selten führten die Diskussionen zu neuen Blickwinkeln und Denkweisen, die die Arbeit verbesserten.

Ebenso dankbar bin ich für die Unterstützung durch meine Familie und Freunde, die durch Korrekturlesen, Motivieren oder andere kleine und große Dinge das Weiterkommen während der Promotion sehr erleichterten.

Abstract

Incremental Reconstruction of Planar Solid Models with Hand-held Depth Cameras

3D models are an essential part in many domains like computer graphics, robotics or computer aided design. Hereby, the sensor-based creation of 3D models provides an easy method to generate a digital copy of an object or scene by using a camera. Compared to a manual creation, no specific knowledge in 3D graphic programs is necessary. To reduce the amount of manual intervention, the ultimate vision is to build an intelligent 3D scanner that provides all required information while being intuitive and easy to handle.

The goal of this work is to enhance the state of the art towards this vision. To achieve this, an approach for an online and incremental reconstruction of planar solid models in terms of a boundary representation model (B-Rep) using a hand-held depth camera is presented. This combination closes the gap that is inherent to current methods: A robust, online-capable acquisition and a simultaneous reconstruction of a model that is rich in geometric and topological information which is the basis for CAD models. The incremental manner facilitates the availability of such models at every point in time during acquisition.

To meet these challenges, multiple subproblems are identified and solved: The concept of partial B-Rep models overcomes the question of a suitable representation for such a system. A procedure for a direct reconstruction and a subsequent fusion of B-Rep models enables an online-capable reconstruction with known camera poses. A method for feature-based registration of single frames extends the system to a *complete simultaneous localization and mapping* (SLAM) framework that can be used with hand-held depth cameras. Approaches for supporting the user simplify the handling and help to create complete models without holes.

An evaluation of each step shows the capabilities and limitations of the presented methods. Additionally, a prototypical implementation for three different sensors and two platforms (desktop computer and mobile device) is described that shows the practical suitability.

Zusammenfassung

Inkrementelle Rekonstruktion von planaren Volumenmodellen mit handgehaltenen Tiefenkameras

In vielen Anwendungsbereichen, wie der Computergrafik, der Robotik oder beim computer-gestützten Entwerfen spielen 3D-Modelle eine große Rolle. Die sensorgestützte Erzeugung von 3D-Modellen ist dabei eine einfache Möglichkeit, Modelle von realen Objekten oder Räumen zu erstellen, indem mit einer Kamera ein digitales Abbild erstellt wird. Im Gegensatz zur manuellen Erzeugung ist dabei kein Expertenwissen im Umgang mit 3D-Grafikprogrammen vonnöten. Um einen manuellen Eingriff so weit wie möglich zu reduzieren, ist die Vision, einen allumfassenden intelligenten 3D-Scanner zu besitzen, der alle benötigten Informationen schnell und in einfacher Art und Weise erfasst.

Diese Arbeit hat zum Ziel, dieser Vision einen Schritt näher zu kommen. Dazu wird ein Verfahren entwickelt, das erlaubt, mit handgehaltenen Tiefenkameras inkrementell und online ein kontinuierliches Volumenmodell in Form eines planaren Boundary-Representation (B-Rep) Modells zu erzeugen. Diese Kombination schließt die in bestehenden Scan-Systemen vorherrschende Lücke von robuster, online-fähiger Erfassung bei gleichzeitiger Erzeugung eines hochwertigen Ausgabemodells mit algebraischer Geometriebeschreibung und Topologiefinformationen, das Grundlage vieler CAD-Systeme ist. Durch die inkrementelle Vorgehensweise ist bereits zu jedem Zeitpunkt während der Rekonstruktion ein hochwertiges partielles Modell verfügbar.

Zur Bewältigung dieser Aufgabe werden mehrere Teilprobleme identifiziert und Lösungen vorgestellt: Das Konzept partieller B-Rep Modelle beantwortet die Frage nach einer geeigneten Modellrepräsentation für eines solches System. Ein Verfahren zur direkten Rekonstruktion und anschließenden Fusion ermöglicht eine online-fähige Rekonstruktion bei bekannter Kamerapose. Eine Methode zur merkmalsbasierten Registrierung von Einzelbildern vervollständigt den Ansatz zu einem kompletten *Simultaneous-Localization-And-Mapping* (SLAM) System, sodass frei bewegliche, handgehaltene Tiefenkameras verwendet werden können. Ansätze zur Unterstützung des Anwenders vereinfachen die Handhabung und helfen, ein Objekt oder eine Szene vollständig zu erfassen.

Anhand einer Evaluation aller Teilschritte werden die Leistungsfähigkeit und die Grenzen des Systems analysiert und diskutiert. Zudem wird eine prototypische Umsetzung für drei unterschiedliche Sensoren und zwei Plattformen (Desktop-Rechner und Mobilgerät) beschrieben, die die Praxistauglichkeit des Ansatzes aufzeigt.

Inhaltsverzeichnis

1	Einleitung	10
1.1	Motivation: Einsatzgebiete von 3D-Scan-Systemen	11
1.2	Eigenschaften von 3D-Scan-Systemen	12
1.3	Vision: Der intelligente 3D-Scanner	18
1.4	Ziele und Beiträge dieser Arbeit	19
1.5	Kapitelübersicht	21
2	Stand der Forschung	22
2.1	Simultaneous Localization And Mapping	23
2.2	Digital Shape Reconstruction	30
2.3	Schlussfolgerungen	35
3	Grundkonzept	38
3.1	Kombination von SLAM und DSR	39
3.2	Partielle Modelle	42
3.3	Parametrisierung	53
3.4	Zusammenfassung	57
4	Rekonstruktion partieller B-Reps	58
4.1	Stand der Forschung	59
4.2	Vorgehensweise	62
4.3	Evaluation	85
4.4	Zusammenfassung	94
5	Fusion partieller B-Reps	96
5.1	Stand der Forschung	97
5.2	Vorgehensweise	99
5.3	Evaluation	108
5.4	Zusammenfassung	115
6	Registrierung partieller B-Reps	118
6.1	Stand der Forschung	119
6.2	Vorgehensweise	121
6.3	Evaluation	135
6.4	Zusammenfassung	143

7	Vervollständigung von partiellen B-Reps und Nutzerrückmeldungen	144
7.1	Stand der Forschung	145
7.2	Übersicht	146
7.3	Bestimmung von Löchern und deren Eigenschaften	154
7.4	Schließen von inneren Löchern	161
7.5	Erzeugung von Nutzerhinweisen	162
7.6	Validierung	166
7.7	Zusammenfassung	173
8	Gesamtsystem	174
8.1	Prototypische Umsetzung	175
8.2	Anwendungen	181
8.3	Zusammenfassung	182
9	Fazit	184
9.1	Zusammenfassung	184
9.2	Ausblick	187
	Verzeichnisse	190
	Abbildungsverzeichnis	190
	Tabellenverzeichnis	193
	Quellenverzeichnis	194
	Eigene Publikationen	207

Symbolverzeichnis

\vec{X}, \vec{x}	Vektor , Ortsvektoren werden in Großbuchstaben, Richtungsvektoren in Kleinbuchstaben notiert
\bar{X}	Vektor, der ein Mittel beschreibt, beispielsweise der Mittelwert einzelner Ortsvektoren oder der Flächenschwerpunkt eines Polygons
x	Skalar
\bar{x}	Skalar, das ein Mittel beschreibt, beispielsweise das arithmetische Mittel einer Menge an Werten
X	Matrix
${}^B T_A$	Transformationsmatrix von Koordinatensystem A nach B
$\vec{x} \circ \vec{y}$	Skalarprodukt zweier Vektoren
$\vec{x} \times \vec{y}$	Kreuzprodukt zweier Vektoren
$X \cdot Y, XY$	Matrixprodukt zweier Matrizen

Abkürzungsverzeichnis

B-Rep	Boundary-Representation (Model)
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAM	Computer Aided Manufacturing
CAQ	Computer Aided Quality Control
CAX	Sammelbegriff für CAD/CAM/CAE/CAQ
CSG	Constructive Solid Geometry
DSR	Digital Shape Reconstruction
GMA	Gaussian Mixture Alignment
GPU	Graphics Processing Unit (Grafikkarte)
ICP	Iterative Closest Points
NURBS	Non Uniform Rational B-Spline
RANSAC	Random Sample Consensus
SLAM	Simultaneous Localization And Mapping

Kapitel 1

Einleitung

Inhalt

1.1	Motivation: Einsatzgebiete von 3D-Scan-Systemen	11
1.2	Eigenschaften von 3D-Scan-Systemen	12
1.2.1	Ausgabe	13
1.2.2	Verarbeitung	14
1.2.3	Erfassung	15
1.3	Vision: Der intelligente 3D-Scanner	18
1.4	Ziele und Beiträge dieser Arbeit	19
1.4.1	Aufgabenstellung	20
1.4.2	Wissenschaftliche Fragestellungen	20
1.5	Kapitelübersicht	21

Als Mensch leben wir in drei Raumdimensionen und nehmen diese auch als solche wahr. Diese bekannte Erfahrung überträgt die Computergrafik auf digitale Systeme, indem sie eine virtuelle Welt auf den zweidimensionalen Bildschirm projiziert. Grundbausteine sind dabei geometrische Modelle, welche die Form eines dreidimensionalen Körpers beschreiben. Die digitale Repräsentation von dreidimensionalen Modellen und deren Erzeugung spielt dabei in vielen Anwendungsgebieten eine große Rolle. Die vorliegende Arbeit beschäftigt sich mit einem 3D-Scan-System, also einem System zur Erzeugung von 3D-Modellen durch Scannen realer Objekte oder Räume. Solche Systeme werden in unterschiedlichen Anwendungsbereichen benötigt (Abschnitt 1.1). Die Leistungsfähigkeit eines solchen Scanners kann eingeordnet werden, indem die Eigenschaften von 3D-Scan-Systemen betrachtet werden (Abschnitt 1.2). Anhand derer zeigt sich, wie in Zukunft ein allumfassender, intelligenter Scanner aussehen könnte (Abschnitt 1.3). Die vorliegende Arbeit hat als Zielsetzung, dieser Vision einen Schritt näher zu kommen. Dazu werden die Ziele und Beiträge dieser Arbeit formuliert (Abschnitt 1.4) und eine Übersicht über die Arbeit gegeben (Abschnitt 1.5).

1.1 Motivation: Einsatzgebiete von 3D-Scan-Systemen

3D-Modelle sind inzwischen allgegenwärtig. Sie stellen ein digitales Abbild realer Objekte oder Welten dar und werden zu unterschiedlichsten Zwecken erzeugt.

Die Unterhaltungsindustrie nutzt 3D-Modelle, um möglichst ansprechende Bilder zu erzeugen, die den Anwender in eine andere Welt versetzen sollen. Dies gilt sowohl für nicht-interaktive Medien, wie beispielsweise animierte Filme, als auch für interaktive Welten, wie sie in Computerspielen üblich sind.

Bei der computergestützten Produktentwicklung (*computer-aided engineering, CAE*) erstellen Entwickler ein geometrisches Modell, das später maschinell gefertigt wird. Die digitale Darstellung bietet dabei viele Vorteile, wie etwa einfache Modifikationen, automatisierte Simulationen, oder die computergestützte Abwicklung des gesamten Schöpfungsprozesses vom Entwurf (*computer-aided design, CAD*) über die Fertigung (*computer-aided manufacturing, CAM*) bis hin zur Qualitätskontrolle (*computer-aided quality control, CAQ*).

In der Robotik sind Modelle nötig, um einem Roboter bestimmte Fähigkeiten zu ermöglichen. Bahnplaner nutzen Umweltmodelle, um kollisionsfreie Bahnen zu berechnen. Virtuelle Modelle von realen Objekten werden benötigt, um diese passend greifen zu können. Werden geometrische Modelle um Zusatzwissen erweitert, so entstehen semantische Modelle, die beispielsweise für eine natürlichsprachliche Bedienung genutzt werden können.

Die Erzeugung von geometrischen Computermodellen kann dabei auf zwei Arten geschehen: virtuell oder sensorgestützt. Bei der virtuellen Erzeugung entsteht das Modell vollständig am Computer, unter Zuhilfenahme geeigneter Software-Werkzeuge. Bei der sensorgestützten Erzeugung werden reale Objekte mit einem Sensor erfasst und aus den Sensordaten ein Modell berechnet.

In den oben genannten Anwendungsgebieten kommen beide Fälle zum Einsatz. Zur virtuellen Erzeugung werden in der Unterhaltungsindustrie 3D-Grafikprogramme genutzt, um die Inhalte nach den Vorstellungen eines Grafikdesigners zu entwerfen und animieren. In der Produktentwicklung kommen CAD-Programme zum Einsatz, die es ermöglichen, parametrisierte und mit Bemessungen versehene Modelle zu generieren. In der Robotik wird die Umgebung eines stationären Roboters modelliert, um kollisionsfreie Bahnen zu planen.

Die sensorgestützte Erzeugung wird eingesetzt, um Abbilder von realen Personen (Avatare) zu erstellen, die dann beispielsweise in Computerspielen mitwirken. Bei der Produktentwicklung wird die sensorgestützte Erzeugung verwendet, um Modelle von existierenden, nicht digital verfügbaren Produkten zu erhalten (*reverse engineering*), die dann beispielsweise als Vorlage für Neuentwicklungen dienen können. Oft werden auch zuerst Tonmodelle erzeugt, die anschließend digitalisiert werden. In der Robotik dient die sensorgestützte Erzeugung dazu, Objekte handhaben zu können, deren Geometrie im Voraus unbekannt ist.

Die drei ausgewählten Anwendungsgebiete zeigen, dass geometrische Modelle und deren Erzeugung in vielen Anwendungsbereichen unverzichtbar sind. Die virtuelle Erzeugung fordert dabei vom Anwender Expertenkenntnisse in der verwendeten Modellierungssoftware und ein gutes dreidimensionales Vorstellungsvermögen. Ist ein real existierendes Objekt vorhanden, das digitalisiert werden soll, so bietet die sensorgestützte Erzeugung klare Vorteile: Der Anwender muss weder Fachwissen bezüglich Modellierung und Bedienung eines Werkzeugs besitzen, noch von Hand Geometrien vermessen. Dies ermöglicht eine zeit- und kostengünstigere Erstellung der Modelle.

Jedoch sind diese Vorteile nur von Nutzen, wenn ein System zur sensorgestützten Erzeugung von 3D-Modellen – auch 3D-Scanner genannt – kein ebenso hohes Fachwissen erfordert und hinreichend einfach zu bedienen ist. Zudem muss ein 3D-Scanner in der Lage sein, die für

die Anwendung nötigen Informationen auch erfassen zu können, beispielsweise Texturen für Avatare in Spielen oder Symmetrien im CAD-Bereich.

Zur Einordnung dieser Arbeit werden im folgenden Abschnitt zuerst die Eigenschaften eines 3D-Scan-Systems untersucht, um die Grenzen existierender Ansätze zu erfassen und offene Problemstellungen zu ermitteln.

1.2 Eigenschaften von 3D-Scan-Systemen

Im Bereich des Reverse Engineering fassen Várady et al. das Ziel der Forschung folgendermaßen zusammen: „*The ultimate goal of reverse engineering systems is to realize an intelligent 3D scanner*“ [Várady97]. In dieser Aussage, die auch auf alle anderen Anwendungsbereiche zutrifft, meint „intelligent“ die Fähigkeit, auch zusätzliche Informationen erkennen zu können, wie beispielsweise geometrische Eigenschaften oder Funktionsweisen eines Objekts. Weitere Ziele sind außerdem eine einfache Bedienbarkeit, ein günstiger Preis sowie eine möglichst große Unabhängigkeit von speziellen Sensoren. Um den Begriff eines intelligenten 3D-Scanners genauer untersuchen zu können, werden nun mögliche Ausprägungen von Eigenschaften eines solchen System im Detail beschrieben.

Die Eigenschaften eines 3D-Scan-Systems werden im Rahmen dieser Arbeit in drei Hauptkategorien aufgeteilt. Angelehnt an die in der Informatik übliche Unterteilung von Algorithmen in Eingabe, Verarbeitung und Ausgabe kann bei Scan-Systemen zwischen Erfassung, Verarbeitung und Ausgabe unterschieden werden. Die Ausgabe beschreibt dabei das resultierende 3D-Modell und dessen Eigenschaften. Unter Verarbeitung fallen die Eigenschaften der zugrunde liegenden Algorithmen. Im Bereich Erfassung sind alle Eigenschaften zusammengefasst, die den Sensor und die Aufnahmeweise des Systems betreffen (Abbildung 1.1).

Je nach Ausprägung einer Eigenschaft wird die Leistungsfähigkeit und damit auch die Komplexität des Gesamtsystems maßgeblich beeinflusst. Dazu werden im Folgenden alle Kategorien und deren Eigenschaften genauer untersucht, beginnend bei der Ausgabe, über die Verarbeitung hin zur Erfassung.

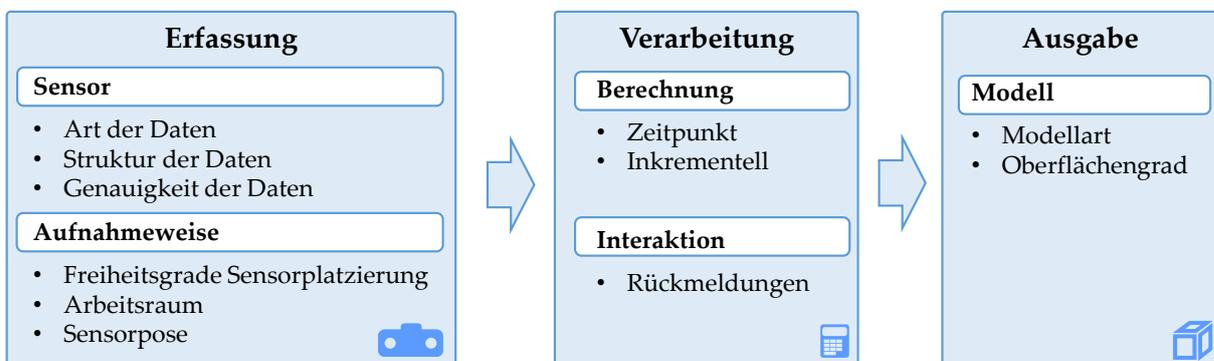


Abbildung 1.1: Die Eigenschaften eines 3D-Scanners lassen sich in drei Kategorien aufteilen: Erfassung, Verarbeitung und Ausgabe.

1.2.1 Ausgabe

Der Informationsgehalt des resultierenden Modells ist eine der maßgeblichen Eigenschaften eines 3D-Scan-Systems. Modelle können sich hinsichtlich ihrer Art sowie des Grades der verwendeten Oberflächen unterscheiden (Abbildung 1.2).

Modellart: Modelle können hinsichtlich ihres Informationsgehaltes unterschieden werden. Geometrische Modelle definieren die Form eines Objekts über Kanten, Oberflächen oder Volumina und deren mathematischer Beschreibung. Sie beinhalten damit höherwertige Informationen als die Vereinigung der reinen Rohdaten (beispielsweise Punktwolken), aber weniger Informationen als semantische Modelle, die zusätzlich zur Form auch noch Wissen zu Funktionen oder Bedeutung bereitstellen (Abbildung 1.3). Geometrische Modelle lassen sich unterteilen in Kanten-, Oberflächen-, Volumen- und Baugruppenmodelle [Lee99].

Kantenmodelle (*wireframe models*) bilden ausschließlich Kanten ab ohne Flächen zu definieren. Die Darstellung kann mehrdeutig bezüglich Oberflächen sein und wird deshalb eher selten genutzt.

Oberflächenmodelle (*surface models*) sind definiert durch eine Menge von Kanten und Oberflächen, die nicht notwendigerweise ein geschlossenes Volumen bilden müssen. Beispiele für Oberflächenmodelle sind Polygonnetze und parametrisierte Oberflächen. Polygonnetze approximieren eine Oberfläche durch ein Netz von stückweise linearen Polygonen (meist Drei- oder Vierecke). Parametrisierte Oberflächen nutzen stückweise polynomiale Flächenstücke (*patches*). Meist wird dabei auf kubische Polynome zurückgegriffen, wie bei Bézier-Oberflächen, B-Spline-Oberflächen oder NURBS-Oberflächen.

Volumenmodelle (*solid models*) bilden im Gegensatz zu Oberflächenmodellen stets ein geschlossenes Volumen ab. Sie lassen sich unterteilen in diskretisierte und kontinuierliche Modelle. Zu den diskretisierten Modellen zählen alle Raumaufteilungsmodelle (*space decomposition, spatial partitioning*). Diese definieren ein Volumen durch die Unterteilung des Raumes in eine Menge von sich nicht schneidenden geometrischen Primitiven. Beispiele für solche Modelle sind Voxelgitter, Octrees oder BSP-Bäume. Aufgrund der Diskretisierung beziehungsweise der linearen Unterteilung können diese Modelle gekrümmte Objekte nur approximativ darstellen. Kontinuierliche Volumenmodelle hingegen enthalten algebraische Formen zur Darstellung von Oberflächen und können so je nach verwendetem Oberflächengrad eine Fläche exakt darstellen. Beispiele für Volumenmodellrepräsentation sind CSG-Modelle (*constructive solid geometry*), die ein Modell mittels boolescher Operationen auf Basis geometrischer Primitive schrittweise aufbauen, und Begrenzungsflächenmodelle (*boundary representation models*), die mit verknüpften Oberflächenstücken ein Volumen umschließen. Üblicherweise werden nur 2-

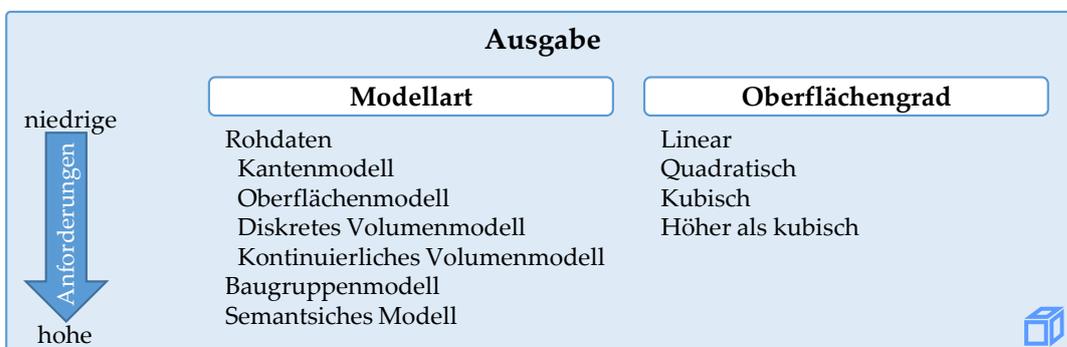


Abbildung 1.2: Bezüglich der Ausgabe kann zwischen Modellart und Oberflächengrad unterschieden werden. Die Ausprägungen sind aufsteigend sortiert nach den Anforderungen an das Gesamtsystem.

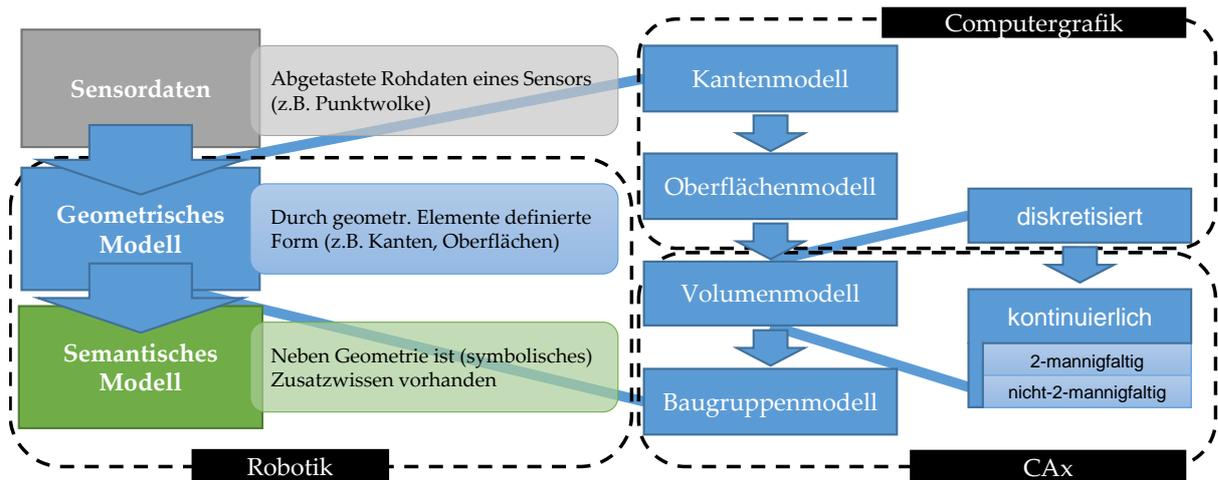


Abbildung 1.3: Geometrische Modelle können bezüglich ihres Informationsgehaltes zwischen Rohdaten und semantischen Modellen eingeordnet werden. Auf der rechten Seite sind zudem weitere Unterteilungen von geometrischen Modellen abgebildet. Die schwarzen Umrandungen markieren die Modellarten, die in den jeweiligen Anwendungsgebieten hauptsächlich eingesetzt werden.

mannigfaltige (*manifold*) Modelle betrachtet, das heißt, dass die Umgebung jedes Punktes auf der Oberfläche homöomorph zu einer Ebene ist [Hoffmann89].

Baugruppenmodelle (*assembly models*) sind Volumenmodelle, die zusätzlich eine Identifizierung von Teilkomponenten, sowie Parametrisierung und Instanziierung erlauben (beispielsweise ein Modell *Tisch* bestehend aus einer Instanz der Komponente *Platte* und vier Instanzen der Komponente *Bein*).

Im Bereich der Computergrafik finden hauptsächlich Oberflächenmodelle sowie diskretisierte Volumenmodelle Anwendung. Der Grund liegt hier im Wunsch, die Daten schnell und oftmals parallel auf Grafikhardware zu verarbeiten. Dazu eignen sich Modelle, die viele gleichartige Elemente besitzen (wie Dreiecksnetze oder Voxelgitter) besser. Im Bereich des computergestützten Entwerfens liegt der Fokus auf einer exakten und vollständigen Darstellung, sodass hier fast durchgängig kontinuierliche Volumenmodelle verwendet werden. Oft werden unterschiedliche Repräsentationen wie CSG-Modelle und Boundary-Representation-Modelle parallel benutzt, um die Vorteile beider Datenstrukturen ausnutzen zu können. In der Robotik hängt die Auswahl des Modells stark von der zu lösenden Aufgabe ab und geht bis hin zu semantischen Modellen, die symbolisches Zusatzwissen beinhalten, das beispielsweise für eine intuitive Mensch-Roboter-Interaktion nötig ist.

Oberflächengrad: Bei Oberflächen-, Volumen- und Baugruppenmodellen können Modelle hinsichtlich des Grades der verwendeten Oberflächen unterschieden werden. Lineare Oberflächen lassen nur eine Approximation mit planaren Flächen zu, Oberflächen höheren Grades bilden auch gekrümmte Bereiche ab. Meist werden keine höheren Grade als kubisch verwendet.

1.2.2 Verarbeitung

Im Bereich Verarbeitung sind alle Eigenschaften zusammengefasst, die die Berechnung des 3D-Modells aus den Sensordaten betreffen (Abbildung 1.4).

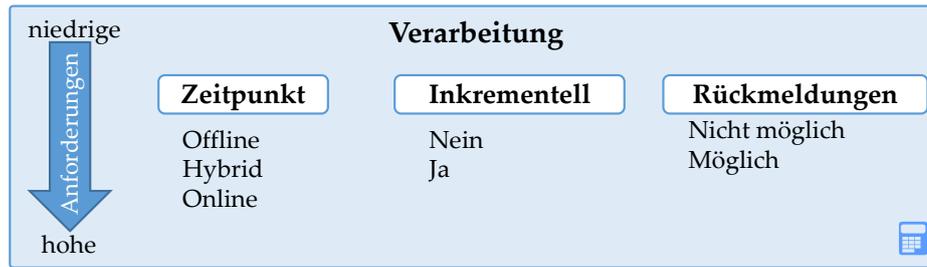


Abbildung 1.4: Bezüglich der Verarbeitung kann unterschieden werden, wann die Rekonstruktion stattfindet, ob das Modell inkrementell erzeugt wird und ob Rückmeldungen an den Nutzer möglich sind. Die Ausprägungen sind aufsteigend sortiert nach den Anforderungen an das Gesamtsystem.

Zeitpunkt: Die Berechnung des finalen Modells kann *offline*, also im Anschluss an den Erfassungsprozess durchgeführt werden. Erfolgt die Berechnung *online*, so wird bereits während der Erfassung ein Modell rekonstruiert, sodass ein Ergebnis vorliegt, wenn die Erfassung abgeschlossen ist. Dies erfordert jedoch, dass die verwendeten Algorithmen hinreichend schnell arbeiten. Zudem kann die Berechnung *hybrid* sein, also eine Mischung aus online und offline.

Inkrementell: Bei einer online oder hybriden Berechnung kann unterschieden werden, ob das resultierende Modell inkrementell erzeugt wird oder nicht. Inkrementell bedeutet, dass zu jedem Zeitpunkt während der Erfassung bereits ein valides Modell vorliegt, das die bis dahin erfassten Informationen enthält. Diese Zwischenergebnisse besitzen dabei die gleichen Eigenschaften (Modellart, Oberflächengrad) wie das endgültige Modell. Je nach Anwendung sind Zwischenergebnisse nützlich, beispielsweise können in der Robotik bei am Endeffektor montierten Kameras während einer Bewegung die Zwischenergebnisse für eine Bahn- oder Greifplanung genutzt werden. Zudem sind die Speicheranforderungen geringer, da nicht alle Einzeldaten bis zum Ende des Erfassungsprozesses vorgehalten werden müssen. Eine nicht inkrementelle Vorgehensweise behält alle Einzeldaten und extrahiert das resultierende 3D-Modell erst am Ende. Dies hat den Vorteil, dass abschließend alle verfügbaren Daten zur Erzeugung vorhanden sind und genutzt werden können.

Rückmeldungen: Ein System kann während der Erfassung Rückmeldungen an den Nutzer geben, die über eine reine Visualisierung des resultierenden Modells hinausgehen, beispielsweise Hinweise zu noch nicht erfassten Bereichen oder Informationen zur Vollständigkeit oder Qualität des Modells. Dies erleichtert dem Anwender die Handhabung.

1.2.3 Erfassung

Bei der Erfassung beeinflussen zwei Faktoren die Anforderungen an das Scan-System: der verwendete Sensor und die Art und Weise der Aufnahme. Die Eigenschaften beider Gruppen werden im Folgenden vorgestellt, beginnend beim Sensor.

Sensor

Der verwendete Sensor beeinflusst maßgeblich die Eigenschaften eines Scan-Systems, da Art, Struktur und Genauigkeit der Daten variieren können (Abbildung 1.5).

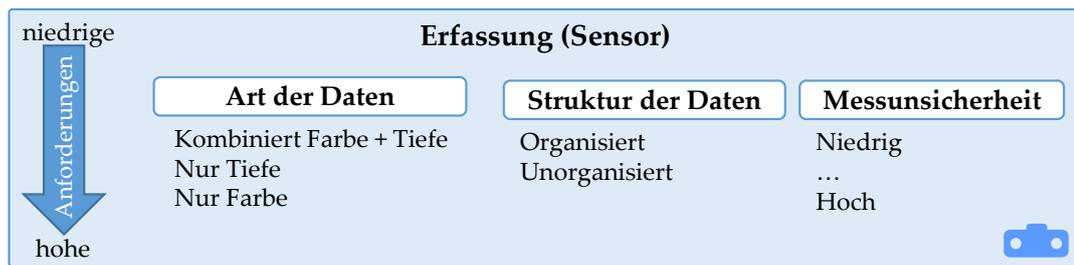


Abbildung 1.5: Die Eigenschaften des Sensors beeinflussen die Komplexität des Scan-Systems. Sensoren unterscheiden sich dabei hinsichtlich der Art und der Struktur der Daten, sowie der Stärke der Messunsicherheit. Die Ausprägungen sind aufsteigend sortiert nach den Anforderungen an das Gesamtsystem.

Art der Daten: Optische Sensoren für Scan-Systeme können grundlegend nach deren Modalität in Farb- und Abstandssensoren unterschieden werden. Abstandssensoren liefern Entfernungen zur Kamera, die direkt für eine Rekonstruktion genutzt werden können. Diese Informationen müssen bei reinen Farbkameras erst mithilfe mehrerer Aufnahmen berechnet werden. Kombinierte Farb-/Tiefensensoren liefern pro Pixel sowohl einen Abstand also auch einen Farbwert.

Struktur der Daten: Die erfassten Daten können in unterschiedlicher Struktur vorliegen. Während Bilder von Farbkameras in einem 2D-Pixelgitter organisiert sind, können Tiefendaten von Abstandssensoren sowohl organisiert als auch unorganisiert, das heißt als ungeordnete Menge von 3D-Messungen, vorliegen. Der Grund sind unterschiedliche Messprinzipien der Sensoren. Stereokameras oder Kameras basierend auf der Erkennung von ausgesendetem strukturiertem Licht nutzen 2D-Farbkameras, die selbst organisierte Daten liefern. Sensoren, die auf eine Laufzeitmessung von Licht aufbauen, können organisierte Daten (wie bei einigen *Time-Of-Flight*-Kameras) oder auch unorganisierte Daten (beispielsweise in Laserscannern, deren Laserstrahl sequentiell umgelenkt wird) generieren. Organisierte Tiefendaten erleichtern das Ermitteln von Nachbarpunkten und können so eine Rekonstruktion vereinfachen und beschleunigen.

Messunsicherheit: Je nach Messprinzip kann sich die Messunsicherheit der Messwerte deutlich unterscheiden. Insbesondere bei Abstandssensoren erreichen Laserscanner eine hohe Genauigkeit. Kostengünstige Sensoren für den Privatbereich hingegen besitzen oft eine geringe Auflösung, große Messabweichungen der Tiefenwerte bedingt durch Rauschen oder Kameraverzerrungen und Lücken um Objektkanten. Je stärker die Ungenauigkeit, desto schwieriger ist die Verarbeitung der Daten.

Einige Beispiele von 3D-Sensoren und deren Eigenschaften sind in Tabelle 1.1 dargestellt.

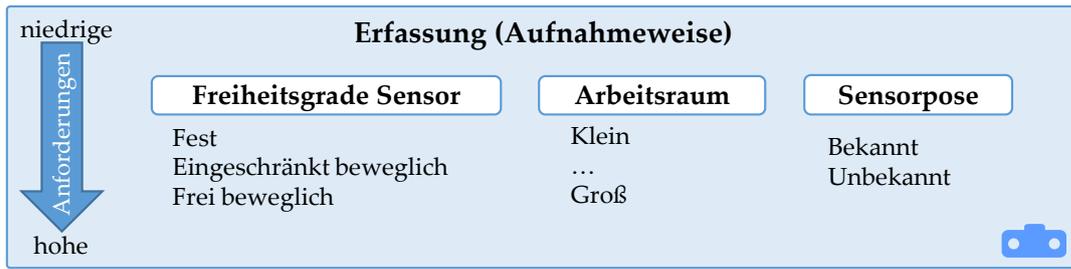


Abbildung 1.6: Bezüglich der Aufnahmeweise unterschieden sich Scan-Systeme in der Bewegungsfreiheit des Sensors, in der Größe des erfassbaren Arbeitsraumes und ob die Pose des Sensors bekannt ist oder nicht. In Tabelle 1.2 sind sechs Beispiele für verschiedenen Ausprägungen aufgelistet.

Aufnahmeweise

Die zweite maßgebliche Eigenschaft bei der Erfassung ist die Art und Weise der Aufnahme, also wie der verwendete Sensor bewegt werden kann, um eine Objekt oder eine Szene zu erfassen (Abbildung 1.6).

Freiheitsgrade der Sensorplatzierung: Scan-Systeme unterscheiden sich im Grad der Bewegungsfreiheit des Sensors. Diese reicht von fest verbauten, unbeweglichen Sensoren bis hin zu frei platzierbaren Sensoren. Je freier die Platzierung, desto komplexere Objekte können aufgenommen werden, da Verdeckungen besser vermieden werden können.

Arbeitsraum: Je nach System kann der erfassbare Bereich eingeschränkt sein, beispielsweise auf wenige Kubikzentimeter, oder nicht. Dies ist meist eine Folge von Einschränkungen bei der Sensorplatzierung, kann aber auch eine Limitierung des zugrunde liegenden Algorithmus sein, wenn hohe Speicheranforderungen das Aufnahmevolumen begrenzen.

Sensorpose: Die Pose des Sensors zum aufzunehmenden Objekt oder zwischen mehreren Blickwinkeln kann bekannt (zum Beispiel bei fest verbauten Sensoren) oder unbekannt und damit zusätzlich zu errechnen sein (beispielsweise bei handgehaltenen Kameras).

Einige exemplarische Fälle für unterschiedliche Art und Weisen der Erfassung und deren Eigenschaften sind in Tabelle 1.2 dargestellt.

Sensor (Beispiele)	3D-Laserscanner	Consumer-Tiefenkamera	Industrielle Tiefenkamera
Art der Daten	nur Tiefe	Farbe + Tiefe	nur Tiefe
Struktur der Daten	unorganisiert	organisiert	organisiert
Messunsicherheit	niedrig	hoch	mittel - niedrig

Tabelle 1.1: Anhand dreier Arten von Sensoren zeigen sich die unterschiedlichen Möglichkeiten der Ausprägungen aus Abbildung 1.5.

Aufnahmeweise (Beispiele)	Drehteller	Käfig	Roboterarm	Mobiler Roboter	Handgehaltener Sensor	Messarm mit Sensor
Freiheitsgrade	eine Achse	fest	frei	frei	frei	frei
Sensor						
Arbeitsraum	gering	mittel	mittel	groß	groß	mittel
Sensorpose	bekannt	bekannt	bekannt	unbekannt	unbekannt	bekannt

Tabelle 1.2: Unterschiedliche Arten von 3D-Scannern besitzen verschiedene Eigenschaften der Aufnahmeweise.

1.3 Vision: Der intelligente 3D-Scanner

Anhand der zuvor beschriebenen Eigenschaften kann nun ein *intelligenter* Scanner, wie ihn *Varady et al.* als höchstes Ziel der Forschung sehen, genauer spezifiziert werden, indem man die Ausprägungen mit den höchsten Anforderungen an das System kombiniert:

Ein solches System kann ein Objekt möglichst exakt (Oberflächengrad *kubisch* oder höher) abbilden und dabei möglichst viele geometrische Eigenschaften (Modellart *Baugruppenmodell*) oder sogar symbolische Eigenschaften (Modellart *Semantisches Modell*) automatisch bestimmen. Die Erfassung kann mit jeder handelsüblichen Tiefenkamera (Struktur der Daten *unorganisiert*, Rauschen *hoch*) oder Farbkamera (Information *nur Farbe*) erfolgen, indem diese frei im Raum bewegt wird (Sensorplatzierung *frei*, Arbeitsraum *groß*, Sensorpose *unbekannt*). Die Rekonstruktion des 3D-Modells erfolgt dabei während der Erfassung (Zeitpunkt *online*) und kann zu jedem Zeitpunkt abgerufen werden (*inkrementelle* Rekonstruktion). Zudem gibt das System dem Nutzer während der Aufnahme direkt Rückmeldungen, beispielsweise zu Qualität und Vollständigkeit des Modells (Rückmeldungen *möglich*). Der Scanner soll dabei so wenig Ressourcen (Speicherplatz, Rechenzeit) wie möglich verbrauchen.

Die Forschung in diese Richtung würde in vielen Anwendungsgebieten einen Fortschritt ermöglichen: Bei der computergestützten Produktentwicklung wird die Nachkonstruktion oder Erstellung eines CAD-Modells erheblich vereinfacht und kann auch von Nichtexperten durchgeführt werden. Zudem können Kosten gespart werden, wenn beispielsweise kein Koordinatenmesssystem mehr benötigt wird, sondern der Sensor in der Hand geführt werden kann [Benkő01]. Ist in einem 3D-Modell die Oberfläche durch analytische Flächengleichungen modelliert, so vereinfacht dies eine (physikalische) Simulation, die direkt auf diesem Modell durchgeführt werden kann [Hsu16]. Zur kollisionsfreien Bahnplanung müssen in der Robotik statische Hindernisse oft explizit von Hand modelliert werden. Ein handgehaltenes System erlaubt dabei eine einfache und schnelle Modellierung der Umwelt. Analytische Oberflächen vereinfachen zudem Optimierungsalgorithmen [Hänel12] oder Kollisionserkennungen [Poutrain01]. Sind auch während der Erfassung zu jedem Zeitpunkt bereits hochwertige Teilmodelle mit analytischen Oberflächen verfügbar, so können diese bei auf Roboterarmen montierten Sensoren (*eye-in-hand setup*) genutzt werden, um bereits während der Bewegung Planungsalgorithmen auszuführen, beispielsweise Greifplanung [Ding01]. Zudem hat eine inkrementelle Verarbeitung den Nutzen einer effizienten Speicherung, da nicht alle Rohdaten bis zum Ende vorgehalten werden müssen. Dies ist beispielsweise in eingebetteten System oder bei mobilen Robotern von Vorteil [Siegwart04].

1.4 Ziele und Beiträge dieser Arbeit

Diese Arbeit hat zum Ziel, der zuvor beschriebenen Vision des intelligenten 3D-Scanners einen Schritt näher zu kommen. Dazu werden in diesem Abschnitt die zentralen Herausforderungen erläutert, sowie die Aufgabenstellung dieser Arbeit dargestellt und abgegrenzt. Dabei werden die untersuchten wissenschaftlichen Fragestellungen beschrieben.

Um einen sinnvollen Schritt in die Richtung der Vision gehen zu können, müssen die Hauptprobleme und -herausforderungen bestimmt werden, die den aktuellen Systemen innewohnen. Im Stand der Forschung in Kapitel 2 wird ausführlich gezeigt, dass eine deutliche Zweiteilung existierender Verfahren vorherrscht: Auf der einen Seite Verfahren mit Fokus auf Erfassung und Verarbeitung, die einfach, robust und in Echtzeit 3D-Modelle mit handgehaltenen Kameras rekonstruieren können, aber ein Defizit beim Informationsgehalt im Ausgabemodell besitzen, da höchstens Oberflächenmodelle in Form von Dreiecksnetzen oder Volumendarstellungen in Form von diskretisierten Voxelräumen erstellt werden. Auf der anderen Seite stehen Verfahren, die viele und hochwertige Informationen extrahieren können und so ein kontinuierliches Volumenmodell erzeugen, aber deutliche Nachteile in der Erfassung und Verarbeitung besitzen, da globale und wenig verrauschte Rohdaten als Eingabe erwartet werden.

Um zur Vision des ultimativen intelligenten 3D-Scanners zu gelangen, muss diese Zweiteilung überwunden werden. Diese Arbeit hat deshalb zum Ziel, eine Brücke zu schlagen zwischen einfacher Handhabung und hochwertigen Ausgabemodellen (Abbildung 1.7). Es soll gezeigt werden, dass ein System sich nicht auf eine der beiden Seiten beschränken muss und eine Kombination möglich ist. Zudem soll untersucht werden, was die Grenzen eines solchen Systems sind. Dazu wird im Abschnitt 1.4.1 unter Verwendung der oben definierten Eigenschaften eines Scan-Systems die Aufgabenstellung dieser Arbeit genauer spezifiziert und in Abschnitt 1.4.2 die Beiträge dieser Arbeit in Form von wissenschaftliche Fragestellungen, die untersucht werden, formuliert.

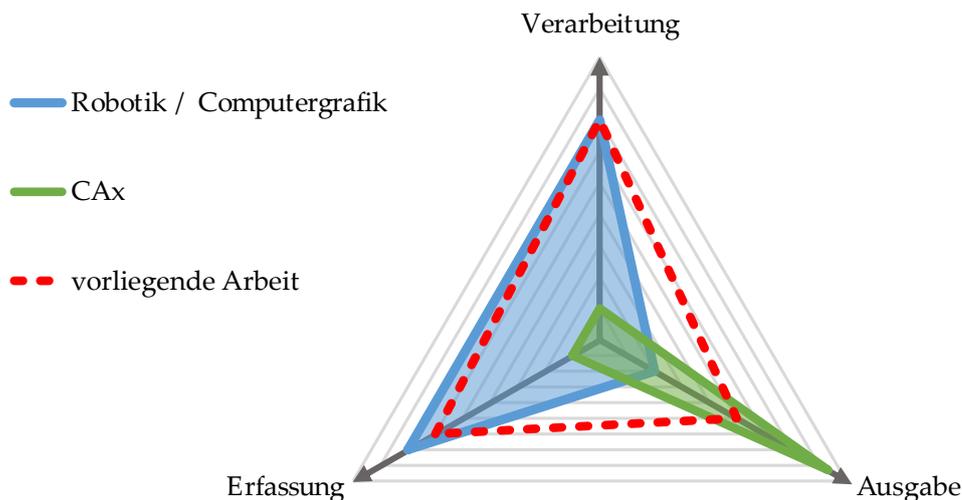


Abbildung 1.7: Schematische Darstellung der Leistungsfähigkeit von 3D-Scan-Systemen aus den Anwendungsgebieten der Robotik/Computergrafik (blau) und des computergestützten Entwerfens, kurz CAx (grün), in den Kategorien Erfassung, Verarbeitung und Ausgabe. Je weiter außen, desto höhere Anforderungen vermag ein System in diesem Bereich zu bewältigen. Die bestehende Zweiteilung zwischen einem Fokus auf Ausgabe und einem Fokus auf Erfassung/Verarbeitung soll mit der vorliegenden Arbeit (rot) überwunden werden.

Erfassung		Verarbeitung	
Art der Daten	nur Tiefe	Zeitpunkt	online
Struktur der Daten	organisiert	Inkrementell	ja
Messunsicherheit	hoch	Rückmeldungen	möglich
Freiheitsgrade Sensor	frei	Ausgabe	
Arbeitsraum	groß	Modellart	kontin. Volumenmodell
Sensorpose	unbekannt	Oberflächengrad	linear

Tabelle 1.3: Die vorliegende Arbeit hat als Aufgabenstellung, ein 3D-Scan-System mit den in der Tabelle beschriebenen Eigenschaften zu entwickeln.

1.4.1 Aufgabenstellung

In dieser Arbeit soll ein 3D-Scan-System entwickelt und evaluiert werden, das die Lücke zwischen einfacher Handhabung und hochwertigen Ausgabemodellen überwindet. Ausgedrückt in Form der anfangs definierten Eigenschaften eines Scan-Systems bedeutet dies ein *kontinuierliches Volumenmodell* als Modellart bei gleichzeitiger Verwendung einer *frei beweglichen* Kamera, deren Sensorpose *unbekannt* ist.

Die weiteren Eigenschaften werden dabei hinsichtlich des Anwendungsbereiches der Robotik sowie einer einfachen Bedienung auch für Nicht-Experten festgelegt: Die Verarbeitung soll *online* und *inkrementell* erfolgen, sodass zu jedem Zeitpunkt während der Erfassung bereits hochwertige (Teil-)Modelle verfügbar sind. Dies bietet mehrere Vorteile: Zum einen kann eine Erfassung jederzeit beendet werden, sobald das rekonstruierte Modell die Anforderungen an eine Aufgabe erfüllt, zum anderen können bereits während der Erfassung Planungs- und Optimierungsalgorithmen ausgeführt werden. Für eine einfache intuitive Bedienung sollen Rückmeldungen an den Nutzer während der Erfassung möglich sein, um den Aufnahmeprozess zu unterstützen. Bei Szenen mit vielen Verdeckungen ist es leicht möglich, dass Löcher im Modell vom Anwender übersehen werden. Da das System schon während der Erfassung Rückmeldungen an den Nutzer geben kann, entfällt eine anschließende Inspektion des Modells und eine erneute Erfassung zum Schließen von Löchern, was zu einer Zeitersparnis führt.

Im Hinblick auf kostengünstige Sensoren und die zunehmende Verbreitung von Sensorik in mobilen Geräten werden als Eingabe Tiefendaten in Form von organisierten Punktwolken angenommen, die aber mit einer nicht vernachlässigbaren Messunsicherheit behaftet sein können.

Als Oberflächengrad des Ausgabemodells wird der einfachste aller Fälle angestrebt, nämlich lineare Oberflächen. Damit können gekrümmte Oberflächen nicht, beziehungsweise nur als Approximation abgebildet werden. Für einige Anwendungen ist dies ausreichend, für andere nicht. Typische Innenraumszenen verfügen jedoch fast immer über eine große Anzahl an planaren Flächen wie Wände, Tische, oder Boden, sodass eine planare Rekonstruktion schon eine gute Approximation der Realität darstellt.

Eine tabellarische Übersicht der genannten Eigenschaften ist in Tabelle 1.3 zu finden.

1.4.2 Wissenschaftliche Fragestellungen

Aus diesen Zielsetzungen ergeben sich mehrere wissenschaftliche Fragestellungen, die in dieser Arbeit untersucht werden sollen. Da das System inkrementell arbeiten soll, ist es nötig, zu jedem Zeitpunkt während der Erfassung bereits Modelle zur Verfügung zu stellen. Diese Zwischenergebnisse sollen dabei den gleichen Informationsgehalt wie das Endergebnis besitzen, also kontinuierliche Volumenmodelle sein. Da zu einem Zeitpunkt ein Objekt oder eine

Szene bis dahin nur teilweise erfasst wurde, kann das Zwischen-Modell noch nicht vollständig sein. Da kontinuierliche Volumenmodelle per Definition immer ein komplettes Volumen umschließen, ist hier eine geeignete Repräsentation zu finden:

F1 In welcher Form können Modelle repräsentiert werden, die noch kein vollständiges Volumen umschließen, aber dennoch die gleichen hochwertigen Eigenschaften besitzen wie kontinuierliche Volumenmodelle?

Ein Schritt zur Überwindung der Lücke zwischen hochwertigen Modellen und einfacher und schneller Bedienung ist, dass solche Modelle inkrementell und online erzeugt werden:

F2 Inwieweit ist es bei bekannter Sensorpose möglich, inkrementell und online solche Modelle zu erzeugen?

Ein solches System kann beispielsweise in der Robotik für *Eye-in-hand* Konfigurationen eingesetzt werden. Für eine vollständig freie Kameraplatzierung in Form einer handgehaltenen Kamera ist es allerdings nötig, die Sensorpose während der Erfassung zu bestimmen:

F3 Inwieweit kann die Pose automatisch bestimmt werden, sodass ein *Simultaneous-Localization-And-Mapping* (SLAM)-System realisiert wird?

Damit auch Nicht-Experten ein solches System intuitiv verwenden können, stellt sich die Frage, wie der Anwender durch das System unterstützt werden kann:

F4 Inwieweit und in welcher Weise kann das System den Anwender bei der Erfassung unterstützen?

Da alle Fragestellungen in einem 3D-Scan-System realisiert werden sollen, ist offensichtlich, dass die vier Fragestellungen nicht unabhängig voneinander zu beantworten sind. Eine geeignete Modellrepräsentation muss kompatibel mit einer inkrementellen und online-fähigen Rekonstruktion sein. Die Posenbestimmung muss schnell genug für eine Online-Ausführung sein.

1.5 Kapitelübersicht

Die folgenden Kapitel dieser Arbeit sind wie folgt strukturiert: In Kapitel 2 wird der Stand der Forschung untersucht und beleuchtet, wie die bereits erwähnte Zweiteilung zwischen Systemen mit Fokus auf einem hochwertigen Ausgabemodell und einfachen online-fähigen Rekonstruktionssystemen zustande kommt.

Anschließend wird in Kapitel 3 das Grundkonzept des in dieser Arbeit vorgestellten Systems dargestellt. Die dort vorgeschlagenen Ansätze bilden die Basis für die folgenden Abschnitte, die einzelne Teile des Systems beschreiben.

Kapitel 4 beschäftigt sich ausschließlich mit der Rekonstruktion eines hochwertigen Modells aus einer einzigen Kamerapose, während in Kapitel 5 die Fusion mehrerer Einzelmodelle unter Annahme einer bekannten Sensorpose betrachtet wird. Diese Einschränkung wird in Kapitel 6 aufgehoben, indem Verfahren zur automatischen Bestimmung der Pose (*Registrierung*) untersucht werden.

Um vollständige Modelle zu erhalten, beschäftigt sich Kapitel 7 mit Löchern und wie diese entweder automatisiert erkannt und geschlossen werden können oder wie der Nutzer auf Löcher während der Erfassung hingewiesen werden kann.

In Kapitel 8 werden dann alle beschriebenen Einzelkomponenten als Ganzes betrachtet und die Umsetzung des Gesamtsystems auf realer Hardware beschrieben.

Als Abschluss fasst Kapitel 9 die Arbeit nochmals zusammen und gibt Ausblicke auf zukünftige Forschungsfragen.

Kapitel 2

Stand der Forschung

Inhalt

2.1 Simultaneous Localization And Mapping	23
2.1.1 Graphbasierte Verfahren	23
2.1.2 Modellbasierte Verfahren	27
2.1.3 Hybride Verfahren	29
2.2 Digital Shape Reconstruction	30
2.2.1 Gesamtsysteme	32
2.2.2 Segmentierungsverfahren	33
2.3 Schlussfolgerungen	35

In diesem Kapitel werden bestehende Ansätze zur Rekonstruktion von 3D-Modellen näher erläutert. Dabei wird im ersten Abschnitt auf *Simultaneous Localization And Mapping* (SLAM)-Systeme eingegangen, die sich mit dem Problem der gleichzeitigen Lokalisierung der Kamera sowie der Kartierung der Umwelt beschäftigen. Den zweiten Abschnitt bilden Systeme aus dem Bereich der *Digital Shape Reconstruction* (DSR), auch *Reverse Engineering* genannt, die sich mit der Erzeugung von informationsreichen 3D-Modellen im Kontext der computergestützten Entwicklung (CAx) beschäftigen. Den Abschluss bildet eine Zusammenfassung und Bewertung der Verfahren hinsichtlich der in dieser Arbeit betrachteten Aufgabenstellung. Einzelne Teile der folgenden Darstellung wurden bereits in [Sand13] und [Sand14] dargestellt und veröffentlicht.

2.1 Simultaneous Localization And Mapping

Als *Simultaneous Localization And Mapping*, kurz *SLAM*, bezeichnet man die gleichzeitige Rekonstruktion der Umwelt und Bestimmung der Sensorpose mit einer bewegten Kamera. Dies ist ein Henne-Ei-Problem, da für die Rekonstruktion die aktuelle Position des Sensors benötigt wird, aber für die Berechnung der Pose eine rekonstruierte Karte. Wird als Sensor ausschließlich eine Kamera benutzt, so spricht man auch von *visual SLAM*, kurz *vSLAM*. Handelt es sich dabei um eine einzelne RGB-Kamera, so wird das Problem als *monocular SLAM* bezeichnet, bei Verwendung einer Stereokamera spricht man von *Stereo-SLAM* oder *RGB-D SLAM*. Im Folgenden wird hauptsächlich auf SLAM mit Tiefenkameras eingegangen. Tiefergehende Informationen zu monokularem SLAM können in den Übersichts-Arbeiten von Younes et al. und Taketomi et al. [Younes16, Taketomi17] gefunden werden.

Grundsätzlich lassen sich die SLAM-Verfahren nach der Repräsentation der Karte in zwei Gruppen einteilen: Graphbasierte und modellbasierte Verfahren. Bei graphbasierten Verfahren (auch *frame-to-frame* Verfahren genannt) existiert während der Aufnahme kein Gesamtmodell, sondern jede Einzelaufnahme wird als Knoten in einem sogenannten Posengraphen repräsentiert. Zwischen Einzelposen werden inkrementell relative Transformationen berechnet, die als Kanten im Graph abgelegt werden. Durch das Schließen von sogenannten Schleifen kann der Graph optimiert werden. Diese Verfahren werden in Kapitel 2.1.1 näher betrachtet.

Bei modellbasierten Verfahren (auch *frame-to-model* Verfahren genannt) wird eine globale Karte als Modell vorgehalten. Die Pose einer neuen Aufnahme wird durch Vergleich zu diesem globalen Modell berechnet. Anschließend wird die neue Aufnahme in das Modell hineinfusioniert. Das Modell wird dadurch vollständiger. Diese Art von Arbeiten wird in Kapitel 2.1.2 genauer untersucht.

Bei der Berechnung der Pose einer Aufnahme, sei es inkrementell oder durch Registrierung auf das Gesamtmodell, kann unterschieden werden zwischen *dense methods* und *sparse methods*. Eine Methode wird als *dense* bezeichnet, wenn zur Berechnung das komplette Sensorbild und alle darin zur Verfügung stehenden Informationen herangezogen werden. Weitere übliche Bezeichnungen für dieses Vorgehen sind *every-pixel-method* oder *direct visual odometry*. Im Gegensatz dazu bezeichnet man die Berechnung als *sparse*, wenn aus den Bilddaten zuerst bestimmte Informationen extrahiert werden, auf Basis derer dann die Berechnung durchgeführt wird. Dies können geometrische Informationen sein, wie Ebenen innerhalb einer Punktwolke, oder markante Punkte innerhalb eines Bildes, die besonders interessant oder aussagekräftig sind (*feature points*). Der Vorteil liegt in der kürzeren Berechnungsdauer, da eine geringere Datenmenge verarbeitet wird. Dafür muss allerdings das Korrespondenzproblem gelöst werden, das einem Merkmalspunkt aus der vorhergehenden Messung einen Merkmalspunkt aus der aktuellen Messung zuordnet. Bei fehlerhafter Zuordnung entstehen schnell große Fehler. Der Vorteil der *dense methods* ist, dass alle verfügbaren Informationen auch zur Berechnung verwendet werden und so kein Informationsverlust auftreten kann.

2.1.1 Graphbasierte Verfahren

Graphbasierte Verfahren (auch *frame-to-frame* Verfahren) erstellen einen sogenannten Posengraph. Jeder Knoten des Graphs repräsentiert eine Aufnahme und enthält die Pose der Aufnahme, also die Translation und die Rotation im Bezug auf ein Weltkoordinatensystem, aus der die Aufnahme erstellt wurde. Kanten verbinden zeitlich aufeinanderfolgende Aufnahmen und enthalten die relative Transformation zwischen den Posen. Die Berechnung dieser inkrementellen Transformationen wird als *Visuelle Odometrie* bezeichnet, da die Posänderung nur anhand der aufgenommenen Bilder der Kamera berechnet wird. Da sich Fehler bei der Berechnung der Odometrie durch die kettenartige Struktur des Graphen fortpflanzen und

so die Gesamtgenauigkeit erheblich verringern können, werden bei graphbasierten Verfahren noch zwei weitere Schritte vorgenommen: Durch sogenannte *Loop Closings* („Schließen einer Schleife“) werden zusätzliche Kanten eingefügt, die Zyklen im Graph erzeugen. Dies ist beispielsweise möglich, wenn ein Ort aufgenommen wird, der bereits zu einem früheren Zeitpunkt schon einmal erfasst wurde. Zwischen diesen nun nicht mehr zeitlich aufeinanderfolgenden Bildern kann ebenfalls eine Transformation berechnet werden, die als zusätzliche Kante im Posegraph eingefügt wird. Dadurch sind nun zusätzliche Informationen vorhanden und die Gesamtgenauigkeit steigt. Dafür müssen die Posen im Graph allerdings durch einen Optimierungsschritt angepasst werden, da die Transformationen der Kanten nicht konsistent sein müssen. Dieser Vorgang wird als Graphoptimierung bezeichnet. Um letztendlich eine komplette Karte zu erhalten, müssen die Daten der einzelnen Aufnahmen mithilfe der nun optimierten Posen in den Knoten fusioniert werden.

In den folgenden Abschnitten werden zuerst Verfahren vorgestellt, die nur eines der eben beschriebenen Teilprobleme des graphbasierten SLAM lösen. Abschließend wird auf deren Kombination im Gesamtverfahren eingegangen.

Visuelle Odometrie

In diesem Abschnitt werden Ansätze dargestellt, die aus zwei Tiefenbildern die relative Transformation zwischen den Posen der beiden Aufnahmen berechnen. Die Methoden können in graphbasierten SLAM-Systemen zur Berechnung der visuellen Odometrie zwischen zwei aufeinanderfolgenden Aufnahmen dienen, sowie zur Berechnung der Transformation zwischen zwei Aufnahmen, die durch das Schließen einer Schleife gefunden wurden. Beide Bilder müssen sich dabei überlappen.

Eine Methode zur Ermittlung der Transformation ist die Annahme der Photokonsistenz [Steinbruecker11]: Ein Weltpunkt taucht in den beiden zu vergleichenden Bildern an verschiedenen Stellen auf und besitzt dort die gleiche Farbintensität. Mithilfe der Tiefenbilder und der intrinsischen Kameraparameter kann eine sogenannte Warping-Funktion aufgestellt werden, die ein Pixel des ersten Bildes auf ein Pixel des zweiten Bildes abbildet. Die Funktion enthält dabei als Variable die unbekannte Kamerabewegung. Diese wird bestimmt, indem die Photokonsistenz maximiert wird, also die Differenz zwischen dem zweiten Bild und dem mit der Warping-Funktion transformierten ersten Bild minimiert wird. Als Eingabe müssen kolorierte Punktwolken vorliegen.

Das Verfahren kann verbessert werden, indem zur Minimierung der Differenz zwischen aktuellem Bild und transformierten Vorgängerbild und damit zur Maximierung der Photokonsistenz ein probabilistisches Verfahren verwendet wird [Kerl13b]. Dabei wird das Differenzbild mithilfe einer Wahrscheinlichkeitsverteilung ausgedrückt und die Kamerabewegung kann mit einer Maximum A Posteriori-Abschätzung ermittelt werden. Der Vorteil des Verfahrens liegt darin, dass nun auch ein Sensormodell und ein Bewegungsmodell mit in die Berechnung einfließen, die die Genauigkeit verbessern.

Während bei den beiden vorhergehenden Ansätzen stets alle Pixel betrachtet wurden, werden bei der merkmalsbasierten Odometrieberechnung zuerst Merkmalspunkte (*feature points*) aus den RGB-Bildern extrahiert. Die Position der Merkmale im Bild ergibt zusammen mit den Tiefeninformationen eine Wolke aus Merkmalspunkten. Um die Transformation zum vorhergehenden Bild zu finden, können die beiden Merkmalspunktfolgen miteinander verglichen und Korrespondenzen gefunden werden [Endres12]. Anschließend wird der RANSAC-Algorithmus [Fischler81] angewandt, um die Transformation zu bestimmen. Um die Pose

noch robuster zu ermitteln, können Merkmale über mehrere Frames hinweg verfolgt werden und dabei mithilfe einer Filtersystems die zuverlässigsten Merkmale ermittelt werden [Domínguez13].

Als Merkmale können nicht nur farb- oder intensitätsbasierte Merkmalspunkte verwendet werden, sondern beispielsweise auch Ebenen, die aus den Punktwolken extrahiert werden. Anschließend werden Korrespondenzen der Ebenen des aktuellen Bildes zu den Ebenen des zu vergleichenden Tiefenbildes gesucht und daraus die Transformation ermittelt. Zum Finden der Ebenen kommen dabei Bereichswachstum (*region growing*) [Pathak09] oder der RANSAC-Algorithmus [Lee12] zum Einsatz.

Einen komplett anderer Ansatz, um die Transformation zwischen zwei Tiefenbildern mit teilweiser Überlappung zu berechnen ist die Verwendung von Spektralanalyse [Bülow13]. Von den zwei zu vergleichenden Tiefenbildern wird dazu die Fourier-Transformation berechnet und das Phasenspektrum verglichen. Anschließend wird zuerst der Gierwinkel bestimmt. In weiteren Schritten folgen dann Roll- und Nickwinkel, sowie die Bestimmung der Translation. Das Verfahren ist sehr robust gegenüber Rauschen, funktioniert allerdings nicht für große Roll- und Nickwinkel zwischen zwei Aufnahmen.

Mit Fokus auf sehr große Karten, beispielsweise bei der Kartierung durch Kameras in Fahrzeugen, schlagen Wang et al. [Wang17] für Stereo-Kameras eine Kombination aus Stereo- und monokularem Verfahren vor. Zur Berechnung der Odometrie eines Frames müssen sowohl beide Augen der Stereokamera (*static stereo*) als auch zeitlich aufeinanderfolgende Bilder (*temporal stereo*) konsistent durch die Pose erklärt werden.

Schließen von Schleifen

Als *Loop Closing* bezeichnet man beim graphbasierten SLAM das zusätzliche Einfügen einer Kante zwischen zwei zeitlich nicht direkt aufeinanderfolgenden Knoten. Dies entspricht der Aufgabe anhand des Bildes zu erkennen, ob man das Objekt oder die Szene im Bild bereits zuvor einmal erfasst hat. Die Schwierigkeit liegt darin, eine kurze Berechnungsdauer für das Erkennen von Schleifen zu erreichen, da es umso mehr Möglichkeiten gibt, je größer die Karte ist. Während viele SLAM-Systeme Schleifen durch eine Suche, die durch mehr oder weniger Zusatzwissen oder Heuristiken geleitet wird, finden, werden in diesem Abschnitt zwei Konzepte dargestellt, die explizit auf das schnelle Finden von Schleifen ausgelegt sind.

Als *Bag of Words* [Galvez-Lopez11] bezeichnet man die Charakterisierung eines Bildes durch sogenannte (visuelle) Worte. Die Zahl der möglichen Worte ist dabei begrenzt. Die Gesamtheit der Worte wird Vokabular benannt und offline durch einen Trainingsschritt bestimmt. Dadurch ist ein Vergleich zwischen zwei Bildern mit Worten schneller möglich als durch einen Vergleich mit Merkmalsdeskriptoren, da die Anzahl an möglichen Worte begrenzt und kleiner als die Anzahl an möglichen Merkmalsdeskriptoren ist. Zur Erkennung von Schleifen werden die Worte des aktuellen Bildes mit den Worten aller vorherigen Bilder verglichen.

Wird die Karte immer größer, so steigt auch die Anzahl der möglichen Partner für ein Loop-Closing mit dem aktuellen Bild. Ab einer bestimmten Kartengröße ist ein Vergleich mit allen vorhergehenden Bildern nicht mehr in Echtzeit zu bewerkstelligen. Ebenso kann der Speicherverbrauch bei großen Karten ein Problem werden. Das von Labbé et al. [Labbé13] vorgestellte Verfahren namens RTABMap (*Real-Time Appearance-Based Mapping*) reduziert die Anzahl der Bilder, die auf das Schließen einer Schleife getestet werden, und lagert die anderen auf die Festplatte aus. Das Kriterium, welche Bilder zum Vergleich behalten werden, ist ein Gewicht,

das beschreibt, wie häufig ein bestimmter Ort schon besucht wurde. Das Vokabular muss dabei nicht wie bei Galvez-Lopez vorher erlernt werden, sondern kann inkrementell erstellt werden.

Graphoptimierung

Bei graphbasierten SLAM-Ansätzen kann die Genauigkeit der Karte durch Optimierung des Posengraphs verbessert werden. In diesem Kapitel sollen kurz einige ausgewählte Algorithmen aufgelistet werden, die in den SLAM-Verfahren des darauf folgenden Abschnitts zum Einsatz kommen.

TORO (Tree-based netwORk Optimizer) [Grisetti09] ist ein Optimierungsframework für Constraint-Netzwerke in graphbasierten SLAM-Verfahren. Knoten besitzen dabei Konfigurationen, die mit den Posen der einzelnen Aufnahmen parametrisiert sind. Kanten beinhalten Bedingungen zwischen zwei Konfigurationen, ausgedrückt durch die Transformation, die mittels Odometrie oder Schleifenerkennung errechnet wurde. Mithilfe eines stochastischen Gradientenabstiegs wird nun die beste Konfiguration aller Knoten gefunden. Dabei werden nicht alle Bedingungen in den Kanten auf einmal optimiert, sondern alle Kanten in zufälliger Reihenfolge nacheinander abgearbeitet.

Das *general graph optimization*-Framework (kurz *g2o*) [Kummerle11] ist ein Framework zur Optimierung von Problemen, die sich als Graph modellieren lassen. Knoten enthalten dabei die Parameter, die optimiert werden. Eine Kante, die zwei Knoten verbindet, besitzt eine Fehlerfunktion, die aus den Parametern der Knoten die Abweichung vom Optimum errechnet. Der Gesamtfehler ist die Summe aus den Fehlerfunktionen der Kanten. Dieser wird durch Anpassung der Knotenparameter minimiert. Das Framework ist dabei nicht wie bei Grisetti auf einen Anwendungsfall spezialisiert, sondern kann an ein Problem angepasst werden, indem der Parameterraum und die Fehlerfunktionen definiert werden. Zur Optimierung stehen mehrere Implementierungen verschiedener Gleichungslöser zur Auswahl, beispielsweise eine Cholesky-Zerlegung oder das Verfahren der konjugierten Gradienten.

Ein Framework zur Graphoptimierung in der mobilen Robotik ist *iSAM* (incremental Smoothing and Mapping) [Kaess08]. Es berücksichtigt dabei den Roboterstatus (inklusive Pose), Steuerbefehle, Landmarken und Messungen von Landmarken. Daraus wird ein Graph erstellt, in dem sowohl die Roboterposen als auch die Positionen der Landmarken optimiert werden. Dafür verwendet man eine QR-Zerlegung, die auch inkrementell durchgeführt werden kann, sobald eine neue Messung eintrifft.

Gesamtverfahren

In den vorhergehenden Kapiteln wurden bereits Lösungen für einzelne Teilprobleme des graphbasierten SLAM vorgestellt. Prinzipiell kann man diese Methoden beliebig kombinieren und so ein graphbasiertes SLAM-System erstellen, wie an den folgenden Arbeiten ersichtlich ist.

In der Arbeit von Endres et al. [Endres12] wird für das aktuelle Bild die visuelle Odometrie zum Vorgängerbild berechnet. Dazu wird das oben beschriebene Vorgehen mit Merkmalsextraktion und RANSAC-Transformationsberechnung angewandt. Für das aktuelle Bild wird die Pose zum Vorgängerbild berechnet sowie zu 20 zufällig ausgewählten früheren Frames. Die Optimierung des Posengraphs geschieht offline im Anschluss an die Rekonstruktion und

verwendet das *g2o*-Framework. Mit den optimierten Posen können nun alle Einzelpunktwolken fusioniert werden.

Das Loop-Closing Verfahren RTABMap [Labbé13] wurde vom Autor zu einem kompletten, graphbasierten SLAM-Verfahren erweitert. Dadurch werden einerseits mehr Schleifen gefunden und andererseits kann sich das System bei verllorener Registrierung durch Finden einer Schleife wieder relokalisieren. Die Berechnung der visuellen Odometrie erfolgt wie bei Endres, zur Graphoptimierung wird jedoch das TORO-Framework verwendet.

Kerl et al. [Kerl13a] verwenden ihr oben beschriebenes, probabilistisches Odometrieverfahren zur Erstellung eines Posengraphs. Dabei wird eine Aufnahme nicht auf ihren direkten Vorgänger registriert, sondern auf einen sogenannten *Keyframe*. Zwischen diesen werden auch Schleifen zu anderen Keyframes gesucht. Zur Optimierung des Posengraphs wird das *g2o*-Framework verwendet. Die Arbeit von Babu et al. [Babu16] verbessert dieses Verfahren nochmals, indem ein anderes probabilistisches Modell zur Berechnung der Odometrie verwendet wird.

2.1.2 Modellbasierte Verfahren

Die zweite Gruppe an SLAM-Verfahren verwendet ein globales Modell als Repräsentation der Karte (auch *frame-to-model* Verfahren genannt). Dabei existiert eine einzige Karte, in die eine Aufnahme sofort hinein fusioniert wird. Die Berechnung der Pose der Aufnahme geschieht durch Vergleich mit der globalen Karte und wird als Registrierung bezeichnet. Um Fehler zu vermeiden, werden oft statistische Methoden angewandt, welche das Rauschen herauszufiltern und das Verfahren robuster zu machen.

Im Folgenden werden drei Gruppen an Verfahren nach ihrer Repräsentation der globalen Karte unterschieden: Volumetrische Modelle, Surfel-Modelle und sonstige Modelle.

Volumetrische Modelle

Basierend auf der Arbeit von Curless and Levoy [Curless96] ist das Modell durch eine Unterteilung des Raumes in Voxel repräsentiert, die die Werte einer Distanzfunktion (*signed distance function, SDF*) enthalten. Diese Funktion gibt den vorzeichenbehafteten Abstand zur nächstgelegenen Oberfläche an. Mithilfe des *Marching-Cubes-Algorithmus* [Lorensen87] kann die Oberfläche, die durch die Nulldurchgänge der Funktion definiert ist, als Dreiecksnetz extrahiert werden. Mithilfe dieser Darstellung ist es möglich modellbasierte SLAM-Systeme zu erstellen.

Newcombe et al. realisieren eine Online-Rekonstruktion namens *KinectFusion* [Newcombe11, Izadi11], die das Voxel-Modell auf der Grafikkarte vorhält. Um eine neue Punktwolke zu integrieren, wird diese mit einem Referenzbild verglichen und daraus die relative Pose berechnet. Als Referenzbild dient dabei ein Tiefenbild mit der Pose des vorhergehenden Kameraframes, das per Raycasting aus der globalen Karte errechnet wurde. Zur Bestimmung der Pose wird ein auf der GPU implementierter *Iterative Closest Points (ICP)*-Algorithmus [Rusinkiewicz01] verwendet. Anschließend wird die Punktwolke mithilfe eines laufenden Mittelwertes in die Karte integriert und per Raycasting wieder ein synthetisches Tiefenbild aus dem Blickwinkel der aktuellen Kameraposition erzeugt. Dies dient sowohl als Referenzbild für die folgende Aufnahme, als auch als Live-Visualisierung für den Nutzer. Eine Ansicht aus anderen Blickwinkeln ist während der Aufnahme nicht möglich.

Nachteilig ist hier der hohe Rechenaufwand, der ohne GPU nicht in Echtzeit zu bewältigen ist, sowie der hohe Speicherbedarf, da sowohl Freiraum als auch Objekte im Voxelgitter

vollständig modelliert werden. Daher muss ein Kompromiss zwischen Genauigkeit und Rekonstruktionsvolumen getroffen werden.

Um diese Nachteile zu kompensieren, wurden einige Ansätze veröffentlicht. Zur Verringerung des Rechenaufwandes können Octrees eingesetzt werden [Steinbruecker13] oder Hashing-Verfahren für Voxel-Blöcke [Nießner13, Kähler15].

Um die Einschränkungen des Rekonstruktionsvolumens aufzuheben, kann der Ort des Voxelgitters verschoben werden [Roth12], sodass sich das Volumen stets mitbewegt. Whelan et al. [Whelan12] schreiben dazu in ihrem System *Kintinuous* die aus dem verschobenen Volumen herausfallenden Daten als Dreiecksnetz auf die Festplatte. Zusätzlich wird ein Posengraph erstellt, der als Knoten alle Posen enthält, an denen eine Verschiebung stattgefunden hat. Zum Schluss können so alle abgelegten Netze mittels des Posengraphs fusioniert werden. Durch Verwendung eines Posengraphs entstehen wieder Probleme wie bei den graphbasierten Verfahren, sodass Loop-Closing und Graphoptimierung nötig werden, um eine Drift-freie Karte zu erhalten [Whelan13, Whelan15b].

Surfel-Modelle

Der Nachteil eines sehr großen Speicheraufwandes von volumetrischen Modellen kann nur durch Verwendung einer anderen Modellart umgangen werden. Keller et al. [Keller13] stellen dazu eine punktbasierte Repräsentation auf Basis von *Surfels* [Pfister00] vor. Ein Surfel, kurz für *surface element*, ist ein ortsgebundenes, punkartiges Primitiv, das zusätzliche Informationen wie eine Normale, eine Kovarianz, einen Radius oder eine Farbe besitzen kann. Mehrere Surfels beschreiben eine Oberfläche, ohne dabei explizite Nachbarn zu besitzen, das heißt ohne Konnektivität. Der Vorteil gegenüber Dreiecksnetzen ist, dass Surfels bei Modifikationen am Modell leichter konsistent gehalten werden können.

Keller et al. [Keller13] verwenden Surfels mit einer Normale, einem Radius und einem Konfidenzzähler als globales Modell. Die Registrierung einer neuen Aufnahme erfolgt dabei analog zu KinectFusion, indem ein Modell aus der letzten Pose gerendert wird, das mithilfe des ICP-Algorithmus mit dem neuen Bild verglichen wird. Bei der Fusion werden Surfels hinzugefügt, aktualisiert oder entfernt, sodass das globale Modell größer oder valider wird. Da nun kein Freiraum mehr modelliert wird, sind diese Modelle speichereffizienter als volumetrische Modelle.

Bei planaren Oberflächen besitzen benachbarte Surfels ähnliche Eigenschaften (z.B. Normale, Konfidenz). Salas-Moreno [Salas-Moreno14] weist diesen Bereichen ein einheitliches Label zu und erkennt so Ebenen, die gesondert behandelt werden und auch komprimierter abgelegt werden können.

Zur Verbesserung der Genauigkeit besonders bei großen Karten führen Whelan et al. [Whelan15a, Whelan16] zusätzlich einen Deformationsgraphen und eine Schleifenerkennung ein. Hierbei werden Schleifen zu Bereichen erkannt, die bereits einmal rekonstruiert wurden. Dadurch ist eine genauere Posenabschätzung möglich. Im Gegensatz zu graphbasierten Verfahren, in denen Schleifen erst bei der Graphoptimierung berücksichtigt werden, wird das Modell sofort mithilfe des sogenannten Deformationsgraphen verbessert, der nicht nur die gerade betrachteten Teile der Karte aktualisiert, sondern auch weiter entfernte Stellen. Durch Aufteilung in Untermodelle erreichen Kähler et al. [Kähler16] noch eine weitere Verbesserung der Genauigkeit.

Eine Kombination volumetrischer Modelle mit Surfels wird von Stückler et al. [Stückler14] vorgeschlagen. In der *Multi-Resolution Surfel Map (MRSMap)* genannten Datenstruktur wird

für jede Voxelseite ein Surfel vorgehalten, in dem Lage und Farbe der Oberfläche codiert sind. Dadurch soll eine einfachere Fusion aus unterschiedlichen Blickwinkeln erreicht werden mit dem Nachteil eines wieder höheren Speicherbedarfs.

Sonstige Modelle

In der auf mobile Roboter ausgelegten Arbeit von Taguchi et al. [Taguchi13] dienen ins 3D projizierte 2D-Farbmerkmale und 3D-Merkmale in Form von erkannten Ebenen als Landmarken, die in einer globalen Karte vorgehalten werden. Für ein neues Bild werden diese Merkmale extrahiert und die Pose durch Vergleich mit den gesammelten Landmarken mithilfe des RANSAC-Algorithmus ermittelt. Anschließend werden gegebenenfalls Landmarken aktualisiert oder hinzugefügt. Je nachdem, welche Landmarken in welchen Frames gesehen wurden, entsteht ein Graph aus Bedingungen in der globalen Karte, der parallel zum SLAM stets optimiert wird, um eine möglichst konsistente Karte zu haben. Da in der globalen Karte nur die Landmarken enthalten sind, ist das resultierende Modell keine Beschreibung der gesamten Oberfläche, sondern nur eine Menge aus unzusammenhängenden Ebenen. Ataer-Cansizoglu et al. [Ataer-Cansizoglu13] verbessern das Tracking, indem noch ein Bewegungsmodell der Kamera eingeführt wird, das allerdings nicht für handgehaltene Kameras geeignet ist, da es eine konstante Bewegungsgeschwindigkeit der Kamera annimmt.

2.1.3 Hybride Verfahren

Das kürzlich vorgestellte Verfahren von Yan et al. [Yan17] versucht die Vorteile der graphbasierten und modellbasierten Ansätze zu vereinen, um eine verbesserte Genauigkeit der Posen und der Rekonstruktion zu erreichen. Basis ist eine probabilistische Surfel-Datenstruktur (*probabilistic surfel map*, PSM), die eine Oberfläche als Surfels mit Unsicherheiten modelliert. Dabei wird einerseits mit Keyframes gearbeitet, das heißt neue Frames werden auf den letzten Keyframe registriert und dort in ein Keyframe-PSM hineinfusioniert. Wurde die Kamera weiter bewegt, wird ein neuer Keyframe erstellt. Die Gesamtheit der Keyframes wiederum bildet einen Graph, zwischen denen auch Schleifen geschlossen werden können. Zusätzlich existiert ein globales PSM, in das alle Keyframe-PSM fusioniert werden. Ein Graphoptimierungsschritt, der das *g2o*-Framework benutzt, optimiert diesen Graph und das globale PSM. Die finale Rekonstruktion erhält man, indem nach der Optimierung nochmals alle Keyframe-PSMs fusioniert werden.

2.2 Digital Shape Reconstruction

Auch im Bereich der Fertigung und CAD-Konstruktion wird sich in den letzten Jahrzehnten verstärkt mit dem Thema der Rekonstruktion von realen Objekten beschäftigt. Die *Society of Manufacturing Engineers* (SME) definiert den Prozess zur Erzeugung eines 3D-Modells aus realen Objekten und das zugehörige Fachgebiet als *Digital Shape Sampling and Processing* (DSSP) [Marks05]. Dies umfasst sowohl die Hard- als auch Softwarekomponenten eines solchen Systems. Die algorithmischen Aspekte eines Systems werden als *Digital Shape Reconstruction* (DSR) zusammengefasst. Diese neuen Termini wurden 2005 von der SME vorgeschlagen, da die bisher verwendeten Begriffe das Thema zu unscharf abgrenzten. Insbesondere der gebräuchliche Begriff des *Reverse Engineering* (RE) soll dadurch abgelöst werden, um die Konnotation des reinen Kopierens oder Nachkonstruierens zu vermeiden und eine enge Verbindung zu industriellen Fertigungsprozessen herzustellen [Marks05].

Ziel der *Digital Shape Reconstruction* ist die Erzeugung eines hochqualitativen 3D-Modells aus Messdaten, um sie innerhalb eines CAE-Prozesses nutzbar zu machen, beispielsweise für die Konstruktion (CAD) oder Simulation. Dementsprechend ist auch ein Ziel konstruktions-spezifische Merkmale neben der reinen Geometrie ableiten zu können, wie zum Beispiel Parametrisierungen oder Symmetrien, bis hin zum Versuch die Entwurfsabsicht (*design intent*) erkennen zu können [Várady08].

Im Gegensatz zu Abschnitt 2.1 liefern die hier beschriebenen Methoden also mindestens ein kontinuierliches Volumenmodell. Dafür wird aber stets von einem globalen Modell als Eingabe ausgegangen. Das heißt, ein zu rekonstruierendes Objekt liegt bereits als Gesamtpunktwolke oder sogar als Dreiecksnetz vor. Insbesondere wird das Problem der Einzelbildregistrierung umgangen, indem oft Koordinatenmesssysteme verwendet werden.

Eigenschaften von DSR-Systemen

Grundsätzlich lassen sich DSR-Systeme hinsichtlich mehrerer Kriterien unterscheiden [Várady08]. Eines davon ist der verwendete Flächentyp im rekonstruierten Modell. Hierbei kann man vier Hauptgruppen identifizieren:

- **Einfache algebraische** Oberflächen, wie Ebenen oder quadratische Flächen (Kugel, Zylinder, Kegel) lassen sich durch eine Gleichung beschreiben.
- **Sweep-Oberflächen** sind Oberflächen, die durch Rotation (*revolution*) oder Translation (*extrusion*) eines 2D-Profiles entlang einer Kurve im Raum entstehen. Einfache algebraische Oberflächen und Sweep-Oberflächen zusammen werden auch als prismatische Oberflächen bezeichnet [Várady08].
- **Freiformflächen** stellen die Oberfläche als Netz von parametrisierten Flächenstücken dar. Meist kommen dabei NURBS (*non-uniform rational B-spline*) Oberflächen zum Einsatz. Durch die Unterteilung in ein Netz können damit komplexere Oberflächenformen dargestellt werden als in den vorhergehenden Kategorien. Allerdings lassen sich in Modellen, die ausschließlich aus Freiformflächen bestehen, keine eindeutigen Teilflächen des Gesamtmodells mehr identifizieren, da stets ein Netz über die Oberfläche gelegt wird.
- Als **Sekundäre Flächen** werden Hilfsflächen bezeichnet, die primäre Flächen auf eine bestimmte Art und Weise verbinden. Beispiele dafür sind Abrundungen an Ecken oder Kanten (*blends*). Das Erkennen von sekundären Flächen ist oft für Modelle erwünscht, die in CAD-Programmen weiter verändert werden sollen.

Ein weiteres Unterscheidungsmerkmal ist die vom Rekonstruktionsverfahren benötigte Genauigkeit der Eingabedaten. Viele Verfahren sind sehr sensibel gegenüber Rauschen, unterschiedlichen Punktdichten oder Unvollständigkeit der Daten. Ebenso relevant ist der Grad der Automatisierung. Einige Verfahren benötigen an mehreren Stellen manuelle Eingriffe, um ein akzeptables Ergebnis zu gewährleisten. Dies ist insbesondere dann der Fall, wenn das Modell die ursprüngliche Entwurfsabsicht (*design intent*) abzubilden versucht. Beispiele dafür sind das eindeutige Erkennen von Hauptflächen gegenüber sekundären Flächen, Symmetrien, Parametrisierungen, oder Abhängigkeiten zwischen Maßen. Ein Modell, das die Entwurfsabsicht berücksichtigt, kann im Anschluss an die Rekonstruktion leicht in einem CAD-Programm modifiziert und weiterverarbeitet werden.

Vorgehensweisen von DSR-Systemen

Innerhalb der *Digital Shape Reconstruction* lassen sich vier grundlegende Herangehensweisen unterscheiden [Várady08], um von Rohdaten in Form einer Punktwolke oder eines Dreiecksnetzes ein Modell zu erzeugen:

- Bei der **manuellen Segmentierung** muss der Nutzer die Rohdaten (meist in Form eines Dreiecksnetzes) manuell segmentieren, indem Trennkurven gezeichnet werden. Dadurch können einzelne Flächenteile identifiziert und angepasst werden und anschließend zu einem Gesamtmodell verbunden werden.
- Beim **Redesign** wird das Modell manuell in einem CAD-Programm nachkonstruiert, die Geometrie wird allerdings direkt aus den aufgenommenen Rohdaten extrahiert. Diese Art der Modellerzeugung sowie die der manuellen Segmentierung ist vergleichsweise zeitaufwändig, da sie viel manuellen Eingriff erfordert. Die Entwurfsabsicht kann allerdings am besten abgebildet werden.
- **Automatic Surfacing** Methoden bilden ein Modell ausschließlich aus Freiform-Oberflächenstücken. Sie besitzen eine hohe Approximationsgenauigkeit und arbeiten vollautomatisiert, können eine Entwurfsabsicht aber nur schlecht abbilden. Da das Modell aus einem Netz aus parametrisierten Flächenstücken besteht, wird es häufig bei Modellen eingesetzt, bei denen es auf die Ästhetik ankommt. Es ist eher ungeeignet für Design-Anpassungen und bauteilabhängige Umstrukturierungen, da topologische Informationen fehlen.
- Methoden, die nach dem Prinzip der **funktionalen Zerlegung** (*functional decomposition*) arbeiten, zerlegen das Modell automatisiert in Einzelflächen, berechnen Kanten und erzeugen so ein Modell. Je nach Vorgehen kann das Verfahren fast vollständig automatisiert ablaufen. Das Resultat ist ein vollständiges CAD-Modell, das auch die Entwurfsabsichten abbildet. Der Nachteil solcher Methoden liegt oft in der Anforderung genauer und vollständiger Eingabedaten.

Auf die ersten beiden Arten der Rekonstruktion wird im Folgenden nicht weiter eingegangen, da sie zu viel manuellen Eingriff erfordern, um in der Aufgabenstellung dieser Arbeit verwendet werden zu können. Methoden zum *automatic surfacing* und der funktionalen Zerlegung bestehen stets aus mehreren Schritten, insbesondere lassen sich drei Teile identifizieren [Chang11]:

1. **Triangulierung:** Liegen die Rohdaten als Punktwolke vor, so werden diese in ein Dreiecksnetz überführt.

2. **Segmentierung:** Das Dreiecksnetz wird in Segmente unterteilt, die später eine einzige Fläche im Volumenmodell repräsentieren. Für jedes Segment wird eine geeignete Repräsentation gefunden (z.B. durch Anpassung eines geometrischen Primitivs). Dies ist der wichtigste Schritt im gesamten Prozess.
3. **Modellierung:** Die Segmentierung wird in ein Volumenmodell überführt. Zusätzlich werden je nach Verfahren sekundäre Flächen, Parametrisierungen oder andere Konstruktionsmerkmale erkannt.

In den folgenden Abschnitten werden nun bekannte Verfahren aus dem Bereich der *Digital Shape Reconstruction* vorgestellt. Dabei existieren nur wenige Verfahren, die den gesamten Prozess abdecken. Diese werden in Abschnitt 2.2.1 einzeln vorgestellt. Da der Segmentierungsschritt den essentiellen Teil eines DSR-Systems darstellt, wird in Abschnitt 2.2.2 näher auf Segmentierungsverfahren eingegangen.

2.2.1 Gesamtsysteme

Es gibt nur wenige Arbeiten, die den gesamten Rekonstruktionsprozess abdecken. Eine der ersten ist von Benkő et al. [Benkő01]: Nach einer Triangulierung werden aus dem Netz Dreiecke entfernt, die an Orten hoher Krümmung liegen. Dadurch entstehen unverbundene Regionen von Dreiecken, die eine Segmentierung des Gesamtmodells darstellen. Dieser Schritt wird als *direkte Segmentierung* bezeichnet. An jedes Segment wird anschließend eine einzige analytische Oberfläche angepasst. Schlägt dies fehl, wird versucht, das Segment weiter zu unterteilen. Bei der Anpassung werden viele unterschiedliche geometrische Tests durchgeführt, um auch rotatorische und translatorische Sweeps sowie Symmetrieachsen zu erkennen. Anschließend wird ein Regionen-Adjazenzgraph erstellt, der die Topologie des späteren Modells widerspiegelt. Damit können nun benachbarte Flächen geschnitten werden, um Kanten und Ecken zu berechnen. Den Abschluss bildet das Anpassen von Rundungen (*blends*) an den Stellen, an denen anfangs Dreiecke entfernt wurden.

Ein ähnliches Vorgehen beschreiben Huang & Menq, das ebenfalls aus den drei Schritten der Triangulierung, Segmentierung und Modellierung besteht [Huang03]. Im Gegensatz zu Benkő werden jedoch zuerst die Einzel-Punktwolken aus verschiedenen Sichten trianguliert und anschließend die Netze verschmolzen, um ein Gesamtnetz zu erhalten. Dieses wird anschließend segmentiert, indem Kanten anhand Unstetigkeiten in der Krümmung erkannt werden [Huang01]. Solche Begrenzungskanten werden anschließend verbunden, sodass das Netz in einzelne Teile zerlegt wird, an die dann Flächen angepasst werden [Huang02]. Nach der Berechnung der Topologie wird zuletzt ein CAD-Modell erzeugt.

Várady et al. gehen von einem Dreiecksnetz als Eingabe aus [Várady06]. Auf dieses wird mithilfe der Morse-Theorie [Edelsbrunner03] ein Netzwerk aus Linien gelegt, das bestimmte Punkte (wie Extrema oder Sattelpunkte einer Krümmungsfunktion) verbindet. Einerseits entsteht dadurch eine Segmentierung in einzelne Bereiche, an die Oberflächen angepasst werden können, andererseits wird dieser Graph später genutzt, um an dessen Knoten und Kanten Abrundungen (*blends*) anzupassen, die die benachbarten Flächenstücke glatt verbinden.

Bénière et al. stellen ein vollautomatisches Verfahren vor, das ein globales, nahezu exaktes Eingabe-Dreiecksnetz in ein CAD-Modell überführt [Bénière13]. Damit adressieren sie den Anwendungsfall des Reverse Engineering von Daten, die aus der Diskretisierung eines bereits bestehenden, aber unter Umständen nicht mehr verfügbaren CAD-Modells oder aus einer mechanischen Abtastung entstanden sind. Zur Rekonstruktion werden zuerst geometrische Primitive gefunden, indem für jeden Punkt des Netzes die Hauptkrümmungsrichtungen

bestimmt werden und benachbarte Punkte mit ähnlicher Krümmungscharakteristik gruppiert werden. Anhand der Nachbarschaften im Mesh wird ein Adjazenzgraph der Primitive erstellt, die dann mit ihren Nachbarn geschnitten werden. Aus den Schnittkanten und den Segmenten werden zuletzt einzelne Flächen eines Begrenzungsflächenmodells erzeugt, die dann zusammengesetzt werden.

Ohne Triangulierung kommen Kyriazis et al. aus [Kyriazis13,Stamati10]. Aus einer globalen Gesamtpunktwolke werden Querschnitte gebildet. In diesen werden im Zweidimensionalen B-Spline Kurven angepasst. Aus dem Stapel an Schnitten werden nun ähnliche Merkmale erkannt, sodass benachbarte Kurven zu einer B-Spline Oberfläche erweitert werden. Das resultierende Modell besteht somit ausschließlich aus B-Spline Oberflächenstücken und ist daher eher für Freiform-Objekte geeignet. Eine Entwurfsabsicht wird nicht erkannt.

2.2.2 Segmentierungsverfahren

Der wichtigste Teil eines DSR-Systems ist die Segmentierung, in der die Eingabe, meist in Form eines Dreiecksnetzes in einzelne Bestandteile unterteilt wird, die später eine Fläche im Modell repräsentieren. Dieser Abschnitt beschäftigt sich mit Verfahren, die ausschließlich die Segmentierung betrachten.

Da die Segmentierung von 3D-Daten in vielen Anwendungsbereichen benötigt wird, existiert eine Vielzahl an Methoden mit teils sehr unterschiedlichen Zielen und Herangehensweisen, von denen nur ein Teil für DSR-Systeme geeignet ist. Die Arbeiten von Agathos et al, Shamir, und Theologou et al. [Agathos07,Shamir08,Theologou15] geben hier einen guten Überblick über die Bandbreite der Methoden. Allgemein lässt sich die Segmentierung in zwei Klassen unterteilen, je nachdem welcher Zweck verfolgt wird (Abbildung 2.1):

- Verfahren, die die Oberfläche in sinnvolle Regionen aufteilen (*surface-based methods*).
- Verfahren, die das gesamte Modell in sinnvolle (volumetrische) Teile aufteilen (*part-based methods*).

Für DSR-Systeme ist dabei die oberflächenbasierte Segmentierung von Relevanz, auf die sich im Folgenden beschränkt wird.

Weiterhin können Segmentierungsverfahren anhand ihrer Herangehensweise unterteilt werden. Im Folgenden wird dabei genauer auf drei Arten eingegangen, da diese besonders gut geeignet sind für die Segmentierung von geometrischen Objekten: Regionenwachstum (*region growing*), Clustering und Methoden zur Anpassung von Primitiven. Für andere Arten, wie topologische Methoden, die auf die Struktur und deren Semantik abzielen, sei auf die Zusammenfassungen von Agathos et al, Shamir, und Theologou et al. [Agathos07,Shamir08,Theologou15] verwiesen.

Regionenwachstum Ausgehend von einem sogenannten *Seed*-Punkt auf der Oberfläche werden benachbarte Elemente (zum Beispiel Punkte oder Dreiecke) zum gleichen Segment hinzugefügt, wenn ein bestimmtes Konsistenz-Kriterium erfüllt ist. Trifft dies für keinen Nachbarn mehr zu, so wird eine neues Segment durch Wachstum ausgehend von einem neuen Seed-Punkt im noch nicht segmentierten Bereich gestartet. Es handelt sich somit um ein gieriges (*greedy*) und lokales Verfahren, bei dem die Wahl der Seed-Punkte und die Reihenfolge der Abarbeitung einen Einfluss auf das Ergebnis haben. Als Konsistenz-Kriterium können beispielsweise Hauptkrümmungsrichtungen [Lavoué05,Denker13] oder Abstände zu Polynomen [Besl88,Sapidis95] dienen. Eine schlechte Wahl der Seedpunkte kann zu einer Übersegmentierung führen, die aber in vielen Fällen durch eine Verschmelzung von Segmenten im Anschluss korrigiert werden kann [Theologou15]. Um bessere Seed-Punkte zu erhalten, kann

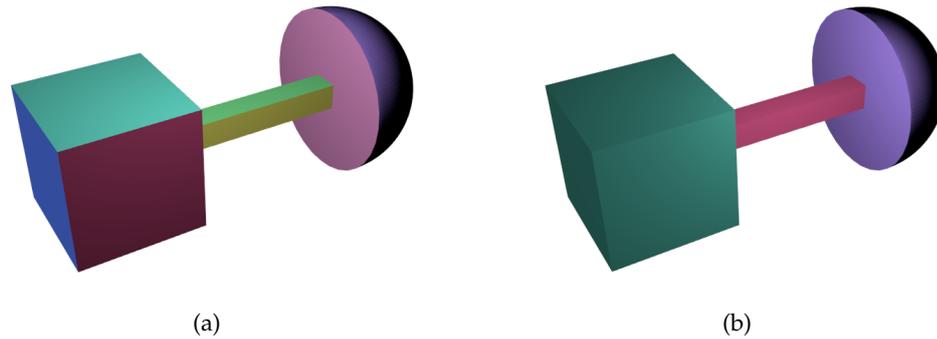


Abbildung 2.1: (a) Segmentierung der Oberfläche (*surface-based*), (b) Segmentierung des Gesamtmodells in volumetrische Teile (*part-based*)

auch ein zweistufiges Verfahren verwendet werden: Anhand einer Analyse der Hauptkrümmungen werden Bereiche mit gleichen Eigenschaften gefunden, die zu Punkten kontrahiert werden, die anschließend als Seed-Punkte für das eigentliche Wachstum dienen [Vieira05].

Clustering Clustering-Methoden benötigen keine Seed-Punkte und vermeiden so das Problem von schlecht gewählten Seed-Punkten. Die Verfahren lassen sich in zwei Gruppen gliedern: Hierarchisches Clustering und Iteratives Clustering.

Beim hierarchischen Clustering unterscheidet man *bottom-up* und *top-down* Methoden. Bei ersterer ist anfangs jedes Dreieck ein einzelnes Cluster. Anschließend werden immer wieder zwei benachbarte Cluster zusammengefasst, sodass ein Baum entsteht, der eine Segmentierung auf verschiedenen Hierarchieebenen darstellt. Die Wahl, welche Cluster zusammengefasst werden, wird durch eine Kostenfunktion bestimmt, in der die gewünschten Eigenschaften eines Segments codiert sind. Beispiele dafür sind Planarität [Garland01] oder die Segmentgröße und Glattheit der Begrenzung [Inoue01]. Das Prinzip kann erweitert werden, sodass unterschiedliche geometrische Primitive [Gelfand04, Attene06] oder Sweep-Flächen [Miandarhoie17] erkannt werden.

Der *top-down*-Ansatz geht von einem einzigen großen Segment aus, das iterativ unterteilt wird. Dazu werden 2D-Begrenzungen gefunden, die zur Teilung verwendet werden [Katz03]. Die Mischung aus *top-down* und *bottom-up* kann auch als *Split-and-Merge* bezeichnet werden. Ausgehend von einem initialen Clustering anhand der Krümmung können benachbarte Segmente geteilt oder verschmolzen werden, sodass ein globales Maß optimiert wird [Taylor03]. Iteratives Clustering basiert auf dem Prinzip des *k-means*-Clustering, auch *Lloyd*-Methode genannt [Lloyd82, Duda00]. Dazu werden folgende zwei Schritte abwechselnd ausgeführt, bis sich keine Änderung mehr einstellt: Erstens wird für jedes Cluster die beste planare Approximation, genannt *Proxy*, bestimmt. Zweitens werden für jeden Proxy die dazu passenden Dreiecke, also neue bessere Cluster bestimmt [Cohen-Steiner04]. Die Anzahl der Cluster muss dabei vorher bekannt sein. Das Verfahren kann erweitert werden, sodass nicht nur planare Approximationen erkannt werden, sondern auch andere Kugeln, Zylinder und Abrundungen (*rolling-ball blends*), indem jeweils das *Proxy* verwendet wird, das am besten passt [Wu05]. Yan et al. erweitern das Verfahren auf Quadriken und lockern die Einschränkung, dass die Anzahl der Cluster vorher bekannt sein muss, indem sie die Anzahl der Cluster schrittweise erhöhen, bis eine Fehlerschwelle unterschritten wird [Yan06, Yan12].

Anpassung von Primitiven Sowohl Regionenwachstums- als auch Clusteringmethoden sind in der Lage geometrische Primitive zu erkennen. In diesem Abschnitt sollten weitere Verfahren basierend auf Primitiven erläutert werden, die sich aber nicht zu einer der beiden vorhergehenden Kategorien zuordnen lassen.

Der RANSAC-Algorithmus [Fischler81] erlaubt es, robust alle Punkte einer Menge zu finden, die ein Modell, also auch Primitive, erklären, indem durch zufälliges Ziehen vieler Stichproben die beste Instanz des Modells ermittelt wird. Zur Segmentierung kann der RANSAC-Ansatz mehrfach hintereinander ausgeführt werden, um mehrere Instanzen unterschiedlicher Primitive zu erkennen [Schnabel07]. Das Ergebnis ist jedoch keine vollständige Segmentierung, da Restpunkte, die von keinem Modell erklärt werden, übrig bleiben können. Da die einzelnen RANSAC-Iterationen unabhängig voneinander sind, kann das Verfahren verbessert werden, indem globale Relationen zwischen Primitiven (wie Parallelität, Orthogonalität, Symmetrie, gleiche Winkel usw.) gefunden werden, die mit einfließen [Li11]. Dabei liegt die Annahme zugrunde, dass viele von Menschen geschaffene Gegenstände solche Relationen aufweisen.

Eine ähnliche Annahme liegt der Arbeit von Le et al. [Le17] zugrunde: Anhand einer Vorsegmentierung in planare Stücke durch iteratives Clustering werden die Hauptrichtungen des Modells erkannt. Anhand derer werden orthogonale Schnitte durchgeführt, sodass 2D-Schnitte entstehen, mithilfe derer es leichter ist, 3D-Primitive wie Zylinder, Kugeln, Kegel und Tori zu erkennen. Das Problem einer unvollständigen Segmentierung wird behoben, indem ein probabilistischer Ansatz zur Lösung des Set-Cover-Problems angewendet wird.

2.3 Schlussfolgerungen

In diesem Abschnitt soll der vorgestellte Stand der Forschung nochmals kurz zusammengefasst werden und gleichzeitig mit den Zielen dieser Arbeit in Verbindung gebracht werden. Dazu werden die in Kapitel 1 identifizierten Eigenschaften eines 3D-Scan-Systems verwendet (Abbildung 1.1).

Simultaneous Localization and Mapping (SLAM)-Systeme im Allgemeinen haben ihre Stärken in der Online-fähigen Berechnung der Rekonstruktion mit frei beweglichen Sensoren und damit unbekannter Sensorpose. Auch Consumer-Tiefenkameras mit relativ starken Messabweichungen sind kein Problem. Graphbasierte Systeme bestehen aus mehreren Teilen, die in der Theorie beliebig kombiniert werden können. Dadurch sind diese Systeme flexibel und leicht erweiterbar. Durch die Graphstruktur wird das resultierende Gesamtmodell jedoch erst am Ende des Aufnahmeprozesses erzeugt, sodass keine Zwischenergebnisse direkt vorliegen, sondern bei Bedarf erst extrahiert werden müssen. Die meisten Veröffentlichungen der letzten Zeit konzentrieren sich auf modellbasierte SLAM-Systeme. Durch ihr globales Modell sind diese sehr robust gegenüber Messabweichungen, benötigen jedoch ein leistungsstarkes System, meist mit GPU. Das Problem eines sehr begrenzten Aufnahmevermögens konnte durch Wechsel von volumetrischen Unterteilungen auf punktbasierte Surfel-Modelle erheblich verbessert werden. Zwischenergebnisse während der Rekonstruktion liegen vor, aber stets nur in der Repräsentation des verwendeten Modells, beispielsweise Surfels. Das Endergebnis beider Arten von SLAM-Systemen sind Oberflächenmodelle in Form von Dreiecksnetzen.

Systeme zur *Digital Shape Reconstruction* (DSR) legen ihren Fokus auf das Ausgabemodell. Kontinuierliche Volumenmodelle, meist in Form von Begrenzungsflächenmodellen, können extrahiert werden. Die Vielzahl an Segmentierungsverfahren zeigt, dass für unterschiedliche Oberflächentypen auch andere Verfahren zum Einsatz kommen, wobei die Erkennung von geometrischen Primitiven am häufigsten adressiert wird. Insgesamt existieren nur wenige Verfahren, die den gesamten Prozess inklusive Erstellung des finalen Modells beschreiben. Des Weiteren wird als Eingabe eine einzige globale Punktwolke oder ein Dreiecksnetz erwartet, die das zu modellierende Objekt vollständig beschreibt. Wird ein Modell aus mehreren Aufnahmen erstellt, so werden zuerst die Rohdaten fusioniert und anschließend das DSR-Verfahren als Post-Processing-Schritt angewandt. Demnach wird auch kein Fokus auf Online-Fähigkeit

		Ausgabe	
		\leq diskretes Volumenmodell	\geq kontinuierliches Volumenmodell
Erfassung Verarbeitung	offline-Berechnung geringe Messunsicherheit Gesamtpunktwolke		Digital Shape Reconstruction
	online-Berechnung hohe Messunsicherheit unregistrierte Einzelbilder	Simultaneous Localization and Mapping	

Tabelle 2.1: Bestehende Systeme lassen eine klare Zweiteilung erkennen: SLAM-Systeme besitzen ihre Stärken in der Erfassung und Verarbeitung, haben aber Defizite beim Ausgabemodell. Bei DSR-Systemen ist es umgekehrt. Diese Arbeit hat zum Ziel diese Lücke zu überwinden und lässt sich deshalb rechts unten einordnen.

der Systeme gelegt. In den meisten Fällen darf zudem kein oder nur wenig Rauschen vorhanden sein, was die Wahl der Sensoren einschränkt.

Insgesamt lässt sich feststellen, dass in den bestehenden Systemen eine klare Zweiteilung mit unterschiedlichen Schwerpunkten herrscht: Einerseits SLAM-Systeme mit Fokus auf robuste, online-fähige Rekonstruktion und andererseits DSR-Systeme mit Fokus auf informationsreiche, hochwertige Ausgabemodelle (Tabelle 2.1). Der Grund ist in der historischen Entwicklung zu vermuten, da SLAM-Systeme hauptsächlich im Bereich der Computergrafik beheimatet sind, DSR-Systeme eher im Bereich der computergestützten Konstruktion. Im Hinblick auf die Vision eines intelligenten 3D-Scanners, der alle Eigenschaften und Probleme gleichermaßen adressiert und löst, muss diese Lücke allerdings überwunden werden.

Kapitel 3

Grundkonzept

Inhalt

3.1 Kombination von SLAM und DSR	39
3.1.1 Bewertung	39
3.1.2 Schlussfolgerungen	40
3.2 Partielle Modelle	42
3.2.1 Repräsentation partieller Modelle	42
3.2.2 Eigenschaften partieller Modelle	45
3.2.3 Formale Beschreibung eines partiellen Begrenzungsflächenmodells . .	46
3.3 Parametrisierung	53
3.3.1 Die Strukturgröße als Universalparameter	53
3.3.2 Berücksichtigung von Messabweichungen	54
3.3.3 Begrenzung der Strukturgröße durch Messabweichungen	56
3.4 Zusammenfassung	57

In diesem Kapitel werden die Grundideen zur Lösung des in dieser Arbeit betrachteten Problems dargestellt und begründet. Um einen Schritt in Richtung der Vision des ultimativen intelligenten 3D-Scanners gehen zu können, muss die im Stand der Forschung ermittelte Lücke zwischen *Simultaneous Localization and Mapping* (SLAM)-Systemen und Methoden zur *Digital Shape Reconstruction* (DSR) überwunden werden. Der erste Abschnitt beschäftigt sich deshalb mit dieser Kombination aus beiden Gebieten und deren Umsetzung und Folgen.

Eine Folge aus dieser Kombination ist die Notwendigkeit von partiellen Modellen. Das sind Modelle mit den gleichen Eigenschaften wie kontinuierliche Volumenmodelle, mit der Ausnahme, dass sie noch kein komplettes Volumen umschließen. Eigenschaften und Repräsentationsarten solcher partieller Modelle werden im zweiten Abschnitt erläutert.

Da ein Ziel eine einfache, auch für Nicht-Experten geeignete Benutzung des Systems ist, wird im dritten Abschnitt kurz auf die Parametrisierung eingegangen. Größere Softwaresysteme leiden oft unter einer großen Anzahl an Parametern, die vom Nutzer in einer sinnvollen Weise eingestellt werden müssen, um gute Ergebnisse zu erhalten. In diesem Abschnitt wird versucht, dieses Problem zu vereinfachen.

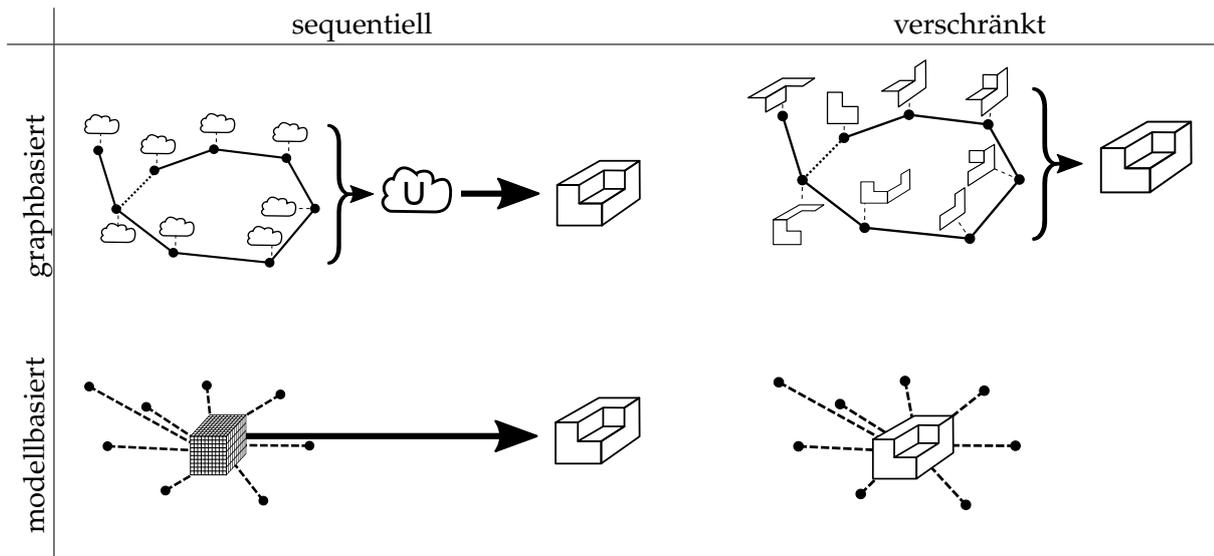


Abbildung 3.1: Schematische Darstellung der vier Kombinationsmöglichkeiten von SLAM-Systemen mit hochwertigen kontinuierlichen Volumenmodellen als Ausgabe. Graphbasierte Systeme speichern die Einzelbilder als Graph, um sie am Ende zu fusionieren. Modellbasierte Systeme fusionieren Einzelbilder sofort in ein globales Modell. Bei sequentieller Ausführung werden die DSR-Methoden nach einer Rekonstruktion mit SLAM-Verfahren angewandt, um ein CAD-Modell zu erzeugen. Bei der verschränkten Variante wird versucht, das Ausgabemodell gleich während des SLAM als Datenstruktur zu nutzen.

3.1 Kombination von SLAM und DSR

Das Hauptaugenmerk dieser Arbeit liegt in der Kombination der einfachen Handhabung durch handgehaltene Kameras mit hochwertigen Ausgabemodellen, um die Lücke, die in den veröffentlichten Verfahren vorherrscht, zu überwinden. Da SLAM-Systeme eine einfache Handhabung ermöglichen und DSR-Systeme eine Rekonstruktion von CAD-Modellen, ist eine Möglichkeit die Hintereinanderausführung beider Methoden. Die Ausgabe des SLAM-Systems ist also die Eingabe des DSR-Systems (*sequentielle* Ausführung). Die zweite Variante ist die Verflechtung von DSR und SLAM, sodass während der Aufnahme bereits kontinuierliche Volumenmodelle rekonstruiert werden (*verschränkte* Ausführung). Beide Varianten können wiederum mit graph- oder modellbasierten SLAM-Methoden realisiert werden (Abbildung 3.1). Im Folgenden werden nun alle vier Möglichkeiten bewertet und der Lösungsansatz für das in dieser Arbeit betrachtete System begründet. Anschließend werden Schlussfolgerungen gezogen, die in zu lösenden Teilproblemen resultieren.

3.1.1 Bewertung

Sequentiell und graphbasiert

Wird ein graphbasiertes SLAM-System und ein DSR-Verfahren hintereinander ausgeführt, so müssen dabei mehrere Probleme gelöst werden: Die Messabweichungen der Gesamtpunktwolke als Zwischenergebnis können je nach verwendeter Kamera ziemlich groß sein. Insbesondere kann die Punktdichte sehr inhomogen sein, da nicht alle Stellen gleich häufig betrachtet worden sein müssen. Dies stellt ein Problem für einige DSR-Algorithmen dar. Zwischenergebnisse während der Rekonstruktion sind nicht verfügbar. Zudem müssen alle Rohdaten bis zum Ende vorgehalten werden, was hohe Speicheranforderungen zur Folge hat. Ein daraus resultierender Vorteil ist jedoch, dass alle zur Verfügung stehenden Daten auch für die Anpassung des geometrischen Modells genutzt werden können.

Sequentiell und modellbasiert

Wird ein modellbasiertes System anstelle eines graphbasierten verwendet, ist weniger Speicher nötig, da die Rohdaten sofort in das globale Modell integriert werden. Das Ergebnis, das als Eingabe für den DSR-Schritt dient, besitzt somit weniger Rauschen und auch eine inhomogene Punktdichte wird verhindert. Da das globale Modell meist im GPU-Speicher vorliegen muss, sind die Aufnahmevermögen beschränkt. Durch die Verwendung von Surfel-Modellen anstatt Voxeln wird diese Einschränkung jedoch schon deutlich reduziert. Wie auch beim vorhergehenden Ansatz sind keine Zwischenergebnisse während der Aufnahme in Form von kontinuierlichen Volumenmodellen möglich, da erst am Ende abstrahiert wird.

Verschränkt und graphbasiert

Man erkennt, dass bei sequentiellen Ansätzen die Speicheranforderung zum Problem werden kann. Kontinuierliche Volumenmodelle stellen durch ihre Abstraktion auf analytische Oberflächen das Modell äußerst kompakt dar, wenn auch komplexer als Punktwolken, Dreiecksnetze, Surfel- oder Voxelmodelle. Beim verschränkten Ansatz kann deshalb direkt das gewünschte Ausgabemodell als Grundlage im SLAM-Teil verwendet werden. Dies spart Speicher und macht den zweiten Schritt, die Umwandlung eines globalen Zwischenmodells in das Zielmodell überflüssig.

Beim graphbasierten SLAM bedeutet dies, dass in jedem Knoten des Graphen eine Umwandlung der Rohdaten in ein partielles Modell erfolgt. Am Ende werden nach der Optimierung des Graphen alle Einzelteile zum Endergebnis fusioniert, was zur Folge hat, dass während der Aufnahme noch keine Zwischenergebnisse vorhanden sind.

Verschränkt und modellbasiert

Bei der verschränkten modellbasierten Variante wird das gewünschte Ausgabemodell direkt als globales Modell verwendet. Dies ermöglicht, dass zu jeder Zeit bereits ein Zwischenstand abgefragt werden kann. Der Speicheraufwand ist nochmals geringer, da nur ein globales Modell vorgehalten werden muss. Die höhere Komplexität des Modells hat allerdings zur Folge, dass die Rekonstruktions- und Fusionsschritte während des SLAMs deutlich aufwändiger sind, da die Datenstruktur stets konsistent gehalten werden muss.

3.1.2 Schlussfolgerungen

Im Hinblick auf die in dieser Arbeit betrachtete Aufgabenstellung (siehe Abschnitt 1.4.1) erscheint eine sequentielle Kombination aus zwei Gründen als ungeeignet. Zum einen ist keine inkrementelle Rekonstruktion eines kontinuierlichen Volumenmodells möglich, da während der Erfassung auf einer anderen Repräsentation gearbeitet wird. Zum anderen sind bestehende DSR-Methoden sensitiv gegenüber Messabweichungen und meist nicht online-fähig.

Die verschränkte Variante kann durch Verflechtung beider Gebiete die jeweiligen Vorteile kombinieren, erfordert jedoch auch neue Ansätze, damit eine Online-Fähigkeit gewährleistet bleibt. Im Vergleich von graphbasiertem und modellbasiertem SLAM besitzt ein globales Modell klar den Vorteil von geringeren Speicheranforderungen sowie der Verfügbarkeit von Zwischenergebnissen durch die inkrementelle Fusion. Es wird sich deshalb dafür entschieden, in dieser Arbeit die vierte Variante zu verfolgen. Im Vergleich zu den anderen Varianten muss hier allerdings das globale Modell stets konsistent gehalten werden, was umso aufwändiger ist, je mehr und höherwertige Informationen das Modell besitzt.

Folgende Schlussfolgerungen können aus dieser Kombination gezogen werden:

1. Die Verwendung der Ausgabemodellart, also eines kontinuierlichen Volumenmodells, als globales Modell für SLAM führt dazu, dass auch noch unvollständige Modelle dargestellt werden können müssen, wenn eine Szene noch nicht komplett erfasst wurde. Dies führt zur Notwendigkeit von partiellen Volumenmodellen. Diese Art von Modellen und die benötigten Eigenschaften werden näher in Abschnitt 3.2 untersucht.
2. Wie bereits erwähnt, ist die Konsistenthaltung dieses Modells aufwändiger als bei anderen modellbasierten Systemen. Kapitel 4 befasst sich deshalb ausschließlich mit der Rekonstruktion von partiellen Modellen aus einzelnen Punktwolken. In Kapitel 5 wird anschließend untersucht, wie sich mehrere partielle Modelle in das globale Modell fusionieren lassen (unter der Annahme einer bekannten Kamerapose).
3. Da in dieser Art von Modell bereits höherwertige Informationen der Oberfläche kodiert sind, kann dies das Problem der Registrierung, also der Berechnung der relativen Transformation zwischen zwei Modellen, vereinfachen. Dies wird in Kapitel 6 näher betrachtet.
4. Da das globale Modell dem Endergebnis entspricht, kann die Güte des Modells direkt während der Erfassung ermittelt werden. Kapitel 7 befasst sich deshalb mit der Unterstützung des Anwenders durch Rückmeldungen mit dem Ziel, ein vollständiges Modell ohne Löcher zu erhalten.

3.2 Partielle Modelle

Durch Verwendung eines verschränkten modellbasierten Ansatzes, ist es nötig partielle Modelle darstellen zu können. Ein partielles Modell ist ein kontinuierliches Volumenmodell, das (noch) kein vollständiges Volumen umschließt, aber ansonsten alle Eigenschaften eines Volumenmodells erfüllt. Diese sind eine eindeutige Orientierung der Oberflächen („davor“ und „dahinter“ ist eindeutig), Abbildung der Topologie (Nachbarschaftsbeziehungen zwischen den geometrischen Elementen) und eine algebraische Repräsentation von Oberflächenstücken.

Die folgenden Beschreibungen und Definitionen von Modellen beschränken sich wie in der Aufgabenstellung (Abschnitt 1.4) genannt auf planare Oberflächentypen. Eine Erweiterung der hier vorgestellten partiellen Modelle auf höhergradige Oberflächentypen ist jedoch möglich.

Im Folgenden werden nun geeignete Datenstrukturen untersucht, mit denen sich partielle Modelle darstellen lassen (Abschnitt 3.2.1), Eigenschaften von partiellen Modellen untersucht (Abschnitt 3.2.2) sowie eine formale Beschreibung der Repräsentation vorgestellt (Abschnitt 3.2.3).

3.2.1 Repräsentation partieller Modelle

Es stellt sich die Frage, wie partielle Modelle repräsentiert werden können. In diesem Abschnitt werden daher bekannte Datenstrukturen von kontinuierlichen Volumenmodellen auf ihre Eignung zur Darstellung von partiellen Modellen untersucht. In den letzten Jahrzehnten haben sich zwei Konzepte zur Darstellung für Volumenmodelle durchgesetzt: Die Modellierung durch boolesche Kombination von Primitiven (*constructive solid geometry*) sowie durch Modellierung der Begrenzungsflächen (*boundary representation*) [Foley96].

Werden komplexe Modelle durch geometrische Primitive, wie Kugeln, Zylinder oder Quader dargestellt, die mittels boolescher Operationen wie Schnitt, Vereinigung und Differenz kombiniert werden, so spricht man von CSG-Modellen (*constructive solid geometry*) [Lee99]. Die Reihenfolge der Operationen wird durch einen Baum dargestellt, dessen Blätter den Primitiven entsprechen (Abbildung 3.2). Die Darstellung ist meist nicht eindeutig, die Extraktion eines Oberflächenmodells - beispielsweise zur Anzeige - ist oftmals rechenaufwändig.

Eine Verwendung als partielles Modell ist nur schwer möglich, da stets Volumenelemente betrachtet werden. Modelliert man ein Objekt, das aus einer einzelnen Ansicht betrachtet wird, indem man das Volumen zwischen Kamera und Oberfläche durch Primitive beschreibt und es anschließend von einer gefüllten Gesamtheit abzieht, so erhält man eine Beschreibung des Raumes, der noch nicht erfasst wurde, also unbekannt ist (Abbildung 3.3c). Durch Verschnei-

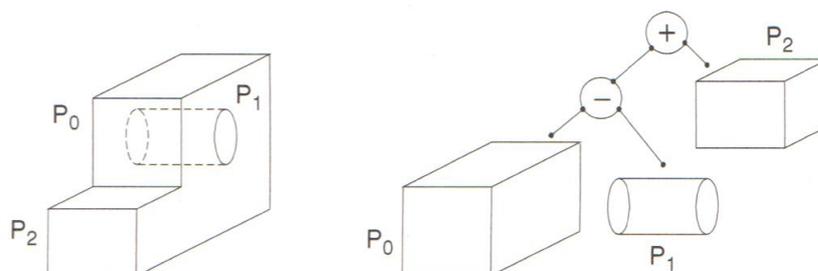


Abbildung 3.2: CSG-Datenstruktur: Ein Objekt wird durch mehrere boolesche Operationen auf geometrischen Primitiven erzeugt (aus [Lee99]).

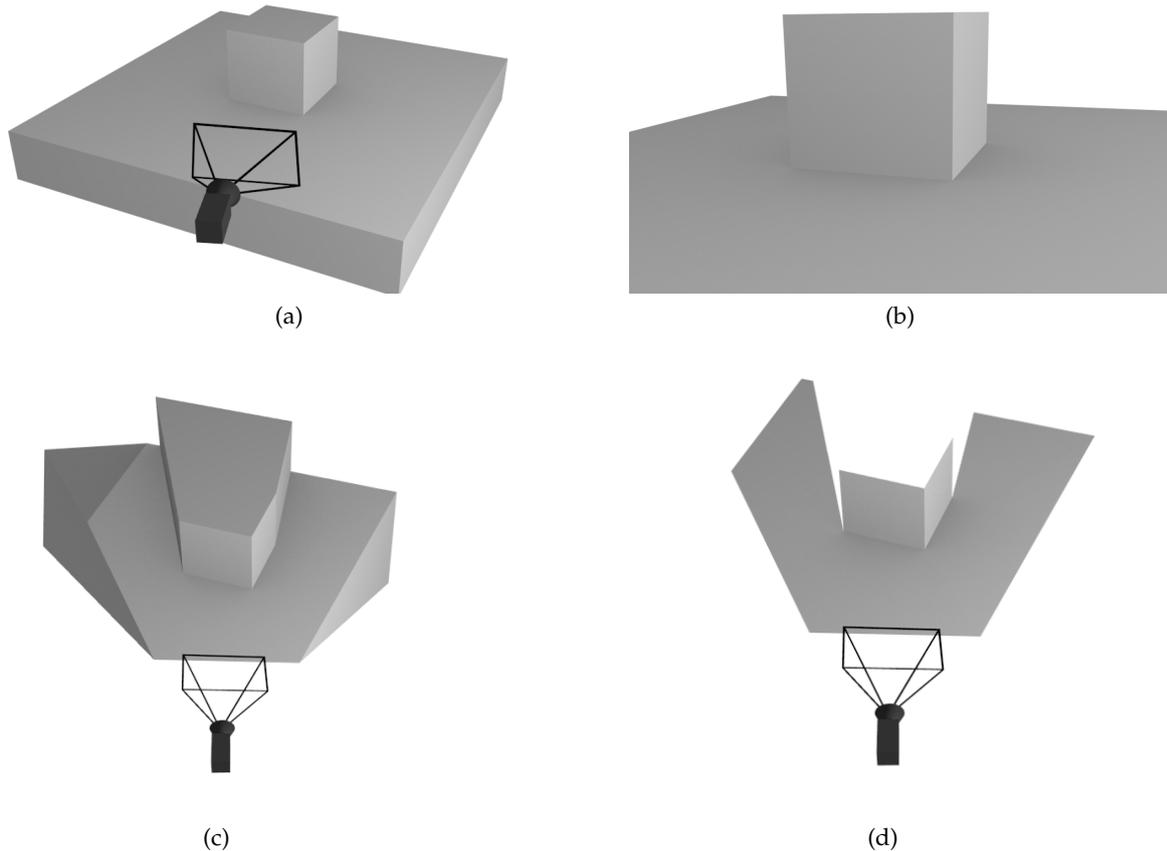


Abbildung 3.3: (a) Szene, die von einer Kamera aus einer bestimmten Ansicht erfasst wird, (b) Szene aus Sicht der Kamera (ein Teil der Oberflächen ist aufgrund des Blickwinkels verdeckt), (c) Repräsentation als CSG-Modell, (d) Repräsentation als Begrenzungsflächenmodell

ding mehrerer solcher Modelle aus verschiedenen Perspektiven erhält man bei ausreichender Anzahl an Bildern ein (gegebenenfalls bis auf die Standfläche) vollständiges Modell des Objekts. Ein einzelnes Modell besitzt jedoch den Nachteil, dass auch unbekanntes Volumen mit modelliert wird.

Begrenzungsflächenmodelle (*boundary representation models, B-Reps*) repräsentieren ein Volumen durch ihre begrenzende Oberfläche, definiert durch Knoten (*vertices*), Kanten (*edges*) und Flächen (*faces*). Flächen können dabei planar, dargestellt durch eine Ebenengleichung, oder gekrümmt sein, repräsentiert durch implizite Gleichungen oder parametrisierte Gleichungen, wie zum Beispiel B-Spline-Flächen. Je höher der Grad der Fläche, desto genauer kann ein Objekt dargestellt werden. Der Übergang von zwei Flächen wird durch eine Kante modelliert, mehrere Kanten treffen sich an Knoten. Neben der Geometrie stellen die Modelle also auch Topologieinformationen zu Verfügung.

Es existieren unterschiedliche Datenstrukturen für Begrenzungsflächenmodelle. Sie unterscheiden sich in der Art und Anzahl von Verzeigerungen zwischen den Knoten, Kanten und Flächen des Modells. Redundante Querverbindungen erhöhen den Aufwand, die Datenstruktur konsistent zu halten, beschleunigen und erleichtern aber die Abfrage von topologischen Beziehungen. Am häufigsten kommt die sogenannte *winged-edge*- [Baumgart72] und die *half-edge*-Datenstruktur [Mäntylä87] zum Einsatz. In Abbildung 3.4 sind beide schematisch dargestellt. Die zusätzliche Verwendung von Begrenzungen (*boundary loops*) als Zwischenschicht zwischen Kanten und Flächen, dient dazu, Flächen mit Löchern repräsentieren zu können,

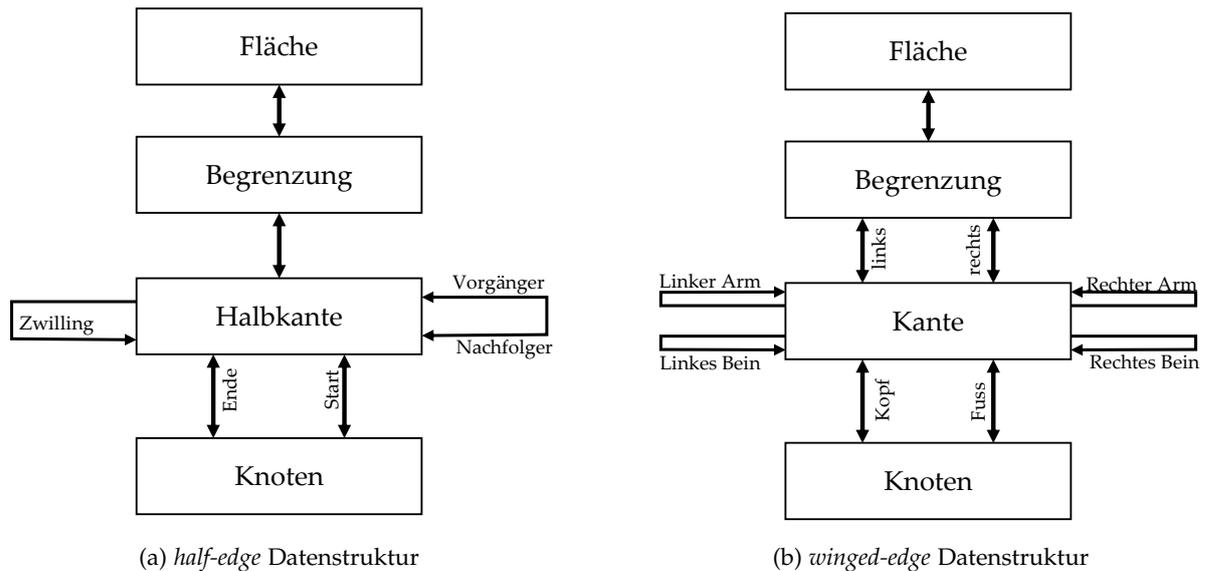


Abbildung 3.4: Schematische Darstellung der *half-edge* und *winged-edge* Datenstruktur

ohne Hilfskanten verwenden zu müssen. Der Hauptunterschied zwischen beiden Varianten besteht in der Modellierung der Kanten. Bei der *winged-edge*-Datenstruktur wird zwischen rechtem und linkem Teil unterschieden. Jeder Teil enthält eine Verbindung zu Vorgänger- und Nachfolgerkante, sowie der zugehörigen Begrenzung. Die Richtung einer Kante ist einerseits relevant für die Definition von links und rechts, aber andererseits nicht festgelegt. Bei der *half-edge*-Datenstruktur wird jede Kante als Paar von Halbkanten (*half-edges*) modelliert. Von außen betrachtet läuft eine äußere Begrenzung stets gegen den Uhrzeigersinn, Begrenzungen von Löchern im Uhrzeigersinn. Dadurch liegt das Innere einer Fläche stets links von einer Halbkante, wenn diese von außen betrachtet wird (Abbildung 3.5a).

Für eine Verwendung als partielles Modell eignen sich Begrenzungsflächenmodelle sehr gut, insbesondere die *half-edge* Datenstruktur, da Halbkanten stets nur zu einer einzigen Fläche gehören. Indem man erlaubt, dass eine Halbkante keine Zwillingkante besitzen muss, ist es möglich, dass eine Fläche keine Nachbarfläche besitzt. Somit ist die Repräsentation partieller Modelle leicht möglich (Abbildung 3.3d). Aus diesem Grund werden in dieser Arbeit Begrenzungsflächenmodelle (B-Reps) mit einer *half-edge*-Datenstruktur als partielle Modelle verwendet.

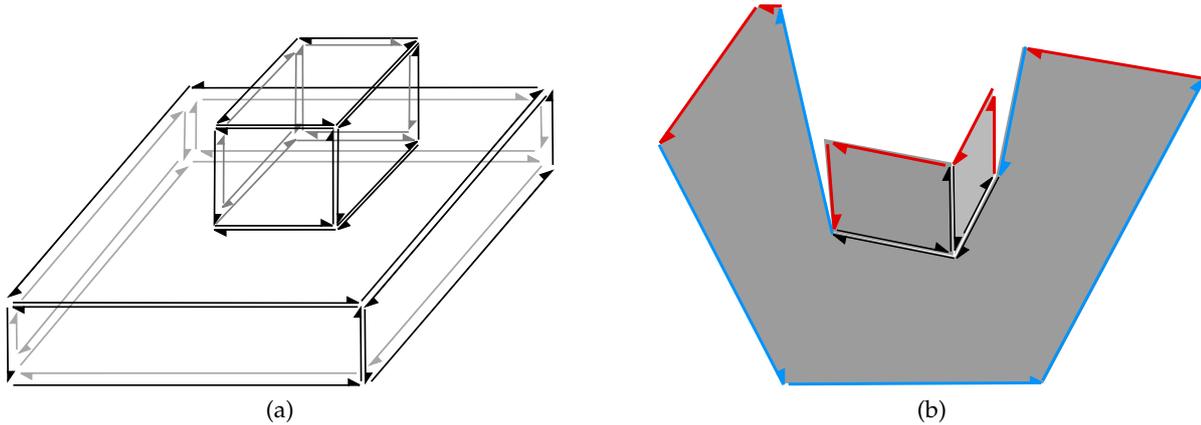


Abbildung 3.5: (a) Das Objekt aus Abbildung 3.3a als vollständiges *half-edge* B-Rep Modell. Die obere horizontale Fläche des unteren Quaders besitzt zwei Begrenzungen: Die äußere Begrenzung sowie eine innere, mit der der obere Quader verknüpft ist. (b) Das partielle Modell aus Abbildung 3.3d mit eingezeichneten Halbkanten unterschiedlichen Typs: Inzidente Kanten (schwarz), pseudo-inzidente Kanten (rot) und virtuelle Kanten (blau)

3.2.2 Eigenschaften partieller Modelle

Ein Beispiel für ein partielles Modell ist in Abbildung 3.5b zu sehen. Man erkennt, dass topologisch zu vollständigen Volumenmodellen ein Unterschied herrscht: Es können nun Kanten existieren, die nicht wie bisher den Übergang zwischen zwei Oberflächenstücken bilden, sondern eine Fläche ausschließlich begrenzen. Daher lassen sich in partiellen Modellen zwei Kantentypen unterscheiden:

1. **Inzidente Kanten** besitzen stets zwei benachbarte Flächen. Geometrisch beschreiben sie eine Unstetigkeit der Oberflächennormalen. Ein vollständiges Volumenmodell besitzt ausschließlich inzidente Kanten.
2. **Terminale Kanten** begrenzen eine Oberfläche, ohne eine Nachbarfläche zu besitzen. An dieser Stelle im Modell ist der weitere Verlauf der Oberfläche unbekannt.

Werden partielle Modelle aus Tiefenbildern erzeugt, so treten terminale Kanten am Bildrand und bei Selbstverdeckungen auf (vgl. Abbildung 3.3b). Bei Selbstverdeckungen kann zwischen terminalen Kanten an verdeckenden und an verdeckten Oberflächen unterschieden werden. Bei verdeckenden Oberflächen verläuft eine terminale Kante an derselben Position wie eine inzidente Kante, wenn das Modell vollständig wäre. An terminalen Kanten in verdeckten Oberflächen kann (und wird vermutlich) die Oberfläche weiterlaufen, sie wurde nur noch nicht erfasst. Somit lassen sich terminale Kanten weiter unterteilen:

- 2.1 **Pseudo-inzidente Kanten** begrenzen eine Oberfläche und beschreiben eine Unstetigkeit in der Oberflächennormalen. Eine zweite adjazente Fläche ist jedoch nicht vorhanden.
- 2.2 **Virtuelle Kanten** sind in der Realität nicht vorhanden. Die Fläche setzt sich fort, wurde jedoch noch nicht rekonstruiert.

In Abbildung 3.5b sind die unterschiedlichen Arten von Kanten farblich dargestellt.

3.2.3 Formale Beschreibung eines partiellen Begrenzungsflächenmodells

In diesem Abschnitt wird ein partielles Begrenzungsflächenmodell (*B-Rep*) formal definiert, um für die folgenden Kapitel eine eindeutige Notation verwenden zu können. Die *half-edge*-Datenstruktur besitzt dabei viele Verzeigerungen, die einen schnellen Zugriff erlauben. Zur formalen Beschreibung werden im Folgenden deshalb innerhalb einer Definition zusätzlich Funktionen (dargestellt in *Festbreitenschrift*) definiert, die diese Verzeigerungen abbilden.

Die Definitionen sind in drei Gruppen unterteilt: Geometrische, strukturelle und semantische. Zusätzlich werden Definitionen für Paare von B-Reps eingeführt, um die Notation in Kapitel 5 (Fusion von B-Reps) zu vereinfachen.

Geometrische Definitionen

Da planare Modelle betrachtet werden, ist es nötig, eine eindeutige Beschreibung von Ebenen zu verwenden:

Definition 3.1

Die **Ebenenkoeffizienten** $\vec{\pi} := \begin{pmatrix} \vec{n} \\ d \end{pmatrix}$ mit $\vec{n} \in \mathbb{R}^3, d \in \mathbb{R}, |\vec{n}| = 1$ beschreiben eine (gerichtete) Ebene im dreidimensionalen Raum mit Normalenvektor \vec{n} , die um $-d$ in Normalenrichtung gegenüber dem Ursprung verschoben ist.

Definition 3.2

Die **Ebene** Π ist die Menge aller Punkte $\vec{P} \in \mathbb{R}^3$, die den Ebenenkoeffizienten $\vec{\pi}$ genügen:

$$\Pi := \{ \vec{P} \in \mathbb{R}^3 : \begin{pmatrix} \vec{P} \\ 1 \end{pmatrix} \circ \vec{\pi} = 0 \}.$$

Definition 3.3

Der **Halbraum** Π^+ ist die Menge aller Punkte $\vec{P} \in \mathbb{R}^3$, die auf der Seite der Ebene liegen, in die die Normale zeigt:

$$\Pi^+ := \{ \vec{P} \in \mathbb{R}^3 : \begin{pmatrix} \vec{P} \\ 1 \end{pmatrix} \circ \vec{\pi} > 0 \}.$$

Definition 3.4

Der **Halbraum** Π^- ist die Menge aller Punkte $\vec{P} \in \mathbb{R}^3$, die auf der Seite der Ebene liegen, in die die Normale nicht zeigt:

$$\Pi^- := \{ \vec{P} \in \mathbb{R}^3 : \begin{pmatrix} \vec{P} \\ 1 \end{pmatrix} \circ \vec{\pi} < 0 \}.$$

Strukturelle Definition

Im Folgenden wird nun ein Begrenzungsflächenmodell (B-Rep) unter Nutzung der *half-edge* Datenstruktur mit allen seinen Elementen definiert:

Definition 3.5

Die Menge \mathcal{V} ist eine **Menge von Knoten**: $\mathcal{V} := \{v_1, \dots, v_{|\mathcal{V}|}\}$.

Ein **Knoten** v ist ein Element, dem ein Punkt im Raum $\vec{P}_v \in \mathbb{R}^3$ zugeordnet ist.

Folgende Funktion sei zusätzlich definiert:

$$\text{ort}: \mathcal{V} \rightarrow \mathbb{R}^3, \quad v \mapsto \vec{P}_v \quad (\text{Zuordnungsfunktion})$$

Definition 3.6

Die Menge \mathcal{H} ist eine **Menge von Halbkanten**: $\mathcal{H} := \{h_1, \dots, h_{|\mathcal{H}|}\}$.

Eine **Halbkante** h ist eine gerichtete Kante von einem Startknoten $v_s \in \mathcal{V}$ zu einem Endknoten $v_e \in \mathcal{V}$.

Folgende Funktionen seien zusätzlich definiert:

$$\begin{aligned} \text{start}: \quad \mathcal{H} &\rightarrow \mathcal{V}, & h &\mapsto v_s && (\text{Startknoten}) \\ \text{ende}: \quad \mathcal{H} &\rightarrow \mathcal{V}, & h &\mapsto v_e && (\text{Endknoten}) \\ \text{richtung}: \quad \mathcal{H} &\rightarrow \mathbb{R}^3, & h &\mapsto \frac{\text{ort}(v_e) - \text{ort}(v_s)}{|\text{ort}(v_e) - \text{ort}(v_s)|} && (\text{normierte Richtung}) \\ \text{strecke}: \quad \mathcal{H} &\rightarrow 2^{\mathbb{R}^3}, & h &\mapsto \{\vec{P} \in \mathbb{R}^3: \vec{P} = \lambda \cdot \text{ort}(v_s) + (1 - \lambda) \cdot \text{ort}(v_e), \quad 0 < \lambda < 1\} && (\text{ohne Endpunkte}) \\ \text{strecke}^*: \quad \mathcal{H} &\rightarrow 2^{\mathbb{R}^3}, & h &\mapsto \{\vec{P} \in \mathbb{R}^3: \vec{P} = \lambda \cdot \text{ort}(v_s) + (1 - \lambda) \cdot \text{ort}(v_e), \quad 0 \leq \lambda \leq 1\} && (\text{mit Endpunkten}) \\ \text{eingehend}: \quad \mathcal{V} &\rightarrow 2^{\mathcal{H}}, & v &\mapsto \{h \in \mathcal{H}: \text{ende}(h) = v\} && (\text{eingehende Kanten}) \\ \text{ausgehend}: \quad \mathcal{V} &\rightarrow 2^{\mathcal{H}}, & v &\mapsto \{h \in \mathcal{H}: \text{start}(h) = v\} && (\text{ausgehende Kanten}) \end{aligned}$$

Die Schreibweise 2^X meint dabei die Potenzmenge und soll die Unterscheidung zwischen einzelnen Elementen und einer Menge an Elementen verdeutlichen. Die Funktion *ort* bildet beispielsweise auf einen einzelnen Punkt im Raum ab, die Funktion *strecke* auf eine Menge an Punkten.

Da Flächen mit Löchern erlaubt sind, ist zuerst eine Definition von Begrenzungen nötig, bevor Flächen definiert werden können.

Definition 3.7

Die Menge \mathcal{B} ist eine **Menge von Begrenzungen**: $\mathcal{B} := \{b_1, \dots, b_{|\mathcal{B}|}\}$.

Eine **Begrenzung** b ist ein Zyklus von Halbkanten $b := (h_0, \dots, h_{|b|-1})$, $h_i \in \mathcal{H}$, die alle innerhalb einer Ebene Π_b mit Ebenenkoeffizienten π_b liegen, und für die die folgenden Eigenschaften gelten müssen:

Sei $0 \leq i, j < |b|$, $i \neq j$.

- $|b| > 2$ (mind. 3 Halbkanten)
- $h_i \neq h_j$ (keine doppelten Halbkanten)
- $\text{ende}(h_i) = \text{start}(h_{(i+1) \bmod |b|})$ (zusammenhängend)
- $\text{strecke}(h_i) \subset \Pi_b$ (in einer Ebene)
- $\text{strecke}(h_i) \cap \text{strecke}(h_j) = \emptyset$ (sich nicht schneidend)
- $\text{ort}(\text{start}(h_i)) \neq \text{ort}(\text{start}(h_j))$ (keine gemeinsamen Eckpunkte)

Eine Halbkante ist stets Teil einer einzigen Begrenzung:

Sei $1 \leq k, l \leq |\mathcal{B}|$, $k \neq l$.

- $\nexists h \in \mathcal{H} : h \in b_k \wedge h \in b_l$ (eindeutige Zuordnung)
- $\bigcup_k b_k = \mathcal{H}$ (keine Halbkante ohne Begrenzung)

Folgende Funktionen seien zusätzlich definiert:

- nach: $\mathcal{H} \rightarrow \mathcal{H}$, $h_i \mapsto h_{(i+1) \bmod |b|}$ (Nachfolger)
- vor: $\mathcal{H} \rightarrow \mathcal{H}$, $h_i \mapsto h_{(i-1) \bmod |b|}$ (Vorgänger)
- umrandung: $\mathcal{B} \rightarrow 2^{\mathbb{R}^3}$, $b \mapsto \bigcup_{0 \leq i < |b|} \text{strecke}^*(h_i)$ (Streckenzug)
- inneres: $\mathcal{B} \rightarrow 2^{\mathbb{R}^3}$, $b \mapsto \{\vec{p} \in \Pi_b : \text{innerhalb von umrandung}(b)\}$ (Inneres)
- inneres*: $\mathcal{B} \rightarrow 2^{\mathbb{R}^3}$, $b \mapsto \text{inneres}(b) \cup \text{umrandung}(b)$ (Inneres + Rand)
- linksDrehend: $\mathcal{B} \rightarrow \{w, f\}$, $b \mapsto \text{"von } \Pi_b^+ \text{ aus betrachtet linksdrehend"}$
- begrenzung: $\mathcal{H} \rightarrow \mathcal{B}$, $h_i \mapsto b : h_i \in b$ (Begrenzung von h)
- ebene*: $\mathcal{B} \rightarrow \mathbb{R}^4$, $b \mapsto \pi_b$ (Ebenenkoeffizienten)

Da eine Ebene orientiert ist, das heißt, ein „Vorne“ und „Hinten“ anhand der Normalenrichtung existiert, kann für Begrenzungen ein Umlaufsinn angegeben werden. Die Funktion linksDrehend mit booleschem Wertebereich $\{w, f\}$ beschreibt diesen Umlaufsinn, indem die Umlaufrichtung vom positiven Halbraum aus betrachtet wird.

Da eine Begrenzung eine geschlossene Kette aus Halbkanten innerhalb einer Ebene darstellt, schließt sie eine Menge an Punkten ein. Dies wird durch Funktion inneres beschrieben. Die Funktion inneres* wird für alle Punkte, die im Inneren oder auf den umrandenden Halbkanten liegen, verwendet.

Definition 3.8

Die Menge \mathcal{F} ist eine **Menge von Flächen** $\mathcal{F} := \{f_1, \dots, f_{|\mathcal{F}|}\}$.

Eine **Fläche** ist ein Tupel von Begrenzungen $f := (b_0, \dots, b_{|f|-1})$, bestehend aus einer äußeren Begrenzung $b_0 \in \mathcal{B}$ sowie optional mehreren inneren Begrenzungen (Löcher) $b_1, \dots, b_{|f|-1} \in \mathcal{B}$, die alle in einer Ebene π_f liegen, und für die alle folgenden Eigenschaften gelten müssen:

Sei $0 \leq i < |f|$, $1 \leq j, k < |f|$, $j \neq k$.

- $|f| \geq 1$ (mind. eine Begrenzung)
- $\text{ebene}^*(b_i) = \pi_f$ (in derselben Ebene)
- $\text{linksDrehend}(b_0) = w$ (äußere Begr. linksdrehend)
- $\text{linksDrehend}(b_j) = f$ (Löcher rechtsdrehend)
- $\text{inneres}^*(b_j) \cap \text{inneres}^*(b_k) = \emptyset$ (Löcher disjunkt)
- $\text{inneres}^*(b_j) \subset \text{inneres}(b_0)$ (Löcher innerhalb)

Eine Begrenzung ist stets Teil einer einzigen Fläche:

Sei $1 \leq l, m \leq |\mathcal{F}|, l \neq m$.

- $\nexists b \in \mathcal{B} : b \in f_l \wedge b \in f_m$ (eindeutige Zuordnung)
- $\bigcup_l f_l = \mathcal{F}$ (keine Begrenzungen ohne Fläche)

Folgende Funktionen seien zusätzlich definiert:

- $\text{rand}: \mathcal{F} \rightarrow 2^{\mathbb{R}^3}, f \mapsto \bigcup_{0 \leq i < |f|} \text{umrandung}(b_i)$ (Rand)
- $\text{inhalt}: \mathcal{F} \rightarrow 2^{\mathbb{R}^3}, f \mapsto \text{inneres}(b_0) \setminus \bigcup_{1 \leq j < |f|} \text{inneres}^*(b_j)$ (Inneres ohne Rand)
- $\text{inhalt}^*: \mathcal{F} \rightarrow 2^{\mathbb{R}^3}, f \mapsto \text{inneres}^*(b_0) \setminus \bigcup_{1 \leq j < |f|} \text{inneres}(b_j)$ (Inneres mit Rand)
- $\text{ebene}: \mathcal{F} \rightarrow \mathbb{R}^4, f \mapsto \text{ebene}^*(b_0)$ (Ebenenkoeffizienten)
- $\text{flaeche}^*: \mathcal{B} \rightarrow \mathcal{F}, b_i \mapsto f$ (Fläche von b_i)
- $\text{flaeche}: \mathcal{H} \rightarrow \mathcal{F}, h \mapsto \text{flaeche}^*(\text{begrenzung}(h))$

Geometrisch ist eine Fläche somit ein nicht notwendigerweise konvexes Polygon mit Löchern. Durch das Erzwingen einer linksdrehenden äußeren und rechtsdrehenden inneren Begrenzung besitzt eine Fläche die Eigenschaft, dass das Innere der Fläche stets links von allen seinen begrenzenden Halbkanten liegt, wenn es aus dem positiven Halbraum der Ebene betrachtet wird.

Die bisherigen Definition reichen aus, um einzelne unzusammenhängende Flächen darzustellen. Um eine vollständige Oberfläche zu erreichen, müssen Flächen noch mit ihren Nachbarflächen verknüpft werden. Dies geschieht durch Definition einer Zwillingsbeziehung auf Halbkanten. Da hier im Speziellen auch partielle, also nicht vollständige Modelle betrachtet werden, muss die Zwillingskantenbeziehung nicht notwendigerweise für alle Halbkanten existieren.

Definition 3.9

Eine Halbkante h_z heißt **Zwillingskante** zu einer Halbkante h mit

$$\text{zwilling}(h) := \begin{cases} h_z & \text{falls existent} \\ \text{undefiniert} & \text{sonst} \end{cases}$$

Dabei muss gelten:

$$\begin{aligned} \text{start}(h) &= \text{ende}(h_z) && \text{(Start und Endknoten vertauscht)} \\ \text{ende}(h) &= \text{start}(h_z) \end{aligned}$$

Da ausschließlich mannigfaltige Modelle betrachtet werden, ist die Zwillingsbeziehung, falls existent, eindeutig und bijektiv:

$$\text{zwilling}(\text{zwilling}(h)) = h$$

Folgende Funktionen seien zusätzlich definiert:

$$\begin{aligned} \text{benachbart}: \quad \mathcal{F} \times \mathcal{F} &\rightarrow \{\text{w}, \text{f}\}, && (f_1, f_2) \mapsto \exists h \in \mathcal{H} : \\ &&& \text{flaeche}(h) = f_1 \wedge \text{flaeche}(\text{zwilling}(h)) = f_2 \\ \text{nachbarn}: \quad \mathcal{F} &\rightarrow 2^{\mathcal{F}}, && f \mapsto \{f' \in \mathcal{F} : \text{benachbart}(f, f')\} \end{aligned}$$

Da das Innere einer Fläche stets links von einer Halbkante liegt, ist auch sichergestellt, dass benachbarte Flächen stets eine konsistente Ausrichtung ihrer Normalen besitzen. Kanten mit Zwilling sind also inzidente Kanten, Kanten ohne Zwilling terminale Kanten (vgl. Abschnitt 3.2.2).

Mit den bisher definierten Elementen lässt sich nun ein partielles B-Rep zusammensetzen:

Definition 3.10

Ein **B-Rep** besteht aus Knoten, Halbkanten, Begrenzungen und Flächen:

$$\text{B-Rep} := (\mathcal{V}, \mathcal{H}, \mathcal{B}, \mathcal{F})$$

Ein B-Rep ist **vollständig**, wenn alle Halbkanten eine Zwillingskante besitzen.

Semantische Definitionen

Die Verwendung von Knoten und Halbkanten in der Datenstruktur erlauben die Repräsentation von partiellen Modellen. Wäre ein Modell stets vollständig, so würde man eher von Kanten und Ecken anstelle von Halbkanten und Knoten sprechen. Dieser Bezug zu vollständigen Modellen wird in den folgenden zwei Definitionen ausgedrückt:

Definition 3.11

Eine Halbkante h heißt **Kante**, wenn h eine Zwillingskante besitzt.
Die Menge aller Kanten heißt \mathcal{K} . Es gilt:

$$\begin{aligned} \exists h_z : h_z = \text{zwilling}(h) \\ \mathcal{K} \subseteq \mathcal{H} \end{aligned}$$

Definition 3.12

Ein Knoten v heißt **Ecke** e , wenn er mindestens zwei adjazente Kanten hat.
Die Menge aller Ecken heißt \mathcal{E} . Es gilt:

$$\begin{aligned} |\text{eingehend}(e) \cap \mathcal{K}| \geq 2 \\ \mathcal{E} \subseteq \mathcal{V} \end{aligned}$$

Folgende Funktion sei zusätzlich definiert:

$$\text{kanten: } \mathcal{E} \rightarrow 2^{\mathcal{K}}, \quad v \mapsto \text{eingehend}(v) \cap \mathcal{K} \quad (\text{adjazente Kanten})$$

Da durch zwei adjazente Kanten mindestens drei Flächen beteiligt sind, ist der Ort der Ecke eindeutig, nämlich am Schnittpunkt der Flächen. Da Flächen, Kanten und Ecken auch bei physikalischen Objekten wahrnehmbar sind, werden diese Elemente besonders bezeichnet:

Definition 3.13

Alle Elemente von \mathcal{E} , \mathcal{K} und \mathcal{F} werden als **physikalische Elemente** bezeichnet.

Definitionen für Paare von B-Reps

Sobald mehr als ein partielles Modell betrachtet wird, stellt sich die Frage nach korrespondierenden Elementen. Dies ist insbesondere bei der Fusion (Kapitel 5) und der Registrierung von partiellen Modellen (Kapitel 6) relevant. Deshalb werden nun Element-Korrespondenzen definiert. Die zwei beteiligten (partiellen) B-Reps sind dabei mit $A = (\mathcal{V}_A, \mathcal{H}_A, \mathcal{B}_A, \mathcal{F}_A)$ beziehungsweise $B = (\mathcal{V}_B, \mathcal{H}_B, \mathcal{B}_B, \mathcal{F}_B)$ bezeichnet.

Definition 3.14

Zwei Flächen $f_A \in \mathcal{F}_A$ und $f_B \in \mathcal{F}_B$ korrespondieren, notiert als $f_A \sim f_B$, wenn sie ein gewisses Korrespondenzkriterium erfüllen. Das Paar $(f_A \sim f_B)$ heißt **Flächenkorrespondenz**. Die Menge \mathfrak{F} ist die Menge aller Flächenkorrespondenzen.

Ein Beispiel für ein Korrespondenzkriterium ist eine identische Ebenengleichung sowie eine nicht leere Schnittmenge der Flächen. Da dies nur bei exakter Geometrie ohne Rauschen sinnvoll ist, werden hier andere Kriterien benötigt. Diese werden in Kapitel 5.2.2 näher erläutert.

Aus Flächenkorrespondenzen können direkt Kanten- und Eckenkorrespondenzen abgeleitet werden:

Definition 3.15

Zwei Kanten $h_A \in \mathcal{K}_A$ und $h_B \in \mathcal{K}_B$ korrespondieren, notiert als $h_A \sim h_B$, wenn die jeweils angrenzenden Flächen korrespondieren, das heißt, wenn gilt:

$$\begin{aligned} (f_{A1} \sim f_{B1}) \in \mathfrak{F}, & \quad \text{mit } f_{A1} = \text{flaeche}(h_A), \quad f_{B1} = \text{flaeche}(h_B) \\ (f_{A2} \sim f_{B2}) \in \mathfrak{F}, & \quad \text{mit } f_{A2} = \text{flaeche}(\text{zwilling}(h_A)), \quad f_{B2} = \text{flaeche}(\text{zwilling}(h_B)) \end{aligned}$$

Das Paar $(h_A \sim h_B)$ heißt **Kantenkorrespondenz**.

Die Menge \mathfrak{K} ist die Menge aller Kantenkorrespondenzen.

Definition 3.16

Zwei Ecken $v_A \in \mathcal{E}_A$ und $v_B \in \mathcal{E}_B$ korrespondieren, wenn die jeweils adjazenten Kanten korrespondieren:

$$(h_{A,i} \sim h_{B,i}) \in \mathfrak{K}, \quad \forall h_{A,i} \in \text{kanten}(v_A), \quad h_{B,i} \in \text{kanten}(v_B)$$

Das Paar $(v_A \sim v_B)$ heißt **Eckenkorrespondenz**.

Die Menge \mathfrak{E} ist die Menge aller Eckenkorrespondenzen.

3.3 Parametrisierung

Ein generelles Problem größerer sensorbasierter Softwaresysteme ist der Umgang mit Parametern. Gerade bei mehrstufigen Verfahren, die Sensordaten unter verschiedenen Umgebungsbedingungen oder unterschiedlicher Hardware verarbeiten, müssen oftmals viele Parameter an die aktuellen Gegebenheiten angepasst werden. Dieser Schritt kann für den Anwender, insbesondere Nicht-Experten, aus folgenden Gründen mühsam sein:

- Die zugrunde liegenden Algorithmen sind dem Anwender oft nicht im Detail bekannt. Dies hat einerseits zur Folge, dass die Auswirkungen der Änderung eines Parameters dem Anwender nicht ersichtlich sind, und andererseits nicht klar ist, welcher Parameter geändert werden muss, um ein Ziel zu erreichen.
- Einzelne Schritte im Verfahren können abhängig voneinander sein. Das heißt, die Änderung eines Parameters verbessert zwar einen Aspekt des Ergebnisses, verschlechtert aber ungewollt einen anderen.
- Einzelne Parameter sind oftmals abhängig, das heißt die Änderung eines Parameters macht nur Sinn, wenn zugleich auch ein anderer Parameter angepasst wird.

Generell stellt sich die Frage, warum überhaupt Parameter notwendig sind. Zwei der Hauptgründe sind folgende:

1. **Messabweichungen:** In dieser Arbeit wird als Eingabe eine organisierte Punktwolke vorausgesetzt. Da die Punkte in Realität nie exakt und fehlerfrei gemessen werden können, sind Parameter nötig, die diesen Umstand berücksichtigen. Je nach Art und Messprinzip des verwendeten Sensors unterscheiden sich Punktdichte, Messrauschen, Messbereich und weitere Eigenschaften deutlich. Um zu einem guten Ergebnis zu kommen, ist es nötig, diese Eigenschaften zu kennen und zu berücksichtigen.
2. **Anwendungsfall:** Sind die Eigenschaften eines Sensors bekannt und berücksichtigt, so können dennoch unterschiedliche Anwendungsfälle betrachtet werden, in denen der Anwender will, dass das System verschieden reagiert. Beispiele dafür sind die gewünschte Rekonstruktionsgenauigkeit oder absichtlich eingeschränkte Messbereiche.

Mit dem Ziel einer einfachen Handhabung soll deshalb in dieser Arbeit darauf geachtet werden, es dem Nutzer möglichst einfach zu machen, das System vor der Benutzung zu konfigurieren. Einfache Konfigurierbarkeit ist dann erreicht, wenn der Nutzer nur sehr wenige unabhängige und eindeutige Parameter einstellen muss oder diese, wenn möglich, automatisiert bestimmen lassen kann. Alle algorithmischen Parameter des Systems sollen dann sinnvoll aus diesen wenigen Parametern ableitbar sein. Im ersten der beiden folgenden Abschnitte wird deshalb eine sogenannte Strukturgröße als Universalparameter bezogen auf Anwendungsabhängige Parameter eingeführt. Der zweite Unterabschnitt beschäftigt sich anschließend mit dem Einfluss von Messabweichungen auf die Parameter.

3.3.1 Die Strukturgröße als Universalparameter

Bei Softwaresystemen im Bereich der Geometrie sind am häufigsten folgende drei Arten von Parametern anzutreffen:

- **Längen-Parameter** geben eine Länge an und sind daher in der Einheit Meter. Durch Potenzieren ist es möglich, auch Flächen- oder Volumeninhalte zu beschreiben.
- **Winkel-Parameter** geben einen Winkel an und sind in der Einheit Radiant.

- **Ganzzahlige Parameter** sind ganzzahlig und beschreiben eine Anzahl von etwas (beispielsweise Anzahl an Pixel). Sie besitzen keine Einheit.

Besteht ein Algorithmus aus mehreren Einzelschritten, die jeweils Parameter benötigen, sind oft logische Abhängigkeiten der Parameter vorhanden. Es ist somit möglich, den Wert eines Parameters in Abhängigkeit eines anderen anzugeben, ohne dass diese Aufgabe dem Nutzer anvertraut werden muss.

Ein grundlegender, anwendungsbezogener Parameter ist die Rekonstruktionsgenauigkeit, die angibt, bis zu welchem Detailgrad geometrische Merkmale rekonstruiert werden sollen. In dieser Arbeit wird ein Längenparameter zur Angabe der Rekonstruktionsgenauigkeit verwendet, der folgendermaßen definiert ist:

Definition 3.17

Die **Strukturgröße** δ_s ist ein vom Anwender zu wählender, metrischer Parameter mit folgender Bedeutung:

Alle geometrischen Elemente der Oberfläche (z.B. Löcher, Kanten, Rundungen), die kleiner sind als δ_s , dürfen ignoriert werden, alle geometrischen Elemente, die größer sind, sollen rekonstruiert werden.

Es ist klar, dass diese Genauigkeit nicht beliebig gewählt werden kann, sondern durch technische Einschränkungen, wie beispielsweise die Auflösung des verwendeten Sensors, beschränkt ist.

Mit dem Ziel einer einfachen Handhabung für Nicht-Experten soll deshalb versucht werden, alle weiteren algorithmischen Parameter auf die Strukturgröße zu beziehen, sodass der Anwender bei Nutzung eines bekannten Sensors nur noch einen Parameter vor der Erfassung festlegen muss. Die Strukturgröße als Längenparameter ist dabei ein Maß, das allgemein verständlich und gut vorstellbar ist.

Im weiteren Verlauf dieser Arbeit werden Längen-Parameter immer mit $d_{\langle\text{Parametername}\rangle}$, Winkel-Parameter mit $\theta_{\langle\text{Parametername}\rangle}$ und ganzzahlige Parameter mit $v_{\langle\text{Parametername}\rangle}$ bezeichnet.

3.3.2 Berücksichtigung von Messabweichungen

Statistische Messabweichungen sind zufällige, nicht systematische und damit nicht korrigierbare Abweichungen des Messwerts vom wahren Wert. Ursachen sind beispielsweise Rauschen oder Quantisierungseffekte bei der Analog-Digital-Wandlung.

Beispiel 1

Gegeben sei eine Ebene $\vec{\pi} = (n_x, n_y, n_z, d)^T$ im dreidimensionalen Raum und eine Menge an Punkten $\vec{P}_i = (x_i, y_i, z_i, 1)^T$. Es soll entschieden werden, welche Punkte zur Ebene gehören. Dazu eignet sich der Abstand zur Ebene in Form folgender Bedingung:

$$|\vec{\pi} \circ \vec{P}_i| < d$$

Bei exakter Messung würde $d = 0$ genügen. Aufgrund von Messabweichungen ist ein Toleranzbereich $d > 0$ nötig.

Die Stärke der statistischen Messabweichung wird meist in Form einer Standardabweichung σ bei angenommener Normalverteilung angegeben. Dabei gilt, dass ca. 95% aller Punkte, die in Wahrheit auf E liegen, innerhalb des Intervalls von $\pm 2\sigma$ um die Ebene liegen. Daher ist für ein Konfidenzniveau von 95 % die Schwelle $d = 2\sigma$ geeignet.

Die Messabweichung des Ortes für Punktwolken eines Tiefensensors ist allerdings nicht konstant. Durch mehrfache Messung und statistische Auswertung kann ein empirisches Fehlermodell erstellt werden. Allgemein lässt sich sagen, dass der Zusammenhang zwischen Messabweichung und Tiefe z quadratisch ist, wie in folgenden Beispielen deutlich wird. Es wird dabei angenommen, dass die Punktwolke im Sensorkoordinatensystem liegt, das heißt, die Kamera befindet sich im Ursprung mit Blickrichtung in Richtung der positiven z -Achse.

Beispiel 2

Für weit verbreitete Sensoren, wie beispielsweise die Microsoft Kinect, sind mehrere Modelle in der Literatur verfügbar, beispielsweise

$$\sigma(z) = 0.0019(z - 0.4)^2 + 0.0012 \quad [\text{Nguyen12}]$$

$$\sigma(z) = 0.0028z^2 \quad [\text{Holzer12}]$$

Unterschiede zwischen Modellen sind dadurch begründet, welche Methode zur empirischen Ermittlung zum Einsatz kam, oder welche Art von Fehler berücksichtigt wurde (Quantisierungsfehler, Abhängigkeit des Winkels einer Ebene zur Sichtachse).

Bei der Verwendung von Parametern, die sich auf fehlerbehaftete Größen beziehen, kann es deshalb sinnvoll sein, die Messabweichung in den Wert des Parameters einfließen zu lassen. Da sich die Messabweichung nur vom verwendeten Sensor, aber nicht von der Geometrie der zu rekonstruierenden Szene abhängt, muss eine einmal bestimmte Messabweichung für den gleichen Sensor nicht erneut bestimmt werden. Meist sind die Werte der Messabweichung in Veröffentlichungen oder Datenblättern zu finden.

Da sich in dieser Arbeit nicht auf eine spezielle Kamera beschränkt wird, aber dennoch Punktwolken betrachtet werden, wird in dieser Arbeit ein einfach quadratisches Fehlermodell mit folgenden Bezeichnern verwendet. Die Punktwolke liege dabei wieder im lokalen Sensorkoordinatensystem.

Definition 3.18

Für stochastische Messabweichungen der Tiefe z der Punkte einer Punktwolke wird folgendes Fehlermodell angenommen:

$$\sigma_z(z) = \sigma_1 \cdot z^2$$

σ_z ist dabei die z -abhängige Standardabweichung einer angenommenen Normalverteilung. σ_1 ist eine pro Kamera feste Konstante. Der Index 1 wurde gewählt, da $\sigma_z(z = 1) = \sigma_1$ der Standardabweichung in einem Meter Entfernung entspricht.

3.3.3 Begrenzung der Strukturgröße durch Messabweichungen

Bei Verwendung einer bestimmten Kamera, für die die Standardabweichung der Messabweichung bekannt ist, kann die Strukturgröße als Anwendungs-abhängiger Parameter nicht komplett frei gewählt werden. Durch die Messabweichungen ist eine untere Schranke gegeben. Folgendes Beispiel illustriert den Umstand anhand einer hypothetischen Nachbarschaftsbestimmung:

Beispiel 3

Zwei Punkte mit Tiefe z heißen benachbart, wenn ihr Abstand kleiner als die Strukturgröße δ_s ist. Durch Rauschen schwanken die Punkte allerdings in dieser Tiefe bereits mit einer Standardabweichung von $\sigma_z(z) = \sigma_1 \cdot z^2$. Unter der Forderung eines 95%-Konfidenzniveaus, also der zweifachen Standardabweichung, kann die Nachbarschaftsbestimmung ab einer Tiefe von

$$z = \sqrt{\frac{\delta_s}{2\sigma_1}}$$

nicht mehr sinnvoll angewendet werden. Umgekehrt ausgedrückt, muss die Strukturgröße mindestens einen Wert von

$$\delta_s = 2 \cdot \sigma_1 \cdot z_{max}^2$$

aufweisen, um für die komplette Punktwolke mit maximaler Tiefe z_{max} korrekt zu arbeiten.

In diesem Beispiel wurde vernachlässigt, dass der Abstand zwischen zwei Punkten sowohl aus einem z -Anteil als auch einem Anteil dx in x oder y -Richtung besteht. Dies ist bei im Pixelgitter benachbarten Punkten einer organisierten Punktwolke allerdings zu vernachlässigen, wie folgende exemplarische Werte für eine Microsoft Kinect - Kamera zeigen:

Beispiel 4

Die Messabweichungen einer **Microsoft Kinect** können mit $\sigma_1 = 0.0028m^{-1}$ [Holzer12] abgeschätzt werden.

Bei einer zur erreichenden Strukturgröße $\delta_{s,1} = 0.04m$ kann bis

$$z_{max,1} = 2.67m$$

die Strukturgröße eingehalten werden (95%-Konfidenzniveau). Will man bis $z_{max,2} = 4m$ korrekt arbeiten, so beträgt die minimale Strukturgröße

$$\delta_{s,2} = 0.0896m.$$

Zwei im Pixelgitter in x benachbarte Punkte, sind bei einer Tiefe von $z_{max,1} = 2.67m$ in x -Richtung

$$dx = \frac{1px \cdot z_{max,1}}{f_x} \approx 0.005m$$

entfernt, wobei f_x die Brennweite in Pixeln ist. Zum Vergleich besitzt die zweifache Standardabweichung in dieser Tiefe einen Wert von

$$2 \cdot \sigma_z(z_{max,1}) = 0.04m.$$

Zusammenfassend lässt sich sagen, dass die Wahl der Strukturgröße abhängig von der verwendeten Kamera gewählt werden muss. Eine niedrige Strukturgröße kann ab einer bestimmten Tiefe nicht mehr eingehalten werden. Je nach Anwendung kann dies unerheblich sein und deshalb ignoriert werden, oder ist wichtig und muss behandelt werden, z.B. durch Einschränkung des Messbereichs.

3.4 Zusammenfassung

In diesem Kapitel wurden drei grundlegende Ansätze vorgestellt, die als Basis für ein 3D-Scan-System dienen, das den Anforderungen der Zielsetzung dieser Arbeit genügt. Die im Stand der Forschung identifizierte Lücke wird durch eine verschränkte modellbasierte Kombination aus SLAM und DSR geschlossen. Dies bietet die Grundlage, um in den folgenden Kapiteln die wissenschaftlichen Fragestellungen F2 und F3 beantworten zu können.

Es wurde gezeigt, dass eine Repräsentation von Modellen, die noch kein vollständiges Volumen umschließen, aber dennoch die gleichen Eigenschaften besitzen wie ein kontinuierliches Volumenmodell (Fragestellung F1), möglich ist. Dazu eignen sich besonders Begrenzungsflächenmodelle (B-Reps) unter Nutzung der *half-edge* Datenstruktur mit der Änderung, dass die Notwendigkeit einer Zwillingshalbkante entfällt. Zudem wurden Eigenschaften eines solchen partiellen Modells dargestellt und eine formale Beschreibung angegeben.

Als drittes wurde das Problem von vielen, für Nicht-Experten schwer zu verstehenden Parametern eines Softwaresystems adressiert. Zur Lösung soll ein Universalparameter, die Strukturgröße, als Basis dienen, auf den alle anderen algorithmischen Parameter bezogen werden. Zudem wurde ein einfaches Modell für Messabweichungen eingeführt, um auch diese im System einheitlich berücksichtigen zu können. Dieses Konzept ist der erste Schritt in Richtung eines einfach zu bedienenden Systems (Fragestellung F4).

Kapitel 4

Rekonstruktion partieller B-Reps

Inhalt

4.1	Stand der Forschung	59
4.1.1	Segmentierung	59
4.1.2	Bestimmung von Segmentbegrenzungen	60
4.2	Vorgehensweise	62
4.2.1	Segmentierung	63
4.2.2	Polygonalisierung	72
4.2.3	B-Rep Erzeugung	83
4.3	Evaluation	85
4.3.1	Güte der partiellen B-Reps	85
4.3.2	Laufzeit	93
4.4	Zusammenfassung	94

Zur Lösung der in dieser Arbeit betrachteten Aufgabenstellung (Abschnitt 1.4) wurde nach Untersuchung existierender Methoden im Stand der Forschung (Kapitel 2) im vorhergehenden Kapitel die grundlegende Vorgehensweise (Abschnitt 3.1) beschrieben. Dieses Kapitel befasst sich mit dem ersten Schritt, der Rekonstruktion eines partiellen, planaren Begrenzungsflächenmodells (*boundary representation model*, B-Rep) aus einer einzelnen organisierten Punktwolke unter Berücksichtigung einer einfachen Parametrisierung wie sie in Abschnitt 3.3 beschrieben wurde.

Im ersten Abschnitt werden dazu bestehende Verfahren zur Segmentierung von Punktwolken untersucht. Der zweite Teil erläutert anschließend die neuen Ansätze zur Rekonstruktion partieller B-Reps. Abschnitt drei evaluiert das vorgestellte Verfahren. Abschließend wird das Kapitel im vierten Teil nochmals zusammengefasst. Teile dieses Kapitels wurden bereits in [Sand16] dargestellt und veröffentlicht.

4.1 Stand der Forschung

Ein partielles, planares B-Rep besitzt Informationen über Geometrie (Flächen, Kanten, Ecken) und Topologie (Nachbarschaften). Inzidente Kanten sind dabei einerseits die geometrische Begrenzung einer planaren Fläche und andererseits das topologische Verbindungsglied zur Nachbarfläche. Im Folgenden befasst sich der erste Abschnitt mit Verfahren, die dem Finden von planaren Regionen in Punktwolken dienen ohne dabei zwangsweise auf die Begrenzung der Region einzugehen. Methoden zur Bestimmung des Randes eines Segments werden im zweiten Abschnitt erläutert.

4.1.1 Segmentierung

Die meisten bekannten Verfahren zur Segmentierung von Punktwolken basieren auf einer der drei folgenden Grundprinzipien: Hough-Transformation, Random Sample Consensus (RANSAC) oder Regionenwachstum. Während die Hough-Transformation für 2D-Probleme zum Finden von Linien oder Kreisen sehr gut geeignet ist, bauen nur sehr wenige Verfahren für die Segmentierung von 3D-Punktwolken auf ihr auf [Borrmann11]. Der Grund liegt in der hohen Laufzeit, die zudem von der Anzahl der resultierenden Ebenen abhängt. Eine Anpassung an eine spezielle Kamera durch Verwendung eines Modells für Messabweichungen kann das Verfahren beschleunigen [Dube11].

Basierend auf dem RANSAC-Prinzip [Fischler81] können Punktwolken iterativ segmentiert werden, indem das am besten passende Modell gefunden wird und dessen zugehörige Punkte aus der Punktwolke entfernt werden. Dies wird mehrfach wiederholt. Auch dieser Weg ist zeitaufwändig und nicht in Echtzeit durchführbar. Zur Beschleunigung wird oft eine Vorsegmentierung oder Vorverarbeitung genutzt, um die eigentliche Anpassung zu vereinfachen. Dabei kommen Unterteilungen anhand verbundener Regionen [Silva02] oder Kanten [Gottardo03], oder volumetrische Vorsegmentierungen mithilfe von Octrees [Schnabel07, Holz11] zum Einsatz. Eine weitere Eigenschaft der Verfahren ist, dass eine gefundene Region nicht zwangsweise verbunden sein muss, da beispielsweise eine Ebene unendlich ausgedehnt ist. Dies kann einerseits im Nachhinein behoben werden, indem Regionen wieder mittels Clustering unterteilt werden [Trevor12] oder andererseits indem die Komponenten bereits während der RANSAC-Iterationen berücksichtigt werden [Gallo11, Qian14].

Beide Arten von Verfahren, Hough-basierte und RANSAC-basierte, benötigen für die Segmentierung keine Informationen über Nachbarschaftsbeziehungen zwischen einzelnen Punkten. Durch Nutzung dieser zusätzlichen Information sind Verfahren auf Basis von Regionenwachstum (*region growing*) deutlich schneller. Hierbei lassen sich zwei Segmentierungsszenarien unterscheiden: unorganisierte und organisierte Punktwolken.

Bei unorganisierten, meist großen Punktwolken wird daher eine volumetrische Unterteilung in Form eines Voxelgitters oder Octrees aufgebaut, um daraus die Nachbarschaftsinformationen zu extrahieren [Deschaud10, Xiao13, Vo15].

Bei organisierten Punktwolken entfällt dieser Schritt, da über die 2D-Gitterstruktur die Nachbarn bereits bekannt sind. Ausgehend von Seed-Punkten wächst eine Region jeweils um ein Nachbarpixel, wenn der neue Punkt kompatibel zur vom Segment repräsentierten Ebene ist. Dabei kann die Kovarianzmatrix der Punkte benutzt werden, die während des Wachsens inkrementell aktualisiert werden kann [Poppinga08]. Statt einer Kovarianzmatrix können auch zuvor Normalen der Punkte berechnet werden und während des Wachsens nur mittlere Normalenrichtungen mitgeführt werden, um die Berechnung zu beschleunigen [Holz14, Arbeiter14]. Holz et al. nutzen dabei zur Bestimmung der Normalen eine schnelle Triangulation auf Basis des 2D-Gitters [Holz14]. Arbeiter et al. verwenden eine feste Maske um die Punkte zur Normalenschätzung und bauen zudem während des Wachsens einen Segmentgraphen

auf, um am Ende benachbarte Segmente nochmals auf eine mögliche Vereinigung zu testen [Arbeiter14]. Dies verhindert eine Übersegmentierung, also eine Unterteilung in zu viele kleine Segmente.

Statt einzelner Punkte können auch mehrere, in einer Zeile oder Spalte zusammenhängende Pixel als Inkrement, mit dem eine Region wächst, genutzt werden. Dies wird hauptsächlich bei Laserscannern eingesetzt. Als Kriterium zum Wachsen kommt dabei der sogenannte *Bearing Angle*, der Winkel zwischen Laserstrahl und Oberfläche, zum Einsatz [Harati07a] oder, wie zuvor, Kovarianzmatrizen zum Test auf Planarität [Georgiev11].

Auch noch größere Inkremente sind möglich, indem die Punkte zu Beginn in regelmäßige rechteckige Stücke aufgeteilt werden, die als Einheit für das Wachsen dienen [Kaushik10, Xiao11]. Auch hier sind Laserscanner das Haupteinsatzgebiet.

Sehr ähnlich zum Regionenwachstum ist eine Segmentierung nach dem Prinzip des Regionenmarkierens (*connected components labelling*). Hier werden die Pixel der Reihe nach durchlaufen und mit einer Markierung (*Label*) versehen. Ein Kriterium gibt an, welche Regionen verschmolzen werden. Durch Verwendung von Winkeln und Abständen kann auch hiermit eine planare Segmentierung erreicht werden [Trevor13].

Ein komplett anderer Ansatz ist die Verwendung von Markov Random Fields (MRF) [Tatavarti17]. Einzelne Punkte werden dabei als Knoten in einem MRF dargestellt, im 2D-Pixelgitter benachbarte Punkte sind auch im MRF verknüpft. Jeder Knoten hat zudem ein Label, das mittels *Bayesian Belief Propagation* so optimiert wird, dass planare Stücke dasselbe Label erhalten.

Es sei angemerkt, dass auch Dreiecksnetze eine Nachbarschaftsbeziehung zwischen Knoten darstellen. Das hier betrachtete Problem ist also verwandt mit den in Abschnitt 2.2.2 vorgestellten Segmentierungsverfahren. Der Unterschied besteht darin, dass in Kapitel 2 eine homogene und vollständige Abtastung der Oberfläche ohne große Messabweichungen vorausgesetzt wird.

4.1.2 Bestimmung von Segmentbegrenzungen

Wenn bekannt ist, welche Punkte sich zu einem planaren Segment gruppieren lassen, so reicht die Bestimmung der Ebenengleichung allein nicht als Approximation der Punkte aus, da eine Ebene per Definition unendlich ausgedehnt ist. Zu diesem Zweck können die Punkte durch ein planares Polygon angenähert werden. Die Bestimmung dieser Begrenzung ist dabei ein 2D-Problem, da alle Punkte in der betrachteten Ebene liegen oder dorthin projiziert werden können.

Die einfachste aller Hüllen ist die konvexe Hülle, die beliebig angeordnete Punkte jedoch nur schlecht approximieren kann. Sind als Innenwinkel des Polygons auch Winkel größer als 180° zugelassen, so spricht man auch von einer konkaven Hülle (*concave hulls*). Eine Variante davon sind die sogenannten *Alpha-Shapes* (Abbildung 4.1a) [Edelsbrunner83]. Zwei benachbarte Punkte im Polygon besitzen die Eigenschaft, dass ein kreisförmiger Leerraum mit dem Radius $1/\alpha$ existiert, der beide Punkte berührt. Der Wert α stellt somit einen Parameter dar, der die Hülle beeinflusst. Für $\alpha = 0$ entspricht die Begrenzung der konvexen Hülle.

Weitere Verfahren für konkave Hüllen ohne direkte mathematische Definition sind der *Swinging-Arm-Algorithmus* [Galton06], der *K-nächste-Nachbar-Ansatz* [Moreira07] (Abbildung 4.1b), sowie die χ -Shapes [Duckham08] (Abbildung 4.1c). Alle liefern ähnliche Ergebnisse, unterscheiden sich aber in der Laufzeitkomplexität sowie in der Möglichkeit von Löchern im Polygon. Ein Vergleich ist in Park et al. [Park12] zu finden, die zudem eine Erweiterung auf höhere Dimensionen vorstellen.

Eine weitere Definition ist die *Alpha-concave hull* [Asaedi13] mit Parameter α (Abbildung 4.1d). Alle inneren Winkel des resultierenden Polygons sind kleiner als $180 + \alpha$ Grad und die Fläche des Polygons ist minimal. Für $\alpha = 0^\circ$ ergibt sich damit die konvexe Hülle, für $\alpha = 180^\circ$

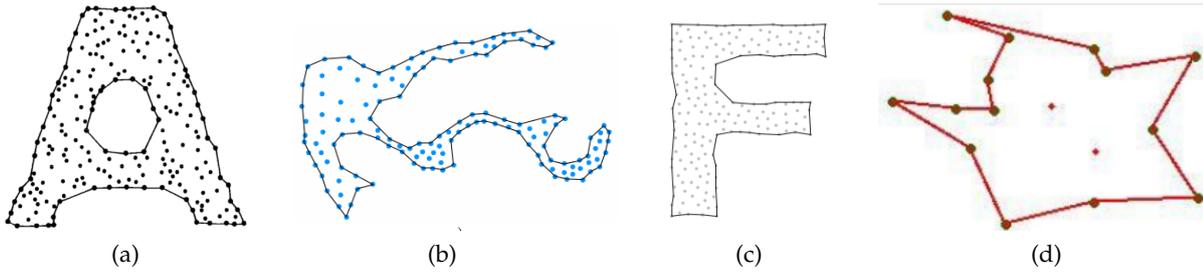


Abbildung 4.1: (a) α -Shape (aus [Edelsbrunner83]), (b) kNN-Ansatz (aus [Moreira07]), (c) χ -Shape (aus [Duckham08]), (d) Alpha-concave hull mit $\alpha = 103^\circ$ (aus [Asaedi13])

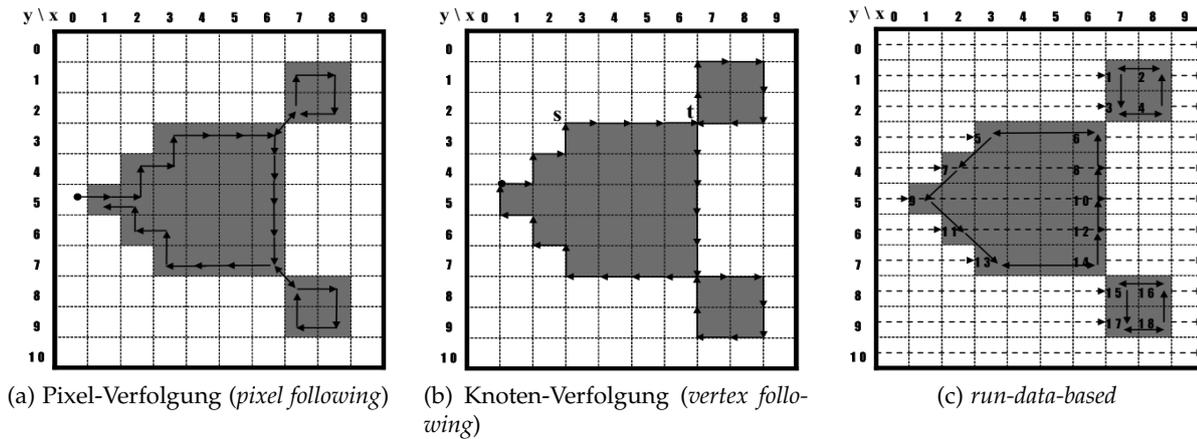


Abbildung 4.2: Arten von Konturverfolgungsalgorithmen (aus [Seo16])

ist das Ergebnis identisch zur Lösung des *min-area travelling salesman problem* (*min-area TSP*). Die Berechnung der Alpha-concave hull ist NP-schwer.

Alle obigen Verfahren gehen von einer unsortierten Punktmenge aus. Im Algorithmus sind dann Schritte zur Bestimmung von nahe liegenden Punkten nötig, beispielsweise durch Sortierung der Punkte oder durch eine k-nächste-Nachbar-Bestimmung. Bei organisierten Punktwolken sind die 3D-Punkte aber bereits in einer 2D-Gitterstruktur abgelegt, sodass Nachbarschaften direkt ableitbar sind. Die Randpunkte des Segments zu finden entspricht damit dem Problem den Rand einer Region in einem 2D-Pixelgitter zu finden. Da später die Randpunkte als Polygon dargestellt werden, ist es wichtig, eine sortierte Liste der Randpunkte zu erhalten. Zu diesem Zweck eignen sich Konturverfolgungsalgorithmen, die sich in drei Gruppen einteilen lassen [Seo16] (Abbildung 4.2): Pixel-Verfolgungsmethoden verfolgen die Randpixel des Vordergrundes, Knoten-Verfolgungsmethoden verfolgen den Rand zwischen Vorder- und Hintergrund, und *run-data*-basierte Methoden arbeiten das Bild zeilenweise ab und erkennen Anfang und Ende des Segments innerhalb einer Zeile. Es existiert eine Vielzahl an Methoden [Grant81, Danielsson81, Pavlidis82, Suzuki85], die sehr ähnliche Ergebnisse liefern und sich oft nur hinsichtlich ihrer (Speicher-) Anforderungen unterscheiden. Zur Entstehungszeit dieser Methoden in den 80er und 90er-Jahren war Speicher ein hartes Kriterium, da schon das doppelte Vorhalten eines Bildes zur Auslagerung des Speichers und damit zu starker Erhöhung der Laufzeit führen konnte [Miyatake97]. Heute ist dieser Punkt eher in den Hintergrund gerückt.

4.2 Vorgehensweise

Das Ziel dieses Kapitels ist es, aus einer einzelnen organisierten Punktwolke ein partielles, planares B-Rep zu erzeugen. Zusammen mit der Fusion von partiellen B-Reps, die im nächsten Kapitel betrachtet wird, bilden diese beiden Schritte die Grundlage des modellbasierten SLAM-Systems. Es ist daher klar, dass die Laufzeit des Rekonstruktionsschrittes die Bildrate des Gesamtsystems stark beeinflusst. Ein Ziel ist daher eine schnelle Rekonstruktion gegenüber einer exakten, insbesondere da die Verschmelzung von Daten aus anderen Blickwinkeln, die die Gesamtrekonstruktion genauer macht, erst im folgenden Schritt erfolgt.

Da die Segmentierung der Eingabepunktwolke der essentielle Schritt ist, scheiden Verfahren auf Basis der Hough-Transformation und des RANSAC-Algorithmus aufgrund ihrer Rechenzeit aus. Zudem liegt eine organisierte Punktwolke vor, sodass es sinnvoll ist, deren Struktur auszunutzen. Im Bereich der Regionenwachstumsverfahren wurde gezeigt, dass die Verwendung von Normalenrichtungen pro Punkt anstelle von Kovarianzmatrizen eine schnelle Segmentierung ermöglicht [Holz14, Arbeiter14]. Der neue Ansatz auf Basis von *Markov Random Fields* ist laut den Autoren vergleichbar schnell, die Ergebnisse zeigen jedoch, dass die planaren Segmente oft nicht kompakt sind und eine Art „Ausleger“ aufweisen [Tatavarti17].

Regionenwachstumsverfahren wie das von Holz et al. und Arbeiter et al. erscheinen deshalb als geeignete Basis für das vorliegende Problem. Da beide Verfahren die Segmentierung zum Ziel haben, muss noch das Finden von Segmentbegrenzungen, das Erzeugen einer topologischen Struktur und des partiellen B-Rep-Modells unter Berücksichtigung der Konzepte zur einfachen Parametrisierung mittels Strukturgröße und Messabweichungen (Abschnitt 3.3) einbezogen werden. Dazu eignet sich das Verfahren von Holz et al. besser: Die initiale Vernetzung erzeugt ein auf dem Pixelgitter angeordnetes Netz, das zur Normalenberechnung, Glättung und für das Regionenwachstum genutzt wird. Dieses Netz kann zusätzlich dazu verwendet werden, ein allgemeines, aber tiefenkorrektes Filterkonzept zu entwickeln, das unter anderem dazu genutzt werden kann, eine strukturgrößen-abhängige Segmentierung zu erreichen (siehe Abschnitt 4.2.1). Das hier entwickelte Verfahren basiert deshalb auf dem Vorgehen von Holz et al. [Holz14]. Nichtsdestotrotz werden die Ideen von Arbeiter et al. verwendet, um eine Übersegmentierung zu verhindern, indem die Topologie bereits während der Segmentierung erstellt und zur nachträglichen Verschmelzung von Segmenten genutzt wird.

Nach der Segmentierung ist es nötig, die Begrenzungen der Flächen zu extrahieren. Aufgrund der Rechenzeit sind hier Algorithmen zur Berechnung der konkaven Hülle klar im Nachteil gegenüber Konturverfolgungsalgorithmen, da diese auch die vorhandene Pixelstruktur ausnutzen. Dabei bieten sich knoten-basierte Verfolgungsverfahren an, da diese den Rand zwischen zwei Pixel legen. Das so entstehende Polygon ist dadurch immer einfach, das heißt zwei Kanten kreuzen oder überlagern sich nicht. Zur Erzeugung eines 3D-Polygons aus der 2D-Kontur wird eine Reprojektion vorgestellt, um den Einfluss von Messabweichungen zu reduzieren. Anschließend werden die unterschiedlichen Kanten eines B-Reps (inzident, pseudoinzident, virtuell) identifiziert und angepasst. Dadurch kann im letzten Schritt leicht ein B-Rep Modell erzeugt werden.

Zusammenfassend lässt sich die Vorgehensweise zur Rekonstruktion planarer, partieller B-Reps aus organisierten Punktwolken durch drei Schritte charakterisieren:

1. Die **Segmentierung** (Abschnitt 4.2.1) unterteilt die Punktwolke in einzelne Segmente. Jedes Segment entspricht später einer Fläche im B-Rep. Das Verfahren von Holz et al. [Holz14] wird dabei erweitert, sodass eine allgemeine tiefenkorrekte Filterung möglich ist. Dies wird dazu genutzt, Messabweichungen zu reduzieren, sowie die Strukturgröße zu berücksichtigen.
2. Die **Polygonalisierung** (Abschnitt 4.2.2) erzeugt aus jedem Segment ein einfaches Polygon (ggf. mit Löchern). Dazu wird ein Knoten-basierter Konturverfolgungsansatz unter

Verwendung von Ansätzen aus Miyatake et al. [Miyatake97] genutzt, der durch Projektion sicherstellt, dass die Polygone auch in 3D einfach sind und trotz Messabweichungen eine gute Approximation darstellen. In den gefundenen Konturen werden anschließend inzidente und pseudo-inzidente Kanten identifiziert, die den späteren Kanten im B-Rep entsprechen.

3. Die **B-Rep Erzeugung** (Abschnitt 4.2.3) erstellt aus den in den vorhergehenden Schritten gesammelten Informationen ein partielles B-Rep. Dies ist unkompliziert, da in der Segmentierung bereits Flächen und in der Polygonalisierung bereits Kanten identifiziert wurden, die direkt verwendet werden können. Nur Ecken müssen noch speziell behandelt werden. Durch ein Optimierungsverfahren wird sichergestellt, dass Flächen, Kanten und Ecken auch global konsistent sind.

4.2.1 Segmentierung

Ziel der Segmentierung ist die Unterteilung der Punktwolke in Regionen. Eine organisierte Punktwolke P (Abbildung 4.3) besteht aus einer Menge an Punkten im dreidimensionalen Raum

$$P = \{\vec{P}_i\}, \quad \vec{P}_i \in \mathbb{R}^3, \quad (4.1)$$

die in einem Pixelgitter der Größe $width \times height$ organisiert sind

$$P(u, v) = \vec{P}_{v \cdot width + u} \quad \text{mit } 0 \leq u < width, \quad 0 \leq v < height, \quad (4.2)$$

und dabei im lokalen Sensorkoordinatensystem liegen. Das bedeutet, dass der Brennpunkt dem Ursprung entspricht und die Blickrichtung die positive z -Achse ist. Unter Annahme einer idealen Lochkamera gilt somit:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \text{round}(\frac{u'}{z'}) \\ \text{round}(\frac{v'}{z'}) \end{pmatrix}, \quad \text{mit } \begin{pmatrix} u' \\ v' \\ z' \end{pmatrix} = K \cdot P(u, v) = K \cdot \vec{P}_{v \cdot width + u}. \quad (4.3)$$

K ist dabei die intrinsische Kameramatrix:

$$K = \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}. \quad (4.4)$$

Einfach ausgedrückt bedeutet dies, dass jeder 3D-Punkt in dem Pixel abgelegt ist, in dem seine Bild-Projektion mittels der Projektionsmatrix K liegt. Es ist zu beachten, dass aufgrund des Messprinzips oder Reflexionen nicht jeder Pixel einen gültigen Punkt enthalten muss (Abbildung 4.3b).

Gesucht ist eine Segmentierung S der Punktwolke in planare Regionen R_j , die in späteren Schritten den einzelnen Flächen des B-Rep Modells entsprechen sollen:

$$S = \{R_1, \dots, R_{|S|}\} \text{ mit } R_j = \{\vec{P}_i\}, \quad R_j \cap R_k = \emptyset, j \neq k. \quad (4.5)$$

Eine Region muss dabei zusammenhängend sein.

Die Segmentierung kann wiederum in drei Einzelschritte unterteilt werden:

1. **Vernetzung:** Die Punktwolke wird mithilfe des Verfahrens von Holz et al. [Holz14] auf Basis des Pixelgitters vernetzt. Aus diesem Netz werden dann Normalenrichtungen für jeden Punkt bestimmt. Zudem ergibt sich zwischen Pixeln eine neue Nachbarschaftsbeziehung, die Tiefensprünge berücksichtigt. Diese wird für alle weiteren Schritte genutzt.

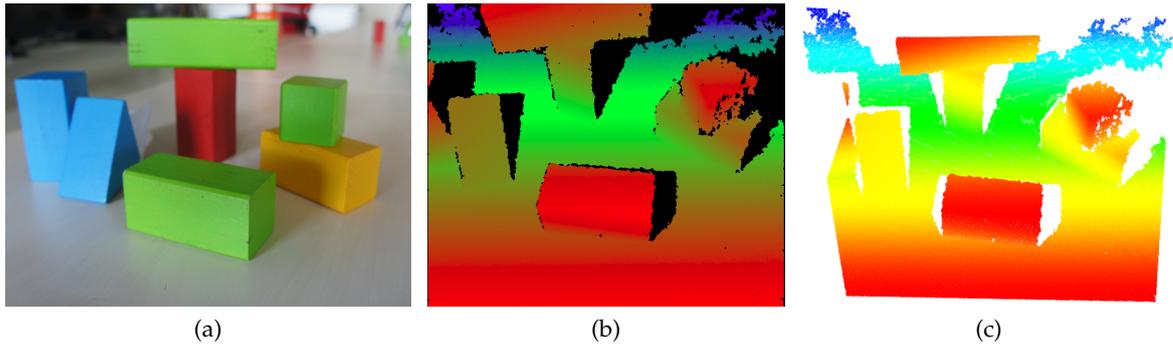


Abbildung 4.3: Eingabepunktvolke zur Veranschaulichung der Rekonstruktionsschritte. (a) Szene, (b) 2D-Darstellung der organisierten Punktvolke (in schwarzen Pixeln sind keine Tiefenwerte vorhanden), (c) perspektivische Darstellung der 3D-Punkte, eingefärbt anhand des z-Werts

2. **Regionenwachstum:** Analog zu Holz et al. wird die Punktwolke mittels Regionenwachstum in planare Teile segmentiert. Zusätzlich wird ein zweiter Wachstumsschritt vorgestellt, der die Qualität der Segmentierung nochmals verbessert.
3. **Filterung:** Auf Basis des Netzes wird ein allgemeines Filterkonzept entwickelt, das tiefenkorrekt und unter Berücksichtigung der Strukturgröße allgemeine Filteroperationen durchführen kann. Dies wird verwendet, um Rauschen in der Segmentierung zu eliminieren.

1. Vernetzung

Im ersten Schritt werden die Punkte der organisierten Punktwolke zu einem Netz verbunden. Holz et al. [Holz14] schlagen dazu vier Typen vor (Abbildung 4.4). Das adaptive Netz ist dabei am besten geeignet, da es die Oberfläche am genauesten approximiert, besonders in der Nähe von Kanten. Zur Erstellung des Netzes werden im Pixelgitter benachbarte Punkte \vec{P}_i und \vec{P}_j mit einer Kante verbunden, wenn zwei Bedingungen erfüllt sind: Die euklidische Distanz muss unter einer Schwelle liegen, und der Winkel zwischen Sichtstrahl und Verbindungsvektor der beiden Punkte muss größer als eine Schwelle sein:

$$\|\vec{P}_i - \vec{P}_j\|_2 < d_{mesh}, \quad (4.6)$$

$$\left| \frac{\vec{P}_i \circ (\vec{P}_i - \vec{P}_j)}{\|\vec{P}_i\|_2 \cdot \|\vec{P}_i - \vec{P}_j\|_2} \right| < \cos(\theta_{mesh}). \quad (4.7)$$

Anschließend wird für jeden Punkt $\vec{P}_i = P(u, v)$ eine Normale $\vec{n}_i = N(u, v) \in \mathbb{R}^3, \|\vec{n}_i\|_2 = 1$ berechnet, indem die Normalen der anliegenden Dreiecke gewichtet mit dem Flächeninhalt gemittelt werden. Im letzten Schritt werden alle Punkte \vec{P}_i und Normalen \vec{n}_i geglättet, indem ein gewichtetes Mittel aus den (maximal neun) benachbarten Punkten bzw. Normalen gebildet wird:

$$\vec{P}_i = \frac{\sum_{j \in NB_i} w_{ij} \vec{P}_j}{\sum_{j \in NB_i} w_{ij}} \quad \text{und} \quad \vec{n}_i = \frac{\sum_{j \in NB_i} w_{ij} \vec{n}_j}{\sum_{j \in NB_i} w_{ij}}. \quad (4.8)$$

Die Menge NB_i beschreibt dabei die Indizes der durch die eingefügten Kanten benachbarten Punkte eines Punktes \vec{P}_i . Die Gewichte w_{ij} setzen sich dabei, wie in Holz et al. beschrieben,

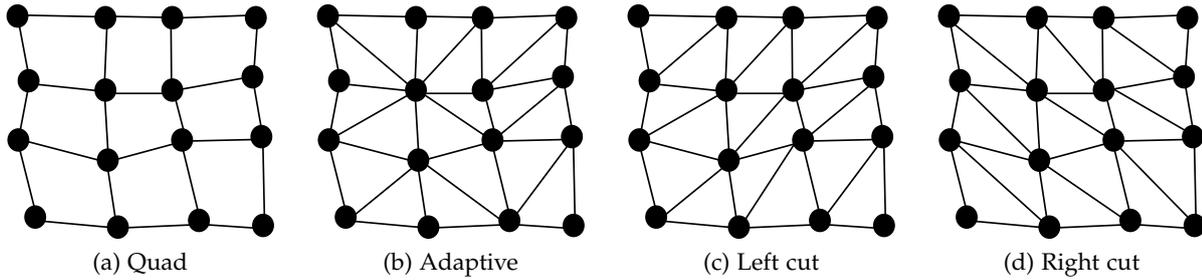


Abbildung 4.4: Verschiedene Arten von Vernetzungen aus [Holz14]

aus Distanz- und Normalentermen zusammen:

$$w_{ij} = e^{-\|\vec{P}_i - \vec{P}_j\|_2} \cdot e^{-\|n_i - n_j\|_1}. \quad (4.9)$$

Für eine Implementierung ist folgendes anzumerken: Eine explizite Berechnung und Speicherung des Dreiecksnetzes ist nicht nötig, da hier wie auch in den folgenden Schritten stets nur die Information über im Netz benachbarte Punkte benötigt wird. Daher genügt es, für jeden Punkt zu speichern, mit welchen Nachbarpunkten er verbunden ist. Da ein Pixel genau acht Nachbarn besitzt, kann diese Information als 8-Bit-Feld (1 Byte) pro Pixel effizient abgelegt werden. Zudem können alle Teilschritte pro Pixel parallelisiert ausgeführt werden. Das Ergebnis der Vernetzung der Punktwolke aus Abbildung 4.3 ist in Abbildung 4.5 dargestellt.

Wie im Grundkonzept (Kapitel 3.3) erläutert, ist ein Ziel, algorithmische Parameter auf möglichst wenige universelle Parameter zu beziehen. Im Vernetzungsschritt werden zwei Parameter benötigt, die Distanzschwelle d_{mesh} und die Winkelschwelle θ_{mesh} . Holz et al. [Holz14] weisen darauf hin, dass es sinnvoll sei, d_{mesh} mit steigendem z -Wert wachsen zu lassen, um die steigenden Messabweichungen einzubeziehen (siehe Kapitel 3.3.2). Bei der Winkelschwelle sei dies nicht nötig.

Da die in Kapitel 3.3.1 definierte Strukturgröße δ_s genau die Länge angibt, unterhalb der keine geometrischen Merkmale erkannt werden sollen, können alle Punkte mit geringerem Abstand als δ_s vernetzt werden. Damit das Vernetzen jedoch sinnvoll anwendbar ist, muss die Schwelle größer sein als die Messabweichung der Punkte in dieser Tiefe. Es ist daher je nach Kamera und gewählter Strukturgröße zu beachten, dass ab einer bestimmten Tiefe z_s die Messabweichungen in der gleichen Größenordnung liegen können (siehe Kapitel 3.3.3).

Es werden deshalb folgende Abhängigkeiten der Parameter vorgeschlagen:

$$d_{mesh} = \max(\delta_s, 2 \cdot \sigma_s(z)), \quad (4.10)$$

$$\theta_{mesh} = const. \quad (4.11)$$

σ_s ist dabei die tiefenabhängige Standardabweichung aus Definition 3.18, wobei die Tiefe z aus beiden zu vernetzenden Punkten \vec{P}_i und \vec{P}_j ermittelt wird: $z = \max(z(\vec{P}_i), z(\vec{P}_j))$ und der Faktor 2 durch die Verwendung des 95%-Konfidenzniveaus entsteht. Insgesamt wird so erreicht, dass die Strukturgröße solange verwendet wird, wie es die Messabweichungen zulassen.

2. Regionenwachstum

In diesem Schritt werden planare Segmente in der Punktwolke gefunden. Dazu wird ein Regionenwachstum-Algorithmus (*region growing*) verwendet, der nach folgendem Prinzip arbeitet:

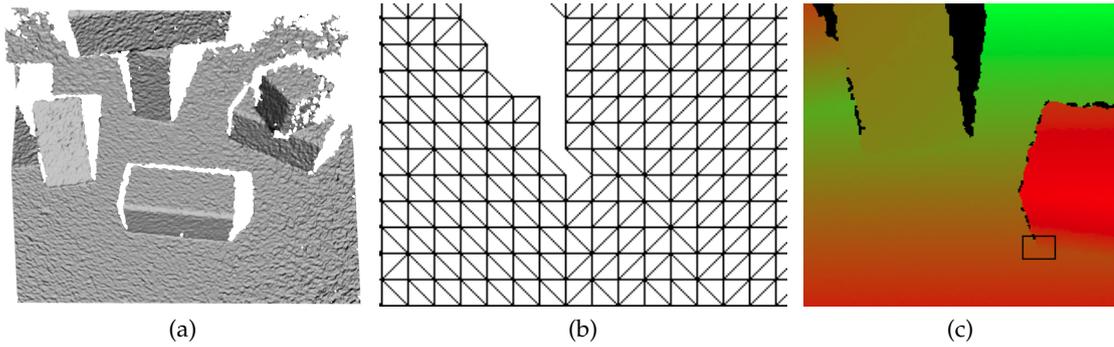


Abbildung 4.5: Adaptives Netz der Punktwolke aus Abbildung 4.3. (a) 3D-Ansicht, (b) 2D-Ansicht des Netzes des in (c) markierten Ausschnitts der Punktwolke

1 wiederhole

- 2 Initialisiere neue Region $R_j := \{\}$ und eine Menge $L := \{\}$.
- 3 Wähle einen noch nicht zu einer Region zugeordneten Punkt als Startpunkt (*seed point*) und füge ihn zu L hinzu.
- 4 **solange** L nicht leer **tue**
 - 5 Entnehme Punkt \vec{P} aus L und füge ihn zu R_j hinzu.
 - 6 **für alle** mit \vec{P} verbundenen Nachbarn \vec{P}_n , die keiner Region zugeordnet sind, **tue**
 - 7 Teste \vec{P}_n auf Kompatibilität zur Region R_j .
 - 8 **wenn** kompatibel **dann** füge \vec{P}_n zur L .
- 9 **bis** alle Punkte zugeordnet sind.

Um planare Segmente zu finden, wird für eine Region der Schwerpunkt \bar{P} und die mittlere Normale \bar{n} verwendet: Ein Punkt \vec{P}_n ist kompatibel zu einer Region R_j , wenn zwei Bedingungen erfüllt sind: Die Distanz des Punktes \vec{P}_n zur Ebene von R_j muss kleiner sein als eine Schwelle d_{grow} und der Winkel zwischen Ebenennormale von R_j und Normale \vec{n}_n des Punktes \vec{P}_n muss kleiner sein als θ_{grow} :

$$|(\vec{P}_n - \bar{P}) \circ \bar{n}| < d_{grow}, \quad (4.12)$$

$$\arccos(\vec{n}_n \circ \bar{n}) < \theta_{grow}. \quad (4.13)$$

Schwerpunkt \bar{P} und Normale \bar{n} können inkrementell angepasst werden, sobald ein Punkt \vec{P} mit Normale \vec{n} zu einer Region aus bisher k Punkten hinzugefügt wird:

$$\bar{P}_{k+1} = \frac{k \cdot \bar{P}_k + \vec{P}}{k+1}, \quad (4.14)$$

$$\bar{n}_{k+1} = \frac{k \cdot \bar{n}_k + \vec{n}}{k+1}. \quad (4.15)$$

Das Ergebnis des Regionenwachstums ist eine Segmentierung S aus den gefundenen Regionen (Abbildung 4.6a):

$$S = \{R_1, \dots, R_j, \dots, R_{|S|}\}. \quad (4.16)$$

Ein Vorteil dieses gierigen Ansatzes liegt in der linearen Laufzeit $O(p)$ mit $p = width \cdot height$, da jeder Pixel nur einmal betrachtet wird. Nachteilig ist, dass die Wahl der Startpunkte einen Einfluss auf das Ergebnis hat. Denn wie im Pseudocode ersichtlich, wird ein Punkt, der einmal zu einer Region R_j zugeordnet ist, nicht erneut beim Wachsen der Regionen R_k , mit $k > j$ berücksichtigt. Daher spielt die Reihenfolge, in der die Segmente erzeugt werden, eine Rolle. In

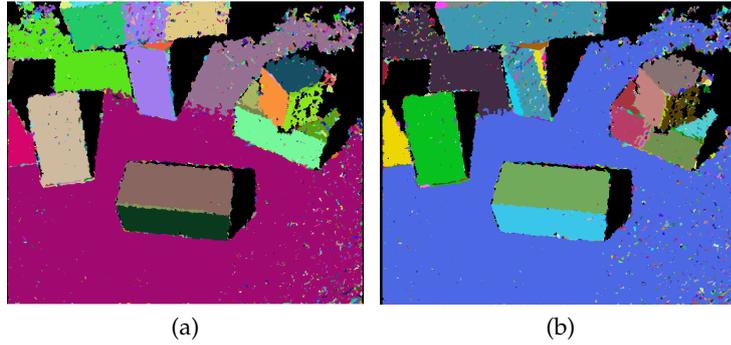


Abbildung 4.6: Ergebnisse des Regionenwachstums. Unterschiedliche Regionen sind mit einer zufälligen Farbe markiert. An schwarzen Pixel ist kein Tiefenpunkt vorhanden. (a) Als Startpunkt wurde der jeweils am weitesten unten beziehungsweise rechts liegende, noch nicht zugeordnete Pixel gewählt. (b) Als Startpunkt wurde der jeweils am weitesten oben beziehungsweise links liegende, noch nicht zugeordnete Pixel gewählt.

Abbildung 4.6a und 4.6b sind zwei Segmentierungen derselben Punktwolke mit unterschiedlicher Wahl der Startpunkte zu sehen.

Ein weiterer Nachteil ist, dass Segmente in benachbarte Segmente, die eine ähnliche Ebenennormale haben, „hineinwachsen“ können, wie in Abbildung 4.6 im violetten beziehungsweise türkisen Segment im oberen Teil des Bildes zu sehen ist.

Bis zu diesem Punkt entspricht das Verfahren dem Ansatz von Holz et al. [Holz14]. Im Folgenden wird nun eine Erweiterung des Regionenwachstums vorgestellt, die unter Beibehaltung der Laufzeitkomplexitätsklasse die eben beschriebenen Nachteile reduziert.

Der Grund für ein Hineinwachsen von Regionen ist, dass ein Punkt, der einmal zu einer Region R_j zugeordnet ist, nicht beim Wachsen der Regionen R_k , mit $k > j$ berücksichtigt wird, obwohl er eventuell besser in diese Region passen könnte. Daher kann im Anschluss an das obige Regionenwachstum die Liste der Regionen noch einmal in umgekehrter Erstellungsreihenfolge, also von $R_{|S|}$ bis R_1 , durchlaufen werden und ein vergleichender Test für Nachbarpunkte ausgeführt werden, der zulässt, dass Punkte von einer Region zu einer anderen wechseln. Folgender Pseudocode verdeutlicht das Vorgehen des erweiterten Regionenwachstums:

```

1  $L := \{\}$ 
2 für  $j = |S| \rightarrow 1$  tue
3   für alle Punkte  $\vec{P}$ , die im Netz benachbart sind zu einem Punkt der Region  $R_j$  tue
4      $R_k =$  Region von  $\vec{P}$ 
5     wenn  $k < j$  dann
6       Füge  $\vec{P}$  zu  $L$  hinzu.
7   solange  $L$  nicht leer tue
8     Entnehme Punkt  $\vec{P}$  aus  $L$ . Sei  $R_p$  die Region des Punktes  $\vec{P}$ .
9     wenn  $\vec{P}$  besser zu  $R_j$  als zu  $R_p$  passt dann
10      Entferne  $\vec{P}$  aus  $R_p$  und füge  $\vec{P}$  zu  $R_j$  hinzu.
11      Füge alle mit  $\vec{P}$  verbundenen Nachbarn, die zu einer Region  $R_k$  mit
         $k < j$  gehören, zu  $L$  hinzu.
12 Teile alle nicht zusammenhängenden Regionen auf.
```

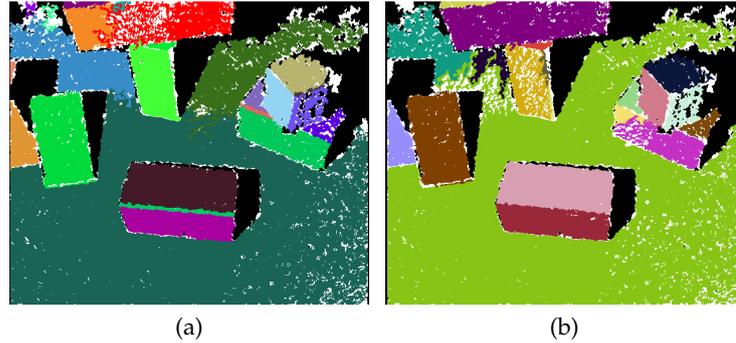


Abbildung 4.7: Segmentierung mit erweitertem Regionenwachstum. Jedes Segment ist in zufälliger Farbe eingefärbt. Alle Segmente mit weniger als 200 Punkte wurden zur besseren Übersicht nicht eingefärbt (weiß). (a) Erweiterung der Segmentierung aus Abbildung 4.6a, (b) Erweiterung der Segmentierung aus Abbildung 4.6b

Der entscheidende Punkt des Vorgehens ist, dass Regionen aus anderen Regionen, die im ursprünglichen Regionenwachstum früher erstellt wurden, nun Punkte, die besser passen, „stehlen“ können. Der Test, ob ein Punkt besser zur Region R_j oder R_p passt, kann mit den gleichen Kriterien wie oben durchgeführt werden, indem Abstand zur Ebene und Winkel zwischen den Normalen verglichen wird: Ein Punkt \vec{P} mit Normale \vec{n} wird aus Region R_p entfernt und zur Region R_j hinzugefügt, wenn gilt:

$$|(\vec{P} - \vec{P}_j) \circ \vec{n}_j| < |(\vec{P} - \vec{P}_p) \circ \vec{n}_p| \quad \wedge \quad \arccos(\vec{n} \circ \vec{n}_j) < \arccos(\vec{n} \circ \vec{n}_p). \quad (4.17)$$

Die Zerteilung von Regionen (Zeile 12 des Algorithmus) ist notwendig, da aus einer Region Punkte gestohlen werden können, sodass die Region nicht mehr zusammenhängend ist. In Abbildung 4.7 ist das Ergebnis des erweiterten Regionenwachstums dargestellt. Es ist anzumerken, dass die Wahl der Startpunkte im ersten Schritt immer noch zu unterschiedlichen Ergebnissen führt, aber keine Regionen mehr auftreten, in denen ein Segment in ein anderes „hineinwächst“.

Die Erweiterung des Regionenwachstum liegt dabei ebenfalls in $O(p)$ mit $p = width \cdot height$, da jeder Pixel höchstens von einem anderen Segment gestohlen werden kann.

Insgesamt werden beim Regionenwachstum zwei Parameter, d_{grow} und θ_{grow} , verwendet, um die Kompatibilität eines Punktes zum Segment zu testen. Im Gegensatz zur Vernetzung wird hier aber nicht der Abstand zweier Punkte beziehungsweise der Winkel zwischen Sichtstrahl und Verbindungsvektor betrachtet, sondern der Abstand eines Punktes zu einem planaren Segment beziehungsweise der Winkel zwischen zwei Normalen. Als Abstandsparameter d_{grow} eignet sich auch hier wieder die Verwendung der Strukturgröße δ_s . Bei hohen Messabweichungen kann diese jedoch nicht eingehalten werden, sodass - wie schon bei der Vernetzung - der Zusammenhang

$$d_{grow} = \max(\delta_s, 2 \cdot \sigma_s(z)) \quad (4.18)$$

sinnvoll ist. Als z -Wert kann dabei die z -Komponente des aktuell betrachteten Punktes verwendet werden. Der Winkelparameter θ_{grow} sollte möglichst klein gewählt werden, um auch unterschiedliche Segmente mit ähnlicher Normale noch gut trennen zu können. Je größer allerdings die Messabweichungen sind, desto mehr verrauscht sind die Positionen der Punkte und demnach auch deren Normale, die anhand der umgebenden Dreiecke im Netz bestimmt wurde. Der Parameter sollte deshalb von der Stärke der Messabweichungen σ_1 (siehe Definition 3.18) abhängen. Da das Netz bereits geglättet wurde und so der quadratische Zusam-

menhang $\sigma_s = \sigma_1 \cdot z^2$ bereits abgeschwächt wurde, hat sich hier der Zusammenhang

$$\theta_{grow} = a \cdot \sigma_1 + b \quad (4.19)$$

ohne Einbeziehung von z empirisch als gut erwiesen, wobei für $a = 97.75$ und $b = 0.34$ gewählt wurde. Für eine gegebene Kamera mit bekannten Wert von σ_1 ist der Parameter also konstant. Dies entspricht auch dem Vorgehen von Holz et al., die ebenfalls einen konstanten Wert für Winkelschwellen benutzen.

3. Tiefenkorrekte, strukturgrößenabhängige Filterung

Ein wichtiges Thema der Bildverarbeitung ist die Reduzierung beziehungsweise die korrekte Berücksichtigung des Messrauschens. Ein Effekt von verrauschten Tiefendaten sind beispielsweise kleine isolierte Segmente, dargestellt als weiße Pixel in Abbildung 4.7. Eine wichtige Klasse von Verfahren der 2D-Bildverarbeitung, die unter anderem auch Rauschen reduzieren können, sind lokale Filtermethoden. Dazu wird eine 2D-Filtermaske über das Bild geschoben und so eine lokale Nachbarschaft erzeugt, die einbezogen wird, um eine Operation auf dem Referenzpixel durchzuführen. Der Vorteil dieser Methoden ist deren Einfachheit trotz einer Vielzahl an möglichen Operationen (beispielsweise Glättungsfilter, Kantenfilter, morphologische Filter).

Da bei der Verarbeitung von organisierten Punktwolken ebenfalls ein Pixelgitter vorhanden ist, soll in diesem Kapitel das Konzept der Filtermasken übertragen werden. Dies hat den Vorteil, dass in den folgenden Schritten der Rekonstruktion bekannte Filteroperationen auf Segmentierungen von Punktwolken angewandt werden können. Damit eine robuste Filterung möglich ist, sollte diese tiefenkorrekt und strukturgrößenabhängig sein.

- **Tiefenkorrektheit:** Es soll berücksichtigt werden, dass benachbarte Pixel nicht notwendigerweise auch in 3D benachbart sind, beispielsweise, wenn an dieser Stelle ein Tiefensprung vorliegt.
- **Strukturgrößenabhängigkeit:** Die Stärke der Filterung soll von der Strukturgröße abhängig beziehungsweise beeinflussbar sein.

Im Folgenden werden nun verallgemeinerte Filtermasken vorgeschlagen, die obige Eigenschaften berücksichtigen. Zuerst wird eine Filtermaskengröße in Abhängigkeit von der Strukturgröße δ_s und der Position (u, v) im Bild berechnet, indem mithilfe der intrinsischen Kameraparameter

$$K = \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \quad (4.20)$$

die Anzahl der Pixel berechnet werden, die der Strukturgröße δ_s in der Tiefe des Punktes an Position (u, v) entsprechen:

$$n_x(u, v) = \frac{\delta_s \cdot f_x}{P_z(u, v)}, \quad (4.21)$$

$$n_y(u, v) = \frac{\delta_s \cdot f_y}{P_z(u, v)}. \quad (4.22)$$

$P_z(u, v)$ ist dabei die z -Komponente des Punktes an Pixelposition (u, v) . Die endgültige Filtermaske $M(u, v)$ besteht anschließend aus allen Pixeln innerhalb einer rechteckigen Filtermaske M_{\square} der Größe $n_x \times n_y$ um (u, v) , die mit dem Referenzpixel (u, v) im Netz durch Kanten innerhalb des rechteckigen Bereichs verbunden sind:

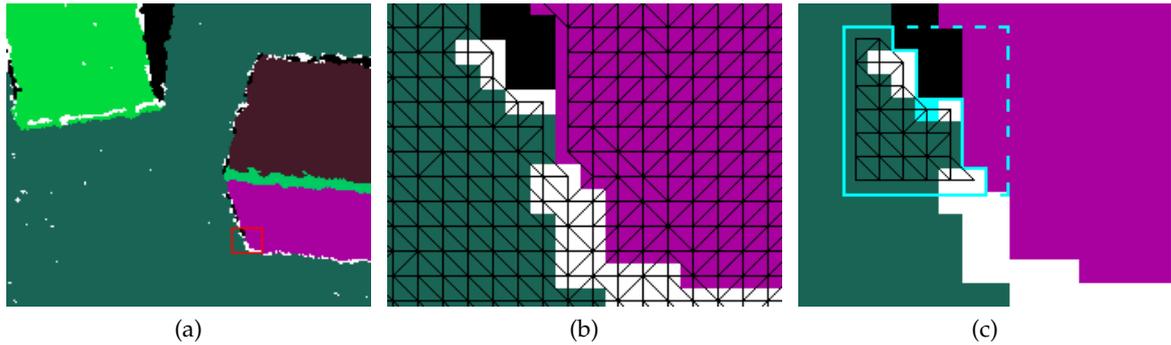


Abbildung 4.8: In (a) ist eine vergrößerte Darstellung der Regionen aus Abbildung 4.7a dargestellt. Schwarze Pixel enthalten keine Tiefenwerte, weiße Pixel sind kleine isolierte Regionen mit weniger als 200 Punkten. Alle anderen Regionen sind zufällig eingefärbt. Der rot markierte Bereich ist in (b) mit eingezeichnetem Netz zu sehen. In (c) ist die Filtermaske (durchgezogene blaue Umrandung) für ein Referenzpixel (blau hinterlegt) zu sehen. Sie entsteht, indem mithilfe der Strukturgröße eine rechteckige Filtermaske erzeugt wird (gestrichelte Umrandung) und darin die mit dem Referenzpixel verbundenen Pixel betrachtet werden.

$$M_{\square}(u, v) = \{ (u', v') : \begin{array}{l} |u' - u| < 0.5 \cdot n_x(u, v) \wedge \\ |v' - v| < 0.5 \cdot n_y(u, v) \end{array} \}, \quad (4.23)$$

$$M(u, v) = \{ (u', v') \in M_{\square} : \exists \text{ Pfad im Netz von } (u', v') \text{ nach } (u, v) \text{ innerhalb } M_{\square} \}. \quad (4.24)$$

Dadurch entstehen nicht rechteckige Filtermasken, wie in Abbildung 4.8 dargestellt. Diese pro Pixel unterschiedlichen Filtermasken sind strukturgrößenabhängig und tiefenkorrekt und können für unterschiedliche Filteroperationen genutzt werden.

Die Berechnung der Filtermasken kann pro Pixel parallel ausgeführt werden. Die Bestimmung der mit dem Referenzpixel verbundenen Pixel ist beispielsweise mit dem *connected-components-labelling*-Algorithmus [Stockman01] möglich. Zudem sei angemerkt, dass die Berechnung der Filtermasken bereits parallel zum Regionenwachstum ausgeführt werden kann, da zur Bestimmung der Masken ausschließlich das Netz und keine Regionen benötigt werden.

3.a Rangordnungsfilter

Durch Rauschen entstehen oft kleine isolierte Segmente (weiße Löcher in Abbildung 4.7). Diese können nun einfach mithilfe der Filtermasken entfernt werden, indem analog zu einem Rangordnungsfilter vorgegangen wird: Innerhalb der Filtermaske wird die am häufigsten auftretende Region ermittelt und diese dem Referenzpixel zugewiesen:

- 1 Bestimme alle Pixel P_{noise} , die zu Regionen mit weniger als $v_{segment}$ Punkten gehören.
- 2 **für alle** $p \in P_{noise}$ **tue**
- 3 Sei M die Filtermaske an der Position von p .
- 4 Bestimme die in M am häufigsten auftretende Region R_{max} .
- 5 Entferne p aus seiner Region und füge p zu R_{max} .

Wie auch von Arbeiter et al. [Arbeiter14] angemerkt, können durch Regionenwachstumsverfahren Übersegmentierungen auftreten, die mittels einer Verschmelzung benachbarter Regio-

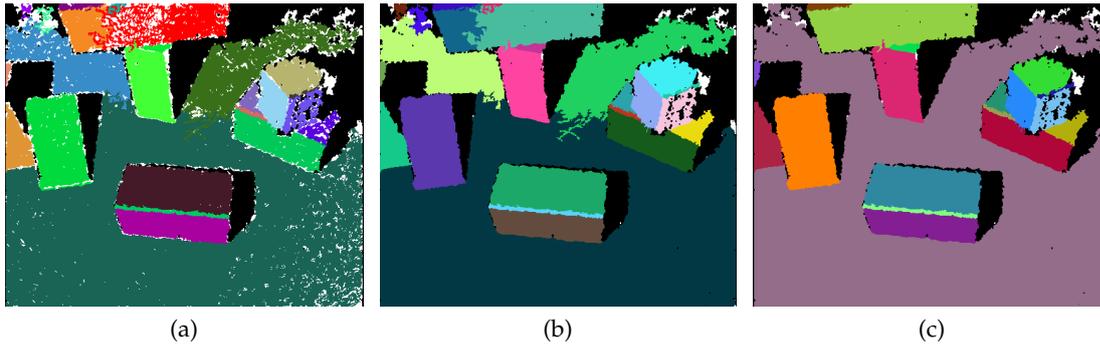


Abbildung 4.9: (a) Ausgangssegmentierung mit vielen kleinen isolierten Segmenten (weiße Pixel), (b) mithilfe des Rangordnungsfilters gefilterte Segmentierung, (c) zusätzliche Verschmelzung von benachbarten Segmenten in der gleichen Ebene

nen mit gleicher Ebenengleichung verbessert werden kann. Abbildung 4.9b zeigt eine solche Übersegmentierung. Ein Grund dafür ist, dass zwei Regionen der gleichen Ebene durch isolierte kleine Regionen getrennt sein könnten und nun, nach der Filterung, zu Nachbarn geworden sind. In Abbildung 4.9c ist das Ergebnis nach der Verschmelzung dargestellt.

3.b Morphologische Filterung

Die gefilterte Segmentierung ist bereits eine gute Unterteilung der Punktwolke in planare Teilstücke, die für die weiteren Schritte der Polygonalisierung und B-Rep-Erzeugung genutzt werden kann. Ein Punkt wurde bisher jedoch noch nicht sichergestellt: Die Beachtung der Strukturgröße (siehe Abschnitt 3.3.1). Es können noch Segmente vorhanden sind, die kleiner oder schmäler als die Strukturgröße sind (Abbildung 4.10a). Das Entfernen dieser Segmente ist mithilfe der erweiterten Filtermasken leicht möglich, indem analog zum morphologischen *Opening*-Filter vorgegangen wird, also einer Hintereinanderausführung von Erosion und Dilatation:

```

// Erosion
1 für alle Pixel p tue
2   Sei M die Filtermaske an der Position von p,  $R_p$  die Region von p.
3   wenn M ein Pixel mit einer Region  $R \neq R_p$  enthält dann
4     Entferne p aus seiner Region.
// Dilatation
5 für alle Pixel p ohne zugeordnete Region tue
6   Sei M die Filtermaske an der Position von p.
7    $R = \{\}$ .
8   für alle Pixel  $q \in M$  mit zugeordneter Region  $R_q$  tue
9     Teste p auf Kompatibilität zu  $R_q$ .
10    wenn kompatibel dann Füge  $R_p$  zu R hinzu.
11   Füge p zur am besten passenden Region in R hinzu.
    
```

Im Erosions-Schritt werden alle Segmente verkleinert, indem Pixel, in deren Nachbarschaft im Netz eine andere Region liegt, entfernt werden (Abbildung 4.10b). Anschließend werden die entfernten Pixel wieder hinzugefügt und zwar zu der Region, zu der sie am besten passen. Dies ist sinnvoll, da es sich hier besonders um Stellen handelt, die zwischen zwei oder mehr Regionen liegen. Wie schon beim Regionenwachstum kann über den Abstand des Punktes \vec{P}

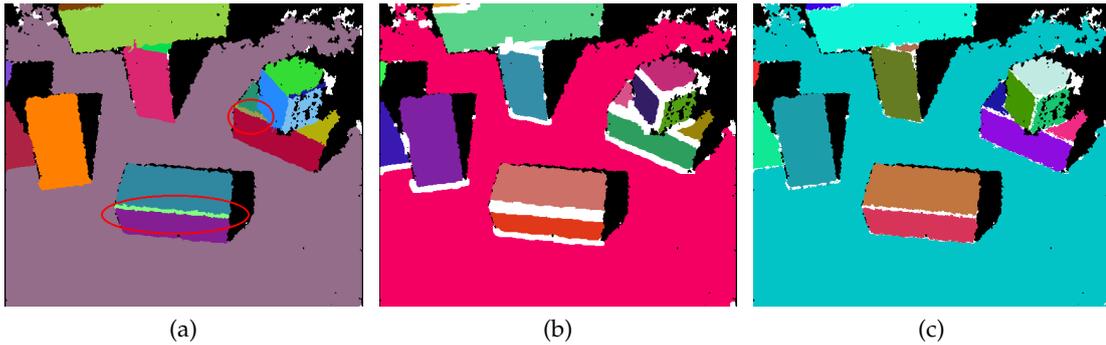


Abbildung 4.10: (a) Segmentierung nach Ausführung des Rangordnungsfilters mit schmalen Segmenten, die kleiner als die Strukturgröße sind (rot markiert), (b) Segmentierung nach dem Erosionsschritt, (c) Segmentierung nach dem Dilatationsschritt

zur Region R mit Schwerpunkt \bar{P} und Normale \bar{n}

$$(\vec{P}, R) = |(\vec{P} - \bar{P}) \circ \bar{n}| \quad (4.25)$$

bestimmt werden, welche Region am besten passt oder ob eine Region kompatibel ist:

$$d(\vec{P}, R_q) < d_{dilatation}. \quad (4.26)$$

Der Parameter $d_{dilatation}$ kann analog zum Regionenwachstum von der Strukturgröße abhängig gemacht werden. Abbildung 4.10c zeigt das Ergebnis der morphologischen Filterung. Im Vergleich zum Ausgangsbild 4.10a bleiben nun nicht zugeordnete Pixel übrig, da diese zu keinem der Nachbarsegmente kompatibel sind. Dies tritt zum Beispiel bei abgerundeten Kanten auf, deren Radius kleiner als die Strukturgröße ist.

4.2.2 Polygonalisierung

Ziel der Polygonalisierung ist es, die zuvor gefundenen flächigen Segmente in Polygone umzuwandeln.

Gegeben ist die aus der Segmentierung erhaltene Menge an Segmenten

$$S = \{R_1, \dots, R_j, \dots, R_{|S|}\}, \quad \text{mit } R_j = \{\vec{P}_i\}. \quad (4.27)$$

Gesucht ist eine Menge an Polygonen

$$T = \{f_1, \dots, f_j, \dots, f_{|S|}\}, \quad (4.28)$$

wobei jedes Polygon f_j ein dreidimensionales, planares, einfaches Polygon mit optionalen Löchern ist. Im Detail bedeuten diese Eigenschaften:

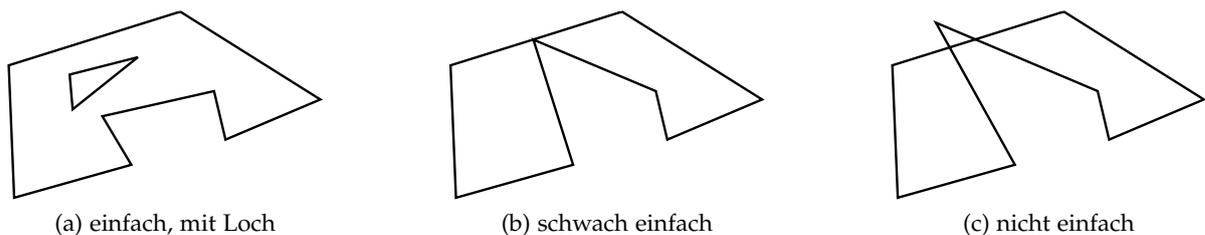


Abbildung 4.11: In einfachen Polygonen dürfen sich Knoten oder Kanten weder berühren noch schneiden. In schwach einfachen Polygonen sind Berührungen erlaubt.

- Mit Löchern: Das Polygon besteht aus einem äußerem geschlossenem Polygonzug sowie optional aus weiteren inneren geschlossenen Polygonzügen, die Löcher repräsentieren.
- Einfach: Kanten oder Ecken des Polygons schneiden oder berühren sich nicht (Abbildung 4.11).
- Dreidimensional: Die Eckpunkte des Polygons liegen im dreidimensionalen Raum (\mathbb{R}^3).
- Planar: Alle Eckpunkte liegen in einer Ebene, nämlich der Ebene des Segments.

Die Repräsentation der Polygone erfolgt durch Angabe der Eckpunkte. Damit der Umlaufsinn eindeutig ist, gelte analog zu den B-Rep-Flächen folgende Definition: Die Eckpunkte des äußeren Rands eines Polygon sind gegen den Uhrzeigersinn, die Eckpunkte eines inneren Polygonzugs sind im Uhrzeigersinn sortiert, wenn das Polygon vom Ursprung aus betrachtet wird. Dies entspricht einer Betrachtung der Oberfläche von außen, also aus der Richtung, in die die Normalen zeigen, da die Punktwolken im lokalen Sensorkoordinatensystem vorliegen.

Auch in diesem Schritt soll ausgenutzt werden, dass es sich um eine organisierte Punktwolke handelt. Das heißt jeder dreidimensionale Punkt hat auch eine eindeutige Position im Pixelgitter, die mathematisch über die Kameramatrix zusammenhängt. Daher wird ein Konturverfolgungsansatz gewählt, der im Vergleich zu Algorithmen zur Bestimmung der konkaven Hülle die Struktur ausnutzen kann (siehe Abschnitt 4.1.2). Insgesamt besteht die Polygonalisierung aus vier Schritten:

1. **Knotenbasierte Konturverfolgung:** Mithilfe eines 2D-Konturverfolgungsansatzes wird für jedes Segment ein 3D-Polygon erzeugt, das alle obigen Eigenschaften erfüllt, aber noch sehr fein ist, d.h. aus sehr vielen Eckpunkten besteht.
2. **Anpassung von inzidenten Kanten:** In diesem Schritt werden die Polygone an Stellen vereinfacht, an denen zwei Segmente benachbart sind, indem Segmente paarweise betrachtet werden und dabei Kanten angepasst werden. Diese Kanten sind die Vorstufe zu inzidenten Kanten eines B-Reps (Definition siehe Abschnitt 3.2.2).
3. **Anpassung von pseudo-inzidenten Kanten:** In diesem Schritt werden Kanten an Stellen angepasst, an denen das Segment einen Tiefensprung besitzt. Diese Kanten sind die Vorstufe zu pseudo-inzidenten Kanten (Definition siehe Abschnitt 3.2.2).
4. **Anpassung von virtuellen Kanten:** Zuletzt werden die übrigen Kanten vereinfacht.

Im Folgenden werden nun alle vier Schritte genauer erläutert.

1. Knotenbasierte Konturverfolgung

Ziel der Konturverfolgung ist es, für ein Segment R ein Polygon f zu bestimmen. Aufgrund der Verfügbarkeit des Pixelgitters bietet es sich an, ein 2D-Konturverfolgungsverfahren zu nutzen. Da Knotenverfolgungsverfahren die Begrenzung auf den Rand zwischen Vordergrund und Hintergrund legen (Abbildung 4.2), liefern sie stets ein einfaches Polygon und sind daher besonders gut geeignet. Im Folgenden wird deshalb ein einfach zu implementierendes Verfahren vorgestellt, das im Rahmen einer am Lehrstuhl des Autors angefertigten Bachelorarbeit [Baumgartl09] entwickelt wurde und der Idee einer Fallunterscheidungstabelle zur Knoten-basierten Verfolgung von Miyatake et al. [Miyatake97] folgt. Grundsätzlich ist aber auch jedes andere Verfahren zur knoten-basierten Konturverfolgung anwendbar. Anschließend wird erläutert, wie durch Projektion aus einer 2D-Kontur ein dreidimensionales Polygon erzeugt werden kann.

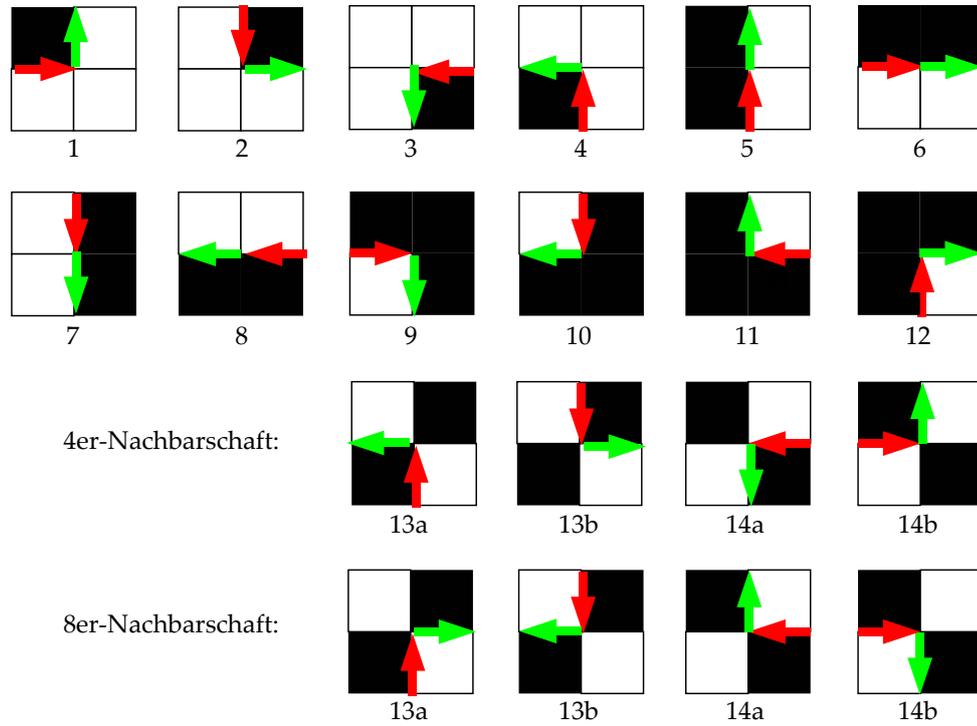


Tabelle 4.1: Fallunterscheidungen bei der Verfolgung einer Kontur. Schwarze Pixel stellen das Segment dar, weiße Pixel den Hintergrund. Der rote Pfeil zeigt die zuletzt verfolgte Richtung an, der grüne Pfeil die nächste Richtung. In den Fällen 13 und 14 wird unterschieden zwischen der vorhergehenden Richtung (Fälle a oder b) und ob eine 4er oder 8er Nachbarschaft betrachtet wird. Auswirkungen unterschiedlicher Nachbarschaften sind in Abbildung 4.12 zu sehen.

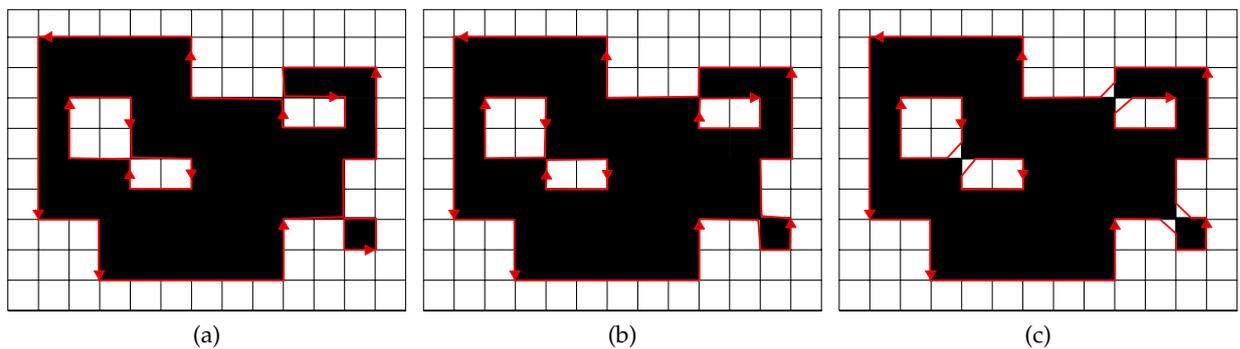


Abbildung 4.12: Beispielsegment (schwarz) und dessen Kontur (rot). (a) Bei Verwendung einer 4er-Nachbarschaft entstehen zwei Polygone und ein Loch. (b) Bei Verwendung einer 8er-Nachbarschaft entsteht ein Polygon mit drei Löchern. (c) 8er-Nachbarschaft unter Verwendung schräger Verbindungen bei den Fällen 13 und 14.

2D-Konturverfolgung Eine Region, dargestellt als 2D-Binärbild, wird durchlaufen, um einen Übergang von Hintergrund zum Vordergrund zu finden. Dieser Übergang wird nun anhand von Fallunterscheidungen (Tabelle 4.1) verfolgt, bis der Ausgangspunkt wieder erreicht ist. Anschließend wird das Bild weiter durchlaufen, bis alle Konturen gefunden wurden. Es entstehen dabei sowohl Konturen, die den äußeren Rand des Segments darstellen, als auch Konturen von Löchern im Segment. Beide Arten können anhand des Umlaufsinn unterschieden werden: Äußere Ränder werden gegen, Löcher im Uhrzeigersinn durchlaufen. Bei der Betrachtung, ob ein Segmentpixel zu einem anderen verbunden ist, kann eine 4er oder 8er-Nachbarschaft verwendet werden. Je nachdem unterscheiden sich zwei der vierzehn Fälle der Tabelle 4.1. Die Auswirkungen auf das Ergebnis sind in Abbildung 4.12 dargestellt. Da durch die Verwendung einer 4er-Nachbarschaft mehrere separierte äußere Konturen an vereinzelt Randpixeln entstehen, ist diese eher ungeeignet für den vorliegenden Anwendungsfall.

Erzeugung von 3D-Polygonen Nun soll die 2D-Konturverfolgung auf die Regionen der Segmentierung angewandt werden. Existiert kein Rauschen, das heißt, liegen alle Punkte eines Segments exakt in einer Ebene, so liefert folgende Methode nahezu korrekte 3D-Polygone:

- M1**
- 1 Fasse ein Segment als Binärbild auf.
 - 2 Führe die 2D Konturverfolgung durch.
 - 3 Ersetze 2D-Pixel (u, v) durch deren 3D-Punkt $P(u, v)$.

Mit dieser Methode erzeugte 3D-Polygone sind allerdings nur schwach einfach. Da an den Eckpunkten der Fälle 13 und 14 die Kontur zweimalig vorbei läuft, berührt sich das resultierende Polygon an dieser Stelle. Um dennoch stets (stark) einfache Polygone zu erhalten, können in diesen beiden Fällen die Ecken um ein halbes Pixel „abgeschrägt“ werden, wie in Abbildung 4.12c zu sehen. Anschließend wird folgende Methode verwendet, die den letzten Schritt durch eine Rückprojektion ersetzt, in der auch nicht-ganzzahlige Pixelkoordinaten verwendet werden können:

- M2**
- 1 Fasse ein Segment als Binärbild auf.
 - 2 Führe die 2D Konturverfolgung durch.
 - 3 Projiziere die 2D-Kontur-Pixel (u, v) mithilfe der Kameramatrix und der Ebenengleichung des Segments zurück nach 3D.

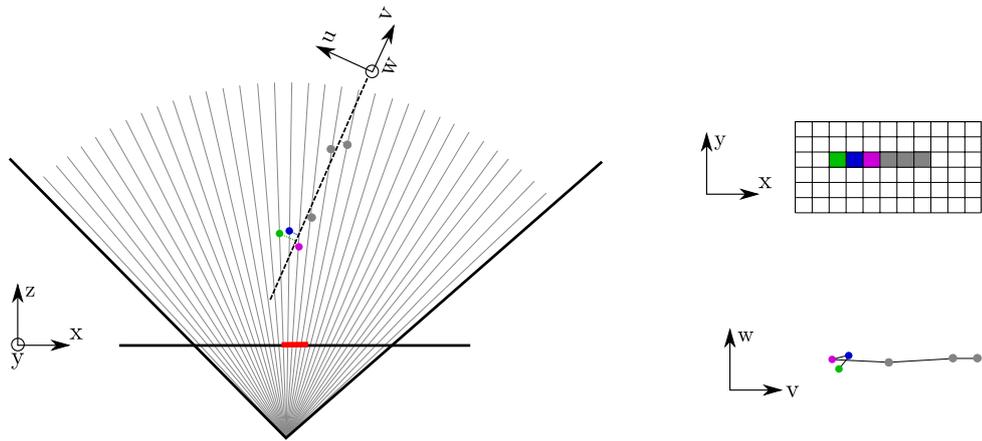
Die so erzeugten Polygone sind nun auch (stark) einfach. Lässt man allerdings Rauschen zu, so erhält man folgendes Resultat:

Bei Methode M1 ist das resultierende Polygon nicht planar, da die zugehörigen 3D-Punkte nun nicht mehr exakt auf einer Ebene liegen. Projiziert man alle resultierenden Eckpunkte anschließend noch auf die Ebene

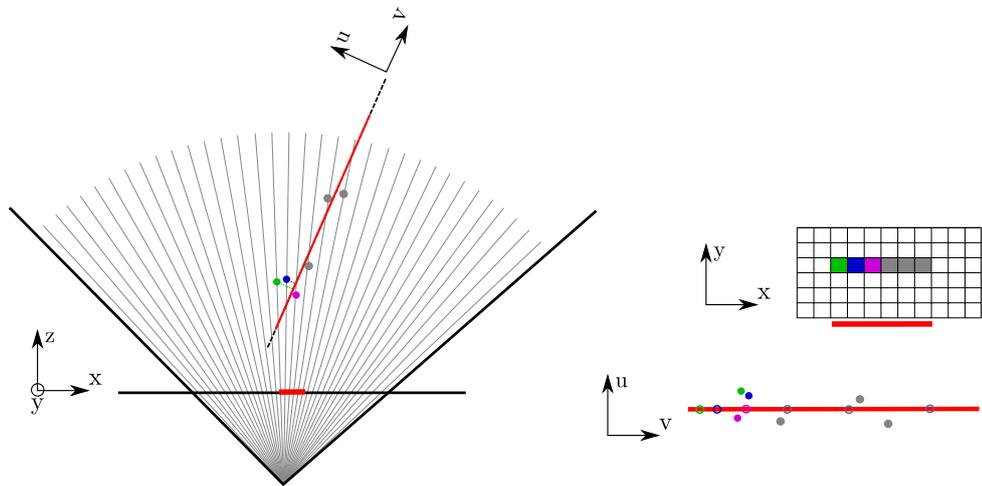
- M3**
- 1 Fasse ein Segment als Binärbild auf.
 - 2 Führe die 2D Konturverfolgung durch.
 - 3 Ersetze 2D-Pixel (u, v) durch deren 3D-Punkt $P(u, v)$.
 - 4 Projiziere 3D-Punkt auf die Ebene des Segments.

so kann es auftreten, dass das resultierende Polygon Selbstüberschneidungen beinhaltet. Der Grund ist in Abbildung 4.13a dargestellt: Durch die Projektion kann die Reihenfolge der Pixel vertauscht werden, sodass die Reihenfolge der 2D-Konturverfolgung Selbstüberschneidungen produziert.

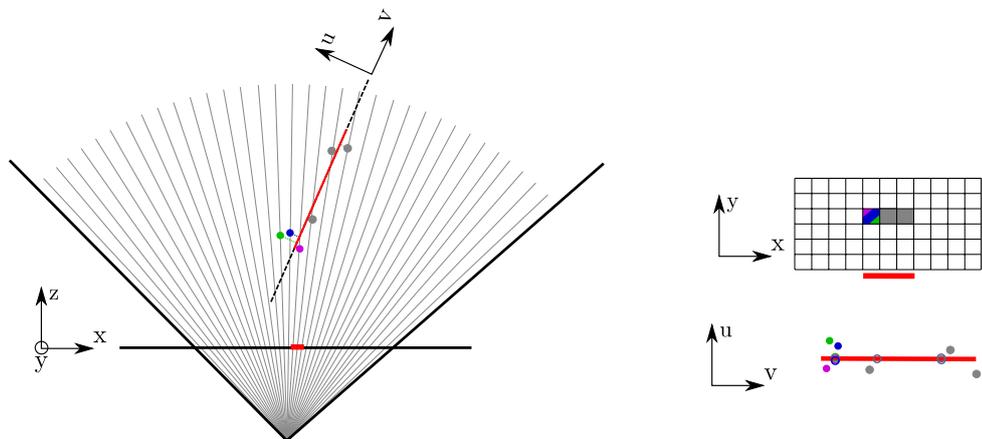
Bei Methode M2 ist das resultierende Polygon korrekt, allerdings beschreibt diese Kontur das



(a) Methode M3: Bei direkter Verwendung der zu einem Pixel zugehörigen 3D-Punkte können nicht einfache Polygone entstehen, da die Reihenfolge der Konturpixel (x -Richtung) nicht der korrekten Reihenfolge der 3D-Punkte (v -Richtung) entsprechen muss.



(b) Methode M2: Bei Rückprojektion der Konturpixel in 3D kann ein zu großes Polygon (rot) entstehen, da der Abstand eines Punktes zum rückprojizierten Punkt größer sein kann als der Abstand zur Ebene. Der Effekt ist umso größer, je mehr die Ebene gegenüber der Bildebene geneigt ist.



(c) Methode M4: Werden die 3D-Punkte zuerst auf die Ebene und anschließend zurück in das 2D-Gitter projiziert, ist das Polygon passend groß und stets einfach.

Abbildung 4.13: Erzeugung eines 3D-Polygons anhand der Kontur. Links: 3D-Punkte im Sichtkegel der Kamera mit Segmentebene (gestrichelt), Bildebene (horizontale Linie) und Strahlen durch die Pixel (grau)

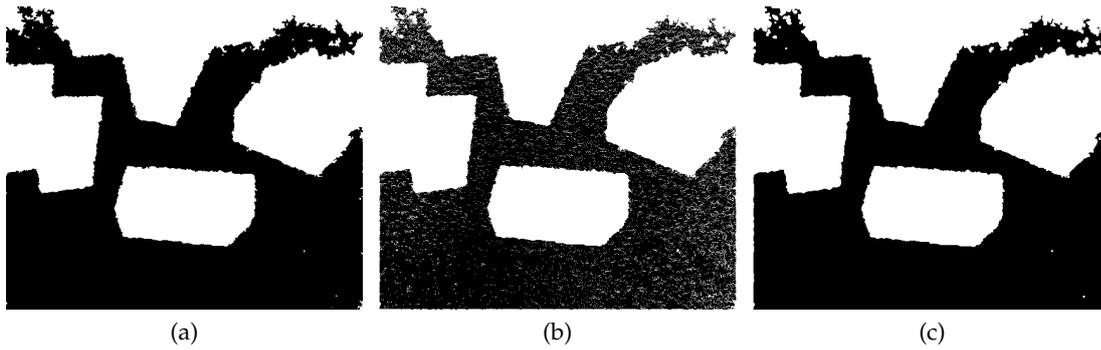


Abbildung 4.14: Das größte Segment aus Abbildung 4.10c als Binärbild: (a) Ausgangssegment, (b) Projektion der Punkte auf die Ebene und anschließende Projektion ins 2D-Pixelgitter, (c) Bild aus (b) mit einem 3x3 Closing-Filter gefiltert

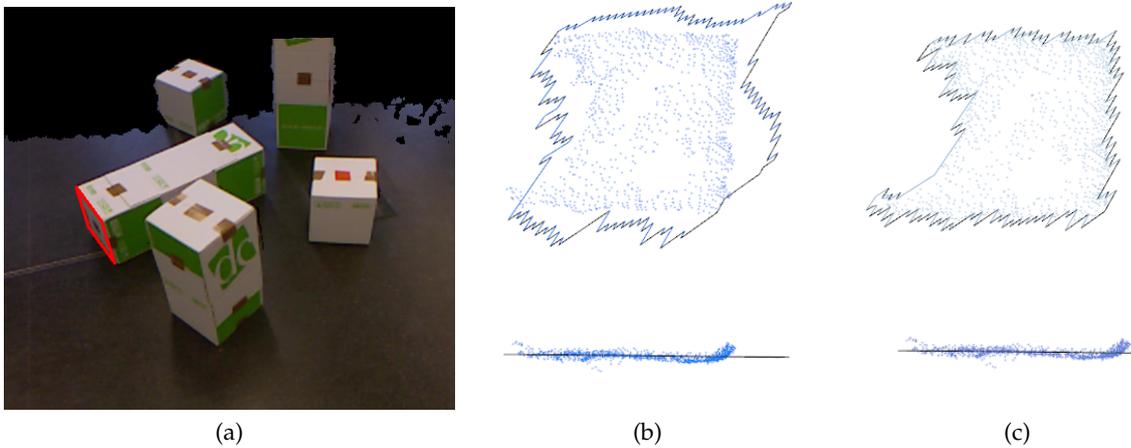


Abbildung 4.15: Die Methoden M2 und M4 unterschieden sich insbesondere bei stark zur Bildebene geneigten Segmenten, wie das rot markierte Segment in (a). Die Kontur bei Variante M2 (b) ist deutlich weiter entfernt von den Punkten (blau) als bei Variante M4 (c). In beiden Abbildungen ist oben jeweils die Frontalansicht, unten eine Draufsicht dargestellt.

Segment bei Ebenen, die stark geneigt gegenüber der Sichtachse sind, nur ungenau, wenn Punkte am Segmentrand verrauscht sind (Abbildung 4.13b). Um dies zu verhindern, muss vor der Konturverfolgung projiziert werden:

- M4**
- 1 Projiziere alle Punkte eines Segments auf dessen Ebene.
 - 2 Projiziere alle Punkte nach 2D und erzeuge so ein Binärbild.
 - 3 Führe einen 3x3-Closing-Filter auf dem Binärbild aus.
 - 4 Führe die 2D Konturverfolgung durch.
 - 5 Projiziere die 2D-Kontur-Pixel (u, v) mithilfe der Kameramatrix und der Ebenengleichung des Segments zurück nach 3D.

In Abbildung 4.13c ist das Ergebnis zu sehen. Der Closing-Filter ist notwendig, da durch das Rauschen die Punkte nach der Projektion auf die Ebene so liegen, dass einzelne Pixel im Inneren des Segments nicht belegt werden (Abbildung 4.14). Im Vergleich zur Methode M2 sind nun allerdings die „überstehenden“ Ränder nicht mehr vorhanden (Abbildung 4.15).

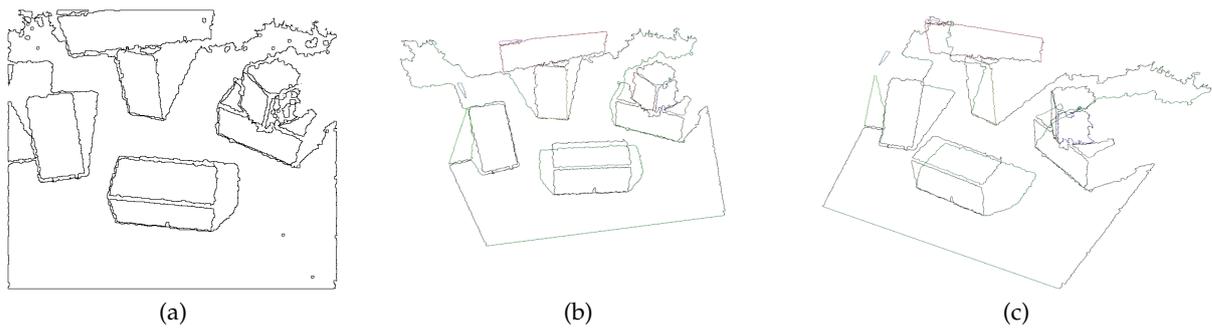


Abbildung 4.16: Ergebnis der Polygonalisierung der Segmentierung aus Abbildung 4.10c mit Methode M4: (a) Konturen im 2D-Pixelgitter, (b-c) 3D-Konturpolygone aus zwei verschiedenen Ansichten. Löcher mit weniger als $v_{Loch} = 200$ Pixeln Flächeninhalt wurden entfernt.

Zusammenfassend lässt sich sagen, dass Methode M2 bei unverrauschten Daten, und Methode M4 bei verrauschten Daten zu wählen ist. Sind in der Punktwolke Pixel vorhanden, für die keine Messung vorliegt (z.B. durch Reflexionen), so können Löcher innerhalb eines Segments entstehen. In diesen Fall können beispielsweise noch alle Löcher verworfen werden, die weniger als v_{Loch} Pixel besitzen. Das Ergebnis der Polygonalisierung für die Segmentierung der Beispielpunktwolke aus dem vorhergehenden Abschnitten ist in Abbildung 4.16 dargestellt.

2. Anpassung von inzidenten Kanten

Durch die Konturverfolgung ist eine Begrenzung der Segmente entstanden, die sehr fein ist, das heißt aus vielen Punkten besteht. In diesen und den folgenden Schritten wird die Anzahl der Konturpunkte reduziert, indem mehrere aufeinander folgende Punkte durch einzelne größere Kantenstücke ersetzt werden. In diesem Schritt werden zuerst Kanten gefunden, an denen sich zwei Segmente berühren. Diese Kanten sind die Vorstufe zu inzidenten Kanten des späteren B-Reps (Abbildung 4.17).

Um diese Kanten anzupassen, werden zunächst alle Paare von benachbarten Segmenten gebildet. Zwei Segmente sind benachbart, wenn sie über Kanten des Netzes verbunden sind ohne durch ein anderes Segment zu laufen. Segmente, die zwar in 2D nebeneinander liegen, aber durch einen Tiefensprung getrennt sind, sind so nicht enthalten. Für jedes Paar (R_A, R_B) mit deren Konturpolygonen f_A und f_B wird folgendes Vorgehen angewandt:

- 1 für alle Paare (R_A, R_B) tue
- 2 Berechne Schnittgerade X
- 3 Finde Intervalle I_i in f_A in der Umgebung von X
- 4 Finde Intervalle J_j in f_B in der Umgebung von X
- 5 Bilde sich überlappende Intervall-Paare (a_i, b_j)
- 6 Ersetze Konturpunkte in f_A und f_B

Die Schnittgerade X kann über die Ebenengleichungen der Segmente berechnet werden und wird dargestellt als

$$X = \vec{A} + t \cdot \vec{v} \quad (4.29)$$

mit Aufpunkt $\vec{A} \in \mathbb{R}^3$, Richtung $\vec{v} \in \mathbb{R}^3$, $\|\vec{v}\|_2 = 1$ und dem freien Parameter $t \in \mathbb{R}$.

Ein Intervall I_i ist definiert als eine Menge von aufeinander folgenden Konturpunkten, deren Abstand zur Geraden X kleiner ist als eine Schwelle $d_{inzident}$. In Abbildung 4.18a sind solche

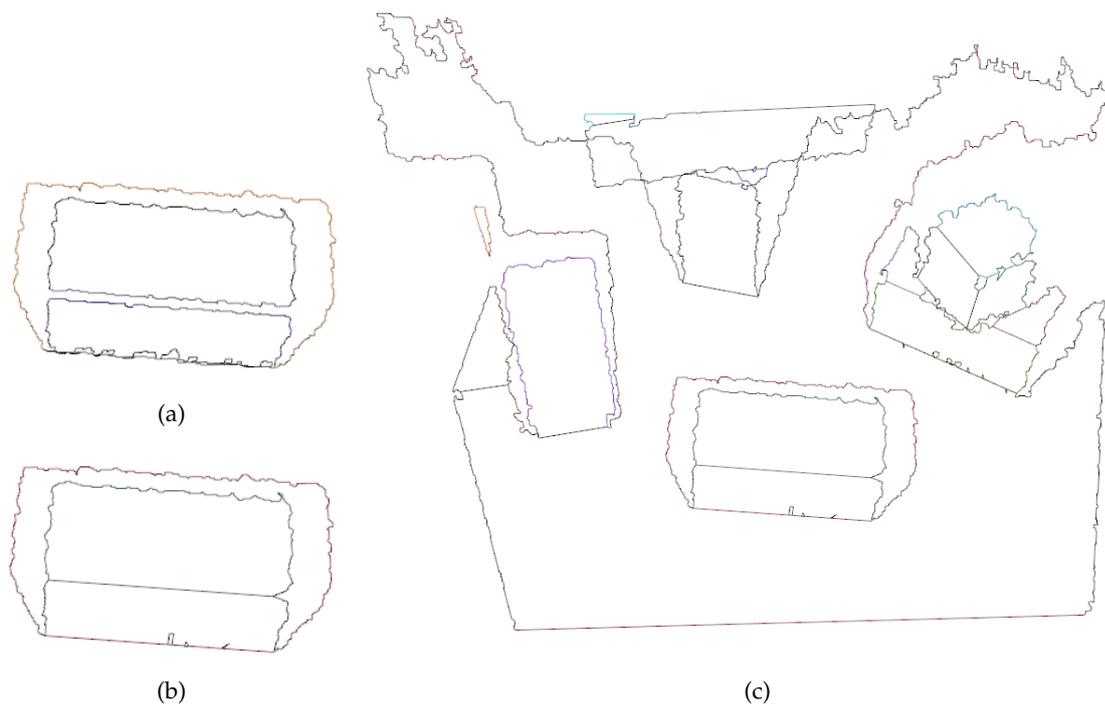
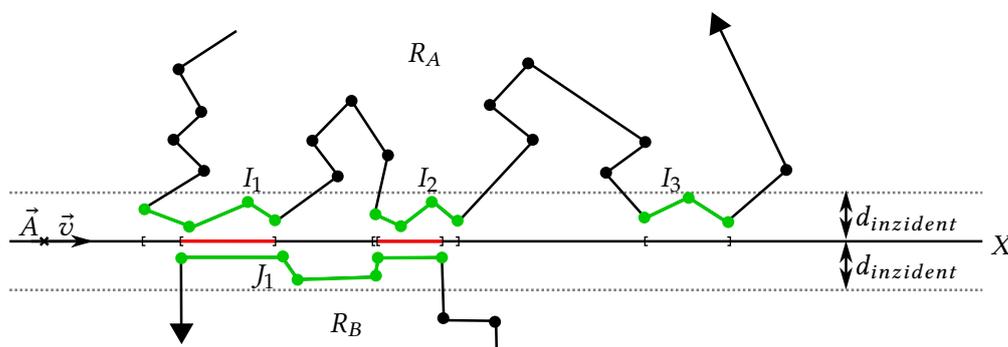
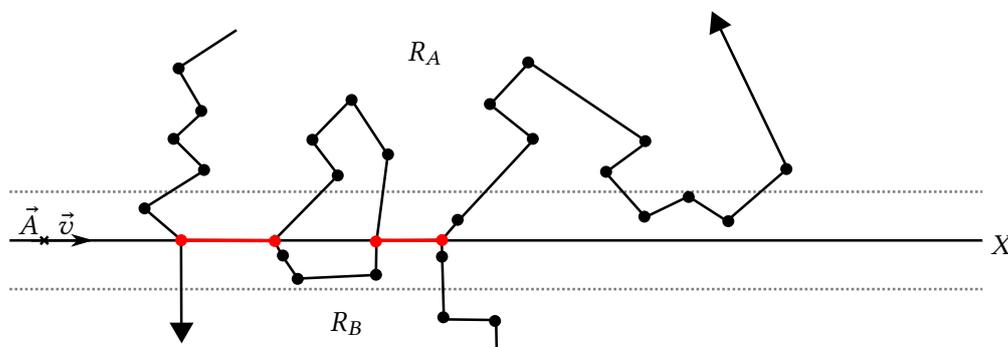


Abbildung 4.17: (a) 3D-Konturen des unteren Teils aus Abbildung 4.16b, (b) 3D-Konturen mit ersetzt inzidenten Kanten, (c) alle Segmente mit ersetzt inzidenten Kanten



(a) Zwei Segmente R_A und R_B und deren Schnittgerade X . Die Punkte der vier grün markierten Intervalle I_1 bis I_3 und J_1 besitzen einen geringeren Abstand als $d_{inzident}$ zur Geraden X . Überlappungen der Intervalle auf X sind rot markiert.



(b) Konturpunkte im Überlappungsbereich werden entfernt und durch eine Kante (rot) auf X ersetzt.

Abbildung 4.18: Anpassung von inzidenten Kanten

Intervalle veranschaulicht. Für jedes Intervall kann ein Bereich $[t_{min}, t_{max}]$ des Geradenparameters angegeben werden, der den abgedeckten Teil der Gerade X beschreibt, indem alle Punkte des Intervalls auf X projiziert und die Werte von t berechnet werden.

Mithilfe der Bereiche können nun Paare von Intervallen aus f_A und f_B gebildet werden, indem die Bereiche auf Überlappung geprüft werden. Es ist zu beachten, dass ein Intervall sich mit mehr als einem Intervall des anderen Segments überlappen kann (siehe rote Bereiche in Abbildung 4.18a).

Abschließend werden alle Intervall-Punkte innerhalb des Überlappungsbereiches gelöscht und durch eine einzige Strecke ersetzt, die auf der Geraden X liegt. Die Endpunkte der Strecken werden gebildet durch die Enden des Überlappungsbereiches (siehe Abbildung 4.18b). Das Ergebnis der Ersetzung von zwei Paaren aus Segmenten ist in Abbildung 4.17a-b dargestellt, das Ergebnis der Anpassung aller Kanten im Beispielfeld aus Abbildung 4.16b ist in Abbildung 4.17c gezeigt.

Der in diesem Schritt verwendete Parameter $d_{inzident}$ kann auch wieder von der Strukturgröße abhängig gemacht werden. Wie man in Abbildung 4.18 erkennt, wird genau dann eine Kante zwischen zwei getrennten Konturen angepasst, wenn beide in einem Korridor von $2 \cdot d_{inzident}$ liegen. Demnach ist es sinnvoll

$$d_{inzident} = 0.5 \cdot \delta_s \quad (4.30)$$

zu wählen. So werden nur Kanten zwischen Konturen angepasst, die höchstens δ_s auseinander liegen.

3. Anpassung von pseudo-inzidenten Kanten

In diesem Schritt werden nun Kanten vereinfacht, die entlang von Tiefensprüngen laufen. Diese Kanten sind die Vorstufe zu pseudo-inzidenten Kanten des späteren B-Reps. Auch hier werden wieder Intervalle in der Kontur gefunden, die anschließend durch größere Kantenstücke ersetzt werden:

- 1 für alle Segmente R_j mit Polygon f_j tue
- 2 Finde Intervalle in f_j , die ein anderes Segment verdecken
- 3 Passe an jedes Intervall einen Polygonzug an
- 4 Ersetze Intervalle durch Polygonzüge

Die einzelnen Schritte des Algorithmus werden im Folgenden genauer erläutert.

Folgende Eigenschaften zeichnen einen Konturpunkt aus, der zu einer pseudo-inzidenten Kante gehört: In der 2D-Pixelnachbarschaft befindet sich ein anderes Segment, das durch einen Tiefensprung vom aktuellen Segment getrennt ist (Abbildung 4.19). Diese Information ist aber bereits implizit im Netz vorhanden: Zwei benachbarte, durch einen Tiefensprung getrennte Pixel sind im Netz nicht durch eine Kante verbunden. Ein Konturpunkt gehört also genau dann zu einer pseudo-inzidenten Kanten, wenn ein nicht verbundenes Nachbarpixel existiert, das zu einem anderem Segment gehört. Da bei Tiefensprüngen, bedingt durch das Aufnahmeprinzip, oft Schatten im Bild entstehen, die keine Messwerte enthalten (schwarze Bereiche in Abbildung 4.19a), muss die Bedingung noch relaxiert werden, sodass beliebig viele Schattenpixel dazwischen liegen dürfen. Da hier nur verdeckende, aber keine verdeckten Kanten gefunden werden sollen, muss zudem überprüft werden, ob es sich um einen Tiefensprung nach hinten oder nach vorne (in z -Richtung) handelt. Ein Intervall aus Zeile 2 des obigen Algorithmus ist also eine Menge aus aufeinanderfolgenden Konturpunkten p , für die ein Punkt q existiert mit folgenden Eigenschaften:

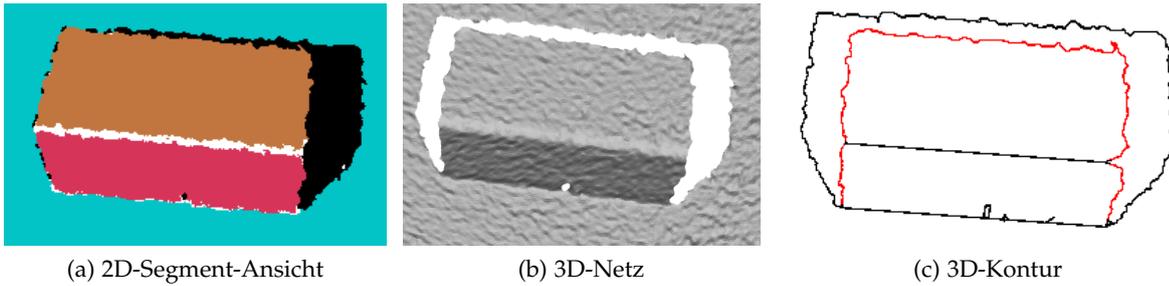


Abbildung 4.19: (a) 2D-Bild (Schattenpixel ohne Messwerte sind schwarz, Pixel ohne Segment, aber mit Messwert sind weiß, die restlichen Pixel zufällig anhand des Segments eingefärbt), (b) 3D-Netz des Ausschnitts, (c) 3D-Kontur mit in rot eingezeichneten gefundenen Konturintervallen. Das obere, im linken Bild braune Segment besitzt ein Konturintervall, das vordere, im linken Bild pinke Segment zwei Konturintervalle.

- q gehört zu einem anderen Segment als p .
- Es existiert ein Weg von p nach q im inversen Netz. Im inversen Netz sind zwei benachbarte Pixel genau dann verbunden, wenn sie im Ursprungsnetz nicht verbunden sind und umgekehrt.
- Der z -Wert des Punktes q ist größer als der von p .

Die resultierenden Intervalle sind in Abbildung 4.19c dargestellt.

Im Gegensatz zu inzidenten Kanten muss in Schritt 3 nun ein offener Polygonzug (*polyline*) anstelle einer Strecke angepasst werden, da die Punkte eines Intervalls nicht notwendigerweise auf einer Geraden liegen müssen. Zur Anpassung eignet sich prinzipiell jeder Approximationsalgorithmus für offene Polygonzüge, als Beispiel sei hier der *Iterative-Endpoint-Fit-Algorithmus* [Douglas73] genannt. Das gesamte Intervall wird hierbei iterativ unterteilt, solange das Ergebnis weiter als eine Schwelle d_{approx} von der ursprünglichen Kontur entfernt ist. Anhand dieser Schwelle kann demnach der Approximationsgrad eingestellt werden. Im letzten Schritt werden die Intervalle durch die angepassten Polygonzüge ersetzt. Das Ergebnis ist in Abbildung 4.20 zu sehen.

Der Parameter d_{approx} sollte dabei wieder von der Strukturgröße abhängen. Bei der Wahl von

$$d_{approx} = 0.5 \cdot \delta_s \quad (4.31)$$

wird erreicht, dass die angepassten Kanten maximal um die halbe Strukturgröße von der Kontur entfernt liegen.

4. Anpassung von virtuellen Kanten

Da die Konturverfolgung eine sehr feine pixel-weise Kontur erzeugt, ist es sinnvoll, auch die übrigen virtuellen Kanten noch zu vereinfachen. Dazu genügt es, auf den bisher noch nicht betrachteten Konturpunkten ebenfalls offene Polygonzüge anzupassen und die Kontur durch diese zu ersetzen (Abbildung 4.21).

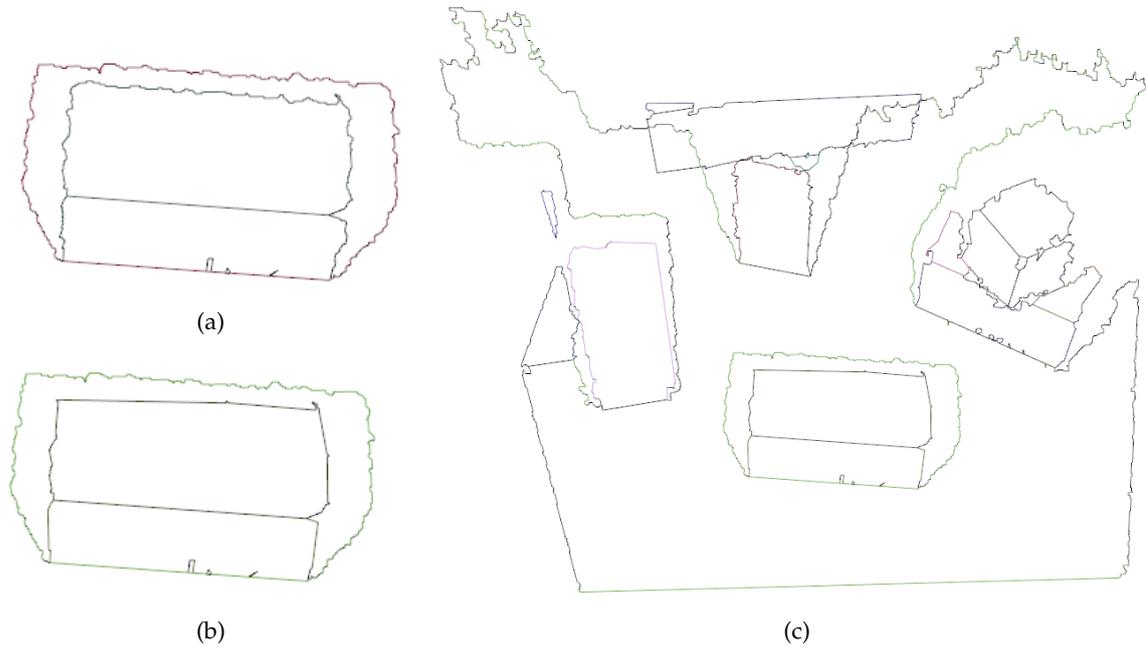


Abbildung 4.20: (a) 3D-Konturen des unteren Teils vor der Anpassung, (b) 3D-Konturen mit ersetzt pseudo-inzidenten Kanten, (c) Alle Segmente mit ersetzt pseudo-inzidenten Kanten

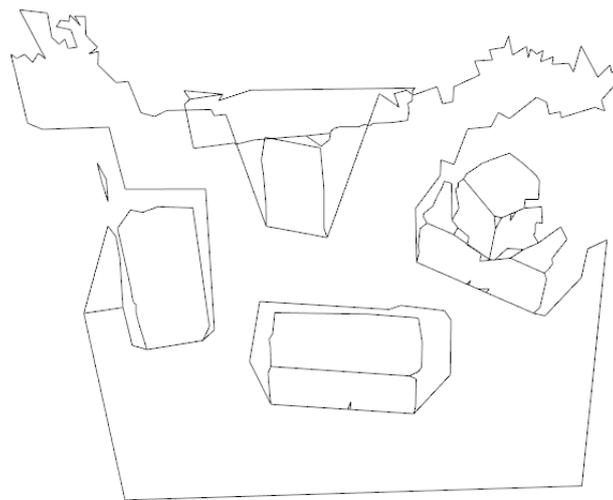


Abbildung 4.21: Konturen nach Anpassung aller drei Arten von Kanten

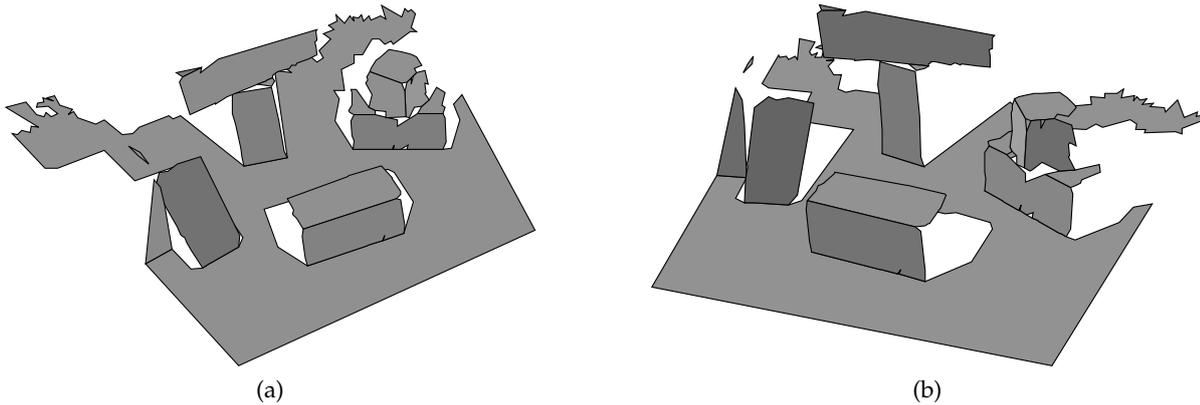


Abbildung 4.22: Zwei Ansichten des erzeugten partiellen B-Rep Modells

4.2.3 B-Rep Erzeugung

Der letzte Schritt der Rekonstruktion partieller B-Reps ist die Umwandlung der Polygone aus dem vorhergehenden Schritt in ein partielles B-Rep-Modell. Durch die Beachtung des Umlaufsinn und die Unterscheidung der Kantenarten in der Polygonalisierung ist die Umwandlung in die B-Rep-Datenstruktur direkt möglich. Durch das paarweise Ersetzen von inzidenten Kanten ist beispielsweise bekannt, welche B-Rep-Kanten als Zwillingskanten verzeigert werden müssen. Zudem werden zwei Informationen für jede Fläche des B-Rep abgelegt: Einerseits die Ebenengleichung einer Fläche, die für Optimierungsschritte genutzt werden kann, und andererseits ein Gewicht. Für eine spätere Fusion von partiellen Modellen ist es wichtig, dass für jede Fläche ein Maß für die Konfidenz dieser Fläche bekannt ist, um gewichtet mitteln zu können. Als Gewicht einer Fläche wird die Anzahl der Punkte des Segments, aus der die B-Rep-Fläche entstanden ist, verwendet. Die Anzahl der Punkte ist aus mehreren Gründen gut als Konfidenz geeignet:

- Die Ebenengleichung einer Fläche kann besser abgeschätzt werden, je mehr Punkte vorhanden sind.
- Flächen, die weit von der Kamera entfernt und damit mehr verrauscht sind, enthalten weniger Punkte als nahe Flächen.
- Flächen, die stark zur Bildebene geneigt und damit mehr verrauscht sind, enthalten weniger Punkte als senkrecht zur Bildebene stehende Flächen.

Das entstandene partielle B-Rep-Modell ist in Abbildung 4.22 zu sehen. Als letztes folgen noch zwei Optimierungsschritte, die sich mit Ecken und konsistenten Ebenengleichungen beschäftigen.

Anpassung von Ecken

Da bei der Anpassung von inzidenten Kanten immer nur zwei Flächen betrachtet wurden, ist es möglich, dass an Ecken, an denen mehrere inzidente Kanten zusammenlaufen, noch ein Loch geringer Größe vorhanden ist (Abbildung 4.23a).

Um korrekte Ecken zu erzeugen, wird deshalb das Loch entfernt und der Eckpunkt auf dem Schnittpunkt der drei Ebenen platziert. Sind mehr als drei Flächen beteiligt, wird der Schnittpunkt durch eine Optimierung der kleinsten Fehlerquadrate berechnet, denn durch Rauschen muss das Gleichungssystem keine eindeutige Lösung besitzen. Um solche fehlenden Ecken zu erkennen, wird eine Schwelle verwendet: Die Enden aller beteiligten inzidenten Kanten

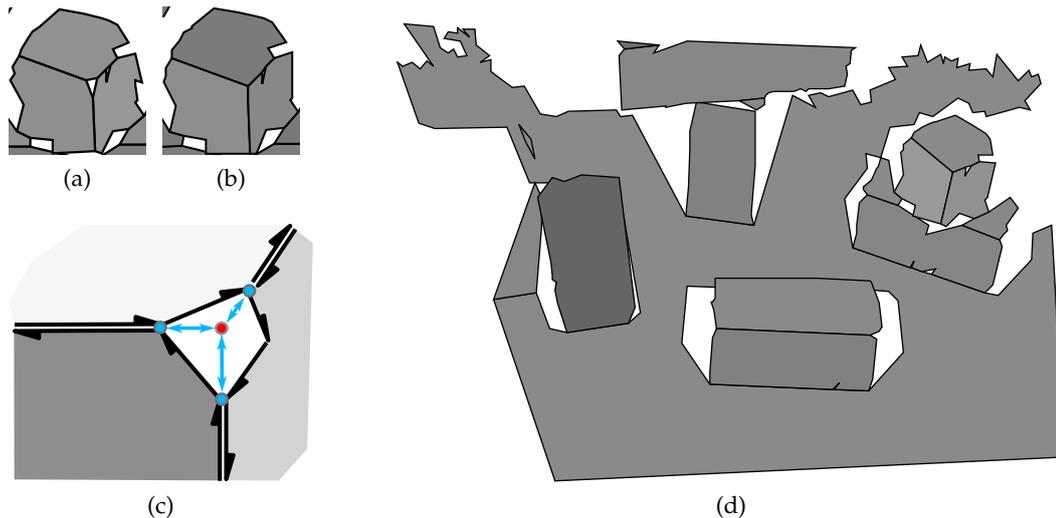


Abbildung 4.23: (a) Fehlende Ecke, (b) korrigierte Ecke, (c) fehlende Ecke mit eingezeichneten Halbkanten. Die Enden aller beteiligten inzidenten Kanten (blau) dürfen maximal um einen Wert d_{corner} vom neuen Eckpunkt (rot) entfernt liegen, um korrigiert zu werden. (d) B-Rep mit korrigierter Ecke

dürfen maximal um einen Wert d_{corner} vom neuen Eckpunkt entfernt liegen, damit die Eckenkorrektur angewandt wird (Abbildung 4.23c). Wählt man den Parameter $d_{corner} = 0.5 \cdot \delta_s$, so werden nie Ecken geschlossen, deren Knoten mehr als die Strukturgröße auseinander liegen.

Optimierung von Ecken

Auch wenn kein Loch geringer Größe an einer Ecke entsteht, so muss der Ort dieser Ecke neu berechnet werden. Da die Anpassung von inzidenten Kanten erstens nacheinander und zweitens nur für Paare von Flächen aufgeführt wird, liegt ein Eckpunkt immer auf der Geraden der inzidenten Kante, die zuletzt angepasst wurde. Auch hier wird deshalb der Eckpunkt neu berechnet, indem alle adjazenten Flächen mittels Optimierung der kleinsten Fehlerquadrate geschnitten werden. Es sei angemerkt, dass bei mehr als drei (mit Rauschen behaftete) Flächen die Eckpunkte nicht mehr auf allen Ebenengleichungen liegen müssen.

Alle anderen Knoten, die keine Ecken sind, müssen nur den Ebenengleichungen genügen. Das heißt, Knoten mit einer adjazenten Fläche (also Knoten an terminalen Kanten) werden auf die Ebene projiziert, Knoten mit zwei adjazenten Flächen (also Knoten an inzidenten Kanten) auf die Gerade.

Optimierung der Ebenengleichungen

Durch die Anpassung und Optimierung der Eckpunkte müssen nicht alle Eckpunkte einer Fläche in einer Ebene liegen, sodass die Fläche nicht mehr planar ist. Deshalb wird für alle Flächen mit drei oder mehr nicht kollinearen Ecken die Ebenengleichung mithilfe der Optimierung der kleinsten Fehlerquadrate neu bestimmt und alle Knoten der Fläche darauf projiziert.

Es ist erkennbar, dass dies ein zyklisches Problem ist, da für die Optimierung der Eckpunkte die Ebenengleichungen der adjazenten Flächen benötigt werden und für die Optimierung der Ebenengleichungen die Ecken. Deshalb wird die Ecken- und Ebenenoptimierung mehrmals hintereinander ausgeführt, bis sich keine Änderung mehr ergibt oder eine Maximalanzahl an Iterationen erreicht ist.

4.3 Evaluation

Die Evaluation des vorgestellten Verfahrens gliedert sich in zwei Teile: Zuerst wird die Güte der partiellen Modelle (Abschnitt 4.3.1) untersucht, um die Qualität der Rekonstruktion zu ermitteln. Anschließend wird die Laufzeit des Verfahrens betrachtet (Abschnitt 4.3.2), um eine Online-Ausführbarkeit des Systems zu testen und Abhängigkeiten zu ermitteln.

4.3.1 Güte der partiellen B-Reps

Zur Evaluation der Güte partieller B-Reps sind Ground-Truth-Daten, also exakte B-Reps, die dem Soll-Zustand entsprechen, vonnöten. Die Erstellung der Ground-Truth ist dabei bei Verwendung von realen Kameras und Objekten schwierig, da nicht nur die Geometrie eines Objektes relevant ist, sondern auch dessen exakte Lage im Kamerakoordinatensystem. Da partielle Modelle rekonstruiert werden, muss zudem für jede Kameraposition eine andere Ground-Truth vorliegen, da unterschiedliche Flächen in unterschiedlichem Maße sichtbar sind. Aus diesem Grund wird die Evaluation der Güte der partiellen B-Reps auf synthetischen Daten durchgeführt. Dies bietet zudem den Vorteil unterschiedlich starke Messabweichungen simulieren zu können.

Im ersten Abschnitt wird dargestellt, wie synthetische Rohdaten und die dazugehörigen partiellen Ground-Truth B-Reps prinzipiell erzeugt werden können. Anschließend werden im zweiten Abschnitt die konkret zur Evaluation verwendeten Datensätze beschrieben. Abschnitt drei erläutert Metriken, mit denen die Güte der Rekonstruktion bewertet werden kann. Den Abschluss bildet Abschnitt vier mit den Ergebnissen.

Erzeugung synthetischer Punktwolken und Ground-Truth Daten

Für die Evaluation werden mehrere Datensätze erzeugt. Für jeden Datensatz wird dabei folgendermaßen vorgegangen:

Zunächst wird eine virtuelle 3D-Szene in einem 3D-Grafikprogramm erzeugt. In dieser werden verschiedene und unterschiedlich viele Objekte platziert. Als Objekte dienen dabei die Modelle aus dem öffentlich verfügbaren SEGCOMP-Benchmark¹ [Hoover96]. Die Autoren stellen hier polyedrische 3D-Modelle, sowie Tiefenbilder aus verschiedenen Ansichten mit Ground-Truth-Daten für Segmentierungsalgorithmen zur Verfügung. Zur Evaluation von partiellen B-Reps eignen sich die verfügbaren Tiefenbilder allerdings nicht, da die Ground-Truth nur als 2D-Segmentierung vorliegt. Für die 3D-Modelle ist nicht mehr bekannt, wie sie in den einzelnen Tiefenbildern positioniert sind. Aus diesem Grund werden die Modelle in eine eigene virtuelle Szene importiert und selbst weiterverarbeitet.

Dazu werden mehrere virtuelle Kamerapositionen manuell erstellt, die das oder die Objekte aus verschiedenen Posen betrachten. Mithilfe der *Blender Sensor Simulation Toolbox*² [Gschwandner11] werden organisierte Punktwolken der Auflösung 640×480 Pixel aus jeder Kamerapose erzeugt. Alle Punktwolken liegen dabei in Sensorkoordinaten vor, also mit positiver z -Achse als Sichtrichtung und Ursprung als Kamerazentrum. Zudem sind alle Punkte markiert, das heißt, für jeden Punkt ist bekannt, zu welcher Fläche des polyedrischen Objektes er gehört. Pro Pose wird eine exakte Punktwolke und mehrere mit unterschiedlich starkem Rauschen generiert. Dazu wird pro Punkt $\vec{P} = (x, y, z)^T$ aus einer Normalverteilung $N(0, \sigma)$ zufällig gezogen, und das Ergebnis s_σ quadratisch mit z erhöht, um das für Tiefenkameras typische

¹<http://marathon.csee.usf.edu/range/seg-comp/SegComp.html>

²<http://www.blensor.org>, Version 1.0.16

Verhalten zu simulieren (siehe auch Definition 3.18 in Abschnitt 3.3.2):

$$s_z = s_\sigma \cdot z^2. \quad (4.32)$$

Mithilfe des Strahlensatzes wird dann die laterale Verschiebung des Punktes bestimmt:

$$s_l = \frac{s_z \cdot l}{z} \quad \text{mit} \quad l = \|\vec{P}\|_2. \quad (4.33)$$

Die verrauschte Position \vec{P}' des Punktes \vec{P} lautet damit:

$$\vec{P}' = \vec{P} + \frac{s_l}{l} \vec{P} = \vec{P} \cdot \left(1 + \frac{s_l}{l}\right) = \vec{P} \cdot \left(1 + \frac{s_z}{z}\right) = \vec{P} \cdot (1 + s_\sigma \cdot z). \quad (4.34)$$

Die Vorgehensweise mit zufälligem Rauschen in z und daraus abgeleitetem Rauschen in x und y entspricht dabei dem Verhalten von Stereokameras, bei denen eine mit Messfehlern behaftete Disparität berechnet wird, die zu verrauschten z -Werten führt und daraus dann x und y bestimmt wird [Khoshelham12]. Zur Evaluation werden in dieser Arbeit fünf verschiedenen Levels an Rauschen erzeugt:

$$\sigma = 0.001, 0.0015, 0.002, 0.0028, 0.004. \quad (4.35)$$

Zum Vergleich sei genannt, dass das Rauschen der Microsoft Kinect-Kamera $\sigma = 0.0028$ beträgt [Holzer12].

Zur Erstellung eines partiellen Ground-Truth-B-Reps wird nun pro Pose das vollständige, exakte 3D-Modell herangezogen und „zugeschnitten“. Dazu werden die Markierungen der aus dieser Pose erzeugten exakten Punktwolke betrachtet. Alle Flächen ohne dazugehörige Punkte werden entfernt.

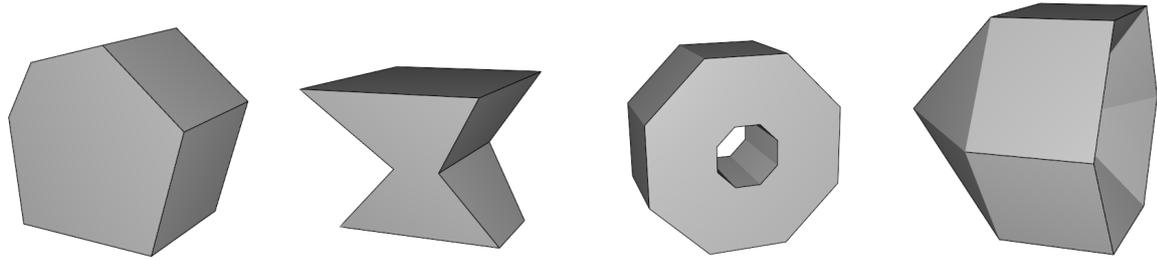
Zudem werden alle Kanten und Ecken im B-Rep entfernt, die nicht aus der aktuellen Kamerapose sichtbar sind. Dieser Schritt wird manuell durchgeführt, da beispielsweise Kanten verdeckt sein können, obwohl beide beteiligten Flächen sichtbar sind. Das Ergebnis ist ein partielles Ground-Truth B-Rep, das nur die Flächen, Kanten und Ecken enthält, die auch in einer aus dieser Kamerapose generierten Punktwolke zu erkennen sind.

Datensätze

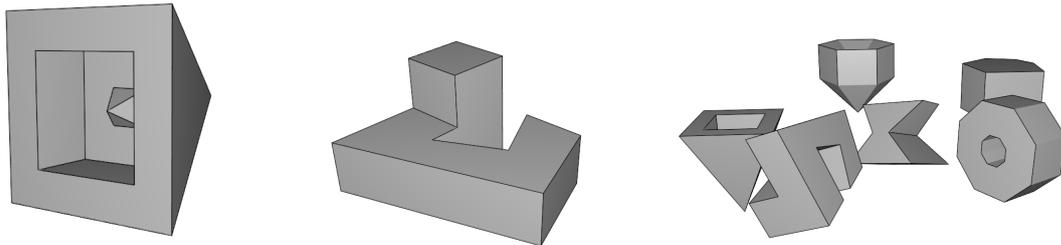
Mit dem eben beschriebenen Verfahren wurden sieben Datensätze erzeugt. Sechs davon beinhalten jeweils die SEGCOMP-Modelle als einzelne Objekte (Abbildung 4.24a-f), ein Datensatz besteht aus sechs Objekten (Abbildung 4.24g), um möglichst viele Verdeckungen zu erzeugen. Pro Datensatz wurden zwischen elf und 22 Kameraposen erzeugt und pro Kamerapose fünf Punktwolken mit den zuvor beschriebenen Levels an Rauschen. Dies ergibt insgesamt 455 Punktwolken zur Evaluation.

Der Messbereich der in der *Blender Sensor Simulation Toolbox* verwendeten virtuellen Kamera beträgt von 0.5 bis 5 Meter. Die Objekte werden deshalb so skaliert, dass die Größe eines einzelnen Objektes ungefähr einen Meter beträgt. Dies ergibt realistische Daten, wie sie beispielsweise auch eine Kinect-Kamera erfassen würde.

Beim Datensatz Mix (Abbildung 4.24g) werden alle Objekte auf eine Größe von circa 0.6 Meter verkleinert, um in einer Kamerapose möglichst viele Objekte erfassen zu können und so möglichst viele Verdeckungen zu erzeugen.



(a) Chest, 11 Ansichten (b) Anvil, 11 Ansichten (c) Nut, 14 Ansichten (d) Engine, 11 Ansichten



(e) Ashtray, 11 Ansichten (f) Crane, 11 Ansichten (g) Mix, 22 Ansichten

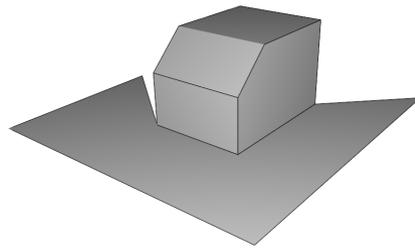
Abbildung 4.24: Zur Evaluation wurden sieben Datensätze erzeugt, indem vom abgebildeten Modell synthetische Punktwolken aus unterschiedlichen Ansichten generiert wurden. Die Modelle stammen aus dem SEGCOMP-Datensatz [Hoover96], von dem auch die Bezeichner der Objekte übernommen wurden.

Auswertung

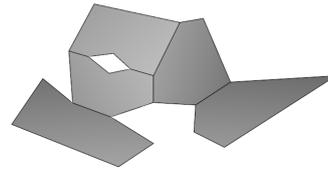
Zur Evaluation rekonstruierter partieller B-Reps wird auf zwei verschiedene Arten ausgewertet: *statistisch* und *quantitativ*.

Die statistische Auswertung misst die Erkennungsrate der Elemente eines B-Reps, also wie viele Flächen, Kanten und Ecken korrekt erkannt wurden. Angelehnt an die statistische Auswertung von Ergebnissen von Segmentierungsalgorithmen [Hoover96] werden folgende Kennzahlen bestimmt:

- **Korrekt** (*correct*): Anzahl der Elemente, die korrekt rekonstruiert wurden.
- **Fehlend** (*missed*): Anzahl der Elemente der Ground-Truth, die in der Rekonstruktion fehlen.
- **Rauschen** (*noise*): Anzahl der Elemente der Rekonstruktion, die nicht in der Ground-Truth vorhanden sind.
- **Übersegmentiert** (*oversegmented*): Anzahl der Elemente der Ground-Truth, die durch zwei oder mehr Elemente in der Rekonstruktion repräsentiert sind.
- **Untersegmentiert** (*undersegmented*): Anzahl der Elemente der Rekonstruktion, die zwei oder mehr Elemente der Ground-Truth zusammenfassen.



(a) Ground-Truth



(b) Rekonstruktion

	Anzahl Ground-Truth	Anzahl Rekonstruktion	Korrekt	Fehlend	Rauschen	Über- segmentiert	Unter- segmentiert
Flächen	5	5	3	1	0	1	0
Kanten	5	6	4	0	0	1	0
Ecken	2	1	1	1	0	-	-

(c) Statistische Auswertung

Abbildung 4.25: Beispiel für die statistische Auswertung einer Rekonstruktion eines partiellen B-Reps im Vergleich zur Ground-Truth. Bei der Anzahl an Kanten und Ecken der Ground-Truth werden Ecken und Kanten nicht mitgezählt, die an Flächen liegen, die in der Rekonstruktion nicht erkannt wurden (hier die Deckfläche). Diese Kennzahlen werden anschließend für alle Kameraposen gemittelt.

Elemente sind dabei Flächen, Kanten und Ecken eines partiellen B-Reps, für die die Kennzahlen getrennt voneinander ausgewertet werden. Bei Ecken können nur die Fälle *Korrekt*, *Fehlend* und *Rauschen* auftreten. Ein Beispiel ist in Abbildung 4.25 zu sehen. Kanten und Ecken von fehlenden Flächen oder zwischen übersegmentierten Flächen werden bei der Auswertung der Kanten und Ecken nicht berücksichtigt, da diese sowieso nicht erkannt werden können (Deckfläche in Abbildung 4.25).

Die quantitative Auswertung misst die Genauigkeit der Rekonstruktion. Dazu wird für alle korrekten, über- und untersegmentierten Flächen, Kanten und Ecken eine Abweichung hinsichtlich Position und Orientierung berechnet und gemittelt. Bei Ecken entfällt die Orientierung.

Für ein Paar aus korrespondierenden Flächen wird als Orientierungsabweichung der Winkel zwischen den Normalen verwendet. Als Positionsabweichung wird der Abstand d_{ij} der Ebenen am Ort der Überlappung verwendet. Eine illustrierte Beschreibung dieses Maßes ist in Abschnitt 5.2.2 (Gleichung 5.7) zu finden.

Für ein Paar aus korrespondierenden Kanten dient der Winkel zwischen den Richtungsvektoren als Orientierungsabweichung, der Abstand zwischen den beiden Strecken als Positionsabweichung. Bei Ecken wird der euklidische Abstand zwischen den Ecken verwendet.

Ergebnisse

Für die Datensätze mit den einzelnen Objekten wird zunächst eine Strukturgröße von $\delta_s = 75mm$ gewählt, für den Datensatz Mix $\delta_s = 45mm$, da die Objekte wie oben beschrieben kleiner skaliert wurden. Zuerst soll nun das Ergebnis gemittelt über alle Levels an Rauschen betrachtet werden (Tabelle 4.2), das zunächst einen groben Überblick über die Resultate liefert. Es zeigt sich, dass die translatorische und rotatorische Abweichung mit Werten meist unter einem Millimeter beziehungsweise einem Grad gering ist. Der Grund ist, dass Ecken anhand Kanten und Kanten anhand Flächen bestimmt werden. Flächen wiederum werden als Least-Squares-Fit an eine Menge an Punkten angepasst, sodass Messfehler oder einzelne falsche

	Anzahl Ground-Truth	Anzahl Rekonstruktion	Korrekt	Fehlend	Rauschen	Über- segmentiert	Unter- segmentiert	Abweichung Position [$10^{-3} m$]	Abweichung Orientierung [$^{\circ}$]
Datensatz: Chest, Rauschen: alle, Strukturgröße: $\delta_s = 75mm$									
Flächen	3.73	3.75	3.67 (98.5%)	0.02	0.00	0.04	0.00	0.23	0.13
Kanten	4.42	4.25	3.91 (88.5%)	0.35	0.00	0.16	0.00	0.82	0.20
Ecken	1.71	1.44	1.44 (84.0%)	0.27	0.00	-	-	1.68	-
Datensatz: Anvil, Rauschen: alle, Strukturgröße: $\delta_s = 75mm$									
Flächen	3.82	3.49	3.40 (89.0%)	0.38	0.00	0.04	0.00	0.29	0.19
Kanten	3.24	3.11	3.04 (93.8%)	0.16	0.00	0.04	0.00	0.89	0.22
Ecken	0.85	0.45	0.45 (53.2%)	0.40	0.00	-	-	2.78	-
Datensatz: Nut, Rauschen: alle, Strukturgröße: $\delta_s = 75mm$									
Flächen	7.57	6.14	5.91 (78.1%)	1.41	0.00	0.07	0.09	0.09	0.10
Kanten	7.63	7.10	7.07 (92.7%)	0.54	0.00	0.01	0.00	0.37	0.28
Ecken	2.93	2.64	2.64 (90.2%)	0.29	0.00	-	-	0.88	-
Datensatz: Engine, Rauschen: alle, Strukturgröße: $\delta_s = 75mm$									
Flächen	7.18	6.11	6.00 (83.5%)	1.02	0.00	0.02	0.07	0.55	0.38
Kanten	5.64	5.58	4.95 (87.7%)	0.47	0.16	0.22	0.00	0.75	0.38
Ecken	2.47	1.27	1.27 (51.5%)	1.20	0.00	-	-	2.12	-
Datensatz: Ashtray, Rauschen: alle, Strukturgröße: $\delta_s = 75mm$									
Flächen	6.00	4.71	4.13 (68.8%)	1.25	0.00	0.15	0.22	0.30	0.28
Kanten	4.13	3.98	3.65 (88.5%)	0.31	0.00	0.16	0.00	0.60	0.33
Ecken	1.25	1.00	1.00 (79.7%)	0.25	0.00	-	-	1.14	-
Datensatz: Crane, Rauschen: alle, Strukturgröße: $\delta_s = 75mm$									
Flächen	5.91	5.78	5.45 (92.3%)	0.29	0.00	0.16	0.00	0.20	0.18
Kanten	6.51	6.40	6.25 (96.1%)	0.18	0.00	0.07	0.00	0.47	0.19
Ecken	2.05	1.82	1.82 (88.5%)	0.24	0.00	-	-	0.87	-
Datensatz: Mix, Rauschen: alle, Strukturgröße: $\delta_s = 45mm$									
Flächen	20.23	13.46	13.21 (65.3%)	6.61	0.04	0.01	0.20	0.22	0.22
Kanten	11.10	10.46	9.89 (89.1%)	0.97	0.05	0.24	0.00	0.39	0.28
Ecken	3.45	2.52	2.52 (72.9%)	0.94	0.00	-	-	0.81	-
Datensätze: alle, Rauschen: alle									
Flächen	9.27	7.08	6.84 (73.8%)	2.17	0.01	0.06	0.10	0.25	0.21
Kanten	6.75	6.44	6.11 (90.6%)	0.50	0.03	0.14	0.00	0.51	0.27
Ecken	2.29	1.74	1.74 (75.8%)	0.56	0.00	-	-	1.12	-

Tabelle 4.2: Statistische und quantitative Ergebnisse der einzelnen Datensätze sowie gemittelt über alle Datensätze. Sämtliche Levels an Rauschen wurden berücksichtigt und gemittelt.

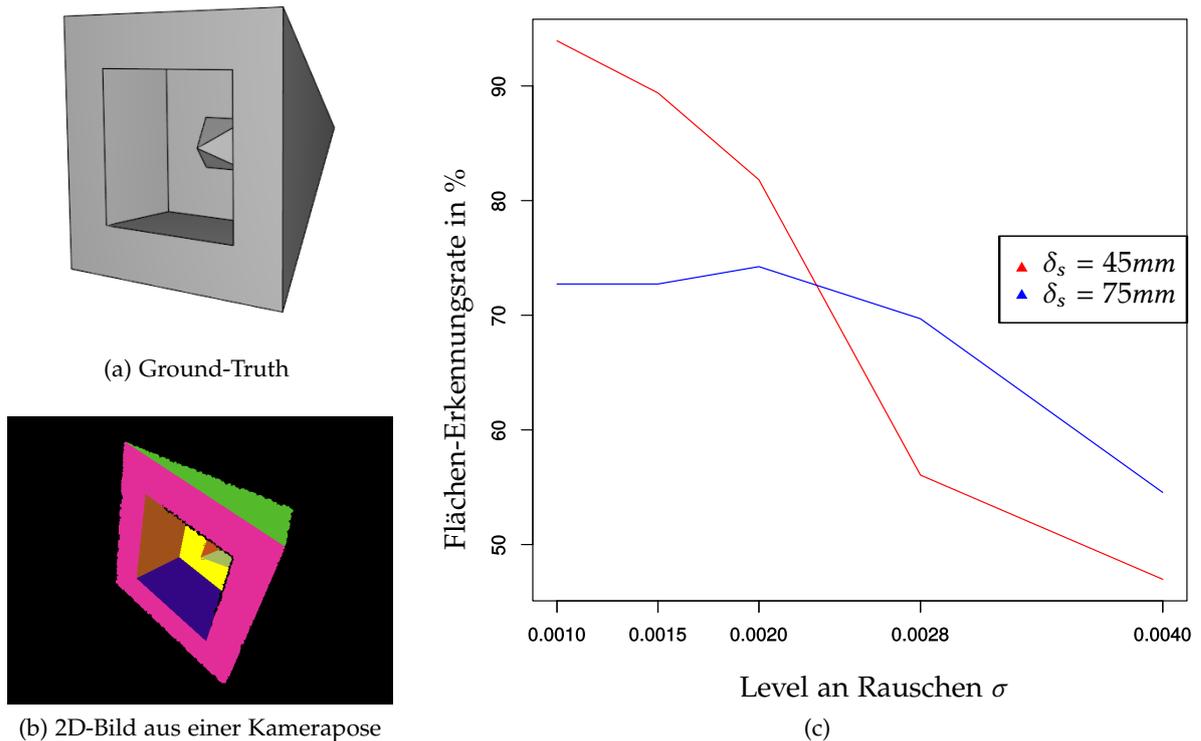


Abbildung 4.26: Der Ashtray-Datensatz (a) enthält kleine Flächen, die aus einigen Kameraposen zudem noch verdeckt werden (b). Diagramm (c) zeigt die Flächenerkennungsrate des Datensatzes in Abhängigkeit der Strukturgröße und des Rauschens.

Punkt-zu-Segment Zuordnungen gut herausgemittelt werden. Die statistische Auswertung hingegen zeigt, dass in dieser Konfiguration die Erkennung der B-Rep-Flächen nicht immer optimal ist und abhängig vom Datensatz sehr schwankt. Nicht korrekte Elemente sind dabei meist fehlend, gelegentlich übersegmentiert. Die Erkennungsrate bei Kanten und Ecken bleibt dabei eher konstant. Dies liegt einerseits daran, dass dabei die bereits gefundenen Flächen und nicht die Rohdaten verwendet werden und andererseits für die Auswertung, wie oben beschrieben, nur die Kanten und Ecken zwischen korrekt erkannten Flächen berücksichtigt werden. Im Folgenden wird deshalb hauptsächlich die Anzahl der korrekt erkannten Flächen verwendet, um Aussagen zu treffen.

Als nächstes wird der Datensatz *Ashtray* näher betrachtet, um die geringe Flächenerkennungsrate im Vergleich zu den anderen Datensätzen zu untersuchen. Als erstes fällt auf, dass das Objekt gleichzeitig kleine und große Flächen enthält (Abbildung 4.26a+b), die durch Selbstverdeckungen oft nur sehr wenige Punkte beinhalten. Deshalb wird die korrekte Flächenerkennungsrate (korrekte Flächen geteilt durch Anzahl Ground-Truth-Flächen) abhängig von gewählter Strukturgröße und dem Level an Rauschen untersucht. Das Ergebnis ist in Abbildung 4.26c dargestellt. Es zeigt sich, dass bei einer geringeren Strukturgröße (45mm) die Erkennungsrate bei geringem Rauschen auch in diesem Datensatz weit über 90% liegt. Für die kleinen Flächen reicht also eine Strukturgröße von 75mm nicht aus. Andererseits ist zu sehen, dass bei einem höheren Niveau an Rauschen die Erkennungsrate bei 45mm stark einbricht, während bei 75mm noch bessere Werte erreicht werden können. Der Grund für dieses Verhalten ist, dass das Rauschen ab einem Punkt so groß ist, dass die Strukturgröße dann nicht mehr eingehalten werden kann. Dieser Umstand wurde bereits in Abschnitt 3.3.3 in den theoretischen Überlegungen vorhergesagt.

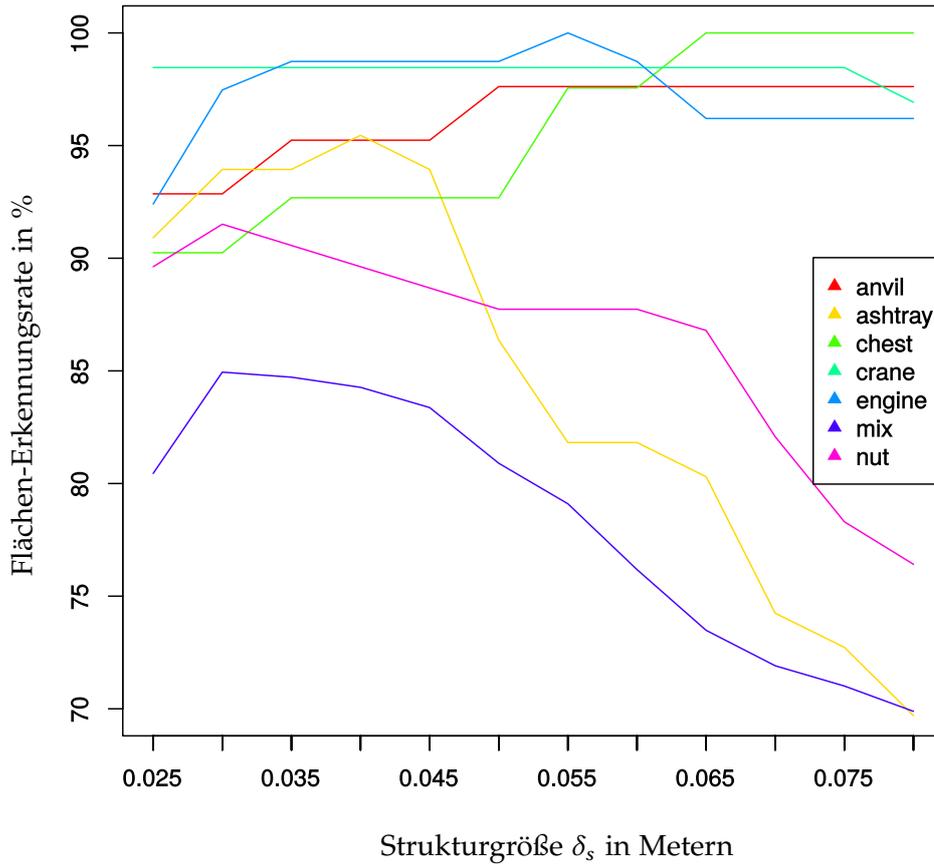


Abbildung 4.27: Die Flächenerkennungsrate bei einem Rauschen von $\sigma = 0.001$ in Abhängigkeit der Strukturgröße

Betrachtet man die Flächenerkennungsrate in Abhängigkeit von der Strukturgröße bei einem festem Wert an Rauschen, so erhält man das in Abbildung 4.27 dargestellte Ergebnis. Man erkennt, dass bei drei Datensätzen die Erkennungsrate deutlich sinkt. Diese Datensätze beinhalten viele kleine Flächen oder viele Verdeckungen, die dazu führen, dass nur kleine Stücke von größeren Flächen sichtbar sind. Diese werden bei einer größer werdenden Strukturgröße nicht mehr erkannt. Bei den übrigen Datensätzen, die nur größere Flächen enthalten, ist die Erkennungsrate stets auf einem hohen Niveau, wobei eine größere Strukturgröße etwas bessere Ergebnisse liefert. Der Grund ist hier, dass kleine durch Rauschen verursachte Fehler umso weniger Einfluss haben je größer die Strukturgröße ist. In Tabelle 4.3 sind die kompletten Ergebnisse aller Datensätze für $\sigma = 0.001$ und einer für jeden Datensatz passenden Strukturgröße dargestellt. Die Flächenerkennungsrate liegt hier bei allen Einzelobjekten über 90%, im Mix-Datensatz bei 85 %.

	Anzahl Ground-Truth	Anzahl Rekonstruktion	Korrekt	Fehlend	Rauschen	Über- segmentiert	Unter- segmentiert	Abweichung Position [10^{-3} m]	Abweichung Orientierung [°]
Datensatz: chest, Rauschen: $\sigma = 0.001$, Strukturgröße $\delta_s = 70mm$									
Flächen	3.73	3.73	3.73 (100.0%)	0.00	0.00	0.00	0.00	0.04	0.02
Kanten	4.45	4.36	4.36 (98.0%)	0.09	0.00	0.00	0.00	0.15	0.03
Ecken	1.73	1.64	1.64 (94.7%)	0.09	0.00	-	-	0.42	-
Datensatz: anvil, Rauschen: $\sigma = 0.001$, Strukturgröße $\delta_s = 50mm$									
Flächen	3.82	3.73	3.73 (97.6%)	0.09	0.00	0.00	0.00	0.05	0.03
Kanten	3.73	3.82	3.55 (95.1%)	0.09	0.00	0.09	0.00	0.23	0.04
Ecken	1.00	0.64	0.64 (63.6%)	0.36	0.00	-	-	0.57	-
Datensatz: nut, Rauschen: $\sigma = 0.001$, Strukturgröße $\delta_s = 30mm$									
Flächen	7.57	6.93	6.93 (91.5%)	0.64	0.00	0.00	0.00	0.02	0.02
Kanten	10.07	10.21	9.07 (90.1%)	0.57	0.00	0.43	0.00	0.07	0.05
Ecken	4.14	3.64	3.64 (87.9%)	0.50	0.00	-	-	0.19	-
Datensatz: engine, Rauschen: $\sigma = 0.001$, Strukturgröße $\delta_s = 55mm$									
Flächen	7.18	7.18	7.18 (100.0%)	0.00	0.00	0.00	0.00	0.08	0.05
Kanten	7.64	8.55	6.45 (84.5%)	0.27	0.00	0.91	0.00	0.30	0.06
Ecken	3.36	1.82	1.82 (54.1%)	1.55	0.00	-	-	0.44	-
Datensatz: ashtray, Rauschen: $\sigma = 0.001$, Strukturgröße $\delta_s = 40mm$									
Flächen	6.00	5.73	5.73 (95.5%)	0.27	0.00	0.00	0.00	0.06	0.06
Kanten	7.00	6.73	6.00 (85.7%)	0.64	0.00	0.36	0.00	0.24	0.25
Ecken	2.91	1.82	1.82 (62.5%)	1.09	0.00	-	-	1.29	-
Datensatz: crane, Rauschen: $\sigma = 0.001$, Strukturgröße $\delta_s = 45mm$									
Flächen	5.91	6.00	5.82 (98.5%)	0.00	0.00	0.09	0.00	0.04	0.03
Kanten	7.18	7.18	7.00 (97.5%)	0.09	0.00	0.09	0.00	0.14	0.05
Ecken	2.36	2.09	2.09 (88.5%)	0.27	0.00	-	-	0.27	-
Datensatz: mix, Rauschen: $\sigma = 0.001$, Strukturgröße $\delta_s = 35mm$									
Flächen	20.23	17.23	17.14 (84.7%)	2.91	0.00	0.00	0.09	0.09	0.09
Kanten	16.36	15.86	14.68 (89.7%)	1.14	0.00	0.55	0.00	0.22	0.17
Ecken	5.18	3.95	3.86 (74.6%)	1.27	0.00	-	-	0.46	-

Tabelle 4.3: Statistische und quantitative Ergebnisse der einzelnen Datensätze bei einem Rauschlevel von $\sigma = 0.001$ und einer für jeden Datensatz passend ausgewählten Strukturgröße

4.3.2 Laufzeit

Um die Laufzeit des Verfahrens und deren Abhängigkeiten zu evaluieren, werden mit der *Blender Sensor Simulation Toolbox* [Gschwandtner11] wie zuvor mehrere synthetische Punktwolken aus mehreren Ansichten erzeugt und die erhaltenen Laufzeiten gemittelt. Dabei werden drei Auflösungsstufen der virtuellen Kamera generiert: 640x480, 320x240 und 160x120. Bei der Festlegung der Kameraposen wird darauf geachtet, dass die organisierte Punktwolke nahezu vollständig belegt ist und keine Pixel leer sind, beispielsweise aufgrund von Über- oder Unterschreitung des Messbereichs.

Für alle hier dargestellten Zeitmessungen wird ein Laptop mit Intel i7-4800MQ-Prozessor (4 Kerne, 8 Threads) mit 32 GB RAM und Ubuntu 14.04 verwendet. Die während der Beschreibung des Vorgehens genannten Parallelisierungsmöglichkeiten (Filtermaskenerzeugung parallel zum Regionenwachstum, pro Pixel parallele Filterung) werden genutzt.

Es ist zu erwarten, dass die Laufzeit von der Auflösung abhängt, da die Vernetzung und Segmentierung auf Pixelebene arbeitet. Die Ergebnisse der Laufzeit der synthetisch erzeugten Daten in Abhängigkeit der Auflösung sind in Abbildung 4.28a dargestellt und lassen wie erwartet einen quadratischen Zusammenhang zwischen Laufzeit und Breite des Pixelgitters erkennen.

Abbildung 4.28b zeigt den Zusammenhang zwischen Laufzeit und Strukturgröße. Bei VGA-Auflösung ist eine Abhängigkeit zu erkennen. Der Grund ist, dass bei einer größeren Strukturgröße die erzeugten Filtermasken größer sind und die Filterung somit mehr Zeit benötigt.

Der prozentuale Anteil der einzelnen Rekonstruktionsschritte an der Gesamtzeit ist in Abbildung 4.29 dargestellt. Zur besseren Übersicht wurde dabei auf die Parallelausführung von Regionenwachstum und Filtermaskenerzeugung verzichtet. Man erkennt, dass 90% der Zeit auf Vernetzung, Regionenwachstum und Filterung verwendet werden. Der Grund liegt darin, dass diese Operationen noch auf einzelnen Pixeln arbeiten. Alle weiteren Schritte betrachten nur noch Segmente.

Die synthetischen Daten eignen sich gut zur Veranschaulichung der Abhängigkeiten von Auflösung und Strukturgröße. Durch Beschränkungen realer Hardware ist jedoch nicht jede Kombination möglich und sinnvoll. Deshalb soll kurz auf die Laufzeiten realer Kameras eingegangen werden.

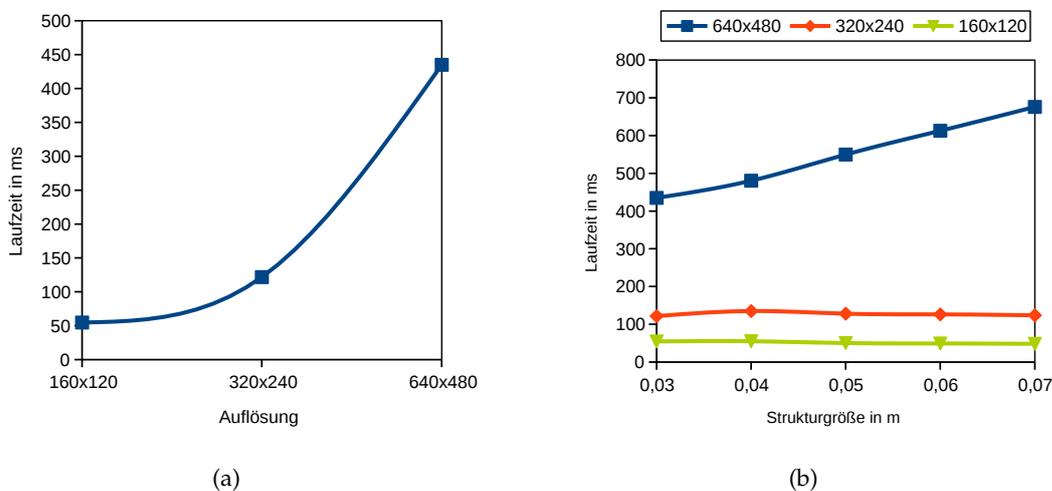


Abbildung 4.28: Abhängigkeit der Laufzeit von der Auflösung (a) und der Strukturgröße (b)

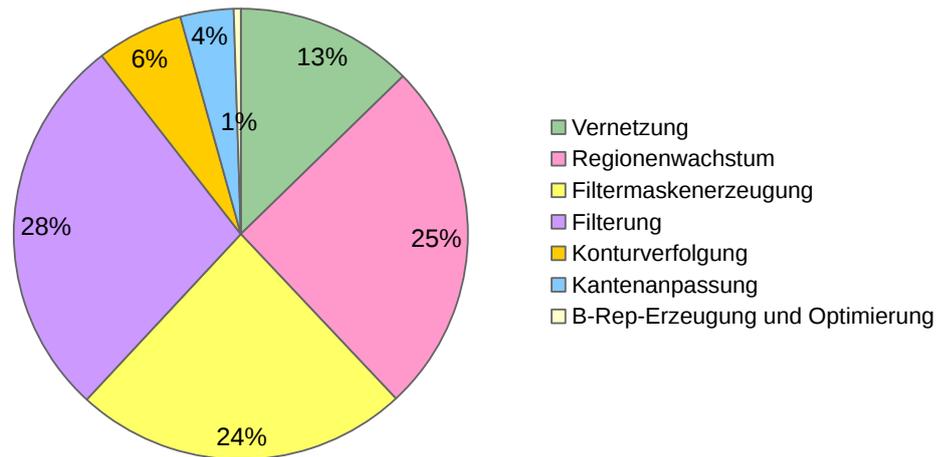


Abbildung 4.29: Prozentuale Verteilung der Laufzeit pro Schritt. Das Regionenwachstum und die Filtermaskenerzeugung können allerdings parallel ausgeführt werden.

Gerät	Auflösung	Strukturgröße [mm]	Laufzeit [ms]
Microsoft Kinect for XBOX 360	640x480	50	540
IDS Ensensio N10	752x480	5	210
Lenovo Phab 2 Pro	224x172	50	65

Tabelle 4.4: Laufzeiten für reale Hardware mit für diese Kamera typischer Strukturgröße

gangen werden. Zur Laufzeitmessung werden Punktwolken einer Consumer-Tiefenkamera (Microsoft Kinect for XBOX 360), einer industriellen Tiefenkamera (IDS Ensensio N10) und eines Smartphones mit integrierter Tiefenkamera (Lenovo Phab 2 Pro) rekonstruiert und die Laufzeiten gemittelt (Tabelle 4.4). Für jede Kamera wird eine typische Strukturgröße gewählt. Die Punktwolken des Smartphones werden zu Vergleichszwecken auf demselben Gerät wie bei beiden anderen Kameras verarbeitet, und nicht auf dem Gerät selbst. Für weitere Informationen bezüglich der Hardware wird auf Kapitel 8 verwiesen, in dem das Gesamtsystem betrachtet wird.

Man erkennt, dass die Kinect-Tiefenkamera mit ca. 500 ms die längste Rekonstruktionszeit besitzt. Durch die starken Messabweichungen kann keine kleine Strukturgröße gewählt werden. Bei der Ensensio-Kamera gelingt die Rekonstruktion bei ähnlicher Auflösung deutlich schneller, da die geringen Messabweichungen eine sehr kleine Strukturgröße ermöglichen. Punktwolken des Lenovo können aufgrund der geringen Auflösung am schnellsten verarbeitet werden. Insgesamt genügen die Laufzeiten für eine Online-Ausführung der Rekonstruktion.

4.4 Zusammenfassung

In diesem Kapitel wurde ein Ansatz zur schnellen Rekonstruktion partieller planarer B-Reps aus einzelnen organisierten Punktwolken vorgestellt. Die Segmentierung ist dabei der erste und wichtigste Schritt. Zwei Verfahren aus der Literatur eignen sich aufgrund ihrer Laufzeit, begründet in der Ausnutzung der Punktwolkenstruktur, besonders gut als Segmentierungsansatz. Das Verfahren von Holz et al. [Holz14] wurde wegen seiner besseren Erweiterbarkeit als Basis für das vorgestellte Rekonstruktionsverfahren verwendet. Zudem wurde eine Erweiterung um einen Filterungsschritt vorgeschlagen. Dabei wurden Tiefen- und Strukturgrößenabhängige Filtermasken vorgestellt, die erlauben, allgemeine Filteroperationen auf den orga-

nisierten Punktwolken und deren Segmentierung durchzuführen. Dies wurde zur Verbesserung der Segmentierung und zur Berücksichtigung des Anwender-abhängigen Parameters der Strukturgröße genutzt.

Der zweite Schritt zur Rekonstruktion ist die Bestimmung des Randes der Segmente. Dazu wurden Konturverfolgungsalgorithmen genutzt. Da diese ebenfalls das 2D-Pixelgitter ausnutzen, sind diese schneller als Algorithmen zur Berechnung einer Hülle. Durch Verwendung eines knotenbasierten Ansatzes und anschließender Rückprojektion in den 3D-Raum wurde sichergestellt, dass das resultierende Polygon einfach ist. In der Kontur wurden anschließend die unterschiedlichen Kantentypen eines partiellen B-Reps identifiziert und angepasst.

Als letztes wird die *half-edge*-Datenstruktur zur Repräsentation des partiellen B-Reps aus den zuvor gesammelten Informationen erstellt. Ein Optimierungsverfahren sorgt dabei für eine globale Konsistenz von Ecken.

Aufgrund der Unvollständigkeit der partiellen Modelle, die durch Verdeckungen aus einzelnen Kameraansichten entstehen, wurde die Güte der Rekonstruktion mit synthetischen Modellen evaluiert. Dabei wurde auf öffentliche verfügbare CAD-Modelle aus dem SEGCOMP-Datensatz zurückgegriffen. Mithilfe der *Blender Sensor Simulation Toolbox* wurde unterschiedliche Kameraposen und verschiedenen Messabweichungen simuliert.

Dabei zeigte sich, dass durch die Mittelung der Punkte innerhalb einer Fläche der Fehler einer Ebenengleichung vergleichsweise gering ist. Schlechte Rekonstruktionen äußern sich eher im Fehlen von Ecken oder einer Über- oder Untersegmentierung von Kanten und Flächen. Besonders bei Objekten mit sowohl kleinen als auch großen Flächen zeigte sich eine Abhängigkeit der Flächenerkennungsrate von Rauschen und Strukturgröße. Eine kleine Strukturgröße ist nötig und sinnvoll, um kleine Flächen gut zu rekonstruieren. Ab einem gewissen Niveau an Rauschen kann eine kleine Strukturgröße allerdings nicht mehr eingehalten werden, sodass die Erkennungsrate stark sinkt. Für den Anwender bedeutet dies, dass für ein gegebenes Objekt die Strukturgröße und das Level an Rauschen der verwendeten Kamera zusammenpassen müssen. Als Abschätzung für diesen Zusammenhang können dabei die Formeln aus Kapitel 3.3.3 herangezogen werden.

Die Auswertung der Laufzeit des vorgestellten Verfahrens zeigte eine Abhängigkeit zur Auflösung der Eingabepunktwolke und zur verwendeten Strukturgröße. Für reale Kameras wurden Rekonstruktionszeiten von 65 bis 540 ms gemessen. Eine Online-Ausführung erscheint deshalb möglich.

Insgesamt ist dies ein großer Schritt zur Beantwortung der Fragestellung F2, ob die Lücke zwischen SLAM und DSR-Systemen überwunden werden kann und bei bekannter Pose online und inkrementell partielle B-Reps erzeugt werden können. Zur vollständigen Beantwortung muss noch die Fusion von partiellen Modellen untersucht werden.

Kapitel 5

Fusion partieller B-Reps

Inhalt

5.1	Stand der Forschung	97
5.2	Vorgehensweise	99
5.2.1	Mittelung von Ebenengleichungen	100
5.2.2	Bestimmung von Flächenkorrespondenzen	101
5.2.3	Vereinigung von Flächen	102
5.3	Evaluation	108
5.3.1	Güte der fusionierten B-Reps	108
5.3.2	Laufzeit	115
5.4	Zusammenfassung	115

Im vorhergehenden Kapitel wurde dargelegt, wie aus einer einzelnen Punktwolke ein partielles B-Rep Modell erzeugt werden kann. Um ein vollständiges Modell zu erhalten, ist es notwendig, die zu rekonstruierende Szene aus mehreren Blickwinkeln zu betrachten. In diesem Kapitel wird untersucht, wie sich mehrere partielle B-Reps aus unterschiedlichen Aufnahmeposen vereinigen lassen. Dabei wird angenommen, dass die Aufnahmepose bekannt ist. Dieser Fall tritt beispielsweise bei *Eye-in-Hand*-Kameras auf, also Kameras, die am Endeffektor eines Roboterarms befestigt und auf das Roboterkoordinatensystem kalibriert sind. Das nachfolgende Kapitel 6 geht dann auf den Fall von unbekanntem Aufnahmeposen ein. Da die Rekonstruktion online ausgeführt werden soll, muss die Fusion inkrementell erfolgen. Das heißt, jedes rekonstruierte Einzel-B-Rep wird sofort in ein Gesamtmodell hinein fusioniert, das damit vollständiger wird. Dadurch bleibt das System auch speichereffizient, da stets nur ein Gesamtmodell als B-Rep vorgehalten werden muss. Für die Fusion bedeutet dies, dass einerseits nur der Fall der Vereinigung von genau zwei B-Rep-Modellen betrachtet werden muss (Gesamtmodell und aktuelles B-Rep), andererseits aber die Fusion auch gewichtet erfolgen muss. Eine Fläche, die im Gesamtmodell bereits sehr oft gesehen wurde und damit bestätigt wurde, darf nicht durch eine einzelne, möglicherweise verlässliche Aufnahme stark verändert werden.

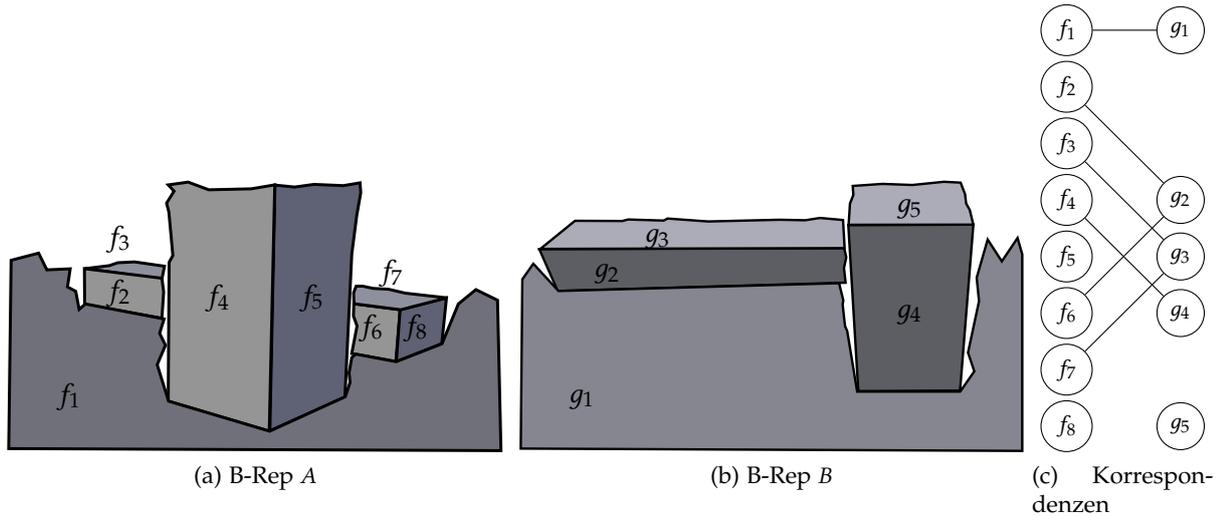


Abbildung 5.1: (a-b) Zwei partielle B-Reps derselben Szene aus unterschiedlichen Blickwinkeln, (c) die Flächenkorrespondenzen als bipartiter Graph. Eine Fläche kann keine, eine oder mehrere korrespondierende Flächen besitzen.

Das Problem der Fusion zweier partieller B-Reps lässt sich folgendermaßen beschreiben:

Gegeben sind zwei partielle B-Reps $A = (\mathcal{V}_A, \mathcal{H}_A, \mathcal{B}_A, \mathcal{F}_A)$ und $B = (\mathcal{V}_B, \mathcal{H}_B, \mathcal{B}_B, \mathcal{F}_B)$ derselben Szene, rekonstruiert aus unterschiedlichen, bekannten Blickwinkeln. Die Transformation vom lokalen Koordinatensystem von B in das von A sei dabei mit ${}^A T_B$ bezeichnet. Zur Vereinfachung der Darstellung wird im Folgenden davon ausgegangen, dass B bereits mithilfe von ${}^A T_B$ ins gleiche Koordinatensystem wie A transformiert wurde. Damit muss im weiteren Vorgehen ${}^A T_B$ nicht mehr einbezogen werden.

Gesucht ist ein valides partielles B-Rep $C = (\mathcal{V}_C, \mathcal{H}_C, \mathcal{B}_C, \mathcal{F}_C)$, das die Informationen, die in beiden Ausgangsmodellen enthalten sind, fusioniert. Im Detail bedeutet dies, dass C im Bereich, in dem keine Überlappung von A und B vorliegt, vollständiger und im Überlappungsbereich genauer ist (z.B. Position von Ecken/Kanten), da mehr Informationen vorliegen. Zudem können neue topologische Informationen abgeleitet werden. Beispielsweise ist für vorher aufgrund von Verdeckungen separierte Flächen nun bekannt, dass diese verbunden sind (Flächen f_2 und f_6 in Abbildung 5.1).

Das Kapitel ist in vier Abschnitte unterteilt. Zuerst werden verwandte Verfahren aus der Literatur betrachtet (Abschnitt 5.1) und mit der vorliegenden Problemstellung verglichen. Danach wird ein neuartiges Verfahren zur Fusion von partiellen B-Reps im Detail vorgestellt (Abschnitt 5.2), das anschließend mit den bereits im vorhergehenden Kapitel beschriebenen Testdatensätzen evaluiert wird (Abschnitt 5.3). Den Abschluss bildet eine Zusammenfassung des Kapitels (Abschnitt 5.4).

Teile des in diesem Kapitel vorgestellten Ansatzes wurden bereits in [Sand16] veröffentlicht.

5.1 Stand der Forschung

In dieser Arbeit wird erstmals eine Kombination aus SLAM und DSR verwendet (siehe Kapitel 3.1), sodass es nötig ist, hochwertige, abstrakte Datenstrukturen zu mitteln und vereinigen und dabei konsistent zu halten. Bestehende Rekonstruktionsverfahren sind diesem Problem nicht ausgesetzt, da nicht auf der Oberflächenrepräsentation fusioniert wird (vgl. Kapitel 2):

Volumetrische Verfahren mitteln Messungen unter Verwendung einer Unterteilung des 3D-Raumes (Voxelgitter, Octree, etc.), die keine topologischen Informationen enthalten [Newcombe11, Nießner13, Whelan15b]. Die Konsistenthaltung während der Fusion entfällt damit. Dasselbe gilt für Surfel-Modelle, da diese ebenfalls keine Verbindungen zwischen den Surfels besitzen [Keller13, Whelan16, Kähler16]. Verfahren zur *digital shape reconstruction* (DSR) gehen entweder von einer einzigen globalen Punktwolke aus oder vereinigen die Rohdaten bereits auf Punktwolkenebene [Várady06].

Betrachtet wird das Problem von überlappenden Oberflächenrepräsentationen im Bereich der Netzreparatur (*Mesh Repair*) in Dreiecksnetzen. Während einige Ansätze auch hier ein Voxelmodell als Zwischenschritt nutzen, um daraus ein neues fusioniertes Mesh zu erstellen, existieren nur sehr wenige Ansätze, die direkt auf der Oberflächenrepräsentation arbeiten [Botsch07]. Die bekannteste Variante ist das *Mesh Zipping*, ein reißverschlussartiges Zusammenfügen der Netze [Turk94]. Für zwei zu vereinigende Dreiecksnetze wird zuerst der Überlappungsbereich mittels einer Distanzschwelle erkannt und entfernt. Anschließend werden beide Netze durch Dreiecke miteinander verknüpft. Die Fusion ist damit vollständig, jedoch wurde die Oberfläche noch nicht gemittelt. Dazu werden die Knoten im Überlappungsbereich verschoben, indem Punkte auf den ursprünglichen Netzen gefunden und gemittelt werden.

Ist für ein aus Sensordaten rekonstruiertes Netz die Aufnahmeweise und Aufnahmepose bekannt, so kann mithilfe eines Rauschmodells auch ein probabilistischer Ansatz zur Mittelung genutzt werden, um genauere Ergebnisse zu erreichen [Cahier12]. Ist die Qualität der beiden zu vereinigenden Netze sehr unterschiedlich, zum Beispiel aufgrund unterschiedlicher Sensoren, kann statt einer Mittelung auch eine reine Entscheidung für das bessere Netz erfolgen [Wuttke12].

Im Bereich der computergestützten Modellierung ist ein grundlegender Konstruktionsschritt zur Erstellung von CAD-Modellen die Ausführung von booleschen Operationen, also die Bildung des Schnitts, der Vereinigung oder der Differenz zweier Modelle. Diese Problemstellung umfasst zwar nicht die Mittelung von Oberflächen, sondern nur die Vereinigung, soll aber dennoch kurz betrachtet werden:

Schon für polyedrische Modelle sind zur Berechnung viele aufwändige Schnitt- und Lagetests nötig [Requicha85, Laidlaw86]. Durch Verwendung von BSP-Bäumen kann die Anzahl der Tests reduziert werden, da durch den Baum der Raum sinnvoll unterteilt wird [Thibault87, Naylor90, Naylor92]. Das größte Problem ist jedoch die numerische Robustheit und Genauigkeit der geometrischen Berechnungen [Hoffmann01, Rossignac07], was zu topologisch inkorrekten Repräsentationen führen kann. Um diesen Effekt zu reduzieren, wird die Standard-Gleitkomma-Arithmetik durch andere Konzepte ersetzt, wie Intervall-Arithmetik [Segal90, Bruderlin91] oder exakter Arithmetik [Hachenberger07], die jedoch zu Lasten einer höheren Berechnungszeit gehen. Beschleunigungen können durch Approximation [Wang11] oder durch zusätzliche Datenstrukturen zur volumetrischen Unterteilung [Barki15] erreicht werden. Für gekrümmte B-Rep-Modelle existieren nur sehr wenige und komplexe Ansätze [Keyser99a, Keyser99b], da die Schnittberechnungen umso aufwändiger sind.

Zusammenfassend lässt sich sagen, dass boolesche Operationen die am schwierigsten zu implementierenden Modellierungsfunktionen sind [Lee99, S. 132]. Aus diesem Grund stellen solche Algorithmen den essentiellen Teil eines Geometrie-Kernels von heutigen CAD-Programmen dar. Bezogen auf das aktuelle Problem liefern diese Algorithmen zwar eine valide B-Rep-Repräsentation der Vereinigung der Ausgangsmodelle, berücksichtigen aber keine Mittelung oder Verbesserung von Flächen, wie sie im Überlappungsbereich aber nötig sind. Zudem gehen die meisten Ansätze von wasserdichten Netzen als Eingabe aus und können so nicht direkt auf partielle B-Reps angewendet werden.

Insgesamt zeigt sich somit, dass sich kein bekanntes Verfahren direkt für die benötigte Fusion

von partiellen B-Reps eignet.

5.2 Vorgehensweise

Geometrisch besteht eine Fläche eines B-Rep aus zwei Teilen, der Oberflächenbeschreibung und der Begrenzung. Im hier betrachteten Fall von planaren B-Reps wird die Oberfläche durch eine Ebenengleichung dargestellt, die Begrenzung ist ein in dieser Ebenen liegendes einfaches Polygon (mit Löchern). Bei der Fusion von partiellen B-Reps können drei Teilprobleme identifiziert werden:

- **Mittelung:** In Bereichen, in denen sich die zwei Ausgangsmodelle überlappen, sind unterschiedliche Informationen vorhanden. Durch Messunsicherheiten kann die Oberflächenbeschreibung einer Fläche in beiden Modellen abweichen. Es muss also eine anhand der Konfidenz gewichtete Mittelung der Oberflächenbeschreibung stattfinden.
- **Vereinigung:** Durch den Blickwinkel kann sich die Begrenzung unterscheiden, wenn in einer Ansicht mehr von einer Fläche erfasst wird als in der anderen. Es ändert sich demnach auch die Begrenzung.
- **Konsistenz:** Bei allen Operationen zur Fusion muss am Ende gewährleistet sein, dass ein topologisch korrektes und valides partielles B-Rep entsteht.

Die Betrachtung des Stands der Forschung zeigt, dass die Vereinigung von vollständigen CAD-Modellen sehr aufwändig ist und keine Mittelung berücksichtigt, sodass eine direkte Anwendung nicht in Frage kommt. Ein wesentlicher Unterschied zu bekannten Verfahren ist, dass hier partielle Modelle betrachtet werden. Die Flächen eines partiellen B-Reps beschreiben bekannte Stücke der Oberfläche, die über Zwillingskantenbeziehungen miteinander verbunden sind. Zur Fusion wird deshalb hier ein 2D-Ansatz verfolgt, das heißt, die Fusion wird pro Fläche durchgeführt. Dies hat folgende Vorteile:

- Die Mittelung vereinfacht sich zu einem gewichteten Mittel planarer Strukturen.
- Die Vereinigung kann mit bekannten 2D-Algorithmen durchgeführt werden, die erheblich weniger Komplexität als 3D-Verfahren aufweisen.
- Die Konsistenthaltung des Modells ist für Elemente, die zu einer Fläche gehören (Halbkanten, Begrenzungen, etc.), leichter, da diese Elemente auch gemeinsam betrachtet werden. Zwillingskantenbeziehungen müssen allerdings gesondert betrachtet werden.

Insgesamt kann die hier vorgestellte Fusion folgendermaßen zusammengefasst werden:

1. Bestimme korrespondierende Flächen, die fusioniert werden sollen.
2. Führe pro Fläche eine 2D-Vereinigung durch, die alle vorhandenen Informationen mittelt, wobei alle Elemente außer Zwillingskanten konsistent bleiben.
3. Stelle die Konsistenz der Zwillingskantenbeziehungen wieder her, indem Kanten zwischen Flächen betrachtet werden.

Um eine 2D-Fusion durchführen zu können, muss eine Ebene im Raum existieren, auf der die Fusion durchgeführt wird. Da zwei korrespondierende Flächen nicht zwangsweise eine identische Ebenengleichung besitzen, beschäftigt sich Abschnitt 5.2.1 mit der Mittelung von Ebenengleichungen. Wie korrespondierende Flächen gefunden werden können, ist in Abschnitt 5.2.2 dargestellt. Der Hauptteil, die eigentliche Fusion wird dann in Abschnitt 5.2.3 im Detail erläutert.

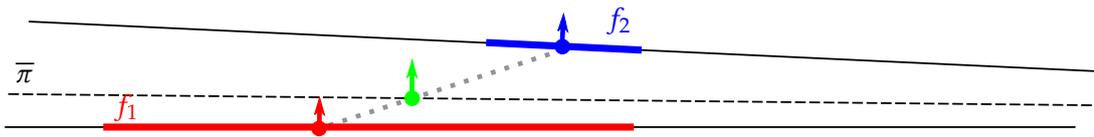


Abbildung 5.2: Die Berechnung der mittleren Ebene $\bar{\pi}$ zweier Flächen f_1 und f_2 mit Gewichten $w_1 = 2$, $w_2 = 1$ erfolgt durch eine gewichtete Mittelung der Normalen und Flächenschwerpunkte beider Flächen.

5.2.1 Mittelung von Ebenengleichungen

Bei der Fusion von B-Reps, insbesondere bei der Fusion von einzelnen Flächen, soll die Güte einer Fläche berücksichtigt werden. Es ist also nötig, ein gewichtetes Mittel zu berechnen. Da grundsätzlich im folgenden Vorgehen das Problem auf zwei Dimensionen reduziert wird, ist es nötig, eine Ebene im Raum zu ermitteln, auf der die 2D-Fusion durchgeführt wird. Es stellt sich also die grundsätzliche Frage, wie eine (gewichtete) mittlere Ebene $\bar{\pi}$ aus zwei oder mehreren Ebenen E_i , in denen die zu vereinigenden Flächen f_i der B-Reps liegen, bestimmt werden kann.

Per Definition ist ein arithmetisches Mittel die Summe einer Menge an Werten dividiert durch deren Anzahl N

$$\bar{m} = \frac{1}{N} \sum_{i=1}^N m_i \quad (5.1)$$

oder im kontinuierlichen Fall

$$\bar{f} = \frac{1}{b-a} \int_a^b f(x) dx. \quad (5.2)$$

Bei einer abgeschlossenen Fläche K mit Flächeninhalt A spricht man auch vom Flächenschwerpunkt

$$\bar{P} = \frac{1}{A} \int_K \vec{P} dA \quad \text{mit} \quad A = \int_K dA. \quad (5.3)$$

Da Ebenen eine unendliche Ausdehnung besitzen, ist diese Definition nicht auf Ebenen anwendbar. Im vorliegenden Fall sind allerdings nicht nur die Ebenen E_i verfügbar, sondern auch die darauf liegenden B-Rep-Flächen f_i . Mit \bar{P}_i als Flächenschwerpunkt, \vec{n}_i als Normale und w_i als Gewicht von f_i kann somit eine mittlere Ebene $\bar{\pi}$ in Form von Aufpunkt \bar{P} und Normale \vec{n} berechnet werden (Abbildung 5.2):

$$\bar{P} = \frac{\sum w_i \bar{P}_i}{\sum w_i}, \quad (5.4)$$

$$\vec{n} = \frac{\sum w_i \vec{n}_i}{\sum w_i}. \quad (5.5)$$

Die Ebenengleichung für $\bar{\pi}$ lautet somit:

$$\bar{\pi} = \begin{pmatrix} \vec{n} \\ -\vec{n} \circ \bar{P} \end{pmatrix} \quad (5.6)$$

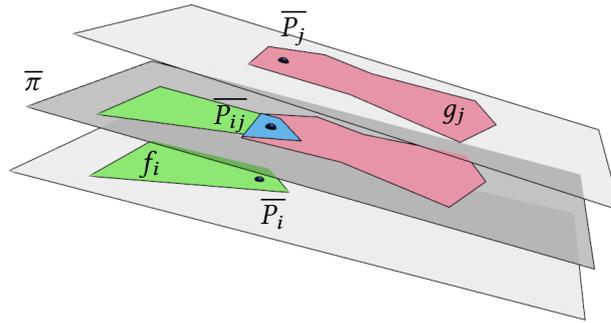


Abbildung 5.3: Als Abstand zweier sich überlappender Flächen f_i und g_j , die nicht exakt in derselben Ebene liegen, eignet sich das Maß $d_{ij} = \|\bar{P}_i - \bar{P}_j\|_2$. Es entsteht, indem beide Flächen auf die gemittelte Ebene $\bar{\pi}$ projiziert werden und das Schnittpolygon mit dessen Schwerpunkt \bar{P}_{ij} bestimmt werden. Durch Rückprojektion entstehen dann die für d_{ij} benötigten Punkte \bar{P}_i und \bar{P}_j .

5.2.2 Bestimmung von Flächenkorrespondenzen

Als erster Schritt der Fusion werden korrespondierende Flächen ermittelt, die später vereinigt werden sollen. Es ist zu beachten, dass einerseits nicht jede Flächen eine korrespondierende besitzen muss, andererseits aber auch N:M-Beziehungen möglich sind (Abbildung 5.1).

Ein einfacher Vergleich der Ebenengleichungen genügt zur Korrespondenzfindung nicht, da auch zwei getrennte Flächen in der gleichen Ebene liegen können. Neben der Normalenrichtung zweier Flächen muss also auch eine Überlappung betrachtet werden. Deshalb wird für jedes Flächenpaar (f_i, g_j) , $f_i \in \mathcal{F}_A, g_j \in \mathcal{F}_B$ zum einen der Winkel α_{ij} zwischen den Ebenennormalen der Flächen f_i und g_j berechnet und zum anderen das Schnittpolygon P_{ij} . Das Schnittpolygon ist der Schnitt der zwei Polygone, die entstehen, wenn f_i und g_j auf eine gemittelte Ebene projiziert werden, die wie im Abschnitt zuvor beschrieben berechnet wird. Von diesem Schnittpolygon kann nun Flächeninhalt A_{ij} und Schwerpunkt \bar{P}_{ij} berechnet werden. Projiziert man den Schwerpunkt \bar{P}_{ij} wieder zurück auf die ursprünglichen Ebenen, sodass die Punkte \bar{P}_i und \bar{P}_j entstehen, so ist

$$d_{ij} = \|\bar{P}_i - \bar{P}_j\|_2 \quad (5.7)$$

ein Maß für den Abstand der beiden Flächen am Ort der Überlappung (Abbildung 5.3). Dieser Abstand existiert nur, wenn eine Überlappung vorhanden, also $A_{ij} > 0$, ist.

Als Kriterium für Flächenkorrespondenzen (siehe Definition 3.14) eignet sich somit folgende Bedingung:

Zwei Flächen $f_i \in \mathcal{F}_A$ und $g_j \in \mathcal{F}_B$ korrespondieren, wenn der Winkel zwischen den Normalen kleiner als eine Schwelle ist und ein Überlappung existiert und der Abstand am Ort des Überlappung kleiner als eine Schwelle ist, also:

$$\alpha_{ij} < \delta_{corr} \quad \wedge \quad A_{ij} > 0 \quad \wedge \quad d_{ij} < d_{corr}. \quad (5.8)$$

Beim Regionenwachstum während der Rekonstruktion (Kapitel 4.2.1) waren ähnliche Schwellen nötig. Dort wurde die Winkelschwelle konstant und die Abstandsschwelle abhängig von der Strukturgröße und der Messabweichung eines Punktes gewählt. An dieser Stelle kann auf die Berücksichtigung der Messabweichung verzichtet werden, da beide Flächen bereits durch Mittelung aus vielen Punkten entstanden sind, sodass direkt $d_{corr} = \delta_s$ verwendet werden kann.

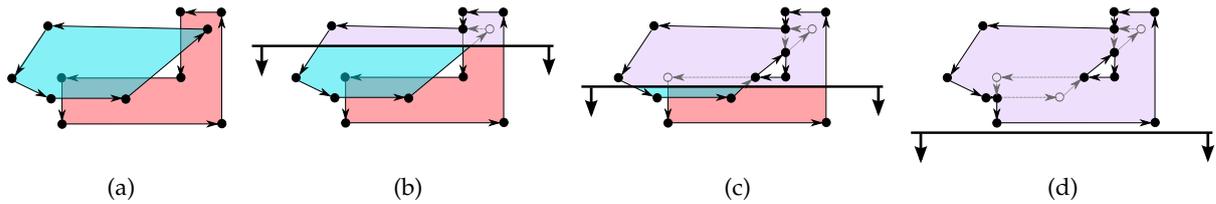


Abbildung 5.4: (a) Zwei Flächen mit ihren Begrenzungs-Halbkanten, (b-d) 2D-Plane-Sweep: Eine *Sweep-Line* durchläuft die Ebene. Alles oberhalb der Sweep-Line ist bereits korrekt vereinigt. Grau eingefärbte Kanten und Knoten liegen innerhalb der Vereinigung und sind nicht mehr Teil der neuen Begrenzung. An Schnittpunkten von Strecken wird ein neuer Knoten eingefügt und die schneidenden Halbkanten werden geteilt. Am Ende ist das Polygon korrekt durch Halbkanten abgegrenzt. Zusätzliche Begrenzungen in Form von Löchern können entstehen.

5.2.3 Vereinigung von Flächen

Nach der Berechnung von Flächenkorrespondenzen startet nun die eigentliche Fusion. Dazu wird die 3D-Fusion reduziert auf 2D-Vereinigungen, indem jede Korrespondenz an Flächen einzeln fusioniert wird. Das Ergebnis dieses Schrittes ist ein bis auf Zwillingkantbeziehungen konsistentes B-Rep-Modell. Um diese zu korrigieren, werden Paare von sogenannten Begrenzungsintervallen gefunden, die anschließend durch ein Paar an Zwillingskanten ersetzt werden. Den Abschluss bildet eine Ecken- und Ebenenoptimierung, wie sie auch bereits bei der Rekonstruktion eingesetzt wurde. Folgende Liste gibt nochmals eine Übersicht über die Schritte, die in den nachfolgenden Abschnitten im Detail dargestellt werden:

1. **2D-Flächen-Vereinigung:** Für alle korrespondierenden Flächen wird eine mittlere Ebene berechnet. Auf dieser Ebene wird jeweils eine 2D-Vereinigung durchgeführt.
2. **Bestimmung von Begrenzungsintervallen:** In jeder Fläche werden Begrenzungsintervalle gefunden, die als Basis für neue Zwillingskanten dienen.
3. **Kantenverknüpfung:** Begrenzungsintervalle aus unterschiedlichen Flächen werden zu Paaren gruppiert. Jedes Paar wird durch ein Paar an Zwillingshalbkanten ersetzt. Das Modell ist nun wieder konsistent.
4. **Optimierung:** Ecken und Ebenengleichungen werden optimiert.

2D-Flächen-Vereinigung

Die nachfolgenden Schritte werden im Folgenden der Übersichtlichkeit halber an zwei zu vereinigenden Flächen erläutert, sind aber genauso auf mehr als zwei Flächen anwendbar. Vor der Fusion wird eine mittlere Ebenengleichung berechnet, auf der die 2D-Fusion durchgeführt wird. Die Berechnung erfolgt gewichtet und wie in Abschnitt 5.2.2 beschrieben. Alle beteiligten Polygone werden auf diese mittlere Ebene projiziert.

Die Fusion selbst erfolgt mithilfe des *Map-Overlay-Algorithmus* [de Berg08], ein 2D-Plane-Sweep-Algorithmus aus dem Bereich der algorithmischen Geometrie. Dazu wird eine Gerade - die *Sweep-Line* - über die Ebene geschoben und Datenstrukturen vorgehalten, in denen alle Strecken sortiert abgelegt sind, die die Sweep-Line gerade schneiden. Dadurch ist bekannt, welche Strecke zu welcher entlang der Sweep-Line benachbart ist und somit auf Schnitt getestet werden muss. Während die Gerade wandert, wird diese Datenstruktur an den sogenannten *Event-Points* aktualisiert und Schnittpunkte zwischen benachbarten Strecken werden

bestimmt. Alle Knoten der Polygone und die während des Sweeps berechneten Schnittpunkte stellen dabei solche Event-Points dar. An keiner anderen Stelle kann sich die Nachbarschaftsbeziehung ändern. Durch diese Vorhaltung der Nachbarn ist diese Vorgehensweise effizienter als alle Strecken untereinander auf Schnitt zu prüfen. Abbildung 5.4 illustriert den Algorithmus grob. Für eine detaillierte Beschreibung des Algorithmus sei auf [de Berg08] verwiesen. Bei der Verwendung dieses Algorithmus sind folgende Punkte hervorzuheben:

- Während des Sweeps ist zu jeder Zeit bekannt, welche Teile der Polygone sich innerhalb und welche Teile sich außerhalb der Vereinigung befinden. Bei der Behandlung von Schnittpunkten ist somit direkt klar, ob diese auch Teil des Randes der Kontur des neuen, vereinigten Polygons sind oder nicht. Später nicht mehr benötigte Elemente können direkt gelöscht werden.
- Die in der *Half-Edge*-Datenstruktur vorhandenen Vorgänger- und Nachfolger-Beziehungen zwischen Kanten einer einzelnen Begrenzung können direkt während des Sweeps aktualisiert werden. Die entstehende Vereinigung ist also bis auf Zwillingkantbeziehungen zu anderen Flächen eine korrekte und konsistente *half-edge* Repräsentation.
- Bei der Vereinigung können neue Begrenzungen in Form von Löchern innerhalb einer Fläche entstehen.
- Die Richtung des Sweeps ist beliebig.

Das Ergebnis der Anwendung des Map-Overlay-Algorithmus auf jede Flächenkorrespondenz ist eine bis auf Zwillingkantbeziehungen konsistente B-Rep-Repräsentation, wobei jede Fläche die Vereinigung aller korrespondierenden Flächen auf einer mittleren Ebene darstellt. Als Gewicht einer vereinigten Fläche wird die Summe des Gewichts der Ausgangsflächen benutzt.

Die Ursache, warum die Zwillingkantbeziehungen fehlerhaft sein kann, liegt in der Lage einer vormals inzidenten Zwillingshalbkante während der 2D-Vereinigung. Dabei können genau drei Fälle auftreten (Abbildung 5.5):

- **A1:** Eine im Ausgangsmodell inzidente Halbkante ist auch nach der 2D-Fusion vollständig in der Begrenzung vorhanden (horizontale Fläche in B-Rep A der Zeile (a) in Abbildung 5.5). Dieser Fall tritt auch auf, wenn zu einer Fläche keine Flächenkorrespondenz besteht, sodass gar keine 2D-Vereinigung durchgeführt wurde (vertikale Fläche in B-Rep A der Zeile (a) und (b) in Abbildung 5.5).
- **A2:** Eine im Ausgangsmodell inzidente Halbkante ist nach der 2D-Fusion nicht (auch nicht teilweise) in der Begrenzung vorhanden, sondern lag bei der Vereinigung komplett im Inneren (horizontale Fläche in B-Rep A der Zeile (b) in Abbildung 5.5).
- **A3:** Eine im Ausgangsmodell inzidente Halbkante wurde bei der 2D-Fusion geteilt, sodass Teile davon in der Begrenzung vorhanden sind, andere Teile nicht (alle Flächen der Zeile (c) in Abbildung 5.5).

Je nachdem, welcher Fall für die zwei Zwillingshalbkanten einer Kante auftritt, kann die Zwillingkantbeziehung korrekt sein oder fehlerhaft aufgrund von fehlenden Partnern oder fehlerhaft aufgrund von Mehrfachverbindungen.

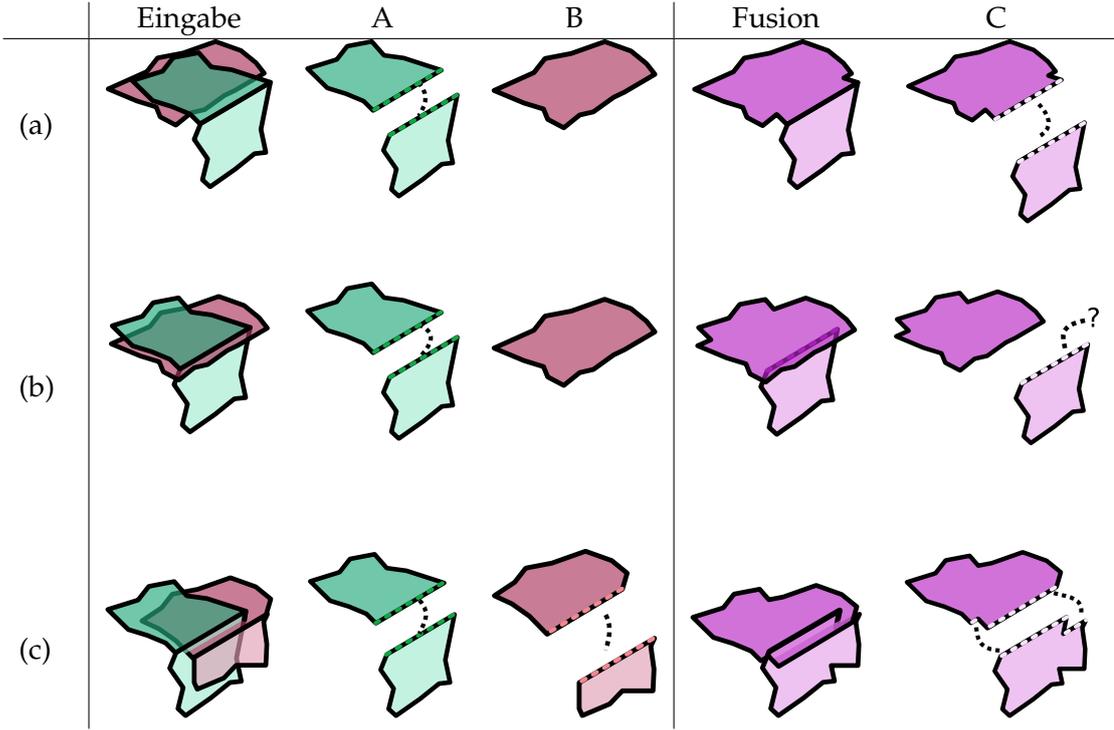


Abbildung 5.5: Beispiele von Zwillingskantenbeziehungen nach der 2D-Flächenvereinigung:
Spalte Eingabe: Zwei zu vereinigende B-Reps A (grün) und B (rot)
Spalte Fusion: B-Rep C nach der 2D-Flächenvereinigung
Spalten A/B/C: Separiert gezeichnete Flächen von A/B/C mit gekennzeichneter Zwillingskantenbeziehung
Zeile (a): Immer noch korrekte Zwillingskantenbeziehung in C (Fall A1)
Zeile (b): Fehlerhafte Zwillingskantenbeziehung in C wegen fehlenden Partners (Fall A2)
Zeile (c): Fehlerhafte Zwillingskantenbeziehung in C wegen Mehrfachverbindung (Fall A3)

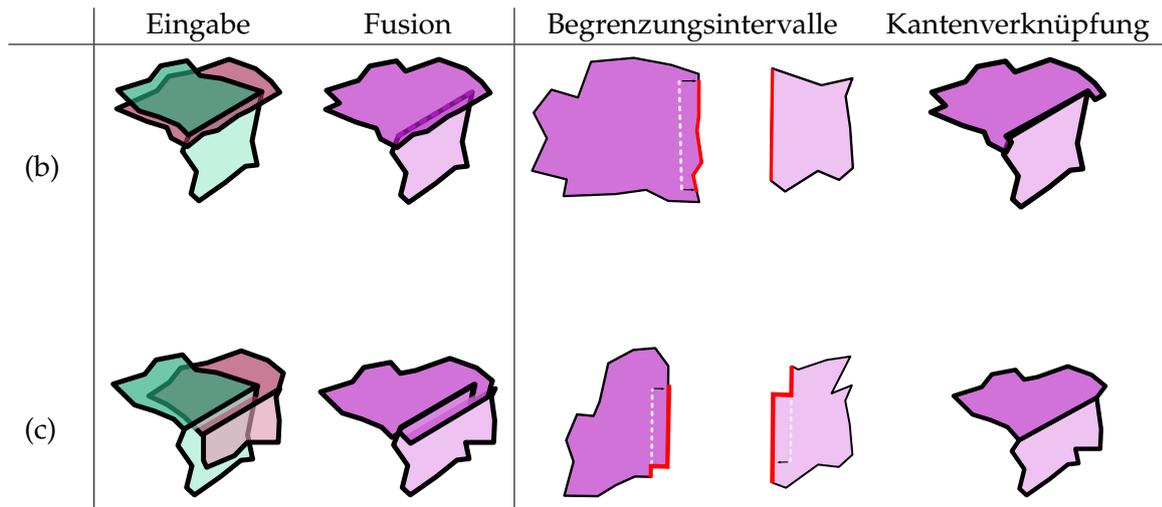


Abbildung 5.6: Beispiel der Korrektur der Zwillingskantenbeziehungen:

Zeile (b) und (c):

Fortsetzung der Beispiele aus Abbildung 5.5. Fall (a) muss nicht betrachtet werden, da die Zwillingskantenbeziehung bereits korrekt ist.

Spalte Eingabe:

Zwei zu vereinigende B-Reps (grün und rot)

Spalte Fusion:

B-Rep nach der 2D-Flächenvereinigung (violett)

Spalte Begrenzungsintervalle:

In beiden Begrenzungen werden durch Projektion von im Inneren der Vereinigung liegenden alten Zwillingshalbanten (gestrichelt) oder durch weiterhin existierende Zwillingshalbanten Intervalle gefunden (rot).

Spalte Kantenverknüpfung:

Diese Intervalle werden durch ein einziges Paar an Zwillingskanten ersetzt, sodass beide Flächen wieder korrekt verknüpft sind.

Bestimmung von Begrenzungsintervallen

Um diese Inkonsistenz zu beheben, werden zuerst Begrenzungsintervalle gefunden. Begrenzungsintervalle sind mehrere aufeinander folgende Halbanten in der Begrenzung nach der 2D-Vereinigung, die im folgenden Schritt der Kantenverknüpfung dann durch ein einziges Paar an Zwillingskanten ersetzt werden (Abbildung 5.6). Die Bestimmung dieser Intervalle erfolgt anhand der oben beschriebenen drei Fälle A1 bis A3:

- **A1** (vollständig in der Vereinigung vorhandene inzidente Halbante): Das Begrenzungsintervall besteht dann ausschließlich aus dieser Halbante (vertikale, helle Fläche in Zeile (b) der Abbildung 5.6).
- **A2** (im Inneren liegende und dadurch nicht in der Vereinigung vorhandene Halbante): Das Begrenzungsintervall wird bestimmt, indem alle Halbanten des fusionierten Polygons ermittelt werden, die geschnitten werden, wenn die im Inneren liegende ursprüngliche Halbante nach außen geschoben wird (horizontale, dunkle Fläche in Zeile (b) der Abbildung 5.6). Da Halbanten gerichtet sind, ist stets eindeutig, in welcher Richtung sich „außen“ befindet.
- **A3** (aufgrund von Teilung nur teilweise in der Vereinigung vorhandene Halbante): Das Begrenzungsintervall wird bestimmt, indem Begrenzungsintervalle für beide Teile mithilfe der obigen zwei Fälle getrennt bestimmt und anschließend vereinigt werden (beide Flächen in Zeile (c) der Abbildung 5.6).

Kantenverknüpfung

Nach dem Bestimmen der Begrenzungsintervalle werden diese nun zu Paaren gruppiert. Da eine inzidente Kante stets zwei adjazente Flächen besitzt, müssen immer genau zwei Intervalle aus benachbarten Flächen zusammen eine neue Kante bilden (Abbildung 5.7). Die Zuordnung zu Paaren ist leicht möglich, da bekannt ist, aus welchen ursprünglichen inzidenten Kanten das Begrenzungsintervall gebildet wurde. In diesen ursprünglichen inzidenten Kanten kann die Zwillingshalbkante ermittelt werden und von dieser ausgehend wieder das andere Begrenzungsintervall bestimmt werden.

Als letzter Schritt werden nun alle Paare von Intervallen durch ein Paar von Zwillingskanten ersetzt. Die Gerade, auf der diese Zwillingskante liegt, wird durch Schnitt der beiden beteiligten Ebenengleichungen ermittelt. Anfangs- und Endpunkt wird bestimmt, indem beide Intervalle auf die Gerade projiziert werden. Das Ergebnis ist in Abbildung 5.7i zu sehen.

Ecken- und Ebenenoptimierung

Wie bereits bei der Rekonstruktion eines Einzel-B-Reps, hängt die Lage eines Eckpunktes von der Reihenfolge ab, in der die Intervallpaare ersetzt wurden. Deshalb wird auch hier die in Abschnitt 4.2.3 beschriebene Optimierung der Position der Eckpunkte und Ebenengleichungen nochmals ausgeführt. Dadurch verbessert sich die Position der Ecken, da die nun gemittelten Ebenengleichungen eingehen.

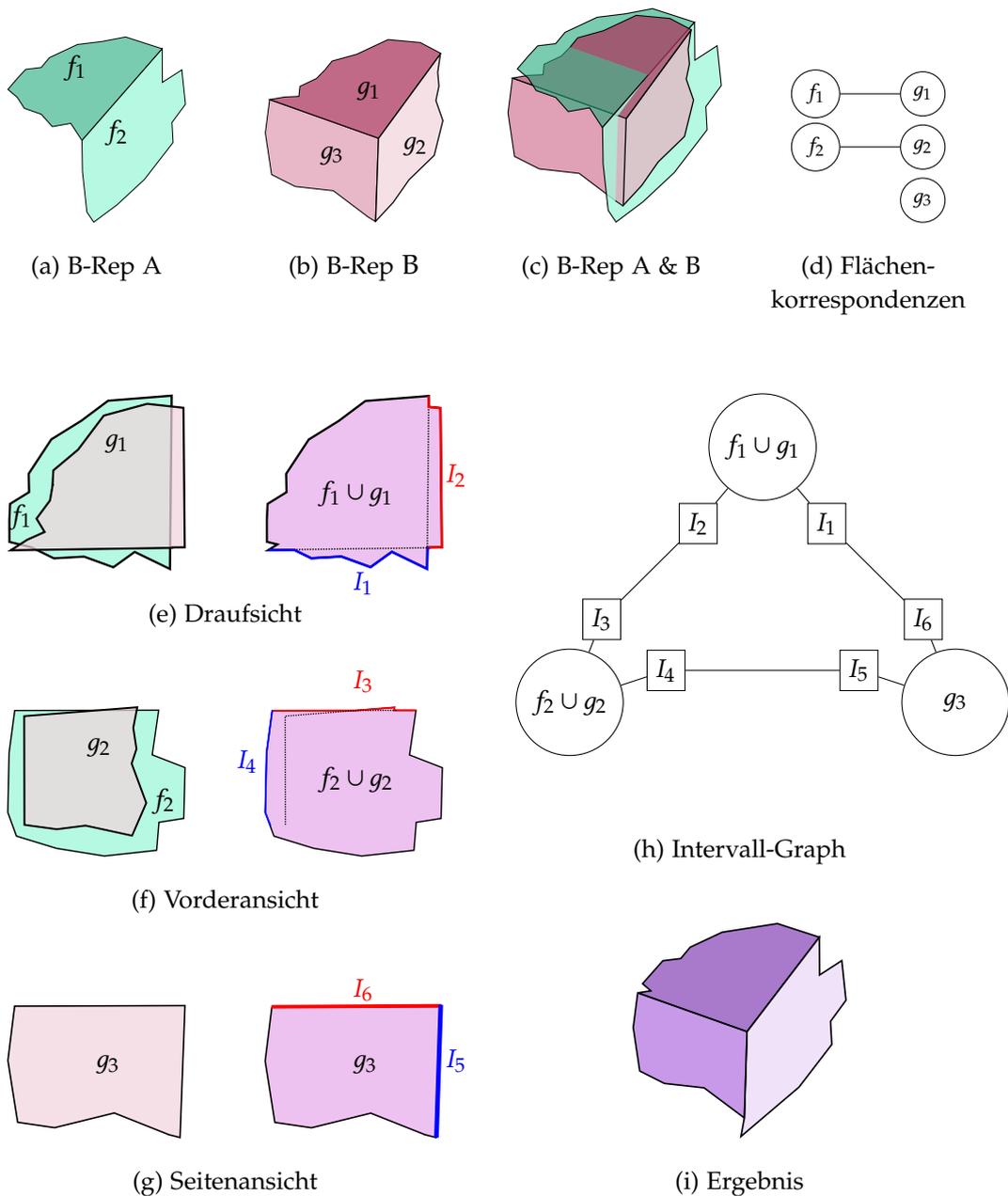


Abbildung 5.7: Beispiel der Korrektur von Zwillingskantenbeziehungen. Zwei zu fusionierende Ausgangsmodelle (a-c) mit ihren Flächenkorrespondenzen (d) werden pro Korrespondenz vereinigt (e-g). In der Vereinigung werden Begrenzungsintervalle (I_1 bis I_6) gefunden. Begrenzungsintervalle aus zwei unterschiedlichen Flächen bilden Paare (h). Nach der Ersetzung der Intervallpaare bei der Kantenverknüpfung ist das B-Rep korrigiert (i).

5.3 Evaluation

Wie schon bei der Rekonstruktion wird einerseits die Güte der B-Reps und andererseits die Laufzeit untersucht, um die Fusion zu evaluieren.

5.3.1 Güte der fusionierten B-Reps

Zur Evaluation der Fusion werden die selben Datensätze wie zur Evaluation der Rekonstruktion (Abschnitt 4.3.1) herangezogen. Pro Datensatz wird dazu folgendermaßen vorgegangen: Für alle Einzelpunktwolken werden, wie zuvor, partielle B-Reps rekonstruiert. Diese werden dann nacheinander fusioniert, sodass am Ende alle Blickrichtungen berücksichtigt werden. Das resultierende Modell wird dann mit dem vollständigen Ground-Truth-Gesamtmodell verglichen und mit denselben Maßen wie bei der Rekonstruktion evaluiert. Der einzige Unterschied besteht darin, dass im Gegensatz zur Rekonstruktion, bei der nur die Ecken und Kanten zwischen den erkannten Flächen verwendet werden, hier in der Ground-Truth stets alle Kanten und Ecken betrachtet werden, um auf ein vollständiges Modell zu prüfen.

Bei der Evaluation der Rekonstruktion wurde eine pro Datensatz passende Strukturgröße ausgewählt und die Einzelmodelle bewertet (Tabelle 4.3). Die Fusion dieser Einzelmodelle ist in Abbildung 5.8 dargestellt, der Vergleich mit dem vollständigen Ground-Truth-Modell ist in Tabelle 5.1 zu finden. Man erkennt, dass die Modelle nahezu vollständig sind, einzelne Ecken und Kanten fehlen. Der Grund für das Fehlen ist, dass während der Fusion alle Elemente eines partiellen B-Reps zwar vereinigt werden, aber keine neuen Kanten und Ecken erzeugt werden. Das heißt, eine Ecke ist im fusionierten Modell nur dann vorhanden, wenn sie auch in mindestens einem Einzelmodell erkannt wurde. Dazu ist es nötig, dass in einer Aufnahme mindestens drei adjazente Flächen sichtbar sind. Ist keine solche Kamerapose vorhanden, so fehlt diese Ecke im Endergebnis. Dies fällt besonders im Datensatz *engine* (Abbildung 5.8d) auf, da hier viele Ecken vorhanden sind, an denen vier oder mehr Flächen zusammenstoßen. Ebenso im *mix*-Datensatz, bei dem viele Ecken aufgrund von Verdeckungen nicht in den 22 verwendeten Kameraposen sichtbar sind.

Als nächstes soll der Einfluss des Rauschens betrachtet werden. Dazu wird der *anvil*-Datensatz verwendet, da dieser keine unterschiedlich großen Flächen enthält. Dadurch wird erreicht, dass eine gleiche Strukturgröße für alle Level an Rauschen verwendet werden kann. Der bei der Rekonstruktion auftretende Effekt, dass eine bestimmte Strukturgröße aufgrund des Rauschens nicht eingehalten werden kann, diese aber für kleine Flächen nötig ist, wird dadurch ausgeschlossen. Somit kann die Güte der Fusion bewertet werden ohne durch ungenügende partielle Rekonstruktionen beeinflusst zu werden. Die Tabelle 5.2 sowie die Abbildungen 5.9 und 5.10 zeigen die Ergebnisse der Fusion im Vergleich zur vollständigen Ground-Truth. Die Genauigkeit und Anzahl der korrekten Elemente nimmt dabei mit zunehmendem Rauschen ab. Betrachtet man die Modelle, so zeigt sich der Grund für die fehlenden Elemente: Die Rekonstruktion einzelner partieller Modelle weicht durch das Rauschen soweit von der Wahrheit ab, dass auch die Fusion dies nicht mehr korrigieren kann.

	Anzahl Ground-Truth	Anzahl Rekonstruktion	Korrekt	Fehlend	Rauschen	Über- segmentiert	Unter- segmentiert	Abweichung Position [10^{-3} m]	Abweichung Orientierung [°]
Datensatz: chest, Strukturgröße $\delta_s = 70mm$, Fusion aus 11 partiellen B-Reps									
Flächen	8.00	8.00	8.00 (100.0%)	0.00	0.00	0.00	0.00	0.03	0.01
Kanten	18.00	18.00	18.00 (100.0%)	0.00	0.00	0.00	0.00	0.04	0.01
Ecken	12.00	11.00	11.00 (91.7%)	1.00	0.00	-	-	0.10	-
Datensatz: anvil, Strukturgröße $\delta_s = 50mm$, Fusion aus 11 partiellen B-Reps									
Flächen	8.00	8.00	8.00 (100.0%)	0.00	0.00	0.00	0.00	0.05	0.03
Kanten	18.00	18.00	18.00 (100.0%)	0.00	0.00	0.00	0.00	0.12	0.03
Ecken	12.00	12.00	12.00 (100.0%)	0.00	0.00	-	-	0.35	-
Datensatz: nut, Strukturgröße $\delta_s = 30mm$, Fusion aus 14 partiellen B-Reps									
Flächen	18.00	18.00	18.00 (100.0%)	0.00	0.00	0.00	0.00	0.02	0.01
Kanten	48.00	48.00	48.00 (100.0%)	0.00	0.00	0.00	0.00	0.03	0.02
Ecken	32.00	32.00	32.00 (100.0%)	0.00	0.00	-	-	0.08	-
Datensatz: engine, Strukturgröße $\delta_s = 50mm$, Fusion aus 11 partiellen B-Reps									
Flächen	18.00	18.00	18.00 (100.0%)	0.00	0.00	0.00	0.00	0.07	0.03
Kanten	30.00	31.00	29.00 (96.7%)	0.00	0.00	1.00	0.00	0.11	0.03
Ecken	14.00	10.00	10.00 (71.4%)	4.00	0.00	-	-	0.26	-
Datensatz: ashtray, Strukturgröße $\delta_s = 40mm$, Fusion aus 11 partiellen B-Reps									
Flächen	14.00	14.00	14.00 (100.0%)	0.00	0.00	0.00	0.00	0.05	0.17
Kanten	28.00	29.00	27.00 (96.4%)	0.00	0.00	1.00	0.00	0.60	0.30
Ecken	18.00	17.00	17.00 (94.4%)	1.00	0.00	-	-	2.09	-
Datensatz: crane, Strukturgröße $\delta_s = 45mm$, Fusion aus 11 partiellen B-Reps									
Flächen	12.00	12.00	12.00 (100.0%)	0.00	0.00	0.00	0.00	0.03	0.01
Kanten	30.00	30.00	30.00 (100.0%)	0.00	0.00	0.00	0.00	0.06	0.02
Ecken	20.00	20.00	20.00 (100.0%)	0.00	0.00	-	-	0.15	-
Datensatz: mix, Strukturgröße $\delta_s = 35mm$, Fusion aus 22 partiellen B-Reps									
Flächen	78.00	77.00	74.00 (94.9%)	1.00	0.00	1.00	1.00	0.12	0.11
Kanten	164.00	143.00	141.00 (86.0%)	22.00	0.00	1.00	0.00	0.28	0.18
Ecken	105.00	76.00	76.00 (72.4%)	29.00	0.00	-	-	0.96	-

Tabelle 5.1: Statistische und quantitative Ergebnisse nach Fusion der partiellen B-Reps pro Datensatz bei einem Rauschlevel von $\sigma = 0.001$ und derselben Strukturgröße, die bei der Rekonstruktion (Tabelle 4.3) verwendet wurde. In Abbildung 5.8 sind die Ergebnisse als Bilder dargestellt.

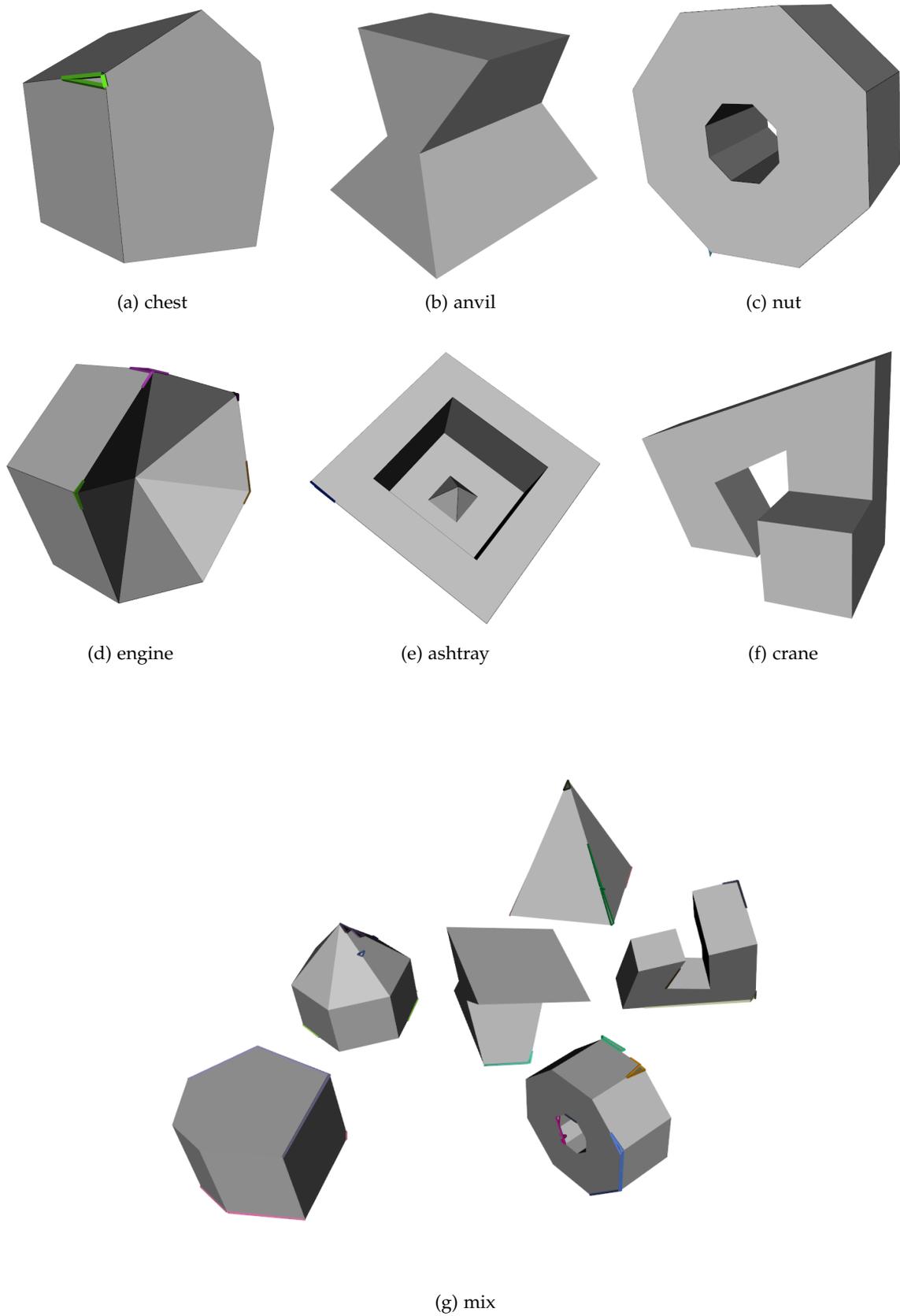


Abbildung 5.8: B-Reps nach der Fusion aller partiellen B-Reps. Löcher im Modell sind farbig markiert. Die Auswertung dieser Modelle ist in Tabelle 5.1 zu finden.

	Anzahl Ground-Truth	Anzahl Rekonstruktion	Korrekt	Fehlend	Rauschen	Über- segmentiert	Unter- segmentiert	Abweichung Position [$10^{-3} m$]	Abweichung Orientierung [$^{\circ}$]
$\sigma = 0.001$									
Flächen	8.00	8.00	8.00 (100.0%)	0.00	0.00	0.00	0.00	0.05	0.03
Kanten	18.00	18.00	18.00 (100.0%)	0.00	0.00	0.00	0.00	0.12	0.03
Ecken	12.00	12.00	12.00 (100.0%)	0.00	0.00	-	-	0.35	-
$\sigma = 0.0015$									
Flächen	8.00	8.00	8.00 (100.0%)	0.00	0.00	0.00	0.00	0.08	0.05
Kanten	18.00	18.00	18.00 (100.0%)	0.00	0.00	0.00	0.00	0.27	0.08
Ecken	12.00	12.00	12.00 (100.0%)	0.00	0.00	-	-	0.87	-
$\sigma = 0.002$									
Flächen	8.00	8.00	8.00 (100.0%)	0.00	0.00	0.00	0.00	0.37	0.24
Kanten	18.00	18.00	18.00 (100.0%)	0.00	0.00	0.00	0.00	0.96	0.33
Ecken	12.00	11.00	11.00 (91.7%)	1.00	0.00	-	-	3.08	-
$\sigma = 0.0028$									
Flächen	8.00	8.00	8.00 (100.0%)	0.00	0.00	0.00	0.00	0.67	0.34
Kanten	18.00	21.00	15.00 (83.3%)	0.00	0.00	3.00	0.00	1.95	0.48
Ecken	12.00	5.00	5.00 (41.7%)	7.00	0.00	-	-	5.11	-
$\sigma = 0.004$									
Flächen	8.00	9.00	8.00 (100.0%)	0.00	1.00	0.00	0.00	1.41	1.01
Kanten	18.00	19.00	13.00 (72.2%)	2.00	0.00	3.00	0.00	5.08	1.45
Ecken	12.00	4.00	4.00 (33.3%)	8.00	0.00	-	-	18.97	-

Tabelle 5.2: Statistische und quantitative Ergebnisse nach Fusion der 11 partiellen B-Reps des *anvil*-Datensatzes mit einer Strukturgröße von $\delta_s = 50mm$. In Abbildung 5.10 sind die Ergebnisse als Bilder dargestellt.

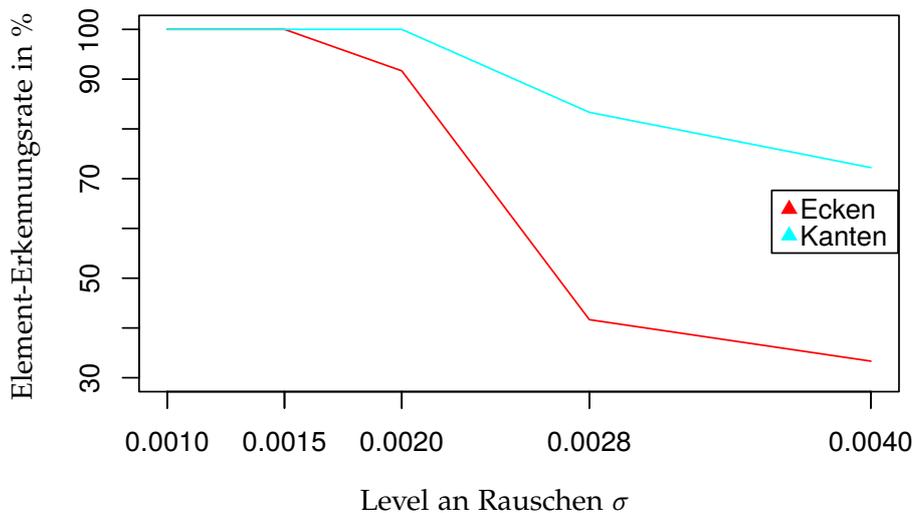


Abbildung 5.9: Kanten- und Eckenerkennungsrate des *anvil*-Datensatzes (11 Einzelmodelle) bei einer Strukturgröße von $\delta_s = 50mm$ in Abhängigkeit des Levels an Rauschen.

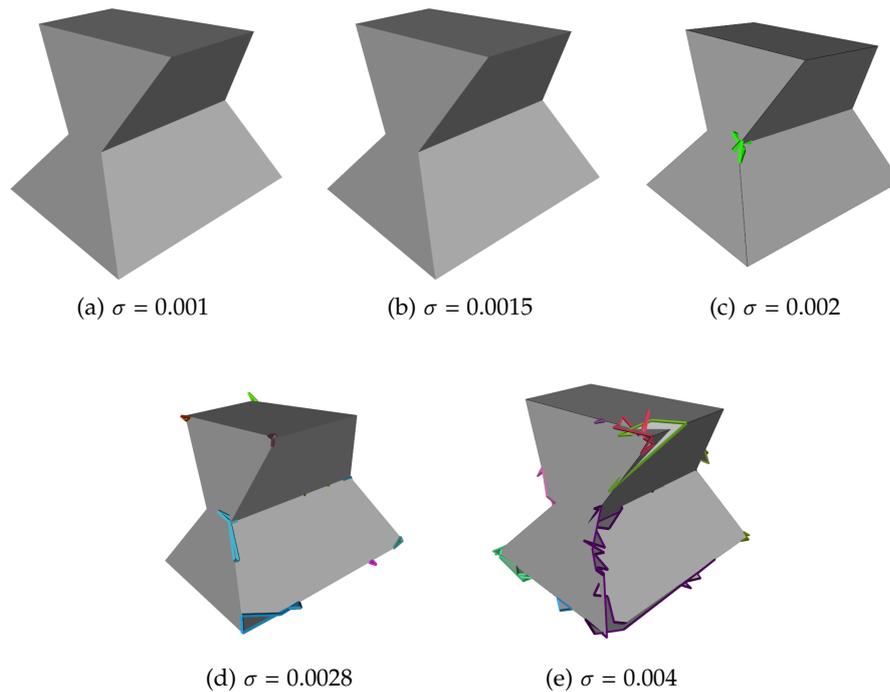


Abbildung 5.10: Resultierende Modelle des *anvil*-Datensatzes nach der Fusion aller 11 Einzelmodelle bei unterschiedlichem Rauschen ($\delta_s = 50mm$). Fehlerhafte Stellen sind farblich hervorgehoben. Die zugehörige Auswertung ist in Tabelle 5.2 zu finden.

Da Rauschen ein stochastischer Effekt ist, werden für den *anvil*-Datensatz weitere 20 Einzelposen mit dem höchsten Rausch-Niveau von $\sigma = 0.004$ erzeugt, sodass letztendlich 31 verschiedene Einzelmodelle vorliegen. Damit kann nun untersucht werden, wie stark die Güte des Fusionsergebnisses von der Anzahl der Einzelmodelle abhängt. Es sei angemerkt, dass die ersten elf betrachteten Aufnahmeasen bereits für eine vollständige Rekonstruktion ausreichend sind, wie man am Rausch-Level $\sigma = 0.001$ erkennt. Die zusätzlichen Posen sind somit redundant. Ausgehend von den elf Einzelmodellen werden nun schrittweise weitere Modelle fusioniert. In Tabelle 5.3 und in den Abbildungen 5.11 und 5.12 ist das Ergebnis dargestellt. Man erkennt, dass die Erfolgsrate mit zunehmender Anzahl an Einzelmodellen zunimmt. Daraus lässt sich schließen, dass bei einem hohen Level an Rauschen mehr Einzelbilder für eine vollständige Rekonstruktion benötigt werden als bei geringem Rauschen, um zum gleichen Ergebnis zu gelangen.

Dass Fehler in den partiellen Modellen durch die Fusion einer größeren Anzahl an Einzelbildern reduziert werden können, gilt allerdings nur für stochastische Fehler, wie ungenaue Ebenengleichungen oder Kanten. Weist ein Einzelbild eine falsche Topologie auf, beispielsweise durch eine fehlerhafte Segmentierung, so kann dies auch durch mehrere Fusionen nicht behoben werden. Abbildung 5.13 zeigt ein solches Beispiel, bei dem bei einem Einzelbild zwei kleine Flächen fälschlicherweise als ein Segment erkannt werden. Nach der Fusion des Modells kann dies auch durch weitere korrekte Einzelbilder nicht mehr korrigiert werden, da der vorgestellte Ansatz kein Teilen von Flächen zur Änderung der Topologie vorsieht. Im schlimmsten Fall bedeutet dies, dass das Gesamtergebnis durch eine einzige, stark falsche Rekonstruktion korrumpiert werden kann.

	Anzahl Ground-Truth	Anzahl Rekonstruktion	Korrekt	Fehlend	Rauschen	Über- segmentiert	Unter- segmentiert	Abweichung Position [10^{-3} m]	Abweichung Orientierung [°]
11 Einzelmodelle									
Flächen	8.00	9.00	8.00 (100.0%)	0.00	1.00	0.00	0.00	1.41	1.01
Kanten	18.00	19.00	13.00 (72.2%)	2.00	0.00	3.00	0.00	5.08	1.45
Ecken	12.00	4.00	4.00 (33.3%)	8.00	0.00	-	-	18.97	-
13 Einzelmodelle									
Flächen	8.00	9.00	8.00 (100.0%)	0.00	1.00	0.00	0.00	1.89	1.20
Kanten	18.00	19.00	13.00 (72.2%)	2.00	0.00	3.00	0.00	6.10	1.89
Ecken	12.00	4.00	4.00 (33.3%)	8.00	0.00	-	-	24.09	-
16 Einzelmodelle									
Flächen	8.00	9.00	8.00 (100.0%)	0.00	1.00	0.00	0.00	1.46	0.81
Kanten	18.00	21.00	13.00 (72.2%)	1.00	0.00	4.00	0.00	5.34	1.27
Ecken	12.00	7.00	7.00 (58.3%)	5.00	0.00	-	-	12.49	-
19 Einzelmodelle									
Flächen	8.00	9.00	8.00 (100.0%)	0.00	1.00	0.00	0.00	1.27	0.60
Kanten	18.00	21.00	15.00 (83.3%)	0.00	0.00	3.00	0.00	4.72	0.94
Ecken	12.00	11.00	11.00 (91.7%)	1.00	0.00	-	-	9.10	-
22 Einzelmodelle									
Flächen	8.00	9.00	8.00 (100.0%)	0.00	1.00	0.00	0.00	2.24	0.95
Kanten	18.00	19.00	17.00 (94.4%)	0.00	0.00	1.00	0.00	5.65	1.31
Ecken	12.00	12.00	12.00 (100.0%)	0.00	0.00	-	-	15.28	-
25 Einzelmodelle									
Flächen	8.00	9.00	8.00 (100.0%)	0.00	1.00	0.00	0.00	2.21	0.95
Kanten	18.00	18.00	18.00 (100.0%)	0.00	0.00	0.00	0.00	4.97	1.36
Ecken	12.00	12.00	12.00 (100.0%)	0.00	0.00	-	-	15.39	-
28 Einzelmodelle									
Flächen	8.00	9.00	8.00 (100.0%)	0.00	1.00	0.00	0.00	1.71	0.68
Kanten	18.00	18.00	18.00 (100.0%)	0.00	0.00	0.00	0.00	3.75	0.98
Ecken	12.00	12.00	12.00 (100.0%)	0.00	0.00	-	-	10.77	-
31 Einzelmodelle									
Flächen	8.00	9.00	8.00 (100.0%)	0.00	1.00	0.00	0.00	1.60	0.64
Kanten	18.00	18.00	18.00 (100.0%)	0.00	0.00	0.00	0.00	3.38	0.90
Ecken	12.00	12.00	12.00 (100.0%)	0.00	0.00	-	-	9.34	-

Tabelle 5.3: Statistische und quantitative Ergebnisse nach Fusion von einer unterschiedlichen Anzahl an Einzelmodellen des *anvil*-Datensatzes mit einer Strukturgröße von $\delta_s = 50\text{mm}$ bei einem Rausch-Level von $\sigma = 0.004$. In Abbildung 5.12 sind die Ergebnisse als Bilder dargestellt.

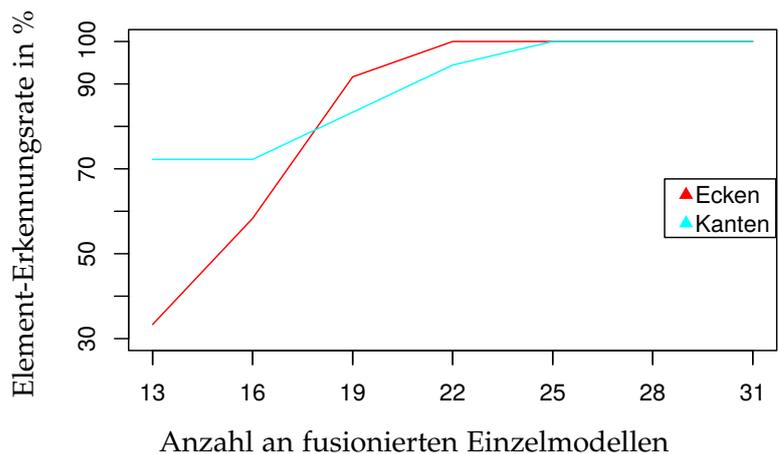


Abbildung 5.11: Kanten- und Eckenerkennungsrate im *anvil*-Datensatzes bei einer Strukturgröße von $\delta_s = 50mm$ und $\sigma = 0.004$ in Abhängigkeit der Anzahl an fusionierten Einzelmodellen.

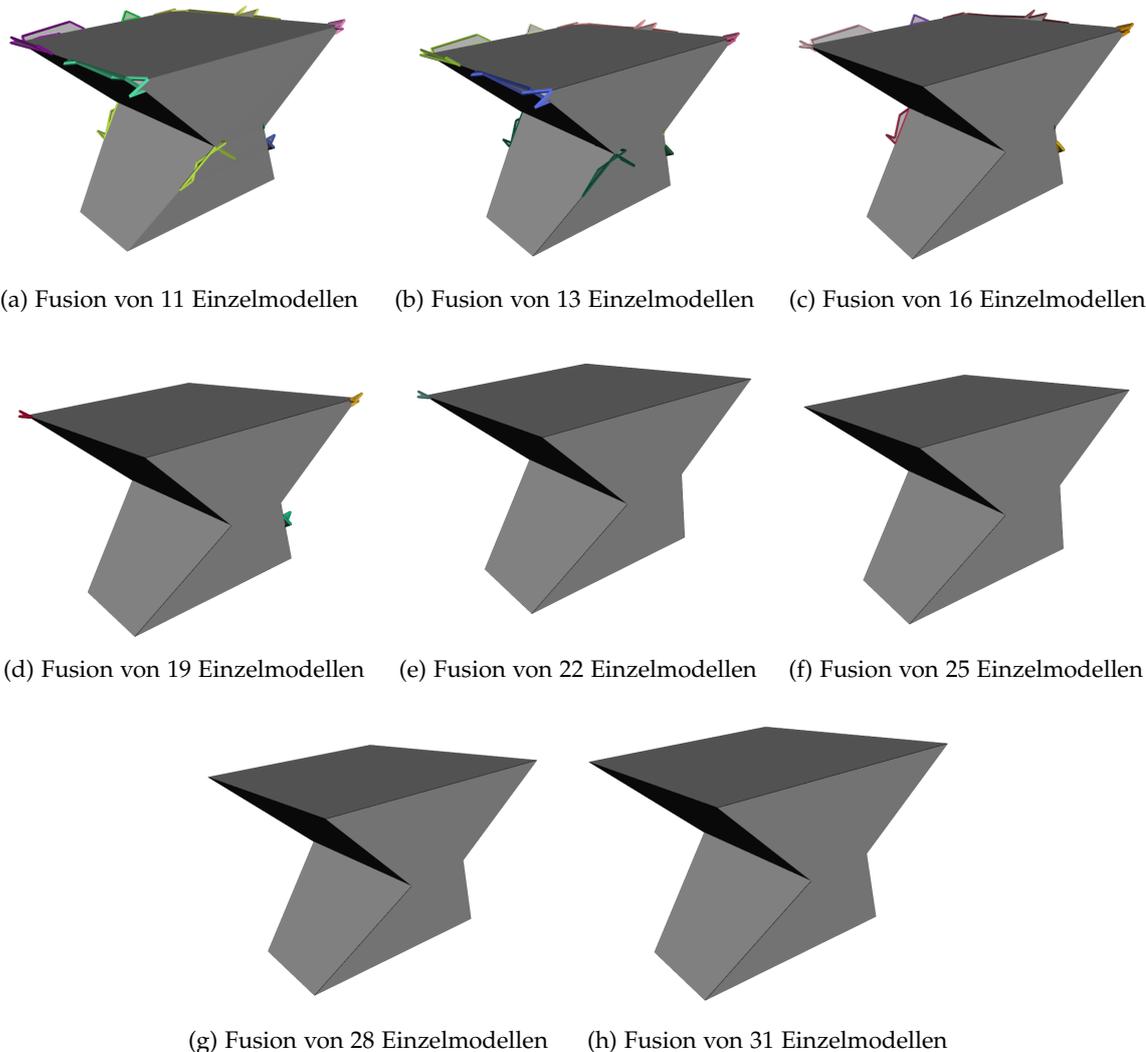


Abbildung 5.12: Resultierende Modelle des *anvil*-Datensatzes nach der Fusion einer unterschiedlichen Anzahl an stark verrauschten Einzelmodellen ($\sigma = 0.004$, $\delta_s = 50mm$). Fehlerhafte Stellen sind farblich hervorgehoben. Die zugehörige Auswertung ist in Tabelle 5.3 zu finden.

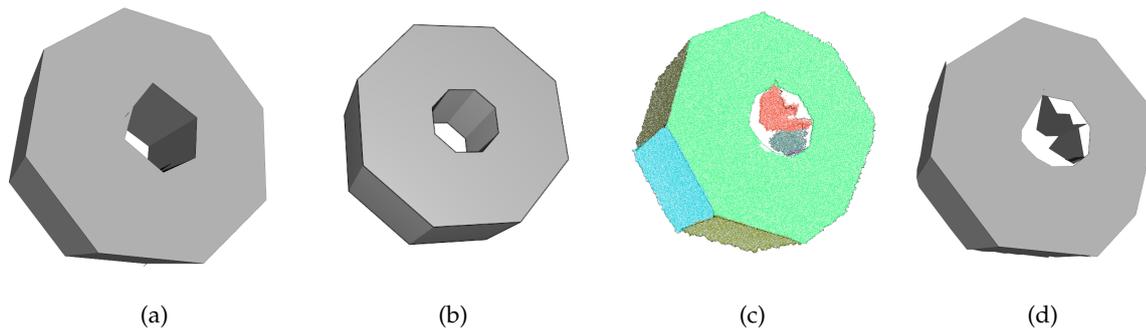


Abbildung 5.13: Das resultierende Modell des *nut*-Datensatzes nach der Fusion bei $\sigma = 0.004$ und $\delta_s = 70mm$ (a) unterscheidet sich von der Ground-Truth (b) durch eine unterschiedliche Anzahl an Flächen. Der Grund ist, dass ein einziges partielles B-Rep eine fehlerhafte Segmentierung (rote Fläche) aufweist (c), sodass eine Untersegmentierung auftritt (d), die auch durch weitere Fusionen nicht mehr korrigiert werden kann.

5.3.2 Laufzeit

Im Gegensatz zur Rekonstruktion ist die Fusion nicht abhängig von der Auflösung der Eingabepunktwolke, da bei der Fusion nur partielle B-Reps und keine Punktwolken betrachtet werden. Vielmehr hängt die Fusion nur von der Anzahl der korrespondierenden Flächen ab, für die der Plane-Sweep-Algorithmus ausgeführt werden muss. Diese Abhängigkeit konnte in der Praxis allerdings nicht gemessen werden, da die Fusion mit im Schnitt 10 Millisekunden zu gering ist, um eine Abhängigkeit sinnvoll darzustellen. Im Vergleich zur Zeit zur Rekonstruktion ist die Zeit zur Fusion somit vernachlässigbar.

5.4 Zusammenfassung

In diesem Kapitel wurde ein Ansatz zur Fusion zweier partieller B-Reps vorgestellt. Verfahren aus der Literatur konnten dabei nicht angewendet werden, da diese entweder nicht für B-Rep Modelle geeignet sind oder stets vollständige Modelle erwarten. Die Fusion muss zudem zwei Aufgaben lösen: Das Resultat soll in den Überlappungsbereichen genauer und in den anderen Bereichen vollständiger sein als die Ausgangsmodelle. Die B-Rep Datenstruktur muss dabei konsistent gehalten werden. Diese Aufgabenstellungen wurden gelöst, indem das Problem der 3D-Fusion auf eine flächenweise 2D-Fusion reduziert wurde. Durch die gewichtete Mittelung von Ebenen werden korrespondierende Ebenengleichungen verbessert. Durch die Fusion mithilfe eines Plane-Sweep-Ansatzes kann die *half-edge*-Datenstruktur zudem während der Fusion bis auf die Zwillingskantenbeziehungen vollständig konsistent gehalten werden. Die anschließende Kantenverknüpfung stellt die topologische Korrektheit wieder her, indem Kanten zwischen benachbarten Flächen angepasst werden.

Die Evaluation wurde auf denselben Daten durchgeführt, die schon zur Rekonstruktion verwendet wurden. Fast alle Modelle sind nach der Fusion vollständig rekonstruiert. Da nicht mehr auf Punktwolken, sondern auf partiellen Modellen gearbeitet wird, ist die Zeit zur Fusion im Vergleich zur Rekonstruktion vernachlässigbar.

Insgesamt lässt sich damit die Fragestellung F2 beantworten: Eine Kombination von SLAM und DSR in einem 3D-Scan-System ist möglich und kann online ausgeführt werden. Durch direkte Rekonstruktion einer organisierten Punktwolke zu einem partiellen B-Rep und anschließende Fusion kann inkrementell ein Modell erzeugt werden. Das Modell wird vollständiger,

je mehr Kameraposen integriert werden. Folgende Grenzen des Verfahren konnten dabei festgestellt werden:

Durch die Fusion werden keine neuen Ecken oder Kanten erzeugt. Eine Ecke oder Kante ist im fusionierten Modell nur dann vorhanden, wenn sie in mindestens einem Einzelbild erfasst wurde. In Kapitel 7 wird deshalb ein Ansatz zur Vervollständigung von B-Reps vorgestellt. Eine Folge daraus ist, dass je höher das Rauschen ist, desto mehr Einzelbilder fusioniert werden müssen, um ein vollständiges Modell zu erhalten, da bei höherem Rauschen öfter Kanten oder Ecken in den Einzelbildern fehlen können.

Durch die Mittelung werden abweichende geometrische Eigenschaften wie Ebenengleichungen, Kanten usw. durch weitere Aufnahmen verbessert. Weist eine Einzelaufnahme allerdings eine fehlerhafte Topologie auf, beispielsweise durch eine Übersegmentierung während der Rekonstruktion, kann dies im vorgestellten Fusionsansatz auch nicht mehr korrigiert werden.

Kapitel 6

Registrierung partieller B-Reps

Inhalt

6.1	Stand der Forschung	119
6.2	Vorgehensweise	121
6.2.1	Merkmale	122
6.2.2	Deskriptoren	123
6.2.3	Korrespondenzen	126
6.2.4	Posenhypothesen	127
6.2.5	Posenbewertung	128
6.2.6	Posenoptimierung	133
6.3	Evaluation	135
6.3.1	Datensätze	135
6.3.2	Auswertung	137
6.3.3	Ergebnisse	137
6.4	Zusammenfassung	143

Mit den in den Kapiteln 4 und 5 dargestellten Methoden lassen sich Modelle erzeugen und fusionieren, wenn die Aufnahmepose (also Position und Orientierung im Raum) bekannt ist. Dies ist beispielsweise bei Kameras der Fall, die an einem Roboterarm montiert sind (*eye-in-hand setup*). Ist die Pose unbekannt, beispielsweise bei handgehaltenen Kameras, so muss diese erst bestimmt werden. In diesem Kapitel wird daher die Problemstellung der Berechnung der relativen Pose zwischen zwei B-Rep Modellen, auch als Registrierung bezeichnet, betrachtet.

Eingabe sind also zwei partielle B-Rep Modelle in verschiedenen lokalen Koordinatensystemen, gesucht ist eine Starrkörpertransformation, die vom Koordinatensystem des einen B-Reps in das des zweiten überführt. Folgende Eigenschaften der Eingabedaten werden dabei angenommen:

- Es existiert ein Überlappungsbereich zwischen beiden B-Reps. Wie groß der Überlappungsbereich sein muss, wird im Laufe des Kapitels weiter untersucht.
- Beide Modelle sind gleich skaliert, d.h. es wird keine Skalierung während der Registrierung betrachtet. Bei der Verwendung von Tiefenkameras, die die Daten in absoluten Maßen (Meter) zurückliefern, ist dies stets der Fall.

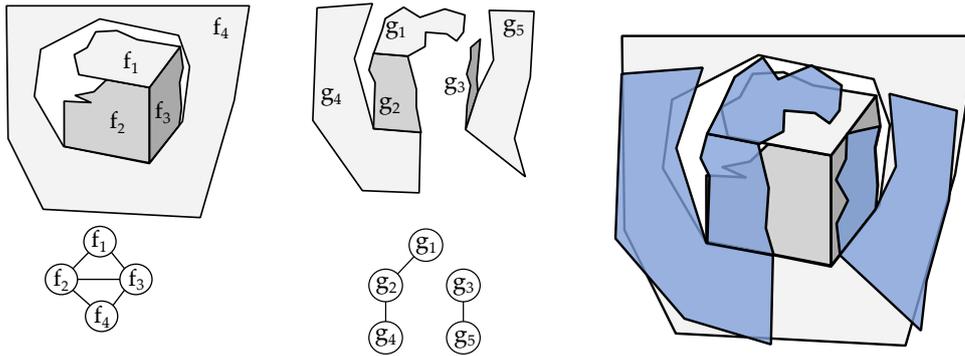


Abbildung 6.1: Als Registrierung wird der Vorgang bezeichnet, eine relative Transformation zwischen zwei partiellen B-Reps (links, mitte) zu berechnen, die beide Modelle ausrichtet (rechts). Obwohl beide Modelle dieselbe Szene zeigen, können die Topologiegraphen durch Verdeckungen komplett verschieden sein.

Folgende Eigenschaften der Eingabedaten werden explizit berücksichtigt, um der Problemstellung dieser Arbeit gerecht zu werden:

- Die Eingabemodelle können durch den Blickwinkel oder durch Verdeckungen unvollständig sein. Dadurch können gleiche Flächen in mehrere Teilflächen aufgeteilt sein oder Kanten und Ecken fehlen (Abbildung 6.1). Beides hat zur Folge, dass die Topologiegraphen sehr unterschiedlich sein können oder keine gemeinsamen Eckpunkte existieren.
- Beide Modelle sind durch Sensorrauschen beeinflusst.

Das Kapitel ist in vier Abschnitte unterteilt. Zuerst werden Registrierungsverfahren aus der Literatur untersucht und bewertet (Abschnitt 6.1). Anschließend wird ein neuartiges Verfahren zur Lösung des Registrierungsproblem vorgestellt (Abschnitt 6.2) und evaluiert (Abschnitt 6.3). Den Abschluss bildet eine Zusammenfassung des Kapitels (Abschnitt 6.4). Teile des in diesem Kapitel vorgestellten Ansatzes wurden bereits in [Sand17] veröffentlicht.

6.1 Stand der Forschung

Die Aufgabe der Registrierung zweier Modelle ist die Berechnung der relativen Transformation, die in diesem Fall eine Starrkörpertransformation ist, da keine Scherungen oder Skalierung betrachtet werden. Dies wird auch als *rigid-alignment* bezeichnet.

Der klassische Ansatz zur Lösung des Registrierungsproblems zwischen zwei Punktwolken ist der *Iterative Closest Points* (ICP)-Algorithmus [Besl92]. Dabei werden, beginnend bei einer Initialpose, Korrespondenzen zwischen Punkten der einen und Punkten der anderen Punktwolke gefunden. Aus diesen Korrespondenzen wird wiederum eine verbesserte Transformation berechnet. Beide Schritte werden nun iteriert, bis die Pose konvergiert. Seit Veröffentlichung wurden etliche Varianten und Verbesserungen vorgestellt, die unterschiedliche Metriken, Suche von Korrespondenzen oder Beschleunigungen vorschlagen. Eine Übersicht ist in Rusinkiewicz et al. [Rusinkiewicz01] und Pomerleau et al. [Pomerleau13] zu finden. Der Nachteil an allen ICP-basierten Methoden ist die Notwendigkeit einer initialen Posenschätzung. Ist keine oder nur eine schlechte vorhanden, so konvergiert das Verfahren nicht oder in ein lokales Minimum, das nicht der gewünschten Lösung entspricht. Dies ist insbesondere bei einer nur partiellen Überlappung der Ausgangspunktwolken der Fall. Aus diesem Grund wird der ICP-Algorithmus auch häufig nur zur Feinregistrierung eingesetzt und für die initiale Pose ein anderes Verfahren verwendet.

Robuster gegenüber der initialen Pose ist die Normalverteilungs-Transformation (*normal-distributions transform*, NDT), die ursprünglich für den \mathbb{R}^2 vorgestellt wurde [Biber03, Biber04]. Dazu wird ein Gitter über eine Punktwolke gelegt und für jede Zelle eine Mischung aus Normalverteilungen an die Punkte angepasst. Dies liefert eine lokale Approximation der Oberfläche, für die zudem Ableitungen berechnet werden können. Diese Umwandlung wird als Normalverteilungs-Transformation bezeichnet. Die zweite Punktwolke wird nun an die Menge von Gauss-Mixturen angepasst, indem die Transformationsparameter mittels Maximum-Likelihood-Abschätzung und Newton-Verfahren bestimmt werden. Eine Erweiterung auf drei Dimensionen ist leicht möglich und ein Vergleich mit ICP zeigt ein robusteres Ergebnis [Magnusson09]. Ein Nachteil ist die Notwendigkeit eines Gitters, da die Zellengröße auch das Ergebnis beeinflusst.

Werden beide zu registrierenden Seiten in Gauss-Mixturen umgewandelt, so spricht man von *gaussian mixture alignment* (GMA). Der Vorteil gegenüber dem ursprüngliche ICP-Algorithmus ist, dass keine expliziten Korrespondenzen zwischen einzelnen Punkten mehr gefunden werden müssen. Dadurch ist das Verfahren robuster bei nicht einheitlicher Punktdichte oder partieller Überlappung [Jian11, Campbell15]. Dennoch ist das Verfahren eine lokale Optimierung, ausgehend von einer Startschätzung.

Um das global Optimum zu finden, wurden Verfahren vorgestellt, die unabhängig von einer Startschätzung den Suchraum effizient durchsuchen, beispielsweise mittels eines *Branch-and-Bound* Ansatzes auf Punktwolken [Yang16] oder auf Gauss-Mixturen [Campbell16]. Durch die Suche sind globale Verfahren rechenaufwändiger als lokale.

Statt alle Punkte einer Punktwolke zu verwenden, ist ein häufig genutzter Ansatz die Verwendung von Merkmalspunkten (*feature points*), die lokale Eigenschaften an besonderen Punkten der Oberfläche beschreiben. Merkmalsbasierte Verfahren sind zudem nicht auf Punktwolken beschränkt, sodass im Folgenden von Ausgangsmodellen statt Punktwolken gesprochen wird. Das grundlegende Vorgehen ist folgendes: Zuerst werden aus beiden Ausgangsmodellen Merkmale extrahiert und deren Eigenschaften in einem sogenannten Deskriptor zusammengefasst. Anschließend werden Korrespondenzen zwischen den Merkmalen des einen Modells zu den Merkmalen des anderen gefunden. Aus diesen Korrespondenzen wird dann die finale Transformation berechnet.

Für Punktwolken und Dreiecksnetze existiert eine Vielzahl an unterschiedlichen Merkmalen, basierend beispielsweise auf Krümmungen oder Momenten. Für eine Übersicht und einen Vergleich sei auf die Übersichts-Arbeiten von Tangelder et al., van Kaick et al. und Tam et al. [Tangelder04, van Kaick11, Tam13] verwiesen. Im Folgenden werden nun merkmalsbasierte Verfahren näher betrachtet, die höherwertige geometrische Informationen als Punkte oder Dreiecke nutzen und damit relevanter für die Registrierung von B-Rep Modellen sind.

Insbesondere bei der Kartierung in der mobilen Robotik werden planare Oberflächenstücke (*planar patches*) verwendet, da diese in Gebäuden häufig auftreten. Eine einfache Korrespondenzfindung ist möglich, wenn angenommen wird, dass große Ebenenstücke stets rechtwinklig zueinander stehen [Kohlhepp06, Harati07b]. Ohne diese Einschränkung kommt das Verfahren von Pathak et al. aus [Pathak10]. Ebenenstücke werden hier durch eine Normale und einen Abstand definiert, für die zusätzlich Unsicherheiten in Form einer Kovarianzmatrix berechnet werden. Daraus können Ebenenkorrespondenzen sowie die Transformation bestimmt werden, die gleichzeitig die Unsicherheit minimiert und die Konsistenz der Ebenen bezüglich der Pose maximiert. Um falsche Ebenenkorrespondenzen früh auszusortieren, werden geometrische Kriterien vorgestellt, wie ein ähnlicher Flächeninhalt oder Überlappung. Die Verwendung der Unsicherheiten in allen Berechnungen macht das Verfahren aufwändig. Im Verfahren von Xiao et al. [Xiao13] werden deshalb nur die Ebenenparameter zum Finden der Korrespondenzen und zur Posenberechnung verwendet. Zudem werden aber noch zusätzliche Bedingungen zur Einschränkung des Suchraumes berücksichtigt, indem eine Pose auf Validität im Bezug auf

die Kinematik des verwendeten mobilen Roboters getestet wird.

Die Berechnung der Pose kann auch auf Basis einer Kombination von Ebenen und Punkten erfolgen. Wie oben werden dazu Ebenen in den Punktwolken ermittelt. Zusätzlich werden in Farbbildern SURF-Keypoints gefunden, deren Projektion in den 3D-Raum zu Punktkorrespondenzen führt. Beide Arten von Korrespondenzen werden gemeinsam benutzt, um die beste Pose zu finden [Taguchi13, Ataer-Cansizoglu13]. Als Eingabe sind kolorierte, organisierte Punktwolken erforderlich.

Eine weitere Kombination ist die Verwendung von Ebenen und Strecken [Cupec15]. Dazu werden aus einem Tiefenbild einerseits Ebenen extrahiert sowie Tiefensprünge erkannt, die in 3D-Streckenstücke umgewandelt werden. Der Grund für die Verwendung von Tiefensprüngen ist, dass diese oft eine Kante eines Objekts darstellen und somit ortsfest sind.

Während Punkte, Strecken und Ebenen natürliche geometrische Primitive sind, können auch künstlich Merkmale zusammengesetzt werden, um die Registrierung zu vereinfachen. Winkelbach nutzt dazu eine Kombination aus zwei Punkten mit deren Normale (genannt *Dipol*), die zufällig mithilfe des RANSAC-Prinzips aus den Ausgangsmodellen gezogen wird [Winkelbach06]. Geometrische Eigenschaften innerhalb des Dipols, wie Winkel und Abstände werden dazu genutzt, um Korrespondenzen zu finden. Im Gegensatz zu den obigen Verfahren reicht ein Paar an korrespondierenden Dipolen aus, um die Pose zu bestimmen.

Einige Verfahren beschäftigen sich explizit mit der Registrierung von Punktwolken auf CAD-Modelle. Dabei betrachtet man die CAD-Modelle jedoch entweder als Dreiecksnetz [Bosché10] oder wandelt diese um. Buchholz et al. [Buchholz15] überführen das Modell durch Abtastung in eine Punktwolke, um anschließend eine Registrierung anhand des Dipol-Ansatzes von Winkelbach ausführen zu können. Korkalo et al. [Korkalo16] verfolgen den Kinect-Fusion-Ansatz [Newcombe11], also der Berechnung der Pose mittels ICP durch Vergleich der Punktwolke mit einem synthetisch generierten Tiefenbild. Der Unterschied ist, dass das generierte Bild nun direkt aus dem CAD-Modell erzeugt wird.

6.2 Vorgehensweise

In diesem Abschnitt soll nun die Vorgehensweise zur Registrierung zweier partieller B-Rep Modelle dargestellt werden. Punktbasierte Verfahren wie ICP oder GMA eignen sich dazu eher schlecht, da sie insbesondere bei partiellen Überlappungen stark von einer guten Anfangsschätzung ausgehen. Wird ein B-Rep zum Zwecke der Registrierung wieder mittels Abtastung wie bei Buchholz in eine Punktwolke überführt, so gehen bereits erkannte geometrische Informationen wie Kanten und Ecken wieder verloren. Ein merkmalsbasierter Ansatz ist daher besser geeignet, vor allem da Merkmale, wie zuvor dargestellt, auf Basis geometrischer Elemente definiert werden können. Besonders Flächen, Kanten und Ecken sind bereits im B-Rep direkt ablesbar. Kein Verfahren aus der Literatur nutzt zudem die Topologie eines B-Reps aus. Es wird daher ein neues Verfahren entwickelt, das sich an die merkmalsbasierten Ansätze anlehnt, zudem aber auch die vorhandenen geometrischen B-Rep-Elemente (Ecken, Kanten, Flächen) sowie Adjazenzinformationen berücksichtigt. Da es sich um partielle Modelle handelt, muss dabei natürlich beachtet werden, dass die Topologiegraphen völlig unterschiedlich sein können (Abbildung 6.1).

Je spezieller die Merkmale sind, desto kleiner ist deren Anzahl. Insbesondere bei partiellen Modellen aus einer einzigen Ansicht ist die Anzahl an Ecken und Kanten deutlich geringer als die Anzahl der ursprünglichen Punkte. Einerseits beschleunigt dies die Verarbeitung, andererseits muss darauf geachtet werden, dass die Bestimmung der Pose eindeutig bleibt. Deshalb werden im Folgenden Merkmale verwendet, die einen hohen Informationsgehalt aufweisen,

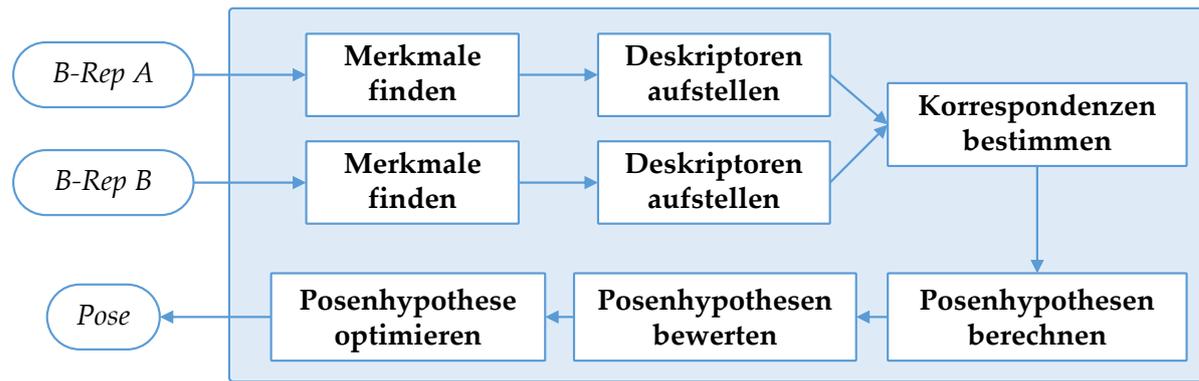


Abbildung 6.2: Vorgehensweise zur Registrierung zweier B-Reps. Der letzte Schritt der Posenoptimierung ist optional.

um zu erreichen, dass, wie bei Winkelbach [Winkelbach06], ein einziges Paar aus korrespondierenden Merkmalen ausreicht, um die Pose eindeutig zu bestimmen.

Abbildung 6.2 zeigt die Vorgehensweise zur Registrierung partieller B-Reps. Die Schritte sind dabei dieselben wie in den meisten merkmalsbasierten Ansätzen (z.B. [Cupec15]), der Unterschied liegt in der Definition der Merkmale selbst. Folgende Schritte werden zur Registrierung durchgeführt:

1. In beiden Eingabe-B-Reps werden alle **Merkmale** ermittelt (Abschnitt 6.2.1). Als Grundlage dienen dabei die Flächen der B-Reps.
2. Für jedes Merkmal wird ein rotations- und translationsinvarianter **Deskriptor** erstellt (Abschnitt 6.2.2). Dabei wird die Topologie des B-Reps ausgenutzt.
3. Mithilfe der Deskriptoren werden **Korrespondenzen** zwischen Merkmalen des ersten und zweiten B-Reps gefunden (Abschnitt 6.2.3).
4. Für jede Korrespondenz kann eine **Posenhypothese** berechnet werden (Abschnitt 6.2.4), da die Merkmale genügend Information besitzen, sodass ein einziges Korrespondenzpaar die Pose ausreichend definiert.
5. Eine Gütemaß **bewertet** die Hypothesen und findet die beste (Abschnitt 6.2.5).
6. Diese Posenhypothese kann optional noch **optimiert** werden (Abschnitt 6.2.6).

6.2.1 Merkmale

Ein B-Rep Modell setzt sich aus Knoten, Halbkanten und Flächen zusammen, die untereinander verzeigert sind. Für eine Verwendung in einem Merkmal ist es sinnvoll, nur solche Elemente eines B-Reps zu verwenden, die auch physikalisch vorhanden sind. Dies sind Kanten, Ecken und Flächen, also alle physikalischen Elemente (siehe Definition 3.13). Diese können stabil auch in anderen Ansichten der Szene wiedererkannt werden. Für andere Elemente gilt dies nicht, da beispielsweise die Lage virtueller Kanten vom Blickwinkel oder von Verdeckungen beeinflusst werden.

Zudem sollen aus einem Merkmal genügend Informationen ableitbar sein, um aus einen Paar von Merkmalen eindeutig eine Transformation bestimmen zu können. Die vier in Tabelle 6.1 dargestellten Kombinationen aus physikalischen Elementen fixieren alle sechs Freiheitsgrade der Transformation, wenn sie linear unabhängig sind.

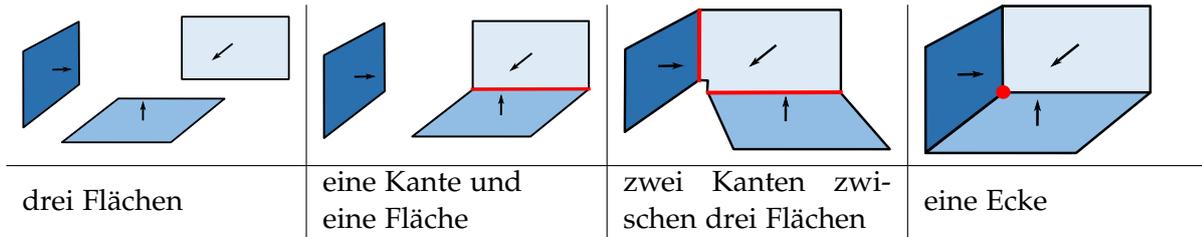


Tabelle 6.1: Kombinationen aus physikalischen Elementen (Fläche, Kante, Ecke) eines B-Reps, die zur Fixierung der sechs Freiheitsgrade genügen (unter der Annahme linearer Unabhängigkeit)

Alle Fälle mit weniger Elementen sind unterbestimmt, alle Fälle mit mehr Elementen sind überbestimmt beziehungsweise enthalten einen der obigen Fälle. Es sei angemerkt, dass alle Fälle implizit drei Flächen enthalten, da eine Kante genau zwei adjazente Flächen, eine Ecke mindestens drei adjazente Flächen besitzt.

Da die Modelle unvollständig sind und sich die Topologie unterscheiden kann, muss eine Ecke oder Kante in B-Rep A nicht zwangsweise in B-Rep B vorhanden sein, auch wenn die korrespondierenden Flächen existieren (beispielsweise die Flächen f_1, f_2, f_3 und g_1, g_2, g_3 in Abbildung 6.1). Um trotzdem eine Korrespondenz herstellen zu können, wird zur Definition eines Merkmals der allgemeinste Fall verwendet, nämlich der der drei unabhängigen Flächen. Existieren zwischen diesen Flächen zudem Kanten oder Ecken, wird diese Information dennoch nicht verworfen, sondern bei der Erstellung der Deskriptoren berücksichtigt (siehe Abschnitt 6.2.2).

Formal ergibt sich folgende Definition eines Merkmals:

Definition 6.1

Ein **Merkmal** M eines B-Reps $A = (\mathcal{V}, \mathcal{H}, \mathcal{B}, \mathcal{F})$ besteht aus drei unterschiedlichen Flächen:
 $M = (f_i, f_j, f_k), \quad f_i, f_j, f_k \in \mathcal{F}, \quad i \neq j \neq k .$

Die Reihenfolge der drei Flächen im Tupel sei dabei so festgelegt, dass die Normalen der Flächen $\vec{n}_i, \vec{n}_j, \vec{n}_k$ ein Rechtssystem bilden. Damit sind drei Schreibweisen für das Merkmal M möglich, die folgendermaßen notiert werden:

$$M^{[0]} = (f_i, f_j, f_k), \quad M^{[1]} = (f_k, f_i, f_j), \quad M^{[2]} = (f_j, f_k, f_i).$$

Ein B-Rep kann damit maximal $\binom{|\mathcal{F}|}{3} = \frac{|\mathcal{F}|!}{6 \cdot (|\mathcal{F}|-3)!}$ Merkmale besitzen. Damit später die Berechnung einer Pose eindeutig ist, werden nur Merkmale zugelassen, deren drei Flächen nicht linear abhängig sind. Dies ist beispielsweise möglich, indem getestet wird, ob die Determinante

$$\det(\vec{n}_i \quad \vec{n}_j \quad \vec{n}_k)$$

gleich null (oder nahe null bei Berücksichtigung von Rundungsfehlern) ist.

6.2.2 Deskriptoren

Ein Deskriptor ist eine translations- und rotationsinvariante Repräsentation eines Merkmals, die zum Finden von Korrespondenzen genutzt werden kann. Eine einfache Möglichkeit hierfür sind Winkel, da sie im Allgemeinen einfach zu berechnen sind und die gewünschten Invarianten besitzen. Im speziellen Fall von B-Reps werden nun Innenwinkel zwischen Flä-

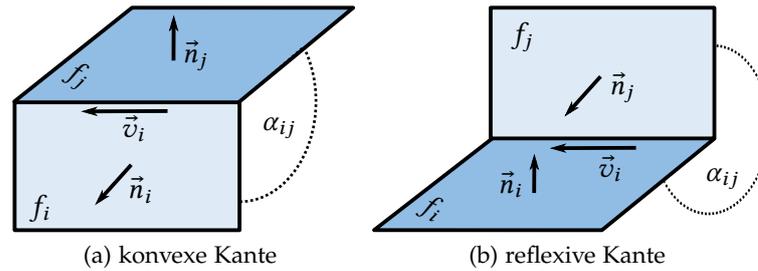


Abbildung 6.3: Verschiedene Arten von Kanten

chen betrachtet. Als Innenwinkel wird derjenige Winkel zwischen zwei Flächen bezeichnet, der im Inneren des Modells liegt. Dies entspricht der Seite von Flächen, von der die Normalen der Flächen wegzeigen.

Da ein Merkmal aus drei Flächen besteht, können drei Innenwinkel zwischen diesen berechnet werden. Sind zwei der Flächen durch eine Kante benachbart, wird diese Information ausgenutzt. Es ist allerdings nicht zwingend notwendig, dass eine Kante existiert. So wird ermöglicht, dass sich beispielsweise eine Korrespondenz bilden kann zwischen drei adjazenten Flächen und drei separierten Flächen, was bei Modellen mit Verdeckungen notwendig ist.

Zur Berechnung des Innenwinkels zwischen zwei Flächen wird zuerst der Fall von zwei adjazenten Flächen und anschließend von zwei nicht-adjazenten Flächen betrachtet:

Innenwinkel zwischen adjazenten Flächen

Zwei adjazente, nicht parallele Flächen f_i und f_j besitzen eine gemeinsame Kante k . Diese ist entweder konvex oder reflexiv. Der Innenwinkel α_{ij} ist der Winkel zwischen f_i und f_j , der im Inneren des Objekts liegt. Dies ist eindeutig, da die Normale einer Fläche stets nach außen zeigt.

Die Berechnung des Innenwinkels α_{ij} ist über folgende Formel möglich:

$$\alpha_{ij} = \begin{cases} \pi - \arccos(\vec{n}_i \circ \vec{n}_j) & \text{falls } (\vec{v}_i \times \vec{n}_i) \circ \vec{n}_j > 0 \quad (\text{konvex}) \\ \pi + \arccos(\vec{n}_i \circ \vec{n}_j) & \text{sonst} \quad (\text{reflexiv}). \end{cases} \quad (6.1)$$

Dabei ist \vec{n}_i, \vec{n}_j die Normale der Fläche f_i, f_j und \vec{v}_i die normalisierte Richtung der Halbkante der Fläche f_i , an der sich beide Flächen berühren (Abbildung 6.3).

Innenwinkel zwischen nicht-adjazenten Flächen

Für zwei nicht-adjazente Flächen ist die Bestimmung des Innenwinkels grundsätzlich nicht eindeutig, da stets eine konvexe oder eine reflexive Verbindung denkbar ist (Abbildung 6.4). Dennoch lässt sich der korrekte Innenwinkel schätzen, sodass beispielsweise in Fällen, in denen durch Verdeckung eine Kante zwischen zwei Flächen nicht sichtbar ist, derselbe Innenwinkel resultiert. Deshalb wird folgendes Vorgehen angewandt:

Gegeben sind zwei Flächen f_i und f_j mit Ebenengleichungen $\vec{\pi}_i = (\vec{n}_i^T \quad d_i)^T$ und $\vec{\pi}_j = (\vec{n}_j^T \quad d_j)^T$.

1. Es wird eine Ebene berechnet, die orthogonal auf beiden Flächen steht: $\vec{\pi}_{ortho} = \begin{pmatrix} \vec{n}_i \times \vec{n}_j \\ 0 \end{pmatrix}$.

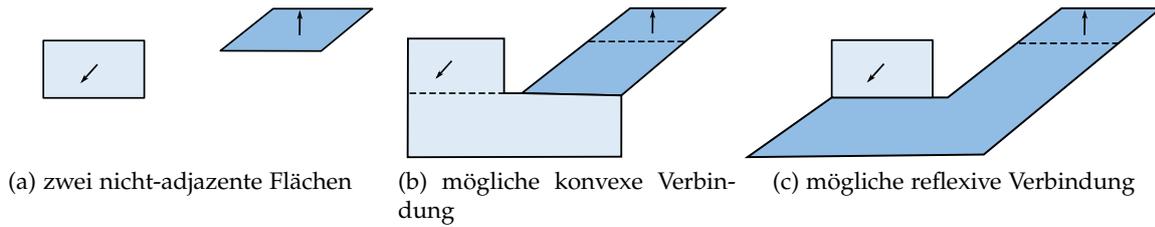


Abbildung 6.4: Für zwei nicht adjazente Flächen (a) ist der Innenwinkel nicht eindeutig bestimmbar, da stets eine konvexe (b) und eine reflexive Verbindung (c) konstruiert werden kann.

2. Die Ebenen $\vec{\pi}_i$ und $\vec{\pi}_j$ werden auf $\vec{\pi}_{ortho}$ projiziert, es entstehen zwei Geraden g_i und g_j , die sich in einem Schnittpunkt \vec{P} schneiden.
3. Die Flächen f_i und f_j werden auf $\vec{\pi}_{ortho}$ projiziert. Dies entspricht einer 2D-Projektion von Polygonen und es entstehen zwei Strecken s_i und s_j , die auf den Geraden g_i beziehungsweise g_j liegen.
4. Anhand der Lage der beiden Strecken bezüglich des Schnittpunktes \vec{P} wird der Innenwinkel mittels der Fallunterscheidung in Tabelle 6.2 bestimmt.

Konvex oder reflexiv		Separiert			
A		Typ: konvex $\alpha_{ij} = \pi - \arccos(\vec{n}_i \cdot \vec{n}_j)$	C		Typ: separiert $\alpha_{ij} = \arccos(\vec{n}_i \cdot \vec{n}_j)$
B		Typ: reflexiv $\alpha_{ij} = \pi + \arccos(\vec{n}_i \cdot \vec{n}_j)$	D		Typ: separiert $\alpha_{ij} = \arccos(\vec{n}_i \cdot \vec{n}_j)$
Unbekannt					
E		Typ: unbekannt			

Tabelle 6.2: Fallunterscheidung zur Berechnung des Winkels α_{ij} anhand der Lage der beiden Strecken s_i und s_j . Die Pfeile symbolisieren die projizierten Normalen der Flächen f_i und f_j .

Liegen beide Streckenstücke vom Schnittpunkt \vec{P} aus gesehen eindeutig auf einer Seite, so können vier Fälle unterschieden werden (Tabelle 6.2 A-D). Fall A und B können als konvex beziehungsweise reflexiv identifiziert werden, da die beiden Flächen eine konvexe beziehungsweise reflexive Kante bilden würden, wenn sie in P verbunden wären. In den Fällen C und D könnten beide Flächen nie eine Kante in P bilden, da die Normalen inkonsistent orientiert wären. Ein Innenwinkel ist somit nicht identifizierbar. Dennoch ist es sinnvoll, einen Winkel zu bestimmen, um auch Korrespondenzen zwischen Merkmalen mit dieser Art von Flächen zu bestimmen. Daher wird als Winkel α_{ij} der Winkel zwischen den Normalen \vec{n}_i und \vec{n}_j verwendet.

Liegen beide Streckenstücke nicht auf einer Seite (Fall E), so kann keine Aussage getroffen werden. In diesem Fall wird das Merkmal verworfen.

Definition Deskriptor

Mithilfe der Innenwinkel kann für ein Merkmal nun ein Deskriptor aufgestellt werden, der rotations- und translationsinvariant ist und daher für das Finden von Korrespondenzen genutzt werden kann:

Definition 6.2

Ein **Deskriptor** $\vec{D}(M)$ eines Merkmals $M = (f_i, f_j, f_k)$ ist ein Vektor aus drei Winkeln

$$\vec{D}(M) = \begin{pmatrix} \alpha_{jk} \\ \alpha_{ik} \\ \alpha_{ij} \end{pmatrix},$$

wobei α_{xy} die oben beschriebenen Innenwinkel zwischen den Flächen f_x und f_y darstellen.

Analog zu Merkmalen sind drei Schreibweisen möglich:

$$\vec{D}(M)^{[0]} = (\alpha_{jk} \ \alpha_{ik} \ \alpha_{ij})^T, \quad \vec{D}(M)^{[1]} = (\alpha_{ij} \ \alpha_{jk} \ \alpha_{ik})^T, \quad \vec{D}(M)^{[2]} = (\alpha_{ik} \ \alpha_{ij} \ \alpha_{jk})^T.$$

6.2.3 Korrespondenzen

In diesem Schritt werden Korrespondenzen von Merkmalen gefunden. Zwei Merkmale M_A und M_B korrespondieren, wenn sich die Winkel im Deskriptor nur wenig unterscheiden. Da es für die Reihenfolge der Flächen im Deskriptor drei Möglichkeiten gibt, existieren auch drei Möglichkeiten, die Winkel zu vergleichen, was in folgender Differenzfunktion ausgedrückt wird:

Definition 6.3

Die **Differenz** Δ_m zweier Merkmale M_A und M_B mit Deskriptoren \vec{D}_A und \vec{D}_B ist definiert als die Maximumsnorm der Differenz der Deskriptor-Vektoren bezüglich einer Reihenfolge/Schreibweise:

$$\Delta_m(M_A, M_B) = \left| \vec{D}(M_A)^{[0]} - \vec{D}(M_B)^{[m]} \right|_{\infty} \quad \text{mit} \quad m = 0, 1, 2.$$

Damit können korrespondierende Merkmale ermittelt werden, indem das Minimum der drei möglichen Differenzen gefunden wird.

Definition 6.4

Zwei Merkmale M_A und M_B **korrespondieren**, notiert als $M_A \sim M_B$, wenn das Minimum der drei Differenzen Δ_0, Δ_1 und Δ_2 unter eine Schwelle δ_c liegt:

$$M_A \sim M_B \iff \min \{ \Delta_0(M_A, M_B), \Delta_1(M_A, M_B), \Delta_2(M_A, M_B) \} < \delta_c.$$

Wenn zwei Merkmale korrespondieren, existieren auch Korrespondenzen zwischen den drei Flächen in den Merkmalen. In Sonderfällen gibt es für zwei korrespondierende Merkmale allerdings mehr als eine Möglichkeit, die Flächen zuzuordnen, beispielsweise bei der Ecke eines Würfels, bei dem alle drei Innenwinkel zwischen den drei Flächen 90° betragen. Diesem Umstand wird in den folgenden zwei Definitionen Rechnung getragen:

Definition 6.5

Zwischen zwei korrespondierenden Merkmalen $M_A = (f_i, f_j, f_k)$ und $M_B = (g_x, g_y, g_z)$, $M_A \sim M_B$, gibt es drei mögliche **Flächenzuordnungen** – notiert als $(M_A \overset{m}{\sim} M_B)$, $m = 0, 1, 2$ – mit folgender Bedeutung: Flächen an der gleichen Position im Tupel korrespondieren bei Benutzung der Schreibweisen $M_A^{[0]}$ und $M_B^{[m]}$.

Beispiel:

$$(M_A \overset{1}{\sim} M_B) \Rightarrow M_A^{[0]} = (f_i, f_j, f_k), M_B^{[1]} = (g_z, g_x, g_y) \Rightarrow f_i \sim g_z, f_j \sim g_x, f_k \sim g_y$$

Definition 6.6

Eine Flächenzuordnung ist **gültig**, wenn gilt:

$$(M_A \overset{m}{\sim} M_B) \text{ gültig} \iff \Delta_m(M_A, M_B) < \delta_c.$$

Aufgrund von Definition 6.4 existiert für zwei korrespondierende Merkmale damit stets mindestens eine Flächenzuordnung.

Beispiel 1

$$\delta_c = 5^\circ$$

$$M_A = (f_1, f_2, f_3), \vec{D}(M_A) = (90^\circ, 10^\circ, 20^\circ)^T$$

$$M_B = (g_5, g_1, g_7), \vec{D}(M_B) = (11^\circ, 19^\circ, 89^\circ)^T$$

$$\Rightarrow \Delta_0 = 79^\circ, \Delta_1 = 1^\circ, \Delta_2 = 79^\circ$$

\Rightarrow eine gültige Flächenzuordnung $(M_A \overset{1}{\sim} M_B)$

mit den drei Flächenkorrespondenzen $f_1 \sim g_7, f_2 \sim g_5$ und $f_3 \sim g_1$

Beispiel 2

$$\delta_c = 5^\circ$$

$$M_A = (f_1, f_2, f_3), \vec{D}(M_A) = (90^\circ, 90^\circ, 90^\circ)^T$$

$$M_B = (g_5, g_1, g_7), \vec{D}(M_B) = (90^\circ, 89^\circ, 91^\circ)^T$$

$$\Rightarrow \Delta_0 = 1^\circ, \Delta_1 = 1^\circ, \Delta_2 = 1^\circ$$

\Rightarrow drei gültige Flächenzuordnungen $(M_A \overset{0}{\sim} M_B), (M_A \overset{1}{\sim} M_B), (M_A \overset{2}{\sim} M_B)$

6.2.4 Posenhypothesen

Für jede gültige Flächenzuordnung von korrespondierenden Merkmalen wird nun eine Posenhypothese bestimmt. Da bei der Bildung der Merkmale auf lineare Unabhängigkeit der beteiligten Flächen geachtet wurde, kann nun aus den drei korrespondierenden Flächen beziehungsweise deren Ebenengleichungen eindeutig eine Starrkörpertransformation berechnet werden.

Für ein Merkmal $M = (f_i, f_j, f_k)$ und dessen drei Ebenengleichungen $\vec{\pi}_i = (\vec{n}_i^T \quad d_i)^T = \text{ebene}(f_i)$ ($\vec{\pi}_j, \vec{\pi}_k$ analog) wird ein lokales Koordinatensystem $K \in \mathbb{R}^{4 \times 4}$

$$K(M) = \begin{pmatrix} \vec{n}_i & \vec{n}_j & \vec{n}_k & \vec{P} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.2}$$

definiert, wobei \vec{P} der Schnittpunkt der drei Ebenen $\vec{\pi}_i, \vec{\pi}_j$ und $\vec{\pi}_k$ ist. Analog zu Merkmalen und Deskriptoren sind drei Schreibweisen möglich:

$$K(M)^{[0]} = K(M), \quad (6.3)$$

$$K(M)^{[1]} = \begin{pmatrix} \vec{n}_k & \vec{n}_i & \vec{n}_j & \vec{P} \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad (6.4)$$

$$K(M)^{[2]} = \begin{pmatrix} \vec{n}_j & \vec{n}_k & \vec{n}_i & \vec{P} \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (6.5)$$

Da die Normalenvektoren nicht orthogonal sein müssen, stellt $K(M)$ eine affine Transformation dar.

Für eine gültige Flächenzuordnung ($M_A \stackrel{m}{\sim} M_B$) kann nun eine Transformation von A nach B berechnet werden:

$${}^B T_A = K^{-1}(M_B)^{[m]} \cdot K(M_A)^{[0]}. \quad (6.6)$$

Besitzen die drei Ebenen in M_A allerdings nicht exakt das selbe Verhältnis der Winkel zwischen den Normalen wie in M_B , so ist die resultierende Transformation affin und keine Starrkörpertransformation:

$${}^B T_A = \begin{pmatrix} A & \vec{t} \\ \vec{0} & 1 \end{pmatrix}. \quad (6.7)$$

Zur Orthogonalisierung kann die Matrix A mittels Singulärwertzerlegung (*singular value decomposition, SVD*) in drei Teile zerlegt werden:

$$A = U \Sigma V^*. \quad (6.8)$$

Dabei ist Σ die Diagonalmatrix aus den Singulärwerten, U und V^* sind unitäre Matrizen, also orthogonal. Damit erhält man den reinen Rotationsanteil von A durch

$$R = UV^* \quad (6.9)$$

und damit die endgültige Starrkörpertransformation

$${}^B T_A = \begin{pmatrix} R & \vec{t} \\ \vec{0} & 1 \end{pmatrix}. \quad (6.10)$$

6.2.5 Posenbewertung

Die Posenbewertung hat zur Aufgabe, aus der Menge der Posenhypothesen die beste zu ermitteln. Grundsätzlich sind hier zwei Strategien denkbar: Eine Bewertung anhand der Häufigkeit oder anhand der Güte.

Bei der ersten Variante geht man davon aus, dass korrekte Posen häufiger auftreten als falsche. Das Ergebnis wird durch Ermittlung von Häufungen ermittelt. Bei der zweiten Variante wird ein Gütemaß verwendet, das zwei Hypothesen vergleicht. So kann eine Rangfolge und damit die beste Hypothese ermittelt werden.

Eine Bewertung über die Häufigkeit hat den Vorteil, dass kein Gütemaß notwendig ist, da nur die Posen selbst betrachtet werden. Das Finden von Häufungen im Fall von 6D-Posen wird auch als Posenclustering (*pose clustering*) bezeichnet. Dabei wird der Parameterraum in Teile aufgeteilt und ein Histogramm erstellt (auch Hough-Transformation genannt). Aufgrund der hohen Dimensionalität ist es jedoch sehr aufwändig und ineffizient [Winkelbach06, S.44]. Zudem besteht eine Pose aus Winkel- und Längenparametern, von denen nicht klar ist, wie diese gemeinsam in einem Abstandsmaß kombiniert werden können. Insgesamt existiert dabei ein hohes Risiko für falsch positive Ergebnisse [Grimson90].

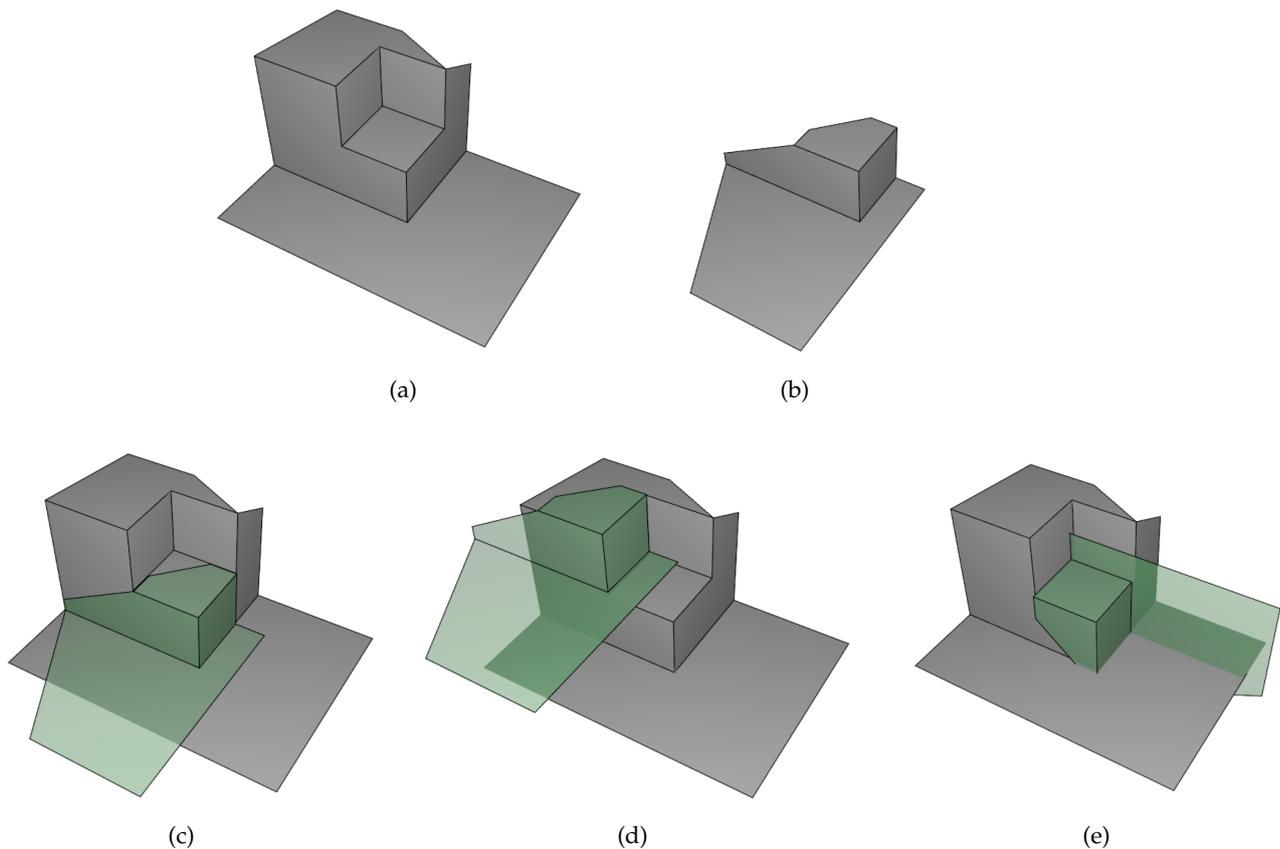


Abbildung 6.5: Zwei zu registrierende partielle B-Reps (a) und (b) mit einer korrekten (c) und zwei falschen Posenhypothesen (d-e)

In dieser Arbeit wird daher die zweite Variante, eine Bewertung anhand eines Gütemaßes bevorzugt. Zur Erstellung eines passenden Gütemaßes werden folgende Beobachtungen ausgenutzt (Abbildung 6.5):

- Aufgrund des Verfahrens zur Erzeugung der Hypothesen existieren drei korrespondierende Flächen.
- Eine korrekte Pose besitzt über alle Flächen gesehen meist eine höhere Überlappung als falsche Pose (vergleiche Abbildung 6.5c und 6.5d).
- Bei korrekten Posen schneidet eine Fläche keine andere Fläche weit im Inneren („Bodenflächen“ in Abbildung 6.5e). Dies wird als (geometrischer) Konflikt bezeichnet.

Im Fall eines SLAM-Systems, bei dem stetig partielle B-Reps registriert und fusioniert werden, ist zudem noch zu beachten, dass falsch positive Registrierungsergebnisse bedeutend schlimmer sind als falsch negative. Kann ein einzelnes partielles Modell nicht registriert und somit nicht fusioniert werden, kann dieser Frame ausgelassen werden. Dies ist weniger problematisch als eine erfolgreiche aber falsche Registrierung, die zu einer Fusion an einer falschen Pose und somit zu einem komplett fehlerhaften Gesamtmodell führt.

Basierend auf diesen Beobachtungen werden deshalb zwei Kriterien genutzt, die in den folgenden Unterabschnitten näher erläutert werden. Die Bewertung durch Überlappung begünstigt Hypothesen mit Überlappung und findet so die beste Hypothese. Die Validierung durch Konflikttest testet, ob die beste Hypothese einen geometrischen Konflikt besitzt, um falsch positive Ergebnisse zu reduzieren.

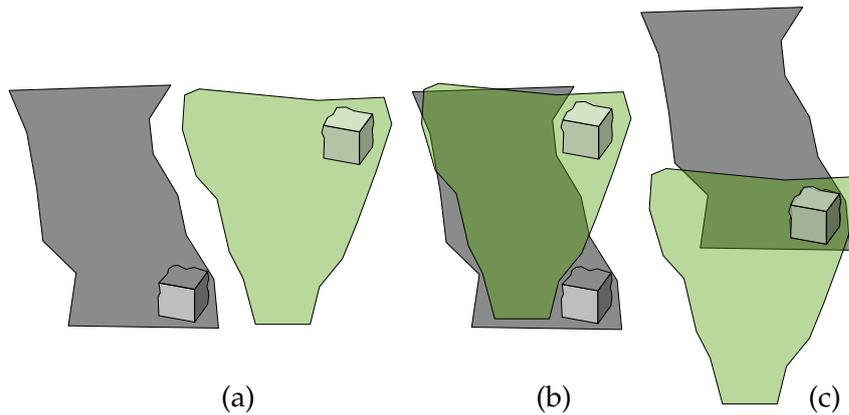


Abbildung 6.6: Zwei partielle B-Reps eines Würfels auf einer Ebene (a) mit zwei möglichen Posenhypothesen (b+c). Das absolute Gütemaß (Gleichung 6.11) favorisiert die große Überlappung der Grundebene (b). Das relative Gütemaß (Gleichung 6.12) bevorzugt die Überlappung vieler kleiner Flächen (c).

Bewertung durch Überlappung

Eine gute Pose zeichnet sich dadurch aus, dass möglichst viele Flächen möglichst gut übereinander liegen. Für eine Posenhypothese mit Pose ${}^B T_A$ kann die Überlappung folgendermaßen bestimmt werden:

1. Transformiere B-Rep $A = (\mathcal{V}_A, \mathcal{H}_A, \mathcal{B}_A, \mathcal{F}_A)$ mithilfe von ${}^B T_A$ ins Koordinatensystem von B-Rep $B = (\mathcal{V}_B, \mathcal{H}_B, \mathcal{B}_B, \mathcal{F}_B)$.
2. Bestimme alle korrespondierenden Flächen $\mathfrak{F} = \{(f_i \sim g_j)\}$, $f_i \in \mathcal{F}_A$, $g_j \in \mathcal{F}_B$ unter Verwendung des Vorgehens wie bei der Fusion (siehe Kapitel 5.2.2). Dabei wird der Schnitt $f_i \cap g_j$ auf einer mittleren Ebene bestimmt.
3. Als Überlappung kann nun der Flächeninhalt des Schnittes $m_{ij} = \text{area}(f_i \cap g_j)$ herangezogen werden.
4. Summiere die Überlappungen aller Flächenkorrespondenzen: $m = \sum m_{ij}$.

Somit können die Posenhypothesen anhand m sortiert werden, die beste Hypothese ist diejenige mit dem größtem Wert von m . Nach obiger Definition besitzt m die Einheit Quadratmeter (m^2).

Die Verwendung des absoluten Maßes

$$m_{ij} = \text{area}(f_i \cap g_j) \quad (6.11)$$

für die Überlappung besitzt folgende Eigenschaft: Hypothesen, die wenige aber große Flächen übereinander legen, werden bevorzugt vor Hypothesen, die viele aber kleine Flächen aufeinander abbilden (Abbildung 6.6). Es wird daher ein zweites Maß vorgeschlagen, das eine hohe Anzahl von passenden Flächen favorisiert:

$$m_{ij} = \frac{\text{area}(f_i \cap g_j)}{\text{area}(f_i \cup g_j)} = \frac{\text{area}(f_i \cap g_j)}{\text{area}(f_i) + \text{area}(g_j) - \text{area}(f_i \cap g_j)}. \quad (6.12)$$

Die einzelnen Werte für m_{ij} stellen die prozentuale Überlappung dar und besitzen daher keine Einheit. Durch die Summierung in Schritt 4 ergeben sich für m aber Werte größer als 1 und eine Überlappung vieler Flächen wird dadurch bevorzugt.

Die exakte Berechnung dieses Maßes ist sehr aufwändig, insbesondere stellt Schritt 2 einen Schnitt zweier nicht-konvexer Polygone mit Löchern dar, der für alle Flächenkorrespondenzen in allen Hypothesen ausgeführt werden muss. Mit h als Anzahl der Hypothesen, f als Anzahl der Flächen im B-Rep, und k als Anzahl der Halbkanten in einer Fläche, bedeutet dies im schlimmsten Fall einen Laufzeit-Aufwand von $O(h \cdot f^2 \cdot k \log k)$, da in der Theorie jede mit jeder Fläche korrespondieren könnte (Aufwand f^2) und dafür der Schnitt bzw. die Vereinigung (Aufwand $k \log k$) berechnet wird.

Im Folgenden wird daher eine approximative Berechnung vorgeschlagen, die 2D-Bounding-Boxes verwendet. Dazu wird als Vorverarbeitungsschritt für jede Fläche eines B-Rep eine 2D-Bounding-Box berechnet, die in der Ebene der Fläche liegt und minimalen Flächeninhalt besitzt. Diese Approximation der Fläche wird als *face-aligned bounding box* (FABB) bezeichnet. Um diese orientierte Bounding-Box zu berechnen, kann der *rotating-calipers*-Algorithmus [Toussaint83] eingesetzt werden. Die Berechnung aller FABBs für ein B-Rep kann dann in $O(f \cdot k \log k)$ durchgeführt werden, mit f gleich der Anzahl der Flächen und k der Anzahl der Halbkanten in einer Fläche. Ein Test auf Schnitt reduziert sich dann auf einen 2D-Bounding-Box Schnittest, der mithilfe separierender Achsen [Schneider03] in konstanter Zeit durchgeführt werden kann. Die Schnittberechnung zweier orientierter Bounding-Boxen ist ebenfalls in konstanter Zeit möglich, da das Schnittpolygon maximal acht Kanten besitzen kann. Insgesamt reduziert sich somit der Berechnungsaufwand auf $O(f \cdot k \log k + h \cdot f^2)$, mit $O(f \cdot k \log k)$ für die einmalige Vorberechnung der Bounding-Boxes pro Fläche und $O(h \cdot f^2)$ für den Schnittest über alle Kombinationen aus Flächen und jeder Hypothese.

Validierung durch Konflikttest

Unter Annahme eines exakten Modells ohne Rauschen fallen alle Flächen bei einer korrekten Hypothese aufeinander. Schneiden sich zwei Flächen (wie in Abbildung 6.5e), so ist dies ein Indiz für eine falsche Hypothese. Dies wird als geometrischer Konflikt bezeichnet (Abbildung 6.7a). In diesem Abschnitt soll getestet werden, ob eine Hypothese einen solchen Konflikt aufweist.

Da mit Messunsicherheiten behaftete Modelle betrachtet werden, treten auch bei korrekten Posen Schnitte zwischen nicht-korrespondierenden Flächen auf (Abbildung 6.7b+c). Diese treten dann allerdings stets am Rand der Fläche auf und nicht wie in Abbildung 6.7a im Inneren einer Fläche. Diese Beobachtung soll genutzt werden, um einen Konflikt zu erkennen.

Die beiden B-Reps werden im folgenden mit B-Rep $A = (\mathcal{V}_A, \mathcal{H}_A, \mathcal{B}_A, \mathcal{F}_A)$ und B-Rep $B = (\mathcal{V}_B, \mathcal{H}_B, \mathcal{B}_B, \mathcal{F}_B)$ bezeichnet. Die Transformation anhand der Posenhypothese sei dabei schon berücksichtigt, das heißt beide liegen im selben Koordinatensystem. Der Schnitt zweier nicht-korrespondierender Flächen $f_a \in \mathcal{F}_A$ und $f_b \in \mathcal{F}_B$ wird mit

$$f_a \cap f_b = \text{inhalt}(f_a) \cap \text{inhalt}(f_b) \quad (6.13)$$

bezeichnet. Da Flächen aus nicht konvexen Polygonen mit Löchern bestehen, ist $f_a \cap f_b$ eine Strecke, mehrere Strecken auf einer Geraden oder die leere Menge.

Mithilfe der Abstandsfunktion

$$d_{Rand}(\vec{X}, f) = \min_{\vec{P} \in \text{rand}(f)} \|\vec{X} - \vec{P}\|_2, \quad (6.14)$$

die den Abstand eines Punktes \vec{X} vom Rand einer Fläche f angibt, kann nun ein Maß bestimmt

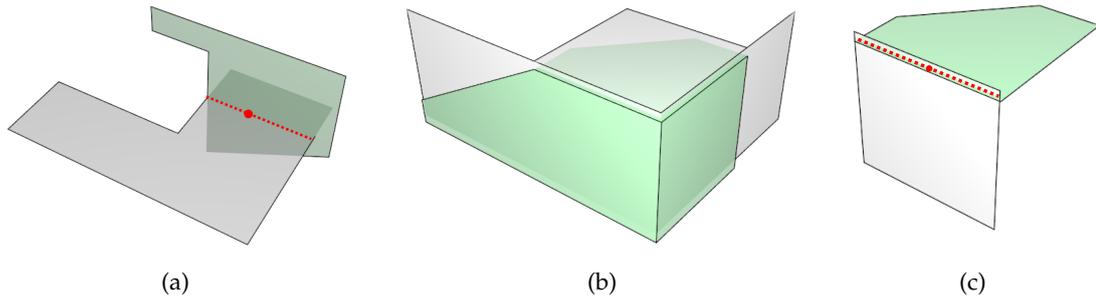


Abbildung 6.7: (a) Bei der falschen Posenhypothese aus Abbildung 6.5e schneiden sich zwei der Flächen so, dass ein Punkt (roter Kreis) auf dem Schnitt (rote Linie) existiert, der weit im Inneren der beiden Flächen liegt. (b) Bei einer korrekten Hypothese wie in Abbildung 6.5c können durch das Rauschen auch Schnitte entstehen. (c) Alle Punkte auf dem Schnitt liegen dann allerdings nah am Rand der Flächen.

werden, wie weit im Inneren sich zwei Flächen f_a und f_b schneiden:

$$d_{\text{Konflikt}}(f_a, f_b) = \max_{\vec{X} \in f_a \cap f_b} 0.5 \cdot (d_{\text{Rand}}(\vec{X}, f_a) + d_{\text{Rand}}(\vec{X}, f_b)). \quad (6.15)$$

In Abbildung 6.7 ist der Punkt \vec{X} , der jeweils das Maximum bildet, rot markiert. Insgesamt wird also ein geometrischer Konflikt zwischen den beiden B-Reps erkannt, wenn

$$\max_{f_a \in \mathcal{F}_A, f_b \in \mathcal{F}_B} d_{\text{Konflikt}}(f_a, f_b) > \delta_{\text{Konflikt}} \quad (6.16)$$

gilt. Bei exakter Geometrie würde $\delta_{\text{Konflikt}} = 0$ genügen, aufgrund von Messabweichungen muss hier jedoch eine höhere Schwelle gewählt werden. Hierbei eignet sich die Strukturgröße $\delta_{\text{Konflikt}} = \delta_s$.

Der Test auf Konflikt ist verhältnismäßig aufwändig zu berechnen, da erstens ein Schnitt von nicht-konvexen Polygonen mit Löchern benötigt wird und dieser für alle Paare von Flächen ausgeführt werden muss. Ähnlich zur Bewertung durch Überlappung kann auch hier das Ergebnis mittels der vorgestellten *face aligned bounding boxes* (FABBs) beschleunigt werden. Dazu wird als Schnitt $f_a \cap f_b$ der Schnitt zwischen den beiden FABBs verwendet, der entweder leer oder eine einzige Strecke ist. Dies kann in konstanter Zeit berechnet werden, da die Bounding Boxes rechteckig sind. Diese Strecke wird nun äquidistant gesampelt und für alle Samples \vec{X} das Maß

$$0.5 \cdot (d_{\text{Rand}}(\vec{X}, f_a) + d_{\text{Rand}}(\vec{X}, f_b)) \quad (6.17)$$

berechnet und das Maximum ermittelt. Dabei wird für den Abstand zum Rand d_{Rand} wieder die echte Begrenzung der Fläche und nicht die Bounding Box verwendet. Für die Auswertung von d_{Rand} wird dann eine vorzeichenbehaftete Abstandsfunktion, das heißt eine Funktion, die positive Werte für Punkte innerhalb der Fläche und negative Werte für Punkte außerhalb der Fläche, verwendet. Dadurch kann getestet werden, ob das Sample \vec{X} auch wirklich innerhalb des echten Schnittes $f_a \cap f_b$ liegt und nicht nur durch die Verwendung der Bounding Boxes erzeugt wurde. Nur Samples mit $d_{\text{Rand}} > 0$ für beide Flächen werden berücksichtigt. Die Berechnung einer vorzeichenbehafteten Abstandsfunktion ist dabei nicht wesentlich aufwändiger, da für jede Halbkante bekannt ist, auf welcher Seite das Innere der Fläche liegt. Die Berechnung des echten Schnittes zweier nicht konvexer Polygone mit Löchern wird dadurch allerdings eingespart.

6.2.6 Posenoptimierung

Bei der Berechnung der Pose in Abschnitt 6.2.4 fließen die drei Paare aus Flächen in die Berechnung mit ein, die in beiden verwendeten Merkmalen vorhanden sind. Dennoch können weitere Korrespondenzen existieren, die auch für die Posenberechnung genutzt werden können. Diese können nun, da eine initiale Pose berechnet ist, leicht gefunden werden (vgl. Kapitel 5.2.2). Da dann mehr Korrespondenzen als nötig vorhanden sind, ist das System überbestimmt und ein Optimierungsverfahren nötig.

Grundsätzlich existieren zwei Arten von Verfahren:

- **Iterative Verfahren** nähern sich der Lösung in mehreren Iterationen an und enden, wenn ein Abbruchkriterium erreicht ist.
- **Direkte Verfahren** berechnen die Lösung direkt, da diese in einer geschlossenen Form angegeben werden kann (*closed form solutions*).

Im Allgemeinen sind direkte Verfahren zu bevorzugen, da iterative Verfahren erstens meist eine längere Laufzeit besitzen, zweitens die Lösung abhängig vom Abbruchkriterium ist und drittens abhängig vom Startwert in einem lokalen Minimum stecken bleiben können. Jedoch existiert nicht für jedes Optimierungsproblem eine geschlossene Lösung, sodass direkte Verfahren nicht allgemein anwendbar sind. Ist das Fehlermaß linear oder quadratisch, wie im Falle von Abständen zwischen geometrischen Größen, wie Punkte, Geraden oder Ebenen, so existiert eine geschlossene Form, die je nach verwendeten Korrespondenzen unterschiedlich ist. Folgende Übersicht listet Optimierungsverfahren, die eine direkte Lösung unter Minimierung eines Abstandsmaßes liefern, in Abhängigkeit der verwendeten Korrespondenzen auf:

- **Punkt-Punkt-Korrespondenzen:** Arun et al. [Arun87] ermitteln aus n Punkt-Punkt-Korrespondenzen die optimale Starrkörpertransformation unter Minimierung des quadratischen Abstandes, indem mittels Singulärwertzerlegung zuerst die Rotation bestimmt und anschließend die Translation ermittelt wird.
- **Ebene-Ebene-Korrespondenzen:** Bestehen Korrespondenzen aus Ebenen, so kann mit der Methode von Khoshelham [Khoshelham16] eine Starrkörpertransformation direkt berechnet werden, die die Differenzen zwischen den Parametern der Ebenengleichungen minimiert. Auch hier werden Rotation und Translation getrennt voneinander berechnet und man erhält eine affine Transformation. Die Rotationsmatrix wird am Ende noch mittels Singulärwertzerlegung orthogonalisiert, um eine Starrkörpertransformation zu erhalten.
- **Punkt-Ebene-Korrespondenzen:** Khoshelham [Khoshelham16] beschreibt ebenfalls eine direkte Methode, um eine Pose aus Punkt-Ebene-Korrespondenzen zu ermitteln, die den Abstand der Punkte zur Ebene minimiert. Nach Berechnung einer vorläufigen Rotation und Translation wird die Rotationsmatrix orthogonalisiert und der Translationsvektor erneut mit der neuen Rotationsmatrix berechnet. Es ist zu beachten, dass die Korrespondenzen unidirektional sind, das heißt von Seite A werden nur Punkte verwendet, von Seite B nur Ebenen. Eine Mischung ist nicht möglich.
- **Gerade-Gerade-Korrespondenzen:** Die Arbeit von Bartoli et al. [Bartoli03] stellt fest, dass diese Art von Korrespondenzen aufgrund ihrer Repräsentation und dem Fehlen eines universellen Fehlermaßes schwieriger zu behandeln sind. Um dennoch zu einer Lösung zu kommen, wird vorgeschlagen, auf Seite A jede Gerade in eine Menge von Punkten umzuwandeln und auf Seite B jede Gerade in eine Menge von Ebenen, wodurch viele Punkt-Ebene-Korrespondenzen entstehen, anhand derer die Transformation bestimmt wird. Je nachdem welche Seite als A und B gewählt wird und wie groß die

Korrespondenzart	Anzahl benötigter Korrespondenzen	Methode	symmetrisch
Punkt-Punkt	3	[Arun87]	ja
Ebene-Ebene	3	[Khoshelham16]	ja
Punkt-Ebene	12	[Khoshelham16]	nein
Gerade-Gerade	2	[Bartoli03]	nein
kombiniert P-P und E-E	3	[Taguchi13]	ja

Tabelle 6.3: Übersicht über direkte (*closed form*) Optimierungsverfahren zur Bestimmung einer Starrkörpertransformation aus unterschiedlichen Arten von Korrespondenzen. Die Anzahl an benötigten Korrespondenzen zur eindeutigen Bestimmung der Transformation geht von einer allgemeinen Lage aus (keine Kollinearitäten usw.). Bei nicht symmetrischen Verfahren hängt die Lösung von einer Entscheidung ab, welche Korrespondenzen auf welcher Seite verwendet werden.

Anzahl der gewählten Punkte und Ebenen ist, können unterschiedliche Lösungen entstehen.

- **kombinierte Punkt-Punkt- und Ebene-Ebene-Korrespondenzen:** Die bisherigen Verfahren betrachten nur eine Art von Korrespondenzen. Taguchi et al. [Taguchi13] schlagen ein Verfahren zur gleichzeitigen Optimierung von Punkt-Punkt- und Ebene-Ebene-Korrespondenzen vor. Die Rotation wird dabei mit einer Singulärwertzerlegung einer Korrelationsmatrix aus Punkten und Normalen bestimmt, die Translation durch Lösen eines Gleichungssystems.

Tabelle 6.3 fasst die beschriebenen Verfahren nochmals zusammen. Insgesamt lässt sich sagen, dass sich zwei Arten eher nicht für einen Einsatz zur Posenoptimierung eignen: Die Methode von Bartoli et al. überführt Gerade-Gerade-Korrespondenzen nur in andere Arten, sodass diese auch direkt genutzt werden können. Khoshelham zeigt in seiner Arbeit, dass Punkt-Ebene-Korrespondenzen weniger robust sind bei schlecht konditionierten Problemen [Khoshelham16]. Zudem wird das Problem dadurch unsymmetrisch. Die drei symmetrischen Korrespondenzarten hingegen sind geeignet.

Angewendet auf partielle B-Reps bedeutet dies, dass aus jeder Flächenkorrespondenz eine Ebenenkorrespondenz gebildet werden kann. Aus korrespondierenden Ecken können Punkt-korrespondenzen erzeugt werden. Durch Verdeckungen und unterschiedliche Blickwinkel kann es jedoch passieren, dass keine oder nur sehr wenige Eckenkorrespondenzen auftreten. Demnach bietet es sich an, das für die jeweilige Situation passende Verfahren zu wählen, also das kombinierte Verfahren von Taguchi et al., falls beide Arten von Korrespondenzen vorliegen oder das Verfahren von Khoshelham et al. für Ebene-Ebene-Korrespondenzen, falls keine Eckenpaare vorliegen.

6.3 Evaluation

Um die Registrierung zu evaluieren, werden Ground-Truth-Daten für die Pose des Sensors benötigt. In den vorhergehenden Kapiteln wurden zur Evaluation synthetische Daten erzeugt, da pro Pose partielle Ground-Truth Modelle benötigt wurden, die nicht oder nur sehr schwer aus realen Objekten erzeugbar sind. Bei der Registrierung entfällt dieses Hindernis, da als Ground-Truth nur die Pose und kein partielles Modell nötig ist. Daher wird die Evaluation der Registrierung auf realen Daten durchgeführt. Diese werden erzeugt, indem eine Tiefenkamera an einem Roboterarm montiert wird. Durch Registrierung der Kamera auf den Arm (*Eye-in-Hand Registrierung*), ist die Position der Kamera im Weltkoordinatensystem bekannt und kann als Ground-Truth verwendet werden. Abbildung 6.8 zeigt den zur Evaluation genutzten Aufbau.

6.3.1 Datensätze

Zur Evaluation werden vier Datensätze erzeugt. Für jeden Datensatz wurde im Arbeitsraum des Roboters eine Szene aus Objekten aufgebaut (Abbildung 6.9): Ein Modell des Empire State Building (ESB), separiert platzierte Bauklötze (BK), zufällig angeordnete Verpackungen (VER) und drei Kanthölzer (HOLZ). Anschließend wurde der Roboter im Gravitationskompensationsmodus manuell geführt, um eine Kamerabewegung eines Anwenders zu simulieren. Dabei wurden pro Szene zwischen 20 und 25 Punktwolken aufgenommen und in jedem Bild ein partielles B-Rep rekonstruiert. Dabei wurde darauf geachtet, dass sich der Roboter zum Zeitpunkt der Bildaufnahme in Ruhe befindet, um Fehler durch eine zeitliche Synchronisation zwischen dem Ermitteln der Roboterpose und Erfassung des Bildes auszuschließen. Eine komplette Umrundung der Objekte war dabei aufgrund des Arbeitsbereiches von Kamera und Roboter nicht möglich.

Um möglichst viele Testfälle zu erzeugen, werden nun Paare von Bildern gebildet, die später aufeinander registriert werden sollen. Es ist klar, dass eine Registrierung nicht für alle Paare möglich ist, beispielsweise wenn auf einen anderen Teil der Szene geblickt wird oder keine gemeinsamen Merkmale vorhanden sind. Deshalb werden drei Varianten an Kombinationen erzeugt:

- **Alle:** Es werden alle möglichen Paare von Bildern gebildet, bei n Posen also $0.5 \cdot n \cdot (n - 1)$ Stück. Dabei sind auch Paare enthalten, bei denen der Blickwinkel so unterschiedlich ist, dass keine Überlappung möglich ist. Somit eignet sich dieser Datensatz für die Evaluation des Konflikttests.



Abbildung 6.8: Zur Evaluation wird eine an einem KUKA LBR 4+ Roboterarm montierte IDS Ensenso N10 Tiefenkamera verwendet, die Testobjekte im Arbeitsraum des Roboters erfasst.

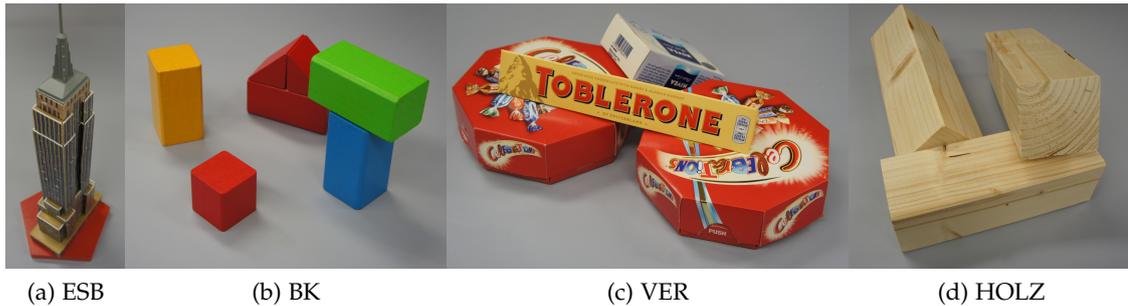


Abbildung 6.9: Die vier zur Evaluation der Registrierung verwendeten Test-Datensätze

Anzahl	ESB	VERP	BK	HOLZ	Summe
Posen	25	22	25	20	92
Kombinationen (alle)	300	231	300	190	1021
davon registrierbar	161	225	298	183	867
Kombinationen (sequentiell)	47	41	47	37	172
davon registrierbar	33	41	47	36	157
Kombinationen (fusioniert)	243	117	248	144	752
davon registrierbar	208	113	248	144	713

Tabelle 6.4: Statistik der Datensätze mit der Anzahl der Posen, aus denen eine Punktwolke aufgenommen wurde und der Anzahl an Kombinationen für die drei beschriebenen Kombinationsmöglichkeiten. Für jede ist zudem die Anzahl an registrierbaren Paaren angegeben, also Paare, die auch gemeinsame Merkmale aufweisen.

- **Sequentiell:** Es werden nur Paare von Bildern gebildet, die während der Bewegung der Kamera höchstens eine Aufnahme dazwischen haben (Beispiel: Während der Führung des Roboters entstehen die Aufnahmen $(1, \dots, i, \dots, n)$. Aufnahme i wird kombiniert mit $i - 2, i - 1, i + 1$ und $i + 2$). Dadurch wird sichergestellt, dass eine Überlappung vorhanden ist.
- **Fusioniert:** Es werden inkrementell die partiellen B-Reps mit der Ground-Truth-Pose fusioniert, sodass bei n Posen ebenfalls n B-Reps entstehen, wobei B-Rep j die Fusion aus B-Rep 1 bis j darstellt. Zur Evaluation werden nun diese n fusionierten B-Reps mit allen Einzel-B-Reps kombiniert, die noch nicht in die Fusion eingegangen sind. Diese Kombinationen sollen den Anwendungsfall einer kontinuierlichen Registrierung und Fusion abbilden.

Eine Registrierung von einem Paar an Bildern ist nur möglich, wenn trotz Überlappung in beiden B-Reps mindestens drei nicht linear abhängige Flächen erkannt werden, deren Deskriptoren zusammenpassen. Ist dies nicht der Fall, kann keine Hypothese gebildet werden und eine Registrierung ist nicht möglich. In Tabelle 6.4 ist eine Statistik der Daten zu sehen mit der Anzahl an registrierbaren Paaren. Auffällig ist, dass bei dem Datensatz ESB (Empire State Building) der Anteil an registrierbaren Paaren deutlich geringer ist als bei den anderen drei Datensätzen. Betrachtet man das Objekt, so zeigt sich der Grund: Das Objekt besteht hauptsächlich aus vertikalen parallelen Flächen. Für eine Registrierung (unabhängig davon, ob diese korrekt ist) sind aber in beiden partiellen Modellen drei linear unabhängige Flächen nötig, deren Deskriptoren zusammenpassen.

6.3.2 Auswertung

Die Güte der Registrierung wird folgendermaßen evaluiert: Für ein Paar aus partiellen B-Reps wird die Pose T_{exp} mithilfe des vorgestellten Registrierungsverfahrens ermittelt und der Unterschied zur Ground-Truth-Pose T_{truth} bestimmt:

$$T_{error} = T_{truth}^{-1} \cdot T_{exp} = \begin{pmatrix} R & \vec{t} \\ 0 & 1 \end{pmatrix}. \quad (6.18)$$

Als Fehler im rotatorischen Anteil δ_{error} wird der Winkel verwendet, der durch Umwandlung der Rotationsmatrix R in eine Achse-Winkel-Darstellung und Weglassen der Achse entsteht. Formal bedeutet dies

$$\delta_{error} = \arccos(0.5 \cdot (R_{00} + R_{11} + R_{22} - 1)). \quad (6.19)$$

Als translatorischer Fehler t_{error} wird der Abstand verwendet, um den sich der Flächenschwerpunkt \vec{P}_c des rekonstruierten B-Reps bei Anwendung von T_{error} verschiebt:

$$t_{error} = \|T_{error} \cdot \vec{P}_c - \vec{P}_c\|_2. \quad (6.20)$$

Der Grund für die Verwendung dieses Maßes anstelle von $\|\vec{t}\|_2$ ist, dass die partiellen B-Reps nicht im Ursprung ihres lokalen Koordinatensystems liegen und so mithilfe von T_{error} die Verschiebung am Ort des B-Reps gemessen wird.

6.3.3 Ergebnisse

Für jedes Paar eines Datensatzes, für das eine Registrierung aufgrund ausreichender korrespondierender Merkmale möglich ist, werden die beiden Fehlerarten berechnet. In der Tabelle 6.5 sind die Ergebnisse dargestellt. Dabei wird hinsichtlich der vorgestellten Posenbewertungsmethoden unterschieden: Bei der Bewertung durch Überlappung zwischen dem absoluten oder relativen Maß, sowie ob die Berechnung exakt oder approximativ durchgeführt wird. Für alle Kombinationen wird zudem überprüft, ob die beste Hypothese einen geometrischen Konflikt aufweist. In der Tabelle wird deshalb unterschieden, ob alle Resultate herangezogen werden, oder nur solche, die keinen geometrischen Konflikt aufweisen.

Folgende Schlussfolgerungen können aus den Ergebnissen gezogen werden:

Der Fehler ist nicht normalverteilt, der Median weicht stark vom Mittelwert ab. Dies ist natürlich, denn die Berechnung der Pose hängt von der Wahl einer korrekten Hypothese ab. Ist diese falsch, so ist die Pose deutlich fehlerhaft. Insgesamt erkennt man daher eine große Anzahl von Datenpaaren mit kleinem Fehler und eine geringe Anzahl mit großem Fehler. Es ist daher sinnvoll, zur Auswertung der Güte der Registrierung die Quantile und nicht den Mittelwert zu betrachten.

Das relative Gütemaß liefert leicht bessere Ergebnisse als das absolute Gütemaß. Da die Objekte auf einer Tischoberfläche stehen, von der in vielen Aufnahmen ein großer Teil zu sehen ist, ist der beschriebene Effekt vom Verhältnis von großen zu kleinen Flächen hier zu beobachten.

Wie erwartet, liefert das exakte Gütemaß bessere Ergebnisse als dessen Approximation, jedoch nur in geringem Maße. Allerdings ist die Berechnungszeit (Tabelle 6.6) deutlich höher, sodass bei zeitkritischen Anwendungen das approximative Maß gewählt werden sollte, da hier der Vorteil der stark verringerten Laufzeiten den Nachteil von leicht schlechteren Ergebnissen deutlich überwiegt.

Alle Datensätze	ohne Konflikttest						mit Konflikttest					
	δ_{error} [rad]			t_{error} [cm]			δ_{error} [rad]			t_{error} [cm]		
Gütemaß	μ, σ	Q_2	Q_3	μ, σ	Q_2	Q_3	μ, σ	Q_2	Q_3	μ, σ	Q_2	Q_3
alle Kombinationen (867)												
exakt absolut	0.46, 0.86	0.012	0.480	5.8, 10.5	0.4	8.7	0.04, 0.26	0.007	0.012	0.7, 3.0	0.2	0.4
approximativ absolut	0.45, 0.86	0.013	0.540	5.8, 10.6	0.4	7.9	0.02, 0.16	0.007	0.012	0.5, 2.4	0.2	0.4
exakt relativ	0.45, 0.86	0.012	0.177	5.2, 10.3	0.3	6.0	0.05, 0.29	0.007	0.012	0.7, 3.3	0.2	0.4
approximativ relativ	0.45, 0.87	0.011	0.143	5.2, 10.4	0.4	5.9	0.05, 0.31	0.007	0.012	0.7, 3.2	0.2	0.4
sequentielle Kombinationen (157)												
exakt absolut	0.07, 0.37	0.006	0.011	1.3, 5.9	0.1	0.3	0.01, 0.01	0.005	0.008	0.2, 0.6	0.1	0.2
approximativ absolut	0.08, 0.37	0.006	0.011	1.3, 5.8	0.1	0.3	0.01, 0.01	0.005	0.010	0.2, 0.6	0.1	0.2
exakt relativ	0.07, 0.37	0.006	0.011	1.2, 5.8	0.1	0.3	0.01, 0.01	0.005	0.009	0.2, 0.6	0.1	0.2
approximativ relativ	0.07, 0.37	0.006	0.011	1.2, 5.8	0.1	0.3	0.01, 0.01	0.005	0.010	0.2, 0.6	0.1	0.2
fusionierte Kombinationen (713)												
exakt absolut	0.31, 0.67	0.008	0.021	4.6, 11.2	0.3	1.5	0.01, 0.00	0.006	0.009	0.3, 0.6	0.2	0.3
approximativ absolut	0.31, 0.67	0.009	0.027	4.9, 11.3	0.3	3.9	0.01, 0.09	0.006	0.009	0.3, 1.4	0.2	0.4
exakt relativ	0.25, 0.65	0.008	0.014	3.5, 10.8	0.2	0.5	0.01, 0.07	0.006	0.009	0.3, 0.6	0.2	0.3
approximativ relativ	0.24, 0.62	0.008	0.013	3.6, 10.9	0.2	0.5	0.01, 0.04	0.006	0.009	0.3, 0.7	0.2	0.3

Tabelle 6.5: Mittelwert μ , Standardabweichung σ , Median Q_2 und oberes Quartil Q_3 der Fehlmaße δ_{error} und t_{error} über alle Datensätze, unterschieden zwischen den drei Kombinationsmethoden sowie ob ein Konflikttest ausgeführt wird, das heißt ob Ergebnisse mit geometrischen Konflikten aussortiert werden

Methode	Mittelwert	Median
exakt absolut	504	18
exakt relativ	478	18
approximativ absolut	4	2
approximativ relativ	4	2

Tabelle 6.6: Zeit in Millisekunden zur Berechnung des Gütemaßes (über alle Kombinationen). Vergleiche dazu auch Abbildung 6.10.

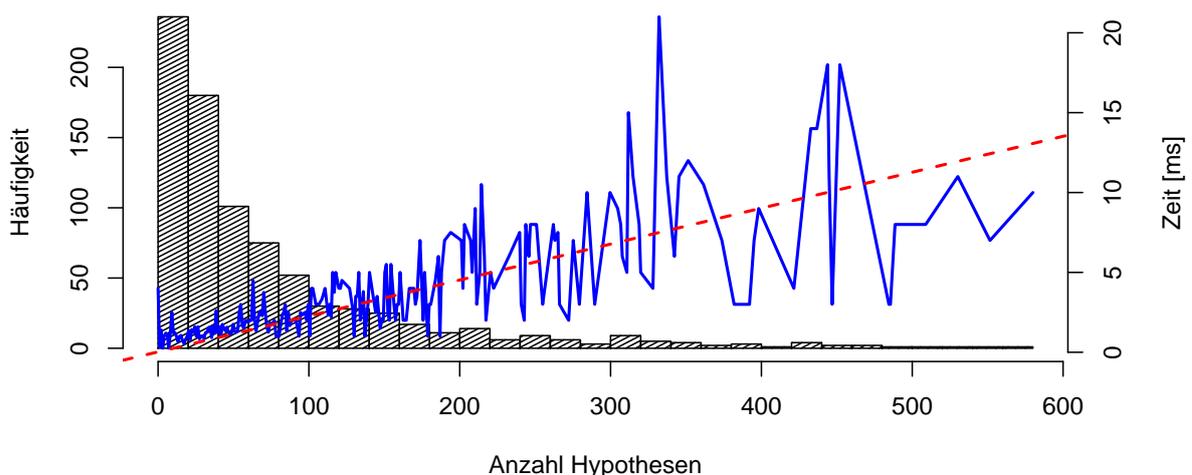


Abbildung 6.10: Häufigkeit einer bestimmten Anzahl an Hypothesen bei der Registrierung eines Paares von partiellen B-Reps (Balken) sowie die Berechnungszeit des approximativen relativen Gütemaßes (durchgezogene Linie) mit eingezeichnete Ausgleichsgerade (gestrichelte Linie)

Insgesamt ist die zur Berechnung des Gütemaßes benötigte Zeit abhängig von der Anzahl der Hypothesen. In Abbildung 6.10 ist dies dargestellt. Eine hohe Anzahl an Hypothesen tritt in den Testdatensätzen allerdings viel weniger oft auf als eine geringe Anzahl, was den Unterschied zwischen Mittelwert und Median in Tabelle 6.6 erklärt.

In Abbildung 6.11 sind die Ergebnisse aus Tabelle 6.5 für das approximative relative Gütemaß nochmals grafisch dargestellt. Durch die Quantil-Verteilung ist die Güte der Registrierung besser ersichtlich: Bei allen Kombinationen aus Paaren kann in ca. 70% der Fälle eine sehr geringe Abweichung von der Ground-Truth erreicht werden, in den restlichen Fällen ist der Fehler sehr groß. In diesen Fällen wurde also eine falsche Hypothese ausgewählt oder es existiert gar keine korrekte Hypothese. Dies ist bei allen Kombinationen aus Paaren auch nicht verwunderlich, da hier einige Paare an partiellen B-Reps vorhanden sind, die nur sehr wenig bis keine Überlappung besitzen. Bei den sequentiellen Kombinationen ist der Fehler in 90% der Fälle kleiner als 0.56 cm und 0.0189 rad (1.08°). Bei den fusionierten Kombinationen wird ein ähnliche Güte in 80% der Fälle erreicht.

Gut erkennbar ist der Nutzen des Konflikttests. Werden nur Paare ohne geometrischen Konflikt zugelassen, so wird auch bei allen Kombinationen in 90% der Fälle ein Fehler kleiner als 0.52 cm und 0.021 rad (1.2 °) erreicht. Der vorgeschlagene Konflikttest ist somit sehr gut geeignet, um falsche Registrierungen auszusortieren. Bei den sequentiellen Kombinationen wird derselbe Fehler sogar in 95% der Fälle eingehalten, bei den fusionierten in 94% .

Um einen Eindruck der statistischen Trefferquote (*recall*) und Genauigkeit (*precision*) des Konflikttests zu erhalten, wird eine Wahrheitsmatrix bestimmt, indem angenommen wird, dass eine Pose korrekt ist, wenn $\delta_{error} < 0.05$ rad und $t_{error} < 1$ cm (Tabelle 6.7). Die Trefferquote des Tests liegt dabei bei zwischen 83% und 99%, die Genauigkeit ist deutlich niedriger. Bei einer Rekonstruktion eines Stroms an Daten ist diese hohe Trefferquote allerdings wesentlich wichtiger als einige falsch positive Tests, die „nur“ dazu führen, dass einzelne Frames ausgelassen werden.

	berechnete Pose		
	falsch	richtig	
Konflikt	261	45	84%
kein Konflikt	19	542	
	92%		

(a) Alle Kombinationen

	berechnete Pose		
	falsch	richtig	
Konflikt	10	9	53%
kein Konflikt	2	136	
	83%		

(b) Sequentielle Kombinationen

	berechnete Pose		
	falsch	richtig	
Konflikt	124	81	60%
kein Konflikt	1	507	
	99%		

(c) Fusionierte Kombinationen

Tabelle 6.7: Statistische Auswertung des Konflikttests. Unterhalb der Tabelle ist die Trefferquote (*recall*) angegeben, rechts daneben die Genauigkeit (*precision*). Die Trefferquote ist der Anteil an falschen Posen, die vom Test erkannt wurden. Die Genauigkeit ist der Anteil an Daten mit Konflikt, die auch tatsächlich eine falsche Pose besitzen. Eine Pose wird als richtig klassifiziert, wenn $\delta_{error} < 0.05$ rad und $t_{error} < 1$ cm.

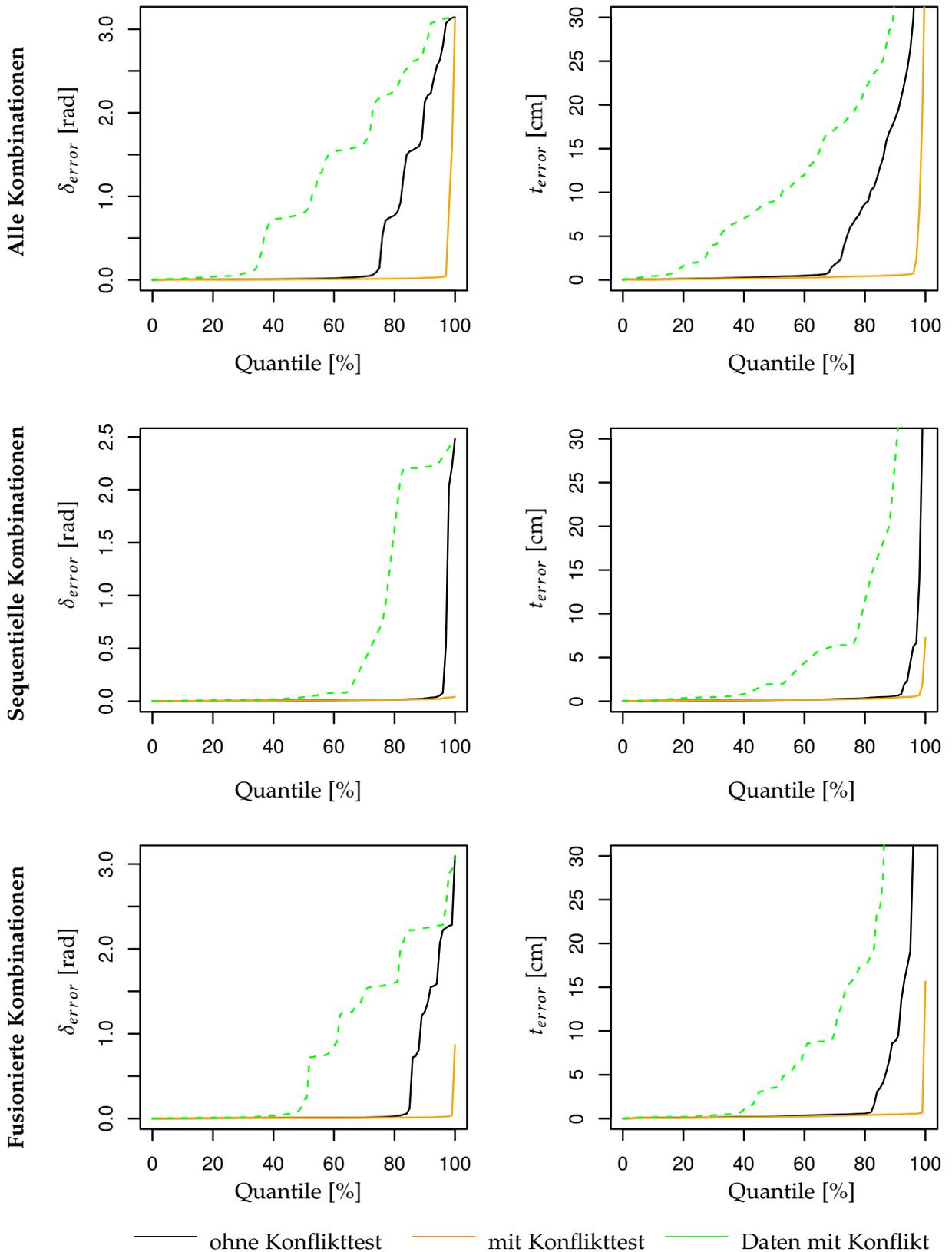


Abbildung 6.11: Quantil-Verteilung des Fehlers δ_{error} (links) und t_{error} (rechts) für die drei Kombinationen unter Verwendung des approximativen relativen Gütemaßes. Die schwarze Kurve zeigt alle Ergebnisse ohne Ausführung des Konflikttests, die orange Kurve zeigt die Verteilung mit Konflikttest, also nur die Daten, bei denen der Test keinen Konflikt festgestellt hat. Zum Vergleich sind alle Daten, die einen Konflikt aufweisen, zudem als grüne Kurve dargestellt.



Abbildung 6.12: Drehsymmetrisches Testobjekt *BigBen*

Es stellt sich die Frage, warum der Test nie 100% korrekte Ergebnisse liefert, beziehungsweise welche Eigenschaften diese falsch registrierten und vom Konflikttest nicht erkannten Paare besitzen. Der Grund sind partielle Symmetrien, das heißt ein partielles B-Rep passt an mehreren Stellen gut und ohne Konflikt. Um diesen Umstand noch weiter zu verdeutlichen, wird ein neuer Datensatz mit einem Objekt erzeugt, das drehsymmetrisch ist (Abbildung 6.12). Dabei werden 20 Aufnahmen erstellt, sodass insgesamt 190 Kombinationen entstehen, wovon 136 gemeinsame Merkmale aufweisen und damit registrierbar sind. Betrachtet man nur die sequentiellen Kombinationen, so sind es 37 Paare, davon 27 registrierbar. Bei den fusionierten Kombinationen gibt es 121 Paare, davon 110 registrierbare.

Die Ergebnisse der Registrierung sind in den Quantil-Diagrammen in Abbildung 6.13 dargestellt. Betrachtet man alle Kombinationen, so erkennt man deutlich, dass sich die Rotationsymmetrie des Objektes in einem hohen Anteil an falschen Registrierungen widerspiegelt, und zwar gehäuft bei einem rotatorischen Fehler von 90° (1.57 rad). Diesmal vermag auch der Konflikttest diese falschen Ergebnisse nicht zu vermeiden. Bei den sequentiellen Kombinationen tritt dieser Effekt nicht auf, da nur in der Aufnahmetrajektorie benachbarte Paare betrachtet werden. Für den Anwendungsfall von einer inkrementellen Fusion von partiellen B-Reps sind die fusionierten Kombinationen interessant. Durch die Symmetrie kann nicht entschieden werden, welche Seite des Testobjektes gerade erfasst wurde. Dies hat zur Folge, dass das Modell nicht korrekt inkrementell registriert und fusioniert werden kann.

Dieser Umstand ist allerdings auch nicht verwunderlich, da die Registrierung ausschließlich die Geometrie der partiellen Modelle betrachtet. Auch für einen Menschen ist es ohne Zusatzwissen nicht entscheidbar, welche Seite des Objektes gerade betrachtet wird.

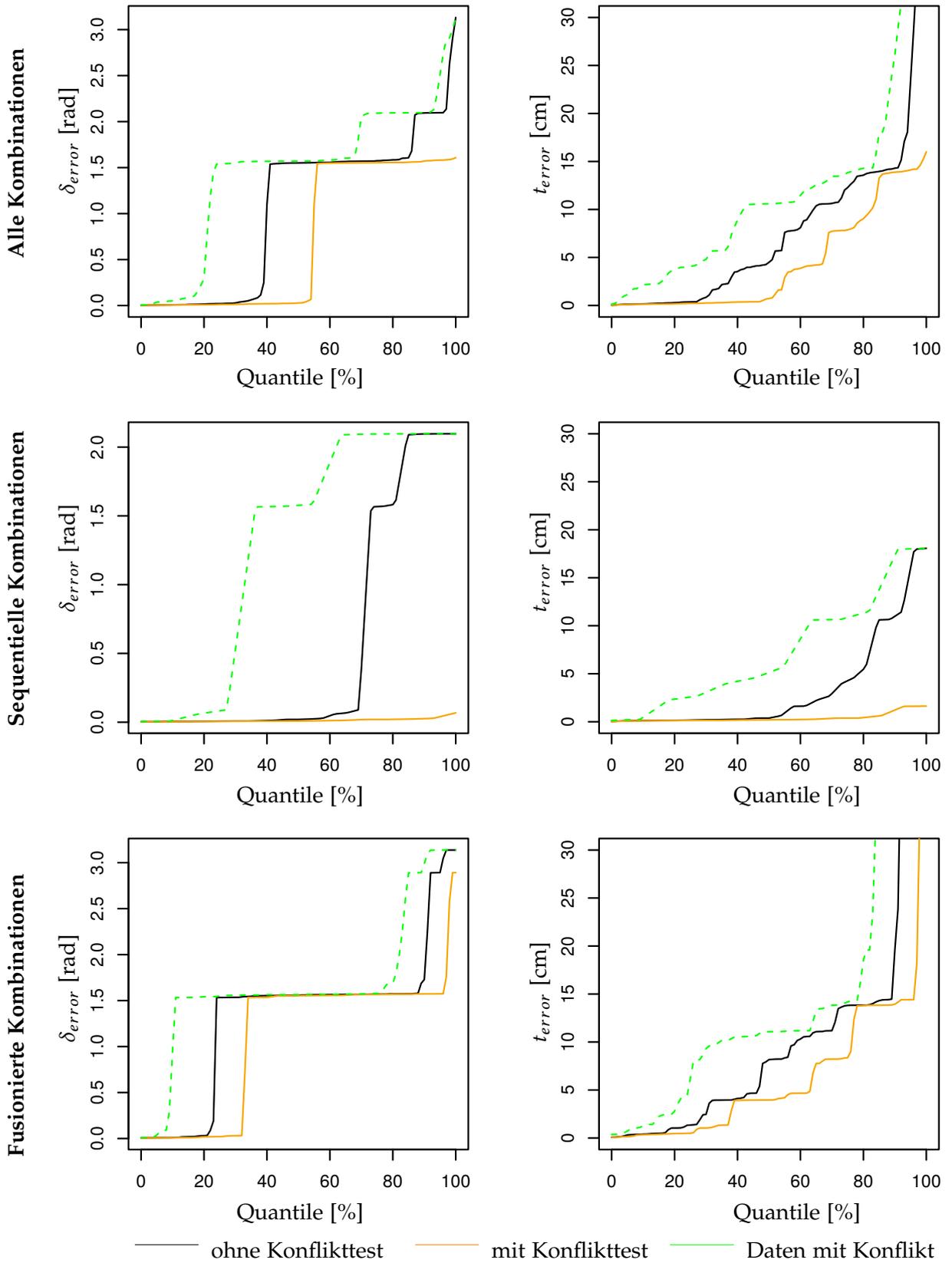


Abbildung 6.13: Quantil-Verteilung des Fehlers δ_{error} (links) und t_{error} (rechts) für den *BigBen*-Datensatz unter Verwendung des approximativen relativen Gütemaßes

6.4 Zusammenfassung

In diesem Kapitel wurde ein Verfahren zur Registrierung partieller B-Reps vorgestellt. Die Kombination aus Rekonstruktion, Registrierung und Fusion stellt somit ein komplettes SLAM-System dar. Dies ermöglicht beispielsweise die Verwendung von handgehaltenen Kameras. Die Registrierung an sich kann auch als reines Lokalisationssystem verwendet werden.

Zur Lösung des Registrierungsproblems wurde ein neuer merkmalsbasierter Ansatz vorgestellt. Da im Gegensatz zu Verfahren aus der Literatur hochwertige partielle B-Reps vorliegen, wurde darauf geachtet, dass die geometrischen Informationen auch ausgenutzt werden. Die Hauptherausforderung stellen dabei Verdeckungen dar, die zu unterschiedlichen Topologien der Modelle führen. Dies wurde gelöst, indem ein Merkmal aus der Kombination von drei unabhängigen Flächen gebildet wurde. Die Existenz von Kanten beziehungsweise Ecken zwischen den Flächen ist dabei nicht zwingen notwendig, wird aber ausgenutzt, falls vorhanden. Die Verwendung dieser Merkmale hat zudem den Vorteil, dass eine einzige Korrespondenz ausreicht, um eine Posenhypothese zu erzeugen. Zur Auswahl der besten Posenhypothese wurden zwei Kriterien, basieren auf Überlappung und geometrischen Konflikt vorgestellt. Da bei der Berechnung der Pose nur drei Paare an Flächen einfließen, sorgt die Posenoptimierung am Ende für ein globales Maximum über alle Flächenkorrespondenzen, die mithilfe der initialen Pose ermittelt werden können.

Zur Evaluation wurden reale Daten unter Verwendung eines Robotersystems zur Generierung von Ground-Truth-Werten genutzt. Dabei wurden unterschiedliche Varianten des Überlappungsmaßes verglichen und die Trefferquote des Konfliktmaßes untersucht. Zudem wurden die Grenzen des Ansatzes getestet. Folgende Schlussfolgerungen können daraus gezogen werden:

Die approximative Variante des Gütemaßes ist deutlich schneller als die exakte und eignet sich für eine online-Ausführung der Registrierung. Im Vergleich des relativen zum absoluten Gütemaß zeigte sich eine leichte Tendenz zum relativen Gütemaß.

Damit eine Registrierung überhaupt möglich ist, müssen in beiden partiellen B-Reps mindestens drei linear unabhängige korrespondierende Flächen vorhanden sein. Diese Einschränkung resultiert aus der Tatsache, dass in dieser Arbeit nur die Geometrie betrachtet wird. Durch zusätzliche Verwendung von Farbmerkmalen ist eine Relaxation dieser Einschränkung denkbar.

Der Test auf geometrischen Konflikt verbessert den Anteil der korrekten Posen deutlich. Dadurch werden falsche Hypothesen verworfen. Anhand der höheren Trefferquote (*recall*) des Tests im Vergleich zur statistischen Genauigkeit (*precision*) erkennt man, dass nur sehr wenige falsch positive Resultate auftreten, falsche negative jedoch häufiger. Gerade für ein SLAM-System mit handgehaltenen Kameras, bei denen ein Strom an Punktwolken verarbeitet wird, ist dies ein sinnvolles Verhalten: Ein fälschlich verworfener Frame ist weniger schwerwiegend als eine nicht erkannte falsche Pose, die nach der Fusion zu einem falschen globalen Modell führen kann. Nicht erkennen lassen sich falsche Posen, die aufgrund von Symmetrien der Szene entstehen, da hier die geometrischen Merkmale identisch sind.

Insgesamt lässt sich damit Fragestellung F3 beantworten: Eine automatisierte Posenberechnung und damit die Realisierung eines SLAM-Systems ist mit dem vorgestellten Verfahren möglich. Die Einschränkungen sind dabei die Notwendigkeit von drei unabhängigen Flächen und Symmetrien in der Szene. Da in dieser Arbeit nur Geometrie und keine Farbe betrachtet wird, kann dies auch nicht behoben werden, denn auch ein Mensch könnte ohne Zusatzwissen nicht entscheiden, welcher Teil eines völlig symmetrischen Objektes betrachtet wird oder wie eine korrekte Pose bei ausschließlich linear abhängigen Flächen lautet. Eine Verwendung von Farbmerkmalen könnte hier in Zukunft eine Verbesserung bringen.

Kapitel 7

Vervollständigung von partiellen B-Reps und Nutzerrückmeldungen

Inhalt

7.1	Stand der Forschung	145
7.2	Übersicht	146
7.2.1	Definition von Löchern und deren Eigenschaften	147
7.2.2	Behandlung von Löchern	153
7.3	Bestimmung von Löchern und deren Eigenschaften	154
7.3.1	Notation	155
7.3.2	Bestimmung der Orientierung bei Kardinalität $k = 2$	155
7.3.3	Bestimmung der Orientierung bei Kardinalität $k = 3$	157
7.3.4	Bestimmung der Konvexität zweier Flächen mit gemeinsamen Punkt	159
7.4	Schließen von inneren Löchern	161
7.5	Erzeugung von Nutzerhinweisen	162
7.6	Validierung	166
7.6.1	Validierung durch vollständige Exploration	166
7.6.2	Validierung anhand Realaufnahmen	171
7.7	Zusammenfassung	173

Mit dem in den vorhergehenden Kapiteln vorgestellten System ist es möglich, auf einfache Weise B-Rep Modelle einer Szene oder eines Objektes aufzunehmen. Üblicherweise soll das Resultat dabei vollständig sein. In der Praxis ist dies jedoch nur schwer zu erreichen. Oft treten Stellen im Modell auf, die fehlen. Mögliche Gründe sind dabei folgende:

- Verdeckungen: Es ist nicht möglich, die Kamera so zu platzieren, dass eine Stelle der Oberfläche gesehen wird.
- Sichtwinkel: Es ist zwar möglich, eine Stelle mit der Kamera zu erfassen, allerdings ist die zu rekonstruierende Oberfläche so stark gegen die Kameraebene geneigt, dass aufgrund des Messprinzips trotzdem keine Messung möglich ist.
- Material: Die zu beobachtende Oberfläche absorbiert das Licht einer aktiven Kamera oder reflektiert es weg vom Sensor.

- Fehlbedienung: Der Nutzer hat vergessen, eine Stelle aufzunehmen.
- Limitierung des Algorithmus: Das Verfahren war (beispielsweise aufgrund von Rauschen) nicht in der Lage eine Stelle vollständig zu rekonstruieren.

Ein wesentlicher Punkt, der zur angenehmen Benutzung des Systems beiträgt, ist deshalb die Unterstützung des Nutzers, um Lücken zu vermeiden und fehlende Bereiche im Modell aufzufüllen. Dieses Kapitel beschäftigt sich deshalb einerseits mit der automatisierten Vervollständigung und andererseits mit der Anzeige von Nutzerhinweisen, die den Anwender schon während der Rekonstruktion unterstützen.

Das Kapitel ist in mehrere Abschnitte unterteilt. Zuerst werden verwandte Arbeiten und deren Umgang mit Unvollständigkeits betrachtet (Abschnitt 7.1). Anschließend wird ein neues Konzept zum Umgang mit Lücken vorgestellt (Abschnitt 7.2). Dabei werden Lücken im Kontext von partiellen B-Reps formal definiert und Eigenschaften von Lücken beschrieben. Der nachfolgende Abschnitt befasst sich dann mit dem Finden aller Lücken in einem unvollständigen Modell und dem Bestimmen der zuvor beschriebenen Lückeneigenschaften (Abschnitt 7.3). Darauf folgt die eigentliche Vervollständigung von Modellen, indem Lücken geschlossen werden (Abschnitt 7.4). Da nicht alle Lücken automatisiert geschlossen werden können, werden für die übrigen Lücken Nutzerhinweise erzeugt, die den Anwender bei der Erfassung unterstützen (Abschnitt 7.5). Den Abschluss bildet eine Validierung des vorgestellten Ansatzes (Abschnitt 7.6) und eine Zusammenfassung (Abschnitt 7.7).

7.1 Stand der Forschung

Das Resultat der Rekonstruktion eines Objekts oder eines Innenraumes mit einer handgehaltenen Kamera ist ein 3D-Modell, das je nach Anwendung unterschiedliche Anforderungen bezüglich Genauigkeit oder Vollständigkeit erfüllen muss. Erfolgt die Kontrolle auf Vollständigkeit am Ende, können mehrere Iterationen an Aufnahme und Prüfung nötig sein, bis das gewünschte Ergebnis erreicht ist. Für eine einfache und intuitive Handhabung ist es daher von Vorteil, schon während des Aufnahmevorganges Rückmeldungen an den Nutzer zu geben, die dieser berücksichtigen kann.

Die einfachste Form einer Rückmeldung an den Nutzer ist die Visualisierung des 3D-Modells während der Rekonstruktion. Dies allein lässt den Anwender schon einen Eindruck über Vollständigkeit und Güte des Ergebnisses gewinnen (zum Beispiel in [Izadi11]). Meist wird das Modell dabei aus der Ansicht der aktuellen Position des Nutzers dargestellt, um eine einfache Orientierung zu gewährleisten. Es wird dem Nutzer überlassen, Fehlstellen oder Lücken zu erkennen und darauf zu reagieren.

Die nächsthöhere Stufe an Rückmeldung ist eine visuelle Markierung bestimmter Teile im Modell, um den Nutzer auf diese Stellen hinzuweisen. Breitbach schlägt dazu vor, die Rückseite des Modells einzufärben [Breitbach15], da diese nur sichtbar ist, wenn ein Loch vorliegt und der Nutzer in das Modell hinein blicken kann.

Die höchste Stufe an Rückmeldungen ist nicht nur eine Markierung eines Loches, sondern auch ein Hinweis, wie dieses geschlossen werden kann, beispielsweise in Form einer vorgeschlagenen Aufnahme-Position und -richtung. Das Problem, mit einer neuen Pose möglichst viel Unbekanntes zu erfassen, wird in der Robotik auch als *Next-Best-View-Problem* bezeichnet. Eine Übertragung auf von Menschen gehaltene Kameras erscheint allerdings wenig sinnvoll, da eine berechnete Pose sowieso nicht exakt vom Menschen eingenommen werden kann. In einer früheren Arbeit des Autors wurden deshalb ungefähre, Achsen-ausgerichtete Posen in eine 3D-Punktwolke eingeblendet [Sand14].

Insgesamt lässt sich sagen, dass zumindest eine Visualisierung des Modells nötig ist, um eine einfache und intuitive Rekonstruktion zu gewährleisten. Breitbach begründet dies mit dem Hand-Auge-Regelkreis nach Schweizer [Schweizer71], der besagt, dass es dem Menschen intuitiv möglich ist, den Unterschied zwischen dem angezeigten Soll-Zustand und dem realen Ist-Zustand wahrzunehmen und darauf so zu reagieren, dass der Unterschied abnimmt [Breitbach15]. Eine Hilfestellung in Form von Markierungen von fehlerhaften Stellen ist also sinnvoll, ein Vorschlag von Aufnahmeposes erscheint nach dieser Argumentation überflüssig. Dem entgegen steht die Tatsache, dass die Orientierung im Raum nicht allen Menschen leicht fällt. Schon zweidimensionale Karten zu lesen und zu verstehen erfordert unterschiedliche Fähigkeiten [Montello98]. Der Prozess des Erkennens von Löchern im Modell hin bis zur Einnahme einer möglichen Position zur Korrektur des Modells erfordert dreidimensionales Vorstellungsvermögen sowie Orientierungssinn, insbesondere dann, wenn nicht nur kleine Objekte, sondern ganze Innenräume rekonstruiert werden. Ein Hinweis in Form einer vorgeschlagenen Aufnahmepose kann daher trotzdem eine deutliche Hilfestellung sein, die den Aufnahmeprozess beschleunigt.

Löcher im 3D-Modell können aber nicht nur in Bereichen entstehen, die der Benutzer unzureichend erfasst hat. Beispielsweise können Ecken oder Kanten fehlen, weil die Rekonstruktion eine Ecke oder Kante nicht zuverlässig erkannt hat oder Material- oder Kameraeigenschaften dies nicht zulassen. Dies sind meist kleine Löcher, an denen das Modell nicht wasserdicht ist. Hier hilft es oft nicht, die Stelle erneut zu erfassen, sondern es wird eine algorithmische Reparatur benötigt. Dies wird in der Computergraphik als *model repair* bezeichnet. Dabei existiert eine Vielzahl an Methoden. Für einen Überblick wird auf die Übersichtsarbeiten von Ju et al. [Ju09] und Guo et al. [Guo18] verwiesen. Die Methoden lassen sich grundlegend unterteilen in Verfahren basierend auf Dreiecksnetzen oder basierend auf volumetrischen Gittern, die aus Dreiecksnetzen erzeugt werden. Eine Übertragung auf partielle B-Reps ist aufgrund der grundlegend anderen Datenstruktur nicht direkt möglich. Verfahren, die speziell für CAD-Modelle ausgelegt sind, berücksichtigen zwar spezifische Merkmale wie Ecken und Kanten, arbeiten aber auch auf tesselierten Modellen in Form von Dreiecks- oder Vierecksnetzen [Barequet97, Bischoff05].

7.2 Übersicht

Die unterschiedlichen Gründe für unvollständige Modelle und verschiedenen Herangehensweisen zeigen, dass ein System auch unterschiedlich auf Unvollständigkeit reagieren muss. Einmal ist ein automatisches Schließen angebracht, einmal ein Hinweis an den Nutzer. Hier wird deshalb der Ansatz verfolgt, die jeweils am besten geeignete Lösungsstrategie zu verfolgen, um den Nutzer schon während der Erfassung zu unterstützen und so unnötige Aufnahmezyklen zu vermeiden. Damit eine Strategie ausgewählt werden kann, müssen Eigenschaften und Kriterien existieren, anhand derer eine Entscheidung getroffen werden kann. Deshalb beschäftigt sich der erste Unterabschnitt damit, wie sich Unvollständigkeiten (in dieser Arbeit als *Löcher* bezeichnet) in einem partiellen B-Rep äußern und welche Eigenschaften diese besitzen (Abschnitt 7.2.1). Anschließend kann abhängig von den Eigenschaften eines Loches die passende Strategie gefunden werden (Abschnitt 7.2.2).

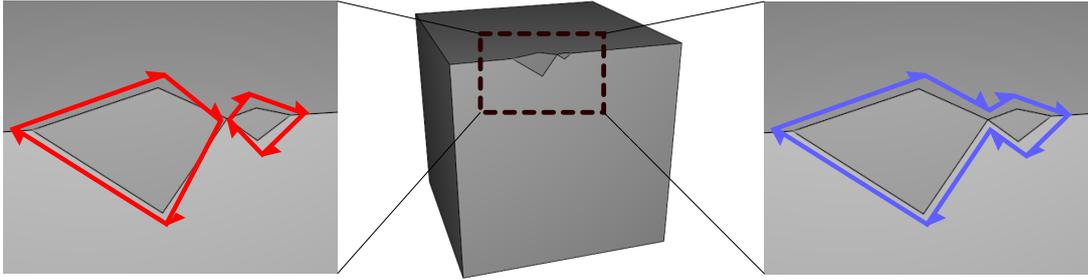


Abbildung 7.1: Der letzte Satz in Definition 7.1 verhindert Mehrdeutigkeiten. Im Fall eines einzelnen Berührungspunktes wird die Konstellation als ein einziges Loch angesehen (rechts). Zwei sich berührende Löcher (links) widersprechen am Berührungspunkt der Forderung, dass eine Nachfolgerkante h_n in der Begrenzung der Fläche entweder eine Zwillingkante besitzt oder Teil desselben Loches ist und dort der Nachfolger von h in der Kette des Loches.

7.2.1 Definition von Löchern und deren Eigenschaften

In Kontext dieser Arbeit werden alle unvollständigen Stellen eines partiellen B-Reps als Loch bezeichnet. Durch das Fehlen einer Zwillingshalbkante kann die Unvollständigkeit leicht getestet werden.

Definition 7.1

Ein **Loch** ist eine geschlossene Kette aus Halbkanten ohne Zwillingkante (siehe auch Def. 3.6 und 3.9). Kette meint dabei, dass der Endknoten einer Halbkante dem Startknoten der nächsten Halbkante entspricht.

Jede Halbkante ohne Zwilling eines partiellen B-Reps ist Teil eines Loches. Halbkanten ohne Zwilling werden im Folgenden deshalb auch **Lochhalbkanten** genannt.

Jede Lochhalbkante h gehört zu genau einer Begrenzung $b = \text{begrenzung}(h)$ einer Fläche. In dieser Begrenzung besitzt h einen Nachfolger $h_n = \text{nach}(h)$. Für h_n muss gelten:

h_n besitzt entweder eine Zwillingshalbkante oder ist ebenfalls Teil desselben Loches wie h und dort der Nachfolger von h in der Kette des Loches.

Der letzte Satz der Definition sorgt für eine Eindeutigkeit aller Löcher eines partiellen B-Reps. So sind beispielsweise mehrere zusammenstoßende Löcher innerhalb einer Fläche abgeschlossen und werden als ein Loch betrachtet (siehe Abbildung 7.1).

Definition 7.2

Die **Kardinalität** $k \in \mathbb{N}$ eines Loches ist die Anzahl an adjazenten Flächen während eines Durchlaufs der Halbkanten eines Loches. Wird eine Fläche mehr als einmal besucht, so wird diese auch mehrfach gezählt.

Beispiele für unterschiedliche Kardinalitäten sind in Abbildung 7.2 dargestellt.

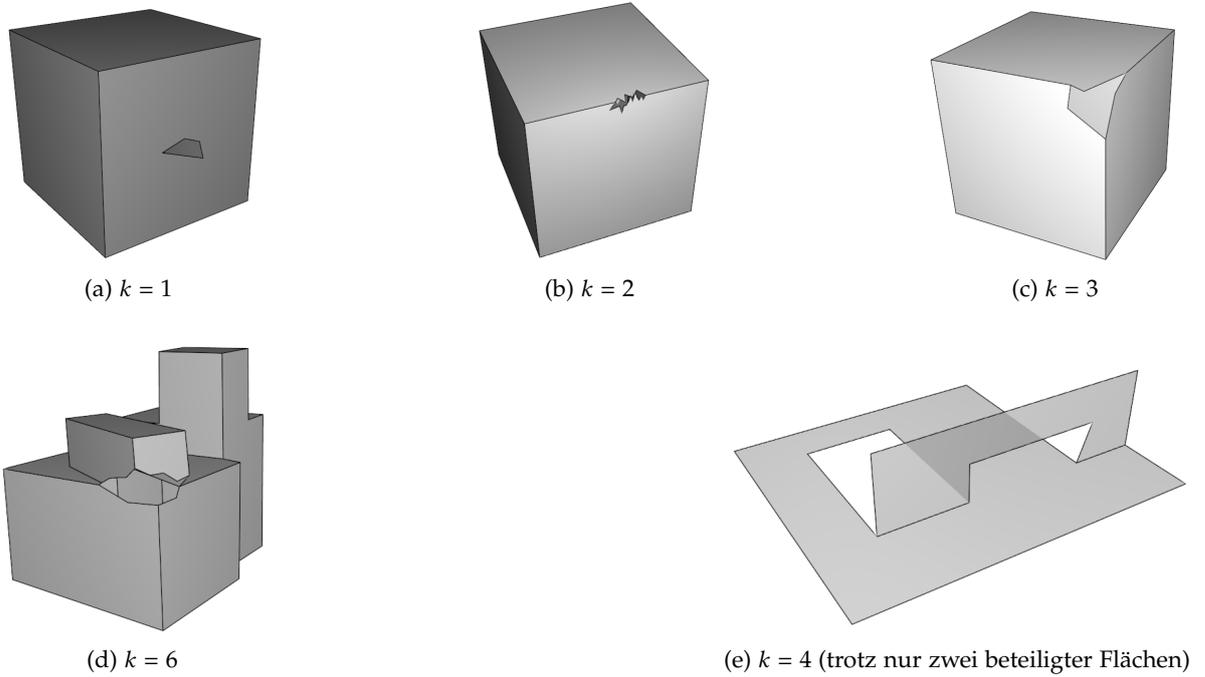


Abbildung 7.2: Beispiele für Löcher mit unterschiedlichen Kardinalitäten k

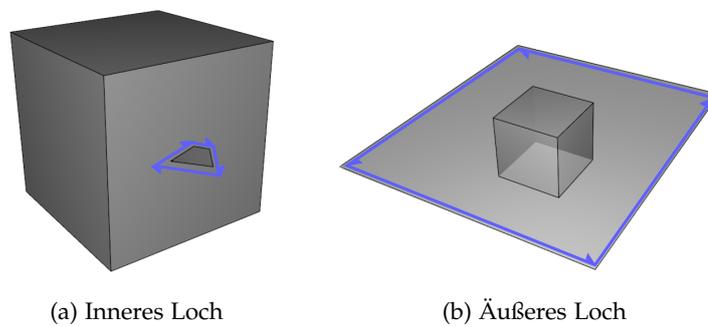


Abbildung 7.3: Beispiele unterschiedlicher Orientierung von Löchern mit Kardinalität $k = 1$

Es ist ersichtlich, dass nicht alle hier definierten Löcher auch Löcher im umgangssprachlichen Sinne sind. Beispielsweise ist der äußere Rand einer Fläche wie in Abbildung 7.3b per Definition auch ein Loch, würde jedoch umgangssprachlich eher nicht so bezeichnet werden. Soll ein Loch geschlossen werden, indem der Nutzer erneut ein Bild aufnimmt und die Rekonstruktion fusioniert, so kann das Loch nur dann vollständig geschlossen werden, wenn alle fehlenden inzidenten Kanten erkannt werden. Dazu ist es nötig, dass die Kamera alle am Loch beteiligten Flächen in dieser Aufnahme sieht. Man erkennt, dass einige Löcher nie mit einer einzigen Aufnahme geschlossen werden können (zum Beispiel Abbildung 7.4d), andere dagegen schon (Abbildung 7.4a). Diesem Umstand wird in der folgenden Definition Rechnung getragen:

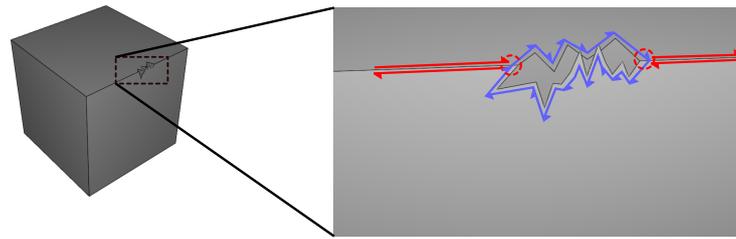
Definition 7.3

Die **Orientierung** $o \in \{\text{innen, außen}\}$ eines Loches beschreibt die Ausrichtung des Loches bezüglich der beteiligten Flächen. Ein Loch ist ein **inneres** Loch, wenn das Loch geschlossen werden kann, indem jede beteiligte Fläche an den am Loch beteiligten Halbkanten vergrößert oder verkleinert wird.

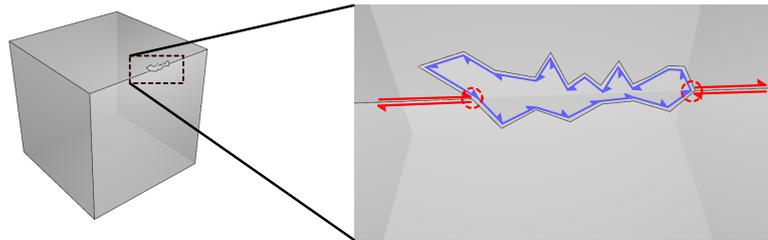
Beispiele für innere Löcher sind

- Löcher innerhalb einer einzigen Fläche (Kardinalität 1, Abbildung 7.3a)
- fehlende Kanten (Kardinalität 2, Abbildung 7.4a-c)
- fehlende Ecken (Kardinalität 3, Abbildung 7.5a+c+e+g).

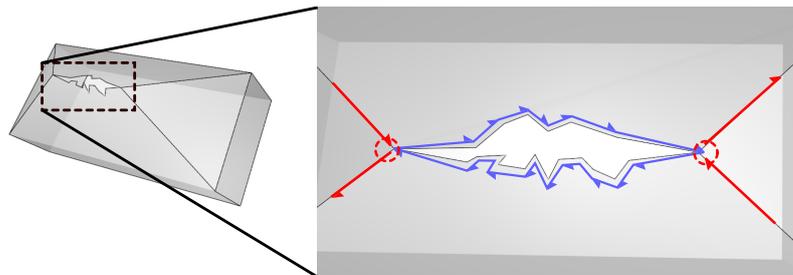
Innere Löcher entsprechen dabei eher dem umgangssprachlichen Gebrauch des Wortes Loch. Äußere Löcher würde man umgangssprachlich eher als Unvollständigkeit bezeichnen. Für innere Löcher der Kardinalität 1 bis 3 existiert eine intuitive Vorstellung, wie diese Löcher geschlossen werden können, in dem die fehlende Ecke, Kante oder Fläche eingefügt wird. Für äußere Löcher ist unklar, wie die Oberfläche an den Lochhalbkanten aussehen könnte.



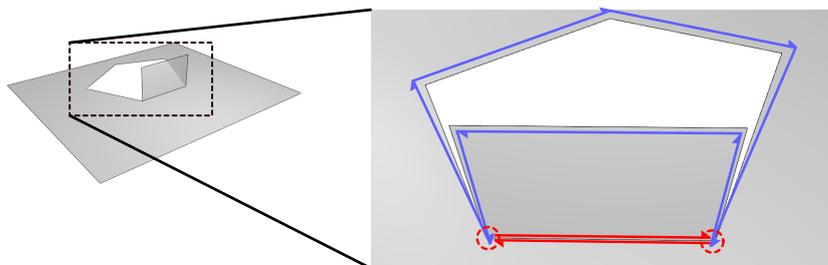
(a) Inneres Loch



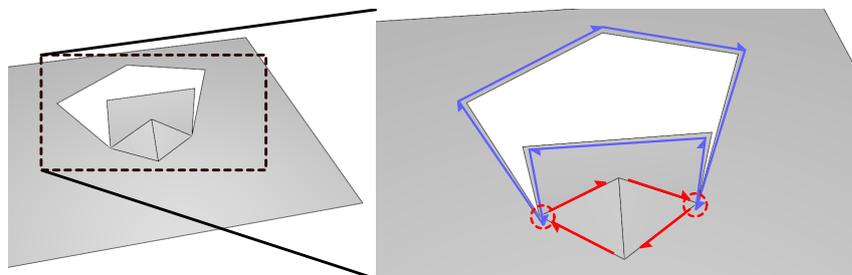
(b) Inneres Loch, bei dem die Flächen überstehen, sodass nicht ins Innere des Würfels geblickt werden kann



(c) Inneres Loch

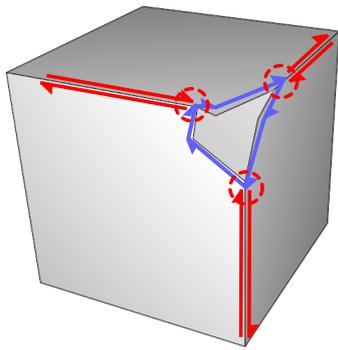


(d) Äußeres Loch

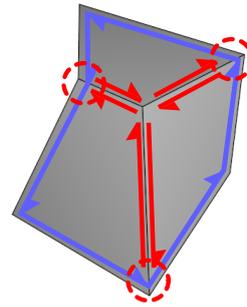


(e) Äußeres Loch

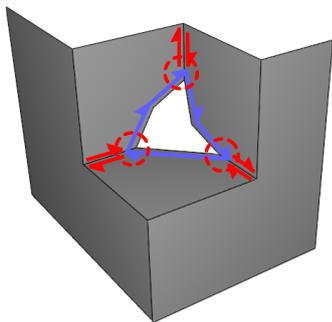
Abbildung 7.4: Beispiele von Löchern mit Kardinalität $k = 2$. Lochhalbkanthen sind blau, Transitionshalbkanthen rot markiert. Transitionsknoten sind rot eingekreist.



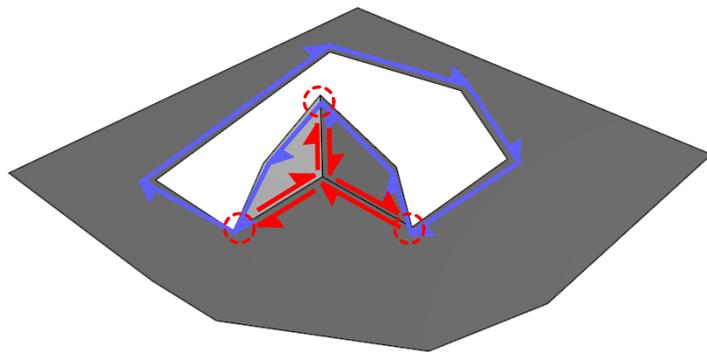
(a) Inneres Loch



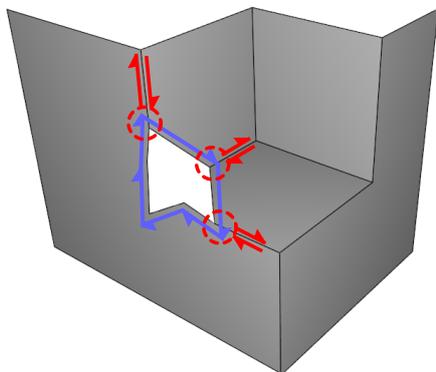
(b) Äußeres Loch



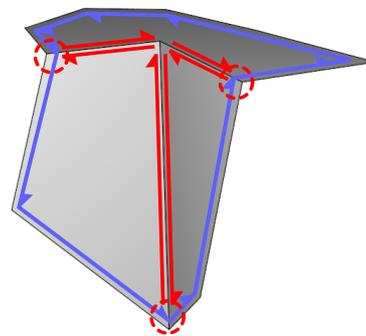
(c) Inneres Loch



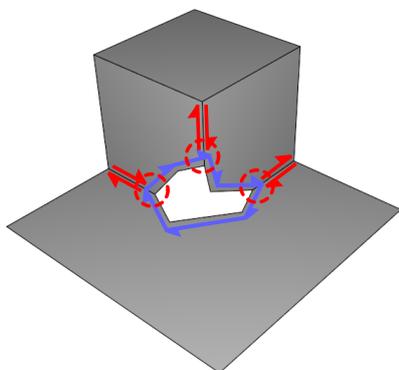
(d) Äußeres Loch



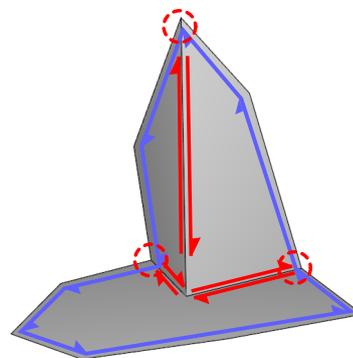
(e) Inneres Loch



(f) Äußeres Loch



(g) Inneres Loch



(h) Äußeres Loch

Abbildung 7.5: Beispiele von Löchern mit Kardinalität $k = 3$. Lochhalbanten sind blau, Transitionhalbanten rot markiert. Transitionsknoten sind rot eingekreist.

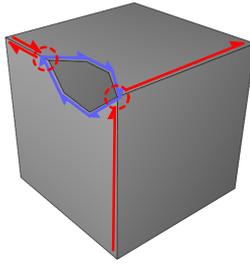


Abbildung 7.6: Ein Loch der Kardinalität $k = 2$ (blau umrandet) besitzt zwei Transitionsknoten (rot eingekreist). An jedem Transitionsknoten existiert eine eingehende und eine ausgehende Transitionshalbkante (rot). Beide können zueinander in Zwillingsbeziehung stehen (Transitionshalbkanten am linken Transitionsknoten) oder getrennt voneinander liegen und Zwillingsbeziehungen zu Kanten anderer Flächen bilden (Transitionshalbkanten am rechten Transitionsknoten). Im ersten Fall werden beide Halbanten als Zwillings-Transitionskanten bezeichnet.

Für die weitere Beschreibung des Vorgehens sind noch zwei Definitionen nötig, die angrenzende Knoten und Halbanten beschreiben.

Definition 7.4

Ein Knoten zwischen zwei aufeinander folgenden Lochhalbanten wird als **Transitionsknoten** bezeichnet, wenn die eingehende Lochhalbante zu einer anderen Fläche gehört als die ausgehende Lochhalbante.

Die Anzahl der Transitionsknoten eines Loches entspricht der Kardinalität des Loches, mit Ausnahme der Kardinalität 1, bei der es keine Transitionsknoten gibt.

Definition 7.5

Eine **Transitionshalbante** ist eine Halbante,

- die einer Lochhalbante, die in einem Transitionsknoten endet, in der Begrenzung ihrer Fläche folgt. Beziehungsweise,
- die einer Lochhalbante, die in einem Transitionsknoten beginnt, in der Begrenzung ihrer Fläche vorhergeht.

Daraus ergeben sich folgende Beobachtungen:

- Transitionshalbanten besitzen immer eine Zwillingshalbante und sind nie Teil eines Loches.
- Zu jedem Transitionsknoten eines Loches gehören genau zwei Transitionshalbanten.
- Diese können zueinander in Zwillingsbeziehung stehen, müssen aber nicht.

Zwei Transitionshalbanten, die in Zwillingsbeziehung zueinander stehen, werden **Zwillings-Transitionskanten** genannt.

In Abbildung 7.6 sind die in den beiden Definitionen beschriebenen Elemente veranschaulicht. Weitere Beispiele sind zudem in den Abbildungen 7.4 und 7.5 zu finden.

7.2.2 Behandlung von Löchern

In diesem Abschnitt soll nun eine Vorgehensweise zur Unterstützung des Nutzers vorgestellt werden. Im Stand der Forschung wurde deutlich, dass zwei grundlegend verschiedene Herangehensweisen existieren: Einerseits die Unterstützung des Nutzers während der Erfassung durch visuelle Rückmeldungen, andererseits das automatisierte Vervollständigen von Modellen. Beide haben zum Ziel, ein vollständiges Endergebnis zu erzeugen.

Bei der vorhergehenden Betrachtung von Unvollständigkeiten (Löcher) und deren Eigenschaften wird zudem deutlich, dass sich einerseits durch die Verwendung von partiellen B-Reps Löcher leicht finden lassen, aber andererseits sich nicht alle Löcher ohne Zusatzwissen sinnvoll automatisiert schließen lassen. Die Tatsache, dass ein Loch automatisiert geschlossen werden kann, heißt noch nicht, dass dieser Schluss korrekt und vom Nutzer gewollt ist.

Demnach lässt sich in einem partiellen B-Reps folgende Einteilung vornehmen:

Es gibt Löcher,

- die automatisch geschlossen werden können und von denen der Anwender auch will, dass diese geschlossen werden,
- die automatisch geschlossen werden können und von denen der Anwender nicht will, dass diese geschlossen werden,
- die nicht automatisch geschlossen werden können.

Die Frage nach dem Wunsch des Anwenders kann im Rahmen dieser Arbeit durch Verwendung der Strukturgröße angenähert werden. Da dieser vom Nutzer gewählte Parameter genau die Dimension angibt, unter der keine geometrischen Merkmale erkannt werden sollen, ist es konsistent, Löcher automatisch zu schließen, die kleiner als die Strukturgröße δ_s sind. Ob ein Loch automatisch geschlossen werden kann, ist einerseits eine algorithmische Frage und andererseits davon abhängig wie viele Annahmen oder Zusatzwissen erlaubt werden. Da hier ein selbstständiges 3D-Scan-System betrachtet wird, bei dem das Modell die Realität abbilden soll, wird als Grundlage nur das partielle Modell und die Eigenschaften der darin enthaltenen Löcher herangezogen. In Abschnitt 7.2.1 wurde dabei deutlich, dass für innere Löcher bis Kardinalität drei ein sinnvoller Schluss anhand Kanten und Ecken möglich ist. Demnach ergibt sich folgende Matrix zur Auswahl einer Strategie zur Unterstützung des Nutzers während des Erfassungsprozesses:

Kardinalität	Orientierung	kleiner als δ_s	größer als δ_s
$k \leq 3$	innen	autom. schließen	visuelle Rückmeldung
$k \leq 3$	außen	visuelle Rückmeldung	
$k > 3$		visuelle Rückmeldung	

Als Größe eines Loches kann dabei dessen maximaler Durchmesser ($k = 1$), der maximale Abstand zur fehlenden Kante ($k = 2$) oder zur fehlenden Ecke ($k = 3$) verwendet werden.

Es bleibt die Frage, welche Art der visuellen Rückmeldung verwendet wird. Die Verwendung einer visuellen Markierung, die eine Aufnahmepose vorschlägt ist, wie im Stand der Forschung beschrieben, verwandt mit dem *Next-Best-View* Problem. Eine Herausforderung dabei ist die schnelle Berechnung einer Pose, da der Suchraum mit sechs Freiheitsgraden sehr groß ist. Im Gegensatz zu anderen Systemen, deren globales Modell auf Punktwolken, Voxelräumen oder Dreiecksnetzen basiert, ist in dieser Arbeit allerdings ein hochwertiges geometrisches Modell verfügbar, das für die Berechnung eines Nutzerhinweises genutzt werden kann. Zudem muss die Pose nicht exakt sein, da ein Mensch diese sowieso nicht genau einnehmen kann. Aus diesen Gründen sollen in dieser Arbeit neben der reinen Visualisierung des Modells Nutzerhinweise in Form von 3D-Pfeilen, die in das Modell eingeblendet werden, zum

Einsatz kommen.

In den folgenden Abschnitten werden nun die benötigten Teilprobleme gelöst: Abschnitt 7.3 befasst sich mit der Bestimmung der Eigenschaften von Löchern, Abschnitt 7.4 mit dem automatisierten Schließen von Löchern. Anschließend wird in Abschnitt 7.5 erläutert, wie die Nutzerhinweise berechnet werden können.

7.3 Bestimmung von Löchern und deren Eigenschaften

In diesem Abschnitt wird erläutert, wie alle Löcher eines partiellen B-Reps ermittelt und deren Eigenschaften bestimmt werden können.

Alle Halbkanten eines Loches können gefunden werden, indem ausgehend von einer Halbkante ohne Zwilling stets die Nachfolgerkante in der Begrenzung der Fläche betrachtet wird. Trifft man auf eine Halbkante, die einen Zwilling besitzt, hat man eine Transitionshalbkante gefunden und wechselt zur Begrenzung der Zwillingshalbkante. Dort wird das Verfolgen fortgesetzt, bis man wieder den Ausgangspunkt erreicht. Folgender Pseudocode verdeutlicht das Vorgehen:

```

1 löcher = { }                                     /* Menge aller Löcher */
2 solange eine nicht besuchte Halbkante ohne Zwilling existiert tue
3     start = eine beliebige nicht besuchte Halbkante ohne Zwilling
4     Markiere start als besucht
5     loch = { start }
6     c = nach(start)
7     solange c ≠ start tue
8         wenn zwilling(c) existiert dann          /* c ist Transitionshalbkante */
9             c = zwilling(c)
10        sonst                                     /* c ist Lochhalbkante */
11            loch = loch ∪ c
12            Markiere c als besucht
13            c = nach(c)
14 löcher = löcher ∪ loch

```

Die Bestimmung der Kardinalität eines Loches ist trivial und kann durch Mitzählen der verschiedenen Flächen während des Findens der Lochhalbkanten ermittelt werden. Schwieriger ist die Bestimmung der Orientierung eines Loches. Bei einer Kardinalität von $k = 1$ ist dies noch einfach. Liegen die Lochhalbkanten in der äußeren Begrenzung der Fläche, ist es ein äußeres Loch, liegen sie in einer inneren Begrenzung, handelt es sich um ein inneres Loch (Abbildung 7.3). Die Kardinalitäten $k = 2$ und $k = 3$ werden in den folgenden Unterabschnitten einzeln betrachtet. Die Orientierung von Löchern mit höherer Kardinalität wird nicht betrachtet, da hier ein automatisches Schließen der Löcher im Allgemeinen nicht eindeutig möglich ist und eine Orientierung deshalb nicht nötig ist.

Zum besseren Verständnis sei angemerkt, dass im Folgenden öfter die Termini *konvex* oder *reflexiv* gebraucht werden. Dabei ist die relative Lage zweier Flächen zueinander gemeint, wie sie von konvexen oder reflexiven Kanten bekannt ist, und nicht die Konvexität eines Polygons in einer Ebene.

7.3.1 Notation

Zur Beschreibung der folgenden Methoden ist es nötig, alle beteiligten Elemente eindeutig zu bezeichnen. In diesem Abschnitt wird daher eine eindeutige Notation für alle folgenden Abschnitte eingeführt. In den Abbildungen 7.7 und 7.9 sind die folgenden Bezeichner für Kardinalität $k = 2$ beziehungsweise $k = 3$ visualisiert.

Für $k > 1$ gilt:

- Die an einem Loch der Kardinalität k beteiligten Flächen heißen f_i mit $i = 1, \dots, k$, wobei für jede Lochhalbkante h der Fläche f_i gelten muss, dass deren Folgekante in der Kette des Loches h_f entweder zur selben Fläche gehört oder zur Fläche f_j mit $j = (i + 1) \bmod k$. In Worten bedeutet dies, dass die Reihenfolge der f_i der Reihenfolge der Flächen entlang der (gerichteten) Lochhalbkanten entspricht.
- Die Normale der Flächen f_i wird mit \vec{n}_i bezeichnet.
- Der Transitionsknoten, an dem die Lochhalbkanten von Fläche f_i auf Fläche f_l mit $l = (i - 1 + k) \bmod k$ wechseln, wird mit t_i bezeichnet, der Ort im Raum heißt $\vec{T}_i \in \mathbb{R}^3$.
- Die Richtung der Transitionshalb-kante, die in t_i endet, wird mit \vec{h}_i bezeichnet.
- Der Vektor von einem Transitionsknoten t_i zum nächsten t_j mit $j = (i + 1) \bmod k$ heißt $\vec{v}_{ij} = \vec{T}_j - \vec{T}_i$.

Für $k = 3$ gilt:

- Der Schnittpunkt der drei Flächen heißt \vec{S} .
- Der Vektor von einem Transitionsknoten t_i zum Schnittpunkt heißt $\vec{s}_i = \vec{S} - \vec{T}_i$.

7.3.2 Bestimmung der Orientierung bei Kardinalität $k = 2$

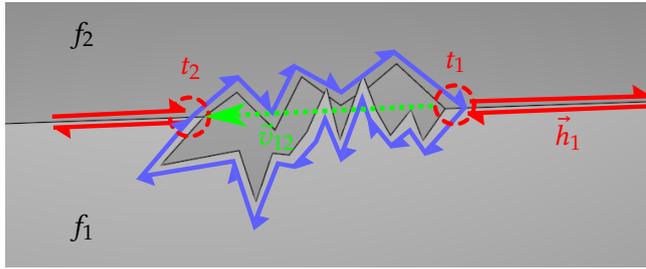
Liegen die Transitionshalb-kanten als Zwillings-Transitions-kanten vor, so ist die Bestimmung der Orientierung direkt möglich. Zeigt v_{12} in dieselbe Richtung wie die Transitionshalb-kante \vec{h}_1 von f_1 , so handelt es sich um ein inneres Loch (Abbildung 7.7a), ansonsten um ein äußeres (Abbildung 7.7b):

$$\text{äußeres Loch} \iff \vec{h}_1 \circ \vec{v}_{12} < 0. \quad (7.1)$$

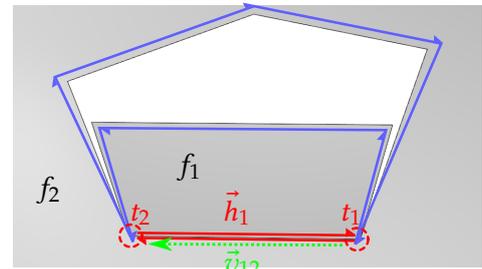
Schwieriger ist die Situation bei Transitionshalb-kanten, die nicht in Zwillingsbeziehung stehen (Abbildungen 7.7c+d). Unter Zuhilfenahme der Normalenvektoren \vec{n}_1 und \vec{n}_2 der Flächen f_1 und f_2 gilt:

$$\text{äußeres Loch} \iff \begin{cases} (\vec{n}_1 \times \vec{n}_2) \circ \vec{v}_{12} < 0 & \text{falls } f_1, f_2 \text{ konvex} \\ (\vec{n}_1 \times \vec{n}_2) \circ \vec{v}_{12} > 0 & \text{falls } f_1, f_2 \text{ reflexiv.} \end{cases} \quad (7.2)$$

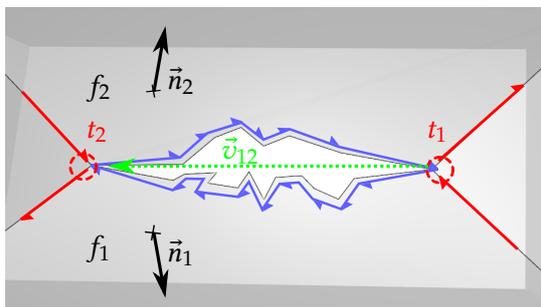
Die Entscheidung über die Orientierung hängt somit davon ab, ob zwei Flächen konvex oder reflexiv zueinander stehen. Der Terminus konvex beziehungsweise reflexiv ist allerdings nur für Kanten definiert, nicht für Flächen, wie in der obigen Formel verwendet. Der Grund ist, dass zwei allgemeine Flächen gleichzeitig sowohl konvexe als auch reflexive Kanten zueinander besitzen können (Abbildung 7.8). Hier benötigt man also die Konvexität am Ort des Loches, also an den Transitionsknoten. Liegen hier Zwillings-Transitions-kanten vor, so ist die Konvexität stets eindeutig. Dies erklärt, warum in diesem Fall die einfache Formel 7.1 gilt, die ein Sonderfall von Gleichung 7.2 ist. Denn dann gilt der Zusammenhang:



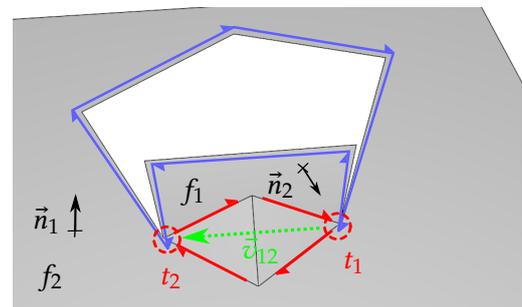
(a) inneres Loch mit Zwilling-Transitionskanten (vgl. Abbildung 7.4a)



(b) äußeres Loch mit Zwilling-Transitionskanten (vgl. Abbildung 7.4d)

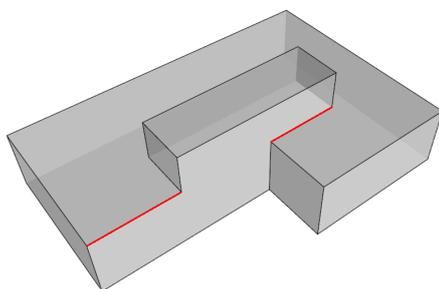


(c) inneres Loch mit getrennten Transitionskanten in konvexer Lage (vgl. Abbildung 7.4c)

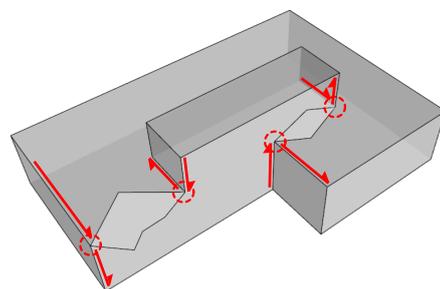


(d) äußeres Loch mit getrennten Transitionskanten in reflexiver Lage (vgl. Abbildung 7.4e)

Abbildung 7.7: Veranschaulichung der verwendeten Bezeichner bei Löchern mit Kardinalität $k = 2$



(a) Zwei Flächen mit einer konvexen und reflexiven Kante (rot markiert)



(b) Zwei innere Löcher

Abbildung 7.8: Zwei Flächen können gleichzeitig zueinander eine konvexe und reflexive Kante besitzen (a). Fehlt die verbindende Kante, so kann dennoch abgeschätzt werden, ob beide Flächen am Ort des Loches konvex oder reflexiv zueinander stehen (b).

$$\vec{h}_i = \begin{cases} (\vec{n}_i \times \vec{n}_j) & \text{falls Kante konvex} \\ -(\vec{n}_i \times \vec{n}_j) & \text{falls Kante reflexiv.} \end{cases} \quad (7.3)$$

Die Bestimmung der Konvexität zweier Flächen an einem gemeinsamen Punkt ohne Zwillings-Transitionsanten ist meist, aber nicht immer, eindeutig bestimmbar. Dieser Umstand wird genauer in Abschnitt 7.3.4 beschrieben. Zuerst folgt jedoch die Bestimmung der Orientierung für $k = 3$, da hier eine analoge Betrachtung möglich ist.

7.3.3 Bestimmung der Orientierung bei Kardinalität $k = 3$

Bei drei beteiligten Flächen unterscheiden sich innere und äußere Löcher darin, ob die drei Flächen an der Stelle des Loches auf den gemeinsamen Schnittpunkt zulaufen oder nicht (Abbildung 7.5). Die Bestimmung der Orientierung für Löcher mit Kardinalität $k = 2$ kann auf $k = 3$ erweitert werden, wie im Folgenden gezeigt wird.

Analog zu $k = 2$, wird zuerst der Fall von Zwillings-Transitionsanten betrachtet, das heißt die Transitionshalbanten liegen auf der Schnittgeraden der beiden am Transitionsknoten beteiligten Flächen (Abbildung 7.9a+b). Die Orientierung kann bestimmt werden, indem die Richtungen von \vec{h}_i und \vec{s}_i verglichen werden:

$$\text{äußeres Loch} \iff \vec{h}_i \circ \vec{s}_i < 0 \quad \forall i \in \{1, 2, 3\}. \quad (7.4)$$

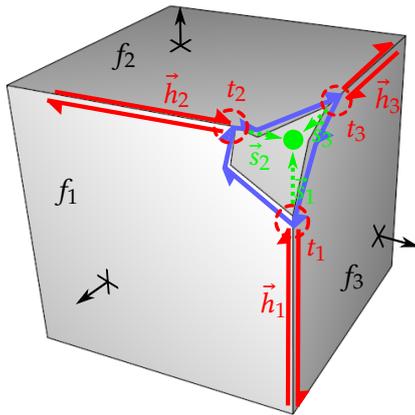
Ebenfalls analog ist die Situation bei getrennten Transitionshalbanten (Abbildung 7.9d+c). Auch hier muss wieder die Konvexität der beiden beteiligten Flächen berücksichtigt werden. Diese muss wieder am Ort des gemeinsamen Transitionspunktes ermittelt werden.

$$\text{äußeres Loch} \iff \begin{cases} (\vec{n}_i \times \vec{n}_j) \circ \vec{s}_i < 0 & \text{falls } f_i, f_j \text{ konvex} \\ (\vec{n}_i \times \vec{n}_j) \circ \vec{s}_i > 0 & \text{falls } f_i, f_j \text{ reflexiv} \end{cases} \quad \forall i \in \{1, 2, 3\}, \quad j = (i - 1 + k) \bmod k. \quad (7.5)$$

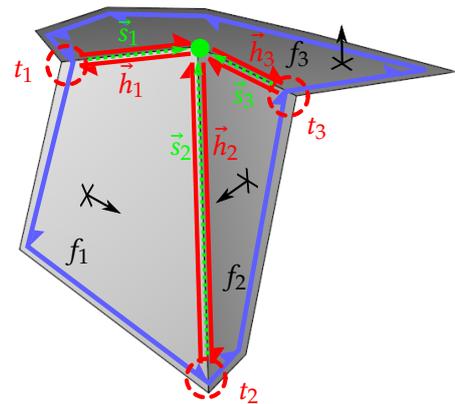
Gleichung 7.4 ist dabei wieder ein Spezialfall von Gleichung 7.5, denn der Zusammenhang 7.3 für Zwillings-Transitionsanten gilt weiterhin.

Wie die Konvexität zweier Flächen mit einem gemeinsamen Punkt bestimmt werden kann, folgt in Abschnitt 7.3.4.

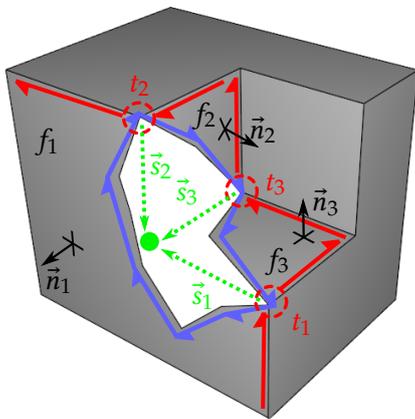
Die zuvor beschriebene Bestimmung der Orientierung schlägt fehl, wenn die beteiligten Flächen linear abhängig sind (Abbildung 7.10). Denn dann kann kein Schnittpunkt berechnet werden, der zur Bestimmung der Orientierung nötig ist. Mit der Tatsache, dass alle inneren Löcher auf einen gemeinsamen Schnittpunkt zulaufen, ist allerdings klar, dass alle Löcher mit linear abhängigen Flächen als äußere Löcher zu klassifizieren sind.



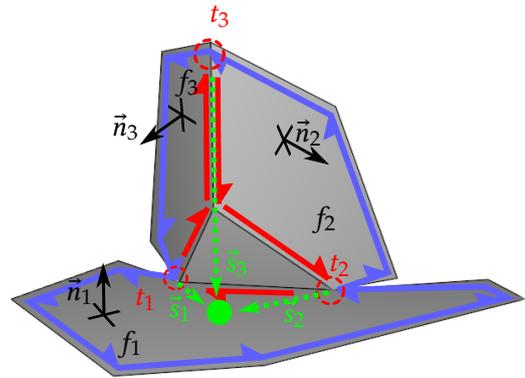
(a) Inneres Loch mit Zwillingstransitionskanten



(b) Äußeres Loch mit Zwillingstransitionskanten



(c) Inneres Loch mit getrennten Transitionshalb-
kanten



(d) Äußeres Loch mit zwei getrennten Transitionshalb-
kanten und einer Zwillingstransitionskante

Abbildung 7.9: Veranschaulichung der verwendeten Bezeichner bei Löchern mit Kardinalität $k = 3$. Lochkanten sind blau, Transitionshalb-
kanten rot markiert. Transitionsknoten sind rot eingekreist. Der Schnittpunkt der drei Flächen und die Vektoren von den Transitionsknoten
zum Schnittpunkt sind grün eingefärbt.

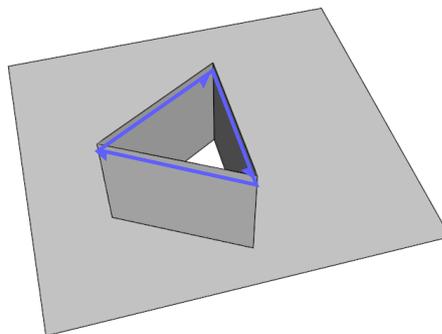


Abbildung 7.10: Ein Loch gebildet durch drei linear abhängige Flächen

7.3.4 Bestimmung der Konvexität zweier Flächen mit gemeinsamen Punkt

Kanten zwischen zwei Flächen f_i und f_j mit den Normalen \vec{n}_i und \vec{n}_j können konvex oder reflexiv sein. Dies ist bestimmbar über die Formel

$$\text{Kante konvex} \iff (\vec{v}_{ij} \times \vec{n}_i) \circ \vec{n}_j > 0, \tag{7.6}$$

wobei \vec{v}_{ij} die Richtung der Halbkante der Fläche f_i darstellt (Abbildung 7.11). Dieser Zusammenhang wurde bereits in Kapitel 6 in Formel 6.1 verwendet. Ebenso wurde dort gezeigt, wie die Konvexität von zwei komplett getrennten Flächen ohne jeglichen Berührungspunkt geschätzt werden kann. Dies muss jedoch nicht immer eindeutig möglich sein.

Im Folgenden soll nun der Fall von zwei sich in einem Transitionsknoten berührenden Flächen betrachtet werden, wie es in den beiden vorhergehenden Abschnitten zur Bestimmung der Orientierung benötigt wird (Abbildung 7.8b).

Es sei angemerkt, dass bei Kardinalität $k = 2$ zwei Transitionsknoten zwischen den beiden betrachteten Flächen existieren, nämlich t_1 und t_2 , während bei Kardinalität $k = 3$ nur ein Transitionsknoten pro Paar existiert (vgl. Abbildung 7.7 und 7.5). Es wird daher im Folgenden von einem Transitionsknoten ausgegangen, wobei folgende Bezeichner verwendet werden:

Wie zuvor heißt die Transitionshalb-kante der Fläche f_i , die im Transitionsknoten t_i endet, h_i ; Zusätzlich wird die Transitionshalb-kante der Fläche f_j , die im Transitionsknoten t_i beginnt, benötigt. Diese wird mit h_j^* bezeichnet (Abbildung 7.12).

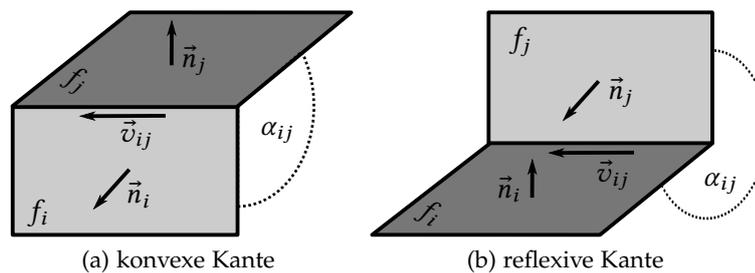


Abbildung 7.11: An eine konvexen Kante (a) ist der Innenwinkel kleiner als 180° , an einer reflexiven Kante (b) größer als 180° .

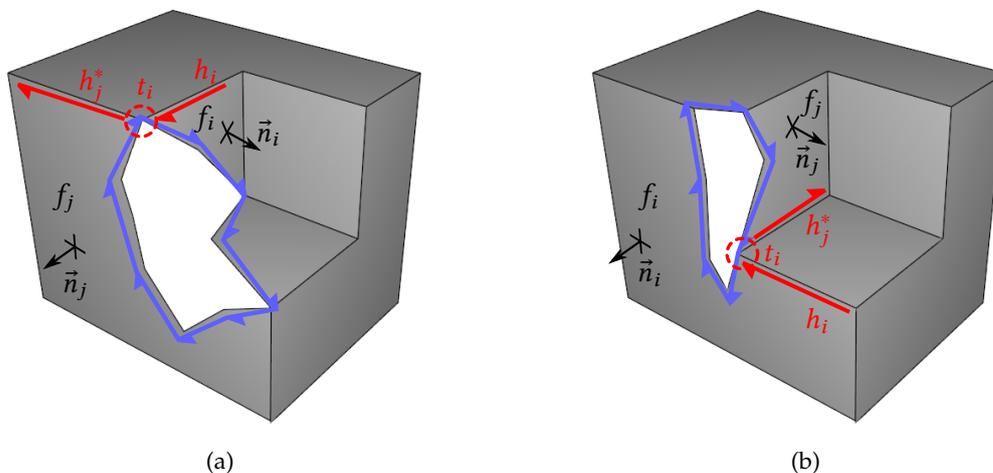


Abbildung 7.12: Bestimmung der Konvexität zwischen den Flächen f_i und f_j

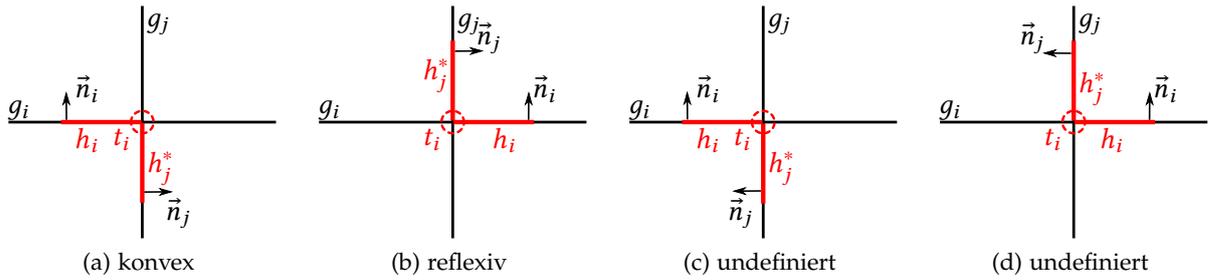


Abbildung 7.13: Es gibt vier mögliche Fälle, wie die Transitionshalbkanthen h_i und h_j zueinander in der Ebene π liegen können (plus die gleichen vier Fälle mit vertauschtem i und j). Das Beispiel aus Abbildung 7.12a resultiert in Fall (a), das Beispiel aus Abbildung 7.12b in Fall (c).

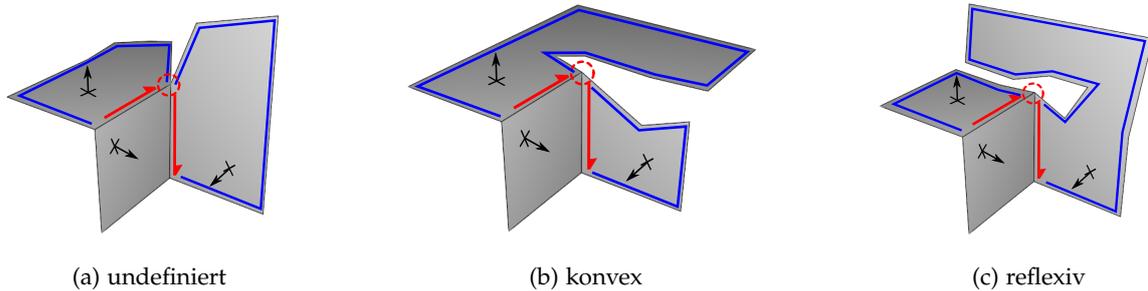


Abbildung 7.14: Bei den undefinierten Fällen aus Abbildung 7.13 ist bei ausschließlicher Betrachtung der beiden Transitionshalbkanthen und der Flächennormalen keine Aussage möglich, denn sowohl eine konvexe (b) als auch eine reflexive (c) Beziehung ist in dieser Konstellation am Transitionsknoten möglich.

Zuerst wird eine Ebene π definiert, die orthogonal auf beiden Flächen steht und den Transitionsknoten t_i enthält:

$$\pi = \left(\begin{array}{c} \vec{n}_i \times \vec{n}_j \\ -(\vec{n}_i \times \vec{n}_j) \circ \vec{T}_i \end{array} \right). \quad (7.7)$$

Projiziert man die Ebenen der Flächen f_i und f_j auf π , so erhält man zwei Geraden, die mit g_i und g_j bezeichnet werden. Für die Lage der beiden Transitionshalbkanthen existieren nun vier Fälle, die in Abbildung 7.13 dargestellt sind (beziehungsweise acht, wenn vertauschtes i und j extra gezählt werden). In den Fällen 7.13a und 7.13b bilden die beiden Halbkanthen eine konvexe beziehungsweise reflexive Verbindung, sodass eindeutig ist, ob der konvexe oder der reflexive Fall für die beiden Flächen vorliegt. In diesen beiden Fällen ist somit die Konvexität am Transitionsknoten bestimmt.

In den Fällen 7.13c und 7.13d ist dies unklar, da die beiden Halbkanthen keine eindeutige Schlussfolgerung zulassen. Abbildung 7.14 zeigt sogar, dass bei ausschließlicher Betrachtung der beiden Transitionshalbkanthen und der Flächennormalen keine Aussage getroffen werden kann, denn sowohl eine konvexe als auch eine reflexive Beziehung ist in dieser Konstellation am Transitionsknoten möglich.

Vergleicht man Abbildung 7.14a mit 7.14b+c, liegt nahe, dass sich in einigen Fällen (b+c) dennoch eine Aussage schätzen lässt, wenn man zusätzlich die Konturen der beteiligten Flächen betrachtet. Projiziert man zusätzlich die Lochhalbkanthen der beiden beteiligten Flächen in die Ebene π , so kann für beide Flächen bestimmt werden, um wie viel die Projektion auf „die andere Seite übersteht“ (Abbildung 7.15). Dieser Wert sei mit $a_i \in \mathbb{R}_0^+$ beziehungsweise $a_j \in \mathbb{R}_0^+$

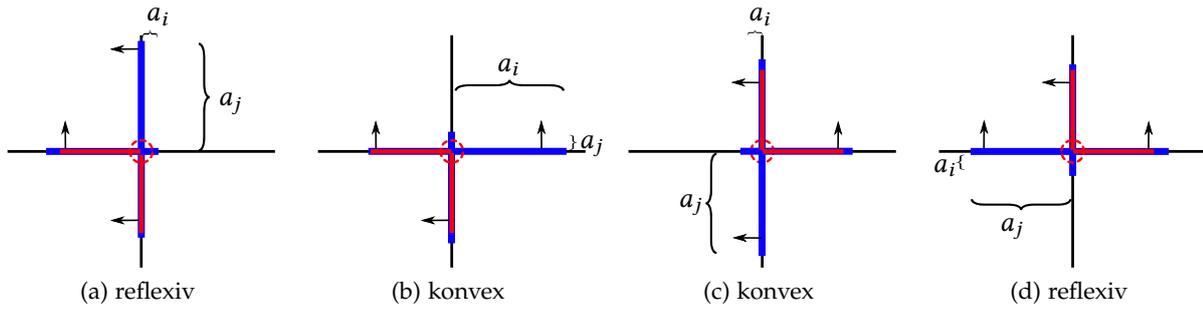


Abbildung 7.15: Projiziert man zu den undefinierten Fällen aus Abbildung 7.13 zusätzlich noch die Lochhalbkanten (blaue Kanten in Abbildung 7.14), so kann die Konvexität geschätzt werden, indem das Maximum von a_i und a_j betrachtet wird. Das Beispiel aus Abbildung 7.14b resultiert in Fall (b), das Beispiel aus 7.14c in Fall (a).

bezeichnet. Je nachdem, welcher Wert größer ist, kann eine konvexe oder reflexive Beziehung angenommen werden.

Es ist ersichtlich, dass es sich dabei nur um eine Schätzung handelt, denn eine eindeutige Lösung muss nicht existieren, wie am Beispiel in Abbildung 7.14a ersichtlich ist. Dennoch kann das Verhältnis

$$r = 1 - \frac{\min(a_i, a_j)}{\max(a_i, a_j)} \quad (7.8)$$

als Konfidenz für die Schätzung verwendet werden. Stehen beide Flächen gleich weit über, so ist $r = 0$.

Bei der Bestimmung der Orientierung von Löchern spielt diese Unsicherheit in der Praxis eher eine untergeordnete Rolle. Damit ein inneres Loch der Kardinalität $k = 3$ fälschlicherweise als äußeres Loch klassifiziert wird, müssten für alle drei Transitionsknoten des Loches getrennte Transitionshalbanten vorliegen, die alle in die undefinierten Fälle aus Abbildung 7.13 fallen und bei denen die Schätzung aus Abbildung 7.15 dann falsch ist.

7.4 Schließen von inneren Löchern

Das Schließen von inneren Löchern bis Kardinalität $k = 3$ ist leicht möglich, da die beteiligten Flächen auf einen gemeinsamen Schnittpunkt oder eine gemeinsame Schnittkante zulaufen beziehungsweise ein Loch innerhalb einer einzelnen Fläche bilden. Folgende Übersicht gibt eine Zusammenfassung, wie die Löcher geschlossen werden können:

- **k=1:** Die innere Begrenzung, in der alle Lochhalbanten liegen, wird gelöscht (Abbildung 7.16a).
- **k=2:** Alle Lochhalbanten der beiden beteiligten Flächen werden jeweils durch eine einzige Halbante ersetzt und diese zueinander in Zwillingbeziehung gesetzt. Falls an einem oder beiden Transitionsknoten bereits Zwilling-Transitionsanten existieren, werden diese mit dem neuen Paar an Halbanten verschmolzen (Abbildung 7.16b).
- **k=3:** Am Schnittpunkt der Flächen wird ein Knoten v_s erzeugt (falls noch nicht existent). Alle Lochhalbanten der drei beteiligten Flächen werden jeweils durch zwei Halbanten ersetzt, wobei als Knoten zwischen den beiden neuen Kanten der Knoten v_s verwendet wird. Halbanten benachbarter Flächen erhalten Zwillingbeziehung. Falls an einem oder mehreren Transitionsknoten bereits Zwilling-Transitionsanten existieren, werden diese mit dem nun benachbarten Paar an Halbanten verschmolzen (Abbildung 7.16c).

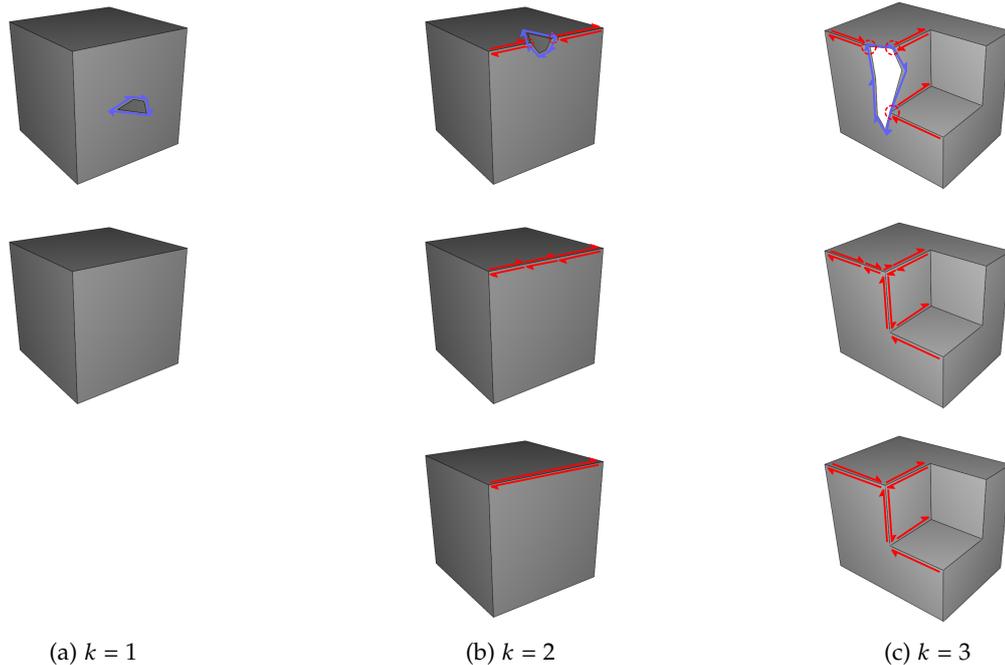


Abbildung 7.16: Schließen von inneren Löchern (oben) durch Ersetzen von Lochkanten (mitte) und Verschmelzen von Kanten (unten), falls nötig

7.5 Erzeugung von Nutzerhinweisen

Ein Nutzerhinweis ist ein Pfeil, der in die 3D-Ansicht des partiellen B-Reps während der Rekonstruktion eingeblendet wird, um den Nutzer auf ein Loch hinzuweisen. Innere Löcher bis Kardinalität 3 können unter Annahme von Verdeckungsfreiheit und Erreichbarkeit mit einer einzigen zusätzlichen Aufnahme geschlossen werden, da alle beteiligten Flächen und Kanten erfasst werden können. In diesem Fall zeigt der Nutzerhinweis die Position und Richtung an, aus der das Loch vollständig mit nur einer einzigen Aufnahme geschlossen werden kann.

Die Berechnung der Position $\vec{p} \in \mathbb{R}^3$ und Richtung $\vec{r} \in \mathbb{R}^3, \|\vec{r}\|_2 = 1$ des Nutzerhinweises wird nun in den folgenden drei Unterabschnitten genauer vorgestellt.

Innere Löcher bis Kardinalität $k = 3$

Eine Ecke oder Kante wird vollständig rekonstruiert, wenn die drei beziehungsweise zwei beteiligten Flächen vollständig sichtbar sind, sodass im Polygonalisierungsschritt der Rekonstruktion nach der Konturverfolgung Kanten zwischen den Flächen angepasst werden können. Bedingt durch das Messprinzip sind bei realen Kameras planare Oberflächen dann am besten zu erfassen, wenn der Winkel zwischen Sichtstrahl und Normale einer Fläche möglichst klein ist, das heißt eine Fläche möglichst senkrecht betrachtet wird. Daraus folgt, dass für alle an einem Loch beteiligten Flächen ein möglichst kleiner Winkel erreicht werden sollte. Es wird deshalb vorgeschlagen, als Richtung des Nutzerhinweises die invertierte Normalensumme der am Loch beteiligten Flächen zu verwenden. Somit wird erreicht, dass alle Flächen unter dem gleichem Winkel betrachtet werden:

$$\vec{r} = \frac{\sum_{i=1..k} -\vec{n}_i}{\left\| \sum_{i=1..k} \vec{n}_i \right\|_2}. \quad (7.9)$$

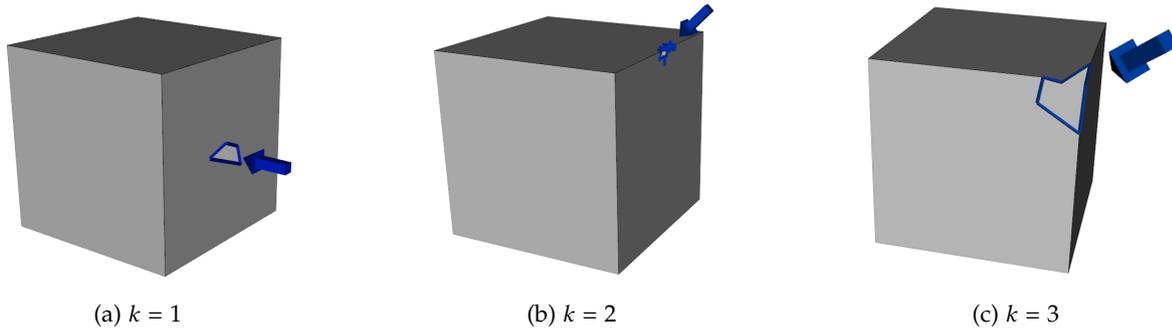


Abbildung 7.17: Nutzerhinweise (blaue Pfeile) für innere Löcher bis $k = 3$

Als Position ist es von Vorteil, wenn direkt auf das Loch geblickt wird. Dazu können die Orte der Transitionsknoten gemittelt ($k = 2, 3$) bzw. der Schwerpunkt \vec{P}_s des Loches ($k = 1$) genutzt werden:

$$\vec{P} = \begin{cases} \vec{P}_s - d \cdot \vec{r} & k = 1 \\ \frac{1}{2}(\vec{T}_1 + \vec{T}_2) - d \cdot \vec{r} & k = 2 \\ \frac{1}{3}(\vec{T}_1 + \vec{T}_2 + \vec{T}_3) - d \cdot \vec{r} & k = 3. \end{cases} \quad (7.10)$$

Der Parameter d entspricht dabei einem Abstand vom Loch in Richtung $-\vec{r}$, der ebenso wie die Größe des Pfeiles an die allgemeine Skalierung des gesamten B-Reps angepasst werden sollte. In Abbildung 7.17 sind die Nutzerhinweise visualisiert.

Äußere Löcher bis Kardinalität 3

Da bei äußeren Löchern die fehlenden Stellen auf der anderen Seite des Objektes liegen als bei inneren Löchern, ist eine Möglichkeit, die umgekehrte Richtung zu verwenden, also

$$\vec{r} = \frac{\sum_{i=1..k} \vec{n}_i}{\left\| \sum_{i=1..k} \vec{n}_i \right\|_2}. \quad (7.11)$$

Dies führt zu Nutzerhinweisen, wie sie in Abbildung 7.18a-c abgebildet sind. Tritt der Fall auf, dass ein Loch entlang einer inneren Begrenzung einer Fläche läuft (horizontale Fläche in Abbildung 7.18c), so kann die Pfeilrichtung unpassend sein: Nimmt man beispielsweise einen Innenraum auf, sodass im Fußboden ein Loch entsteht, dann schlägt der Pfeil eine Richtung von unterhalb des Bodens vor. Dies kann behoben werden, indem berücksichtigt wird, ob ein Loch entlang einer inneren oder äußeren Begrenzung einer Fläche läuft:

$$\vec{r} = \frac{\sum_{i=1..k} a_i \cdot \vec{n}_i}{\left\| \sum_{i=1..k} \vec{n}_i \right\|_2} \quad \text{mit} \quad a_i = \begin{cases} +1 & \text{falls Loch entlang äußerer Begrenzung der Fläche } f_i \text{ läuft} \\ -1 & \text{falls Loch entlang innerer Begrenzung der Fläche } f_i \text{ läuft.} \end{cases} \quad (7.12)$$

Dies führt zu den in Abbildung 7.18d+e abgebildeten Nutzerhinweisen. Die Position \vec{P} der Pfeile kann identisch zu inneren Löchern bestimmt werden.

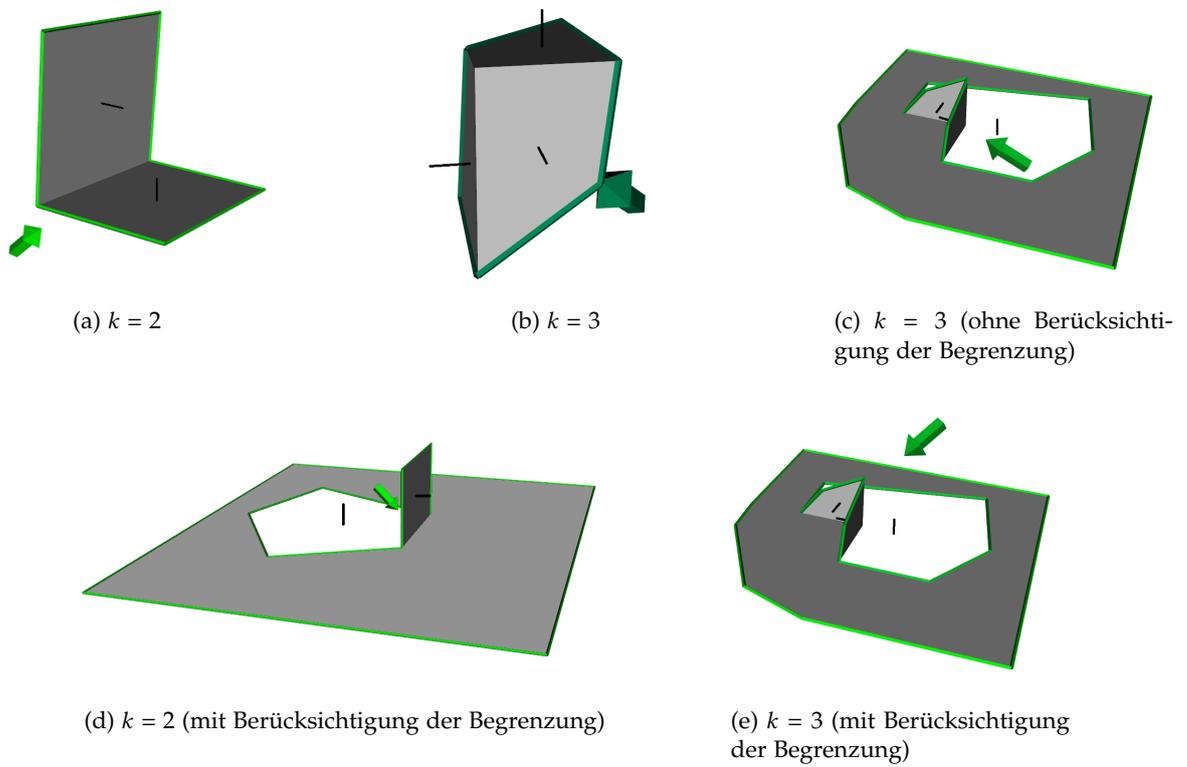


Abbildung 7.18: Nutzerhinweise (grüne Pfeile) für äußere Löcher bis $k = 3$. Zur Verdeutlichung der Außenseite ist am Flächenschwerpunkt die Normale jeder Fläche durch eine schwarze Linie angedeutet.

Löcher ab Kardinalität 4

Für Löcher der Kardinalität $k > 3$ ist nicht klar, wie diese geschlossen werden können, da mehr als drei Flächen nicht auf einen gemeinsamen Schnittpunkt zulaufen müssen. Eine Bestimmung der Orientierung nach obigen Verfahren ist aufgrund des fehlenden Schnittpunktes ebenfalls nicht möglich. Es wird deshalb eine andere Variante zur Bestimmung der Position und Richtung eines Nutzerhinweises vorgeschlagen. Die Idee ist, ein Loch mit gedachten Dreiecken zu schließen und eine Richtung zu ermitteln, mit der diese virtuellen Dreiecke gesehen werden können (Abbildung 7.19). Dazu wird der Mittelpunkt \vec{M} der Transitionsknoten berechnet

$$\vec{M} = \frac{1}{k} \sum_i \vec{T}_i \quad (7.13)$$

und damit Dreiecke mit zwei benachbarten Transitionsknoten erzeugt (Abbildung 7.19b). Für jedes dieser Dreiecke kann die Normale berechnet werden, die analog zu den Flächen nach außen zeigt:

$$\vec{\Delta}_i = (\vec{M} - \vec{T}_i) \times \vec{v}_{ij} \quad \text{mit } j = (i + 1) \bmod k. \quad (7.14)$$

Indem der Normalenvektor der Dreiecke hier nicht normiert wird, kann mit der Formel

$$\vec{r} = - \frac{\sum_i \vec{\Delta}_i}{\left\| \sum_i \vec{\Delta}_i \right\|_2} \quad (7.15)$$

ein mit dem Flächeninhalt eines Dreiecks gewichtetes Mittel der Dreiecksnormalen gebildet und als Richtung für den Nutzerhinweis verwendet werden. Als Ort, auf den der Pfeil zeigt, kann \vec{M} verwendet werden. Das Resultat ist in Abbildung 7.19c zu sehen.

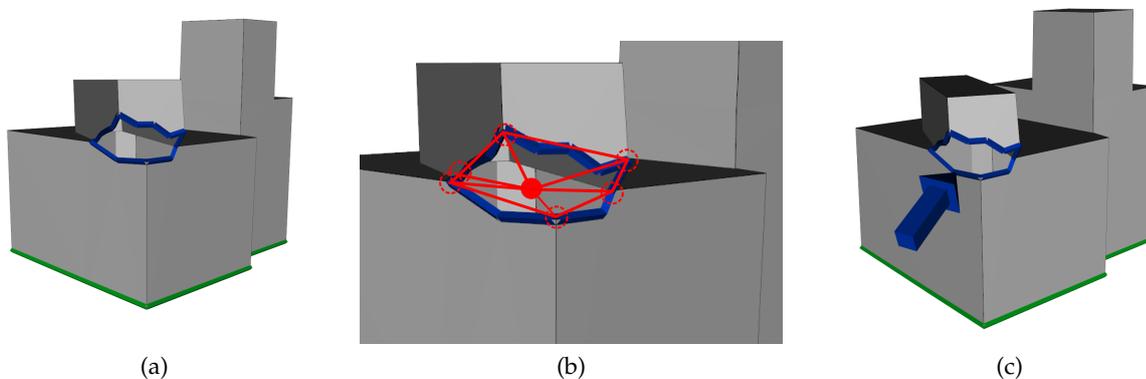


Abbildung 7.19: (a) Ein Loch der Kardinalität $k = 6$, (b) Vergrößerung des Loches mit Transitionsknoten (gestrichelt), Mittelpunkt der Transitionsknoten und virtueller Triangulierung, (c) berechneter Nutzerhinweis

7.6 Validierung

In diesem Abschnitt soll das beschriebene Verfahren zum Schließen von Löchern und zur Erzeugung von Nutzerhinweisen validiert werden. Dazu werden zuerst alle möglichen Locharten bis Kardinalität $k = 3$ vollständig aufgezählt und die Korrektheit des Schließens von Löchern und der Nutzerhinweise gezeigt (Abschnitt 7.6.1). Anschließend wird die beschriebene Methode auf Realweltdaten angewendet und die Ergebnisse präsentiert (Abschnitt 7.6.2).

7.6.1 Validierung durch vollständige Exploration

In diesem Abschnitt soll das beschriebene Verfahren zum Schließen von Löchern und zur Erzeugung von Nutzerhinweisen validiert werden, indem gezeigt wird, dass es für alle möglichen Fälle bis Kardinalität $k = 3$ ein korrektes Ergebnis liefert. Die Korrektheit wird dabei manuell ermittelt, indem überprüft wird, ob ein Nutzerhinweis eine sinnvolle Richtung zur Vervollständigung anzeigt. Beim Schließen von Löchern wird überprüft, ob ein valides B-Rep ohne Loch entsteht. Bis Kardinalität $k = 3$ kann der Problemraum vollständig exploriert werden, da die Anzahl der möglichen Fälle begrenzt ist. Auf eine Untersuchung höherer Kardinalitäten wird verzichtet, da für diese einerseits kein automatisiertes Verfahren zum Schließen der Löcher möglich ist und andererseits die Anzahl der möglichen Fälle exponentiell steigt. Zudem ist die Validität der Nutzerhinweise anhand einiger Beispiele in den Realweltdaten im Folgeabschnitt sichtbar.

Basis für die korrekte Auswahl der Strategie und für Nutzerhinweise ist die Bestimmung der Orientierung, die wiederum von der Lage der Flächen zueinander abhängt. Bei $k = 2$ gibt es hier zwei Möglichkeiten (konvex oder reflexiv), bei $k = 3$ existieren vier Fälle, abhängig davon, wie viele der drei Flächenpaare konvex sind (3x konvex, 2x konvex + 1x reflexiv, 1x konvex + 2x reflexiv, 3x reflexiv). Bei der Bestimmung der Konvexität können die Transitionshalbkanten in Zwillingbeziehung stehen oder getrennt sein. Da Letzteres der allgemeinere Fall ist, wird im Folgenden ausschließlich der Fall von getrennten Transitionshalbkanten betrachtet. Bei der Berechnung von Nutzerhinweisen von äußeren Löchern wird zudem verwendet, ob sich die Lochhalbkanten einer Fläche in der inneren oder äußeren Begrenzung befinden. Demnach muss bei äußeren Löchern unterschieden werden, für wie viele der beteiligten Flächen dies der Fall ist. Dabei werden drei Möglichkeiten unterschieden: Keine, eine, oder mehr als eine der Flächen besitzen Lochhalbkanten, die Teil der inneren Begrenzung der Fläche sind. Insgesamt entstehen somit 26 Fälle, die in Tabelle 7.1 nochmals aufgelistet und in Tabelle 7.2 summiert sind.

Kardinalität	1	2	3
Konvexität	-	konvex reflexiv	3x konvex 2x konvex, 1x reflexiv 1x konvex, 2x reflexiv 3x reflexiv
Orientierung	innen außen	innen außen	innen außen
Begrenzung	-	0x innere Begrenzung 1x innere Begrenzung 2x innere Begrenzung	0x innere Begrenzung 1x innere Begrenzung ≥2x innere Begrenzung

Tabelle 7.1: Mögliche Ausprägungen der verschiedenen Möglichkeiten der Behandlung von Löchern

Kardinalität	Konvexität	Anzahl Fälle		Summe
		Orientierung	Begrenzung (nur bei äußeren Löchern)	
1	1	2	1	2
2	2	2	3	8
3	4	2	3	16

Tabelle 7.2: Anzahl der möglichen Fälle bei der Behandlung von Löchern der Kardinalität $k \leq 3$. Die Gesamtzahl berechnet sich über $\#Konvexität \cdot (1 + \#Begrenzung)$, da die Betrachtung der Begrenzung nur bei äußeren Löchern relevant ist.

Für alle möglichen Fälle wurden künstliche Beispielm Modelle erzeugt und manuell überprüft, ob der erzeugte Nutzerhinweis eine sinnvolle Richtung darstellt. Für innere Löcher wurde zudem getestet, ob das Loch korrekt geschlossen wurde. In den Abbildungen 7.20 bis 7.24 sind einige der Fälle dargestellt. Es wurde festgestellt, dass das vorgestellte Verfahren alle Löcher korrekt klassifiziert, schließt und Nutzerhinweise berechnet, solange nicht einer der folgenden Fälle auftritt:

Für Löcher, deren Lochhalbanten in mehr als einer inneren Begrenzung verlaufen (Abbildung 7.25a), ist der Nutzerhinweis wenig aussagekräftig. Für Löcher, bei denen eine Fläche um eine andere gewunden ist (Abbildung 7.25b), wird die Orientierung falsch bestimmt, was aber keine Auswirkung auf den Nutzerhinweis hat. Beide Fälle spielen aber in der Praxis eine untergeordnete Rolle, da genau diese Konstellation auftreten muss, ohne dass weitere Flächen am Loch beteiligt sind, die die Kardinalität ändern. Insbesondere Löcher, die in mehr als einer inneren Begrenzung verlaufen, sind schon in der Theorie schwierig zu konstruieren (siehe Abbildung 7.25a).

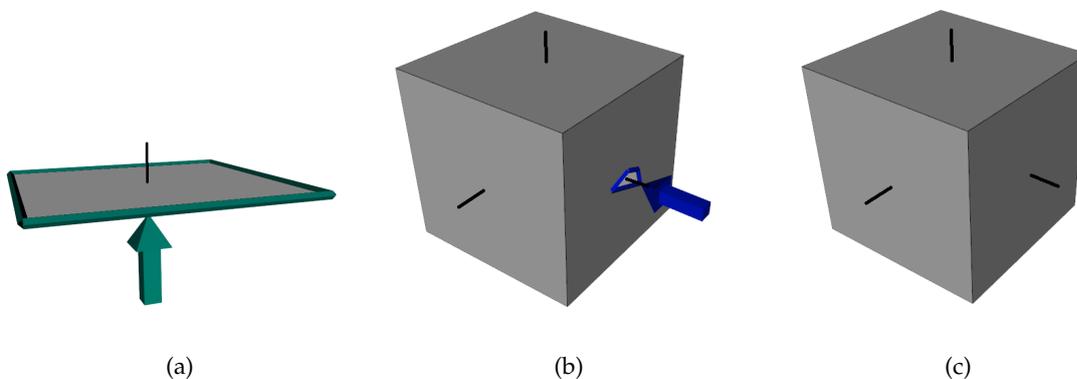


Abbildung 7.20: Löcher der Kardinalität $k = 1$: (a) Äußeres Loch, (b) Inneres Loch, (c) Modell nach Schließen des inneren Loches. Die Richtungen der Flächennormalen sind durch schwarze Linien angedeutet.

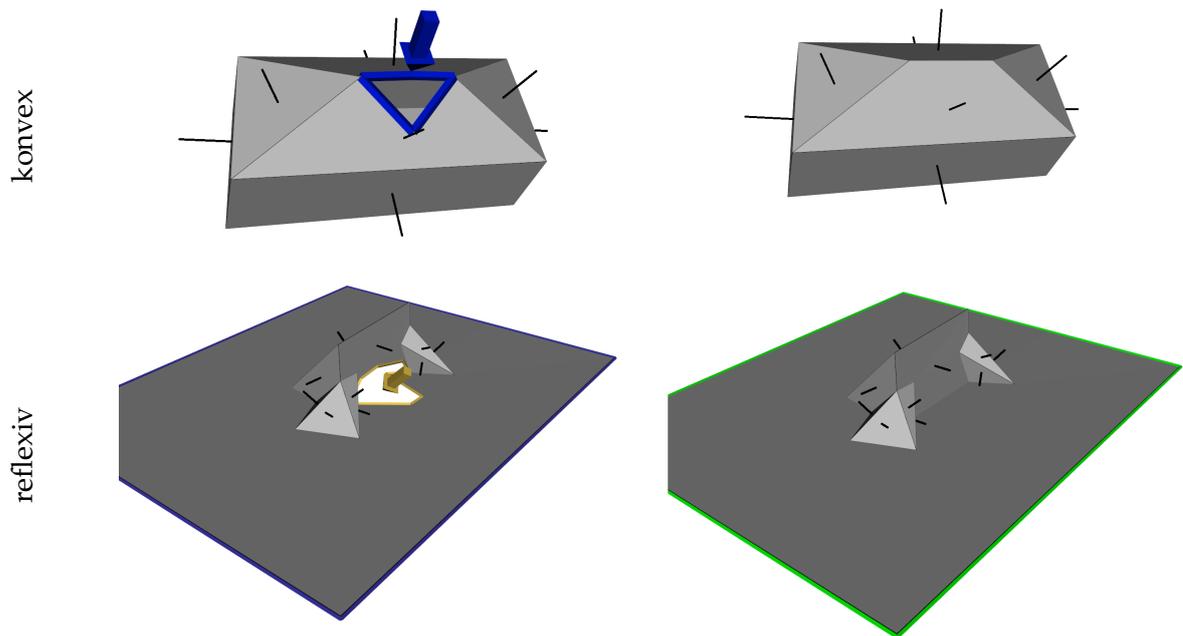


Abbildung 7.21: Innere Löcher der Kardinalität $k = 2$ für reflexive und konvexe Kanten. Links: Nutzerhinweise. Rechts: Modell nach dem Schließen des Loches. Die Richtungen der Flächennormalen sind durch schwarze Linien angedeutet.

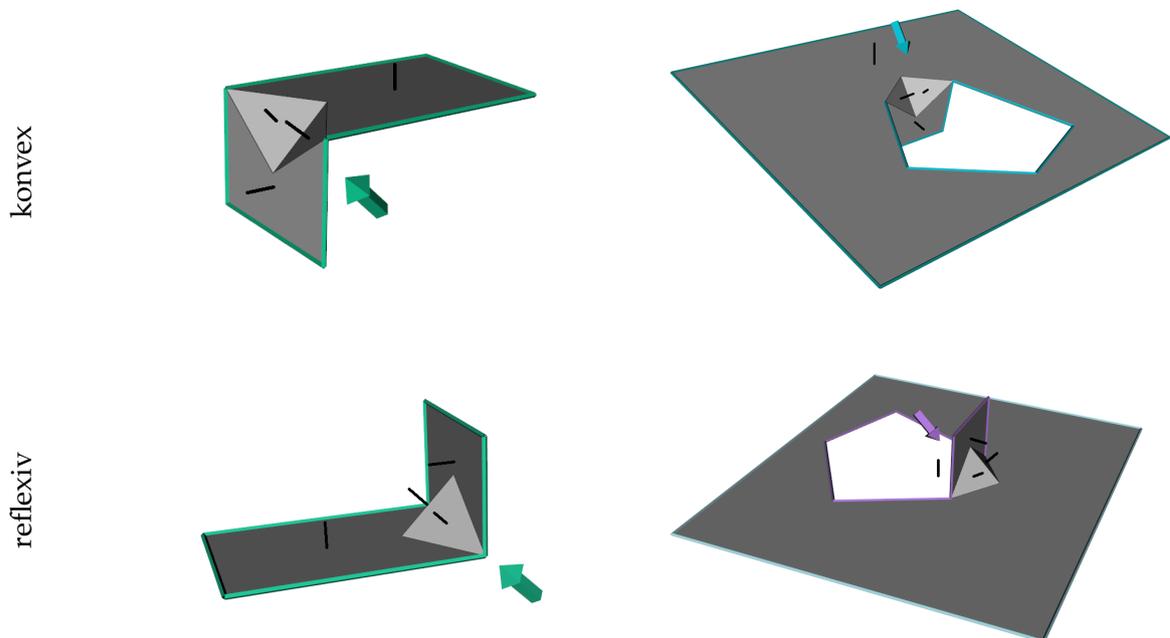


Abbildung 7.22: Äußere Löcher der Kardinalität $k = 2$ mit Nutzerhinweisen für reflexive und konvexe Kanten. Links: Lochkanten liegen nur auf äußeren Begrenzungen. Rechts: Lochkanten liegen auf einer inneren Begrenzung. Die Richtungen der Flächennormalen sind durch schwarze Linien angedeutet.

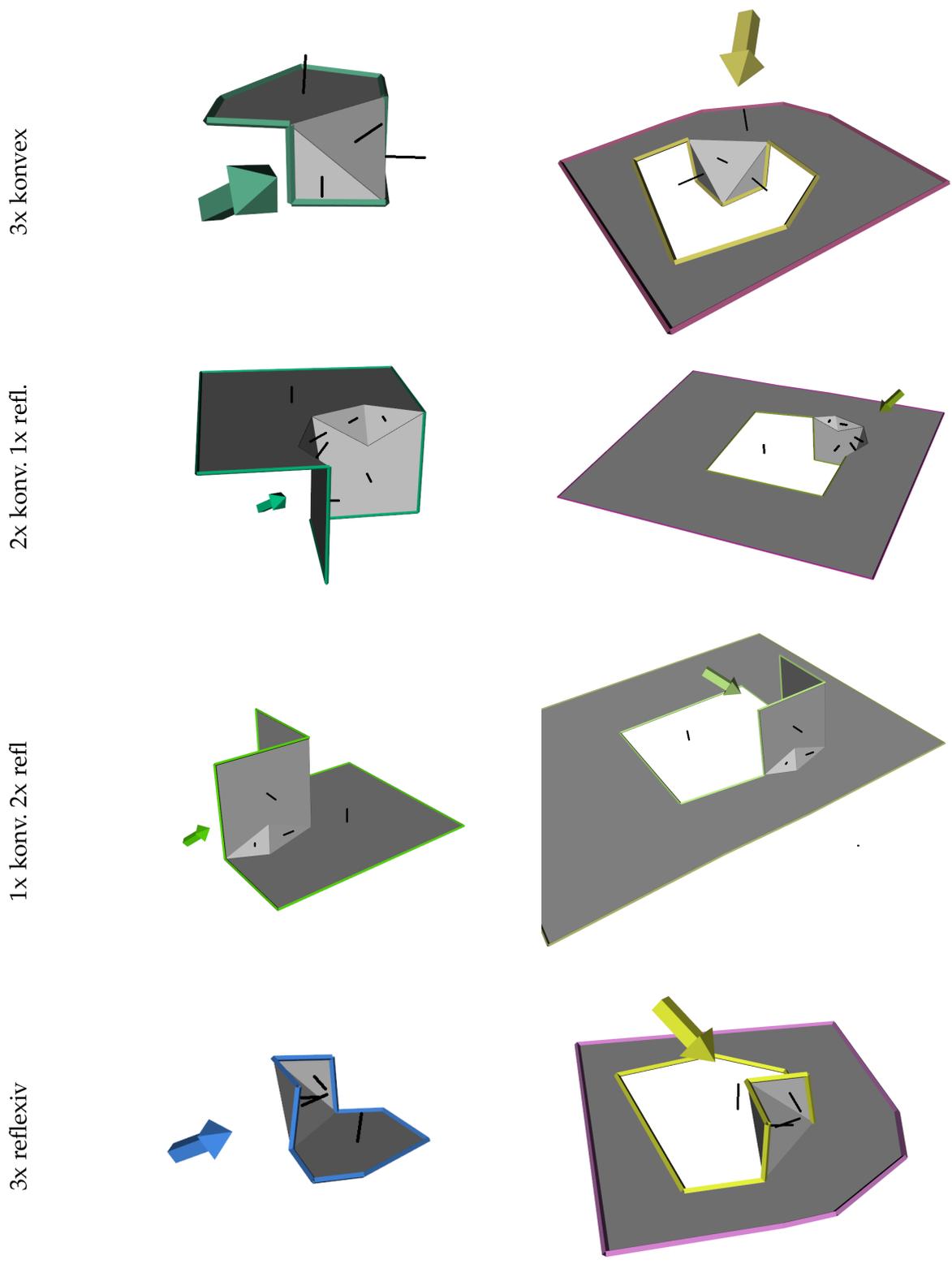


Abbildung 7.23: Äußere Löcher der Kardinalität $k = 3$ mit Nutzerhinweisen für die vier möglichen Fälle an Konvexität der drei Kanten. Links: Lochkanten liegen nur auf äußeren Begrenzungen. Rechts: Lochkanten liegen auch auf einer inneren Begrenzung. Die Richtungen der Flächennormalen sind durch schwarze Linien angedeutet.

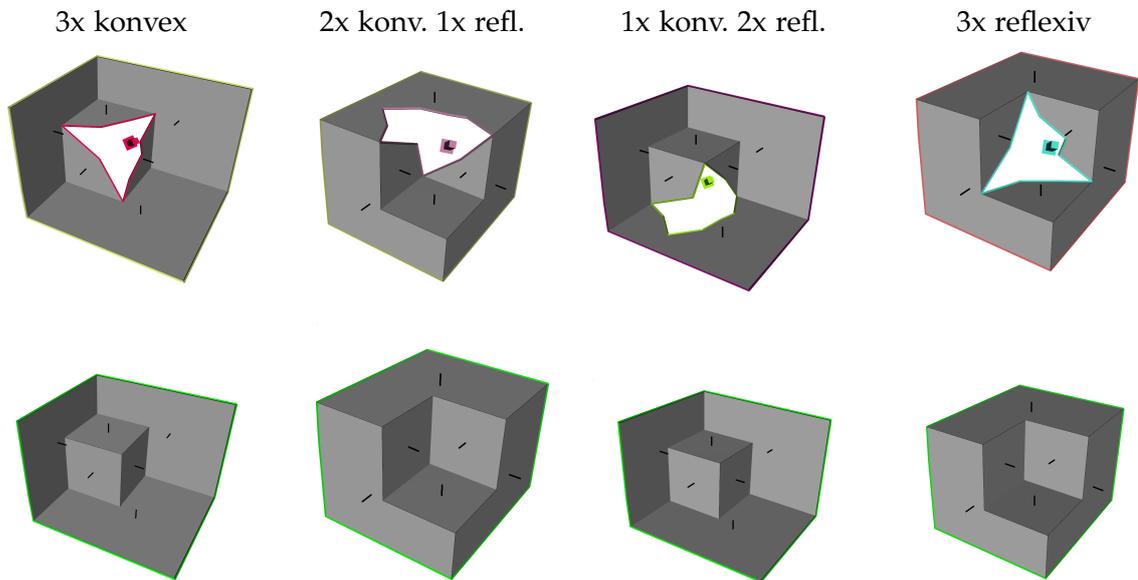


Abbildung 7.24: Innere Löcher der Kardinalität $k = 3$ für die vier möglichen Fälle an Konvexität der drei Kanten. Oben: Nutzerhinweise. Unten: Modell nach dem Schließen des Loches. Die Richtungen der Flächennormalen sind durch schwarze Linien angedeutet.

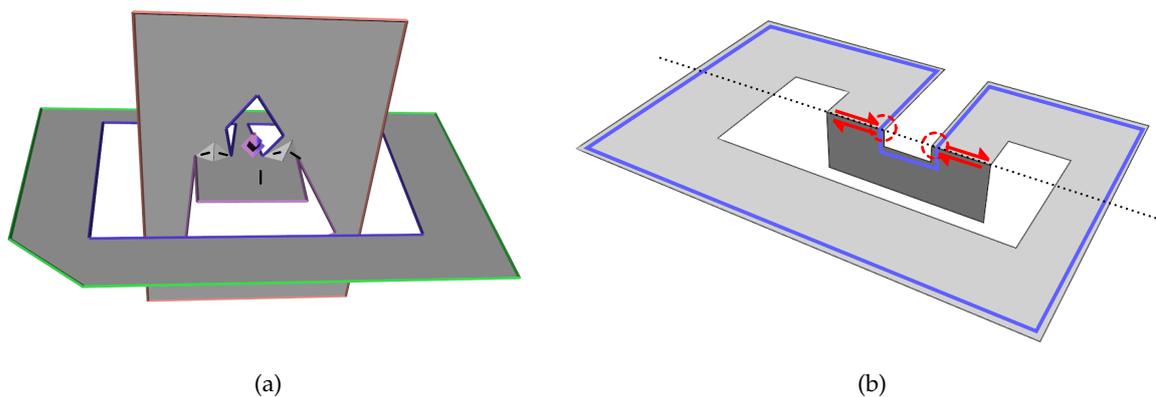


Abbildung 7.25: Die Berechnung der Nutzerhinweise ist wenig aussagekräftig bei Löchern, deren Lochhalbanten in mehr als einer inneren Begrenzung liegen (a). Bei Löchern, bei denen eine Fläche um eine andere Fläche gewunden ist, wird die Orientierung falsch bestimmt (b).

7.6.2 Validierung anhand Realaufnahmen

Neben den theoretischen Überlegungen soll nun der vorgestellte Ansatz an realen Daten dargestellt werden. Dazu wird zunächst eine Szene mit einer handgehaltenen Kamera rekonstruiert und die Rohdaten gesichert. Anschließend wird die Rekonstruktion am Rechner erneut durchgeführt und zwar einmal mit und einmal ohne dem automatischen Schließen von Löchern. Das Ergebnis ist in Abbildung 7.26 dargestellt. Vergleicht man die Ergebnisse, so sind im Groben keine Unterschiede festzustellen. In einer Detailansicht zeigt sich jedoch, dass viele kleine, mit dem Auge schlecht wahrnehmbare Löcher geschlossen werden, hauptsächlich an Ecken und Kanten. Der Grund ist, dass im B-Rep eine Kante nach der Fusion nur vorhanden ist, wenn sie in mindestens einer Einzelaufnahme erfasst werden konnte (vergleiche Evaluation im Kapitel 5). Ist dies nicht der Fall, so fehlt die Kante. Ist die Rekonstruktion der Nachbarflächen trotzdem gut, so ist das Loch kaum wahrnehmbar. Das automatisierte Schließen hilft hier, diese Löcher zu entfernen.

Als zweites wurde eine Szene aus Kisten aufgebaut und rekonstruiert. Dabei wurden Nutzerhinweise in die Live-Visualisierung eingeblendet. Abbildung 7.27 zeigt die Hinweise zu verschiedenen Zeitpunkten während der Erfassung. Man erkennt, dass die Anzahl der Hinweise anfangs zunimmt und ihr Maximum erreicht, sobald das Modell einmal umrundet wurde, aber noch einige Löcher vorhanden sind. Der Nutzer kann darauf reagieren und die betroffenen Stellen sukzessive erneut erfassen. Die Anzahl der Hinweise sinkt, bis am Ende das Modell vollständig rekonstruiert wurde.

Eine weitere Beobachtung ist, dass sich Pfeile mit anderen Pfeilen oder Objekten schneiden können, da bei der Berechnung der Hinweise kein Kollisionstest mit anderen Pfeilen oder dem Modell durchgeführt wird. Dieser Effekt tritt jedoch nicht häufig auf.

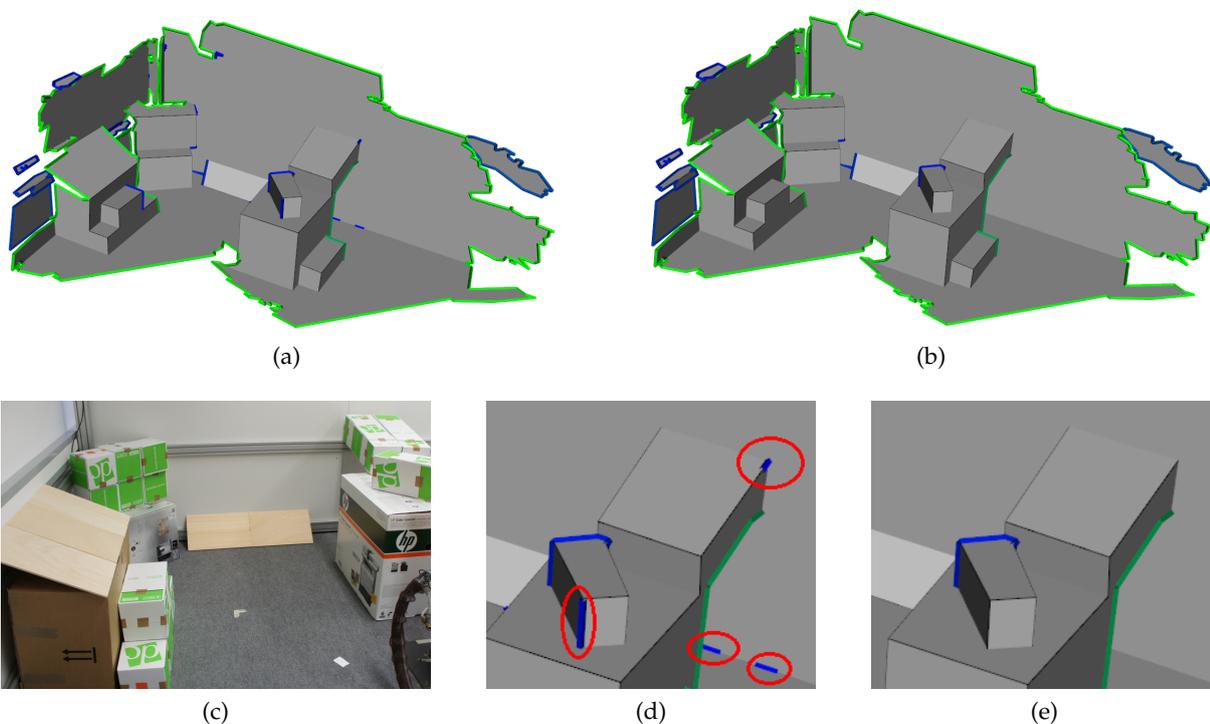
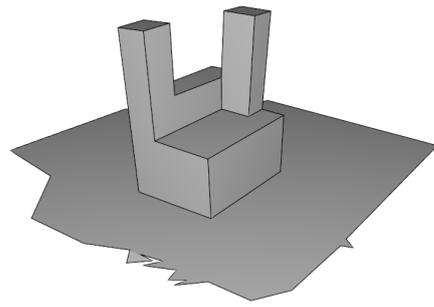


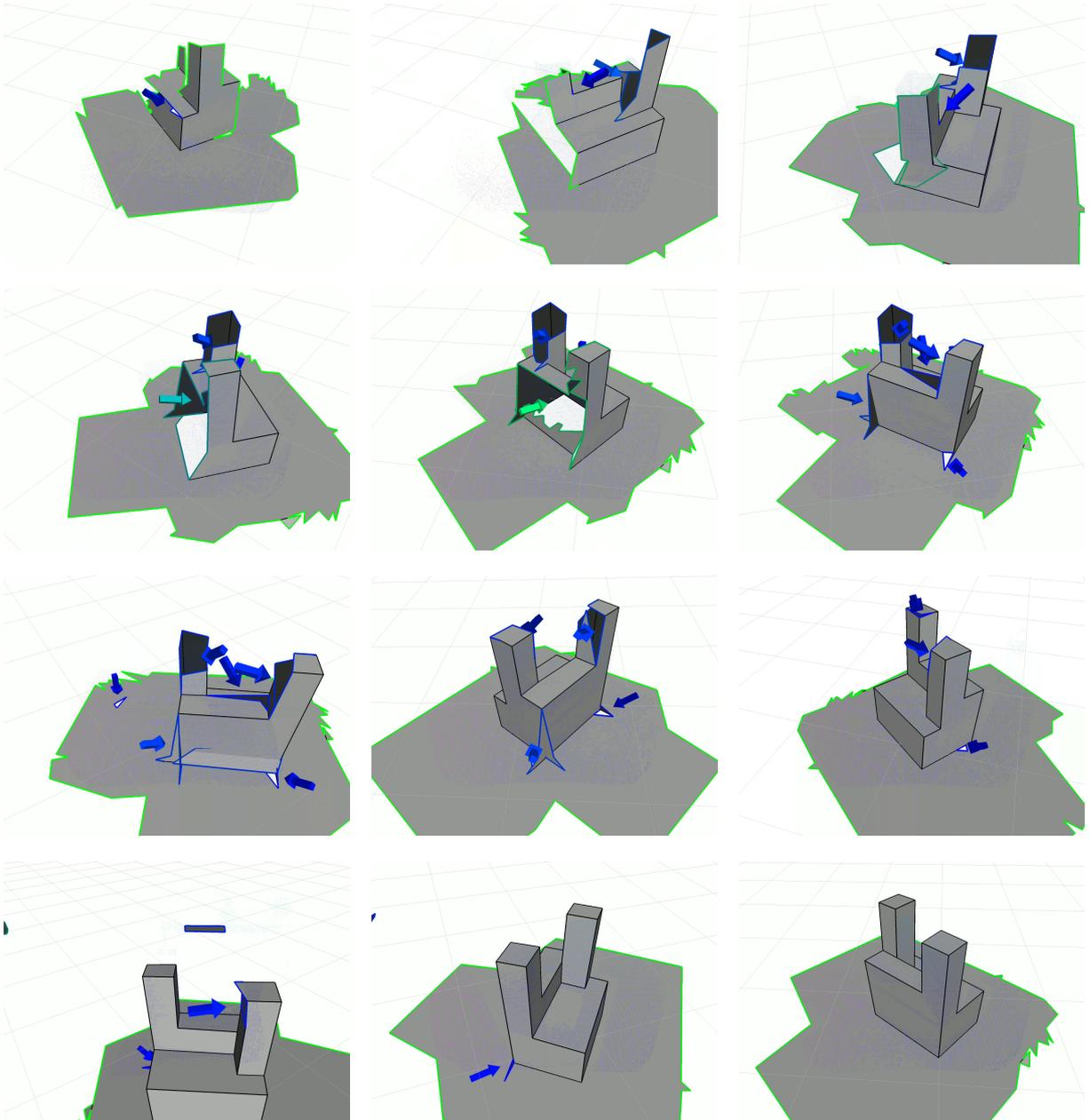
Abbildung 7.26: Eine Testszene (c) wurde mit einer Microsoft Kinect for XBOX rekonstruiert. Modell (a) ist das Resultat ohne, Modell (b) mit automatischem Schließen von Löchern. Die Detailansicht (d+e) zeigt, dass viele kleine Löcher geschlossen wurden.



(a)



(b)



(c)

Abbildung 7.27: Testszene (a) und mit dem Lenovo Phab 2 Pro rekonstruiertes Modell (b). Während der Erfassung wurden die Kisten umrundet, dabei dynamisch Nutzerhinweise erzeugt und angezeigt (c).

7.7 Zusammenfassung

Dieses Kapitel befasste sich mit der automatischen Vervollständigung von partiellen B-Reps und der Erzeugung von Nutzerhinweisen. Es wurde gezeigt, dass unterschiedliche Arten von Unvollständigkeiten in partiellen Modellen auftreten können, die auch unterschiedlich behandelt werden müssen. Einerseits wird das Modell durch eine automatisierte Vervollständigung mittels Schließen von Löchern vollständiger, andererseits unterstützt das Anzeigen von Nutzerhinweisen in der 3D-Live-Visualisierung den Nutzer, Löcher schon während der Erfassung zu beheben. Um die jeweils am besten passende Strategie auswählen zu können, wurden Unvollständigkeiten eines partiellen B-Reps in Form von Löchern formal definiert und Eigenschaften bestimmt, die es erlauben, die Löcher zu kategorisieren. Für innere Löcher bis Kardinalität $k = 3$, die kleiner als die Strukturgröße sind, ist ein automatisiertes Schließen sinnvoll. Alle weiteren Löcher werden durch Nutzerhinweise markiert. Dabei wurden 3D-Pfeile verwendet, die auf die entsprechende Stelle hinweisen und die bei inneren Löchern zudem einen Vorschlag für eine Pose geben, mit der das Loch geschlossen werden kann, wenn sie vom Nutzer eingenommen wird.

Das vorgeschlagene Verfahren wurde validiert, indem alle möglichen Konstellationen von Löchern mit Kardinalität $k \leq 3$ erzeugt wurden und auf einen sinnvollen Hinweis beziehungsweise auf einen korrekten Lochschluss getestet wurden.

Insgesamt wurde somit das beschriebene 3D-Scan-System um eine Komponente zur Vereinfachung des Aufnahmeprozesses erweitert, die den Nutzer unterstützt (Fragestellung F4). Anhand von Realdaten wurde gezeigt, dass das Modell weniger, insbesondere schlecht erkennbare, kleine Löcher enthält und dass die Nutzerhinweise einen guten Überblick über die verbleibenden Löcher geben. Die Abwesenheit von Pfeilen zeigt zudem an, dass das Modell vollständig ist.

Für zukünftige Arbeiten könnte zur Verbesserung des Verfahrens ein Kollisionstest durchgeführt werden, um Schnitte von Pfeilen mit anderen Pfeilen oder mit dem Modell zu verhindern. Zudem könnte mit einer Benutzerstudie die Zeitersparnis zur Erzeugung eines vollständigen Modells ermittelt werden.

Kapitel 8

Gesamtsystem

Inhalt

8.1	Prototypische Umsetzung	175
8.1.1	Microsoft Kinect for XBOX 360	175
8.1.2	IDS Ensensio N10	177
8.1.3	Lenovo Phab 2 Pro	178
8.2	Anwendungen	181
8.2.1	ENACT	181
8.2.2	SIMERO	181
8.3	Zusammenfassung	182

In diesem Kapitel sollen die zuvor beschriebenen Einzelteile eines 3D-Scan-Systems im Ganzen betrachtet werden und die prototypische Umsetzung dargestellt werden. Abbildung 8.1 gibt einen Überblick über die einzelnen Komponenten des Gesamtsystems. Die Rekonstruktion erzeugt aus einer organisierten Punktwolke ein partielles B-Rep. Mithilfe der Registrierung werden das partielle Modell und das Gesamtmodell verglichen und eine Pose ermittelt. Schlägt die Registrierung fehl, beispielsweise aufgrund fehlender Merkmalskorrespondenzen oder eines positiven Konflikttests, wird das partielle Modell verworfen. Bei erfolgreicher Registrierung wird das Modell mit dem Gesamtmodell fusioniert. Zur Vervollständigung werden kleine innere Löcher direkt geschlossen. Das Gesamtmodell wird während der Erfassung live visualisiert, dabei werden Löcher markiert und Nutzerhinweise zu den verbleibenden Löchern angezeigt. Nach dem Ende der Erfassung kann der Nutzer verbleibende große innere Löcher noch (mit Bestätigung) schließen lassen. Als Anwendungs-abhängiger Parameter muss der Anwender nur die Strukturgröße festlegen. Kameraparameter (Kameramatrix, Stärke der Messabweichungen) müssen für eine gegebene Kamera einmalig ermittelt werden. Ist die Pose der Eingabepunktwolke bereits bekannt, beispielsweise bei am Roboter montierten Kameras oder Geräten mit Positionssensorik, entfällt der Registrierungsschritt.

Der vorgestellte Ansatz wurde prototypisch für unterschiedliche Hardware realisiert. In Abschnitt 8.1 wird die Realisierung im Detail erläutert und Ergebnisse werden vorgestellt. In Abschnitt 8.2 werden kurz Anwendungen des 3D-Scan-Systems aus dem Bereich der Robotik genannt, in denen das System integriert wurde. Den Abschluss bildet Abschnitt 8.3 mit einer Zusammenfassung.

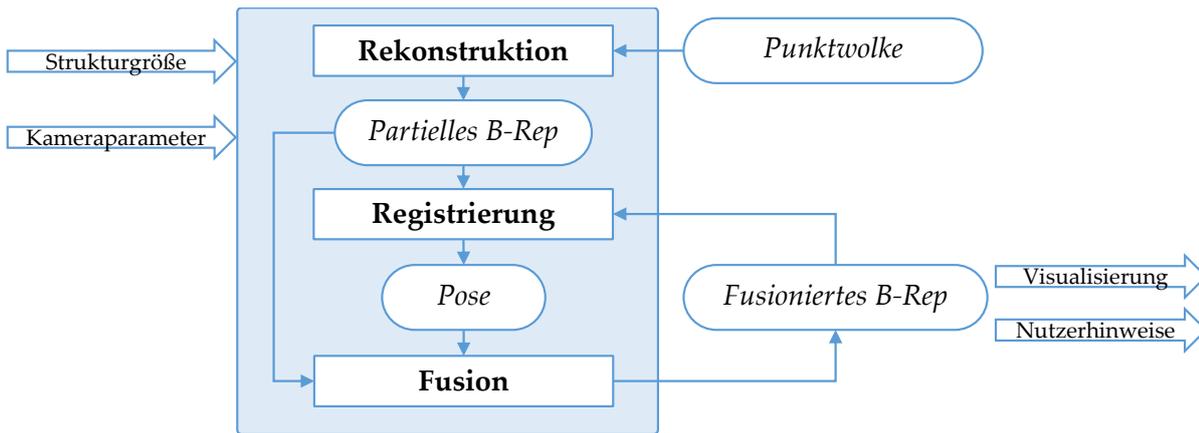


Abbildung 8.1: Übersicht über die Komponenten und Daten des entwickelten 3D-Scan-Systems. Neben den Rohdaten benötigt das System als Eingabe die vom Anwender zu wählende Strukturgröße, sowie Kameraparameter (intrinsische Kameramatrix, Konstante σ_1 zur Beschreibung von Messabweichungen). Die Kameraparameter müssen für eine verwendete Kamera nur einmal bestimmt werden. Die Ausgabe ist das fusionierte B-Rep, das live visualisiert und mit Nutzerhinweisen angereichert wird.

8.1 Prototypische Umsetzung

Das vorgestellte Rekonstruktionsverfahren wurde mit drei unterschiedlichen Tiefenkameras realisiert, einer Microsoft Kinect for XBOX 360, einer IDS Senso N10 und dem Lenovo Phab 2 Pro, einem Smartphone mit integrierter Tiefenkamera. In jedem der folgenden Unterkapitel werden die für diese Kamera verwendeten Parameter beziehungsweise Anpassungen erläutert. Zudem werden beispielhafte Rekonstruktionen gezeigt und die Benutzerfreundlichkeit des Systems dargestellt.

8.1.1 Microsoft Kinect for XBOX 360

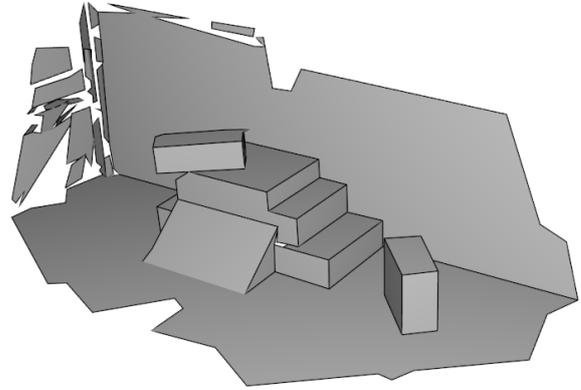
Die Microsoft Kinect for XBOX 360 ist eine Consumer-Tiefenkamera, die durch Projektion eines bekannten Infrarot-Punktmusters und Triangulation eine organisierte Punktwolke der Größe 640×480 erzeugt. Der Messbereich liegt bei 0.8 bis ca. 4 Meter, sodass sich die Kamera für Innenräume und nicht zu kleine Objekte eignet. Die Messunsicherheiten sind im Vergleich zu industriellen Kameras hoch. Als Konstante für das Fehlermodell (Definition 3.18) wurde wie in Holzer et al. der Wert $\sigma_1 = 0.0028$ gewählt [Holzer12].

Die Kamera muss zum einen über ein USB-Kabel mit einem Rechner verbunden werden, zum anderen ist ein separates Stromkabel nötig. Bei Verlängerung des USB-Kabels ist ein aktiver USB-Hub nötig, da sonst kein zuverlässiges Streaming der Punktwolken möglich ist.

Durch den Mindestabstand und die relativ großen Messabweichungen ist die Kinect für mittlere bis große Objekte geeignet. In Abbildung 8.2 sind mehrere Rekonstruktionen abgebildet. Dabei wurde eine Strukturgröße von 5 cm verwendet. Planare Oberflächen können dabei zuverlässig rekonstruiert werden. Nicht polygonale Oberflächen (Roboter in Abbildung 8.2c) können nur bedingt rekonstruiert werden, da sie im Rekonstruktionsprozess durch mehrere planare Stücke approximiert werden. Unterscheidet sich die planare Approximation weite- rer Ansichten zu stark, so ist eine Fusion dieser Teile nicht mehr möglich. Dies führt zur einer unvollständigen Rekonstruktion des Roboters (Abbildung 8.2d). Ein weiterer Nachteil



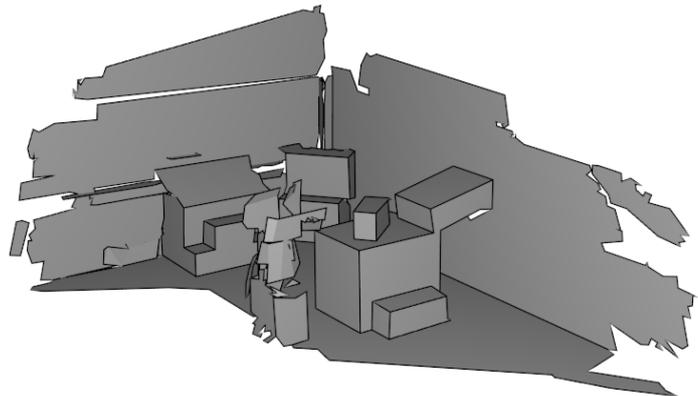
(a)



(b)



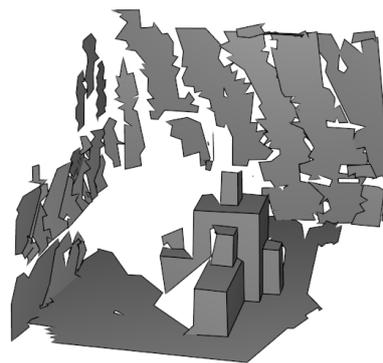
(c)



(d)



(e)



(f)

Abbildung 8.2: Mit der Kinect for XBOX 360 rekonstruierte Szenen (links) und das zugehörige B-Rep Modell (rechts)

der Kinect ist die Notwendigkeit eines Strom- und USB-Kabels, das nicht beliebig verlängert werden kann. In der Szene in Abbildung 8.2e wurde versucht einen kompletten Raum mit Hindernissen in der Mitte zu rekonstruieren. Diese konnten auf der Rückseite aufgrund der Einschränkungen durch die Kabel nicht erfasst werden. Die Wände des Raumes können ebenfalls nicht vollständig rekonstruiert werden, da keine Registrierung möglich ist, wenn nur eine Wand im Bild ist und demnach alle sichtbaren Flächen linear abhängig sind.

8.1.2 IDS Ensenso N10

Die IDS Ensenso N10 ist eine industrielle Tiefenkamera, die nach dem Stereo-Prinzip arbeitet. Die Erzeugung der organisierten Punktwolke wird zusätzlich durch Projektion eines Infrarot-Punktmusters unterstützt. Die Auflösung liegt bei 752×480 Pixeln bei einem Messbereich von 0.2 bis 0.9 Metern. Damit ist die Kamera eher für kleine bis mittelgroße Objekte geeignet. Die Messabweichungen sind verhältnismäßig gering, als Konstante für das Fehlermodell (Definition 3.18) wurde ein Wert von $\sigma_1 = 0.00417$ empirisch ermittelt ¹.

In Abbildung 8.3 ist die Rekonstruktion eines Spielzeugmodells mit einer Strukturgröße von 0.5 cm abgebildet. Eine fünfeckige Fläche wurde dabei untergelegt, um mehr linear unabhängige Flächen in einem Bild zu erhalten. Durch die höhere Auflösung, die geringeren Messabweichungen und den Messbereich sind wesentlich feinere Strukturen rekonstruierbar als mit der Kinect-Kamera. Auch hier sind die Einschränkungen der Registrierung spürbar. Aufgrund der Symmetrie konnte die Rückseite des Modells, aufgrund von ausschließlich linear abhängigen Flächen der obere Teil des Modells nicht rekonstruiert werden.

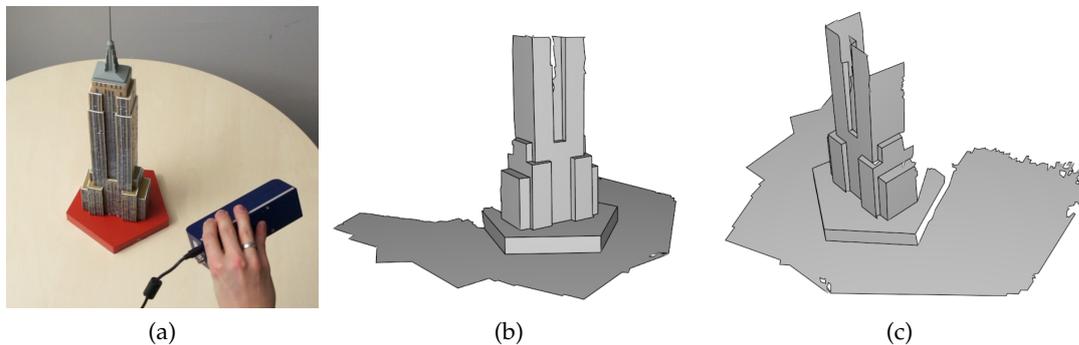


Abbildung 8.3: (a) Rekonstruktion eines Modells des Empire State Buildings mit einer handgehaltenen IDS Ensenso N10 Tiefenkamera, (b+c) rekonstruiertes Modell. Die Rekonstruktion der Rückseite ist nicht möglich, da aufgrund der Symmetrie die Registrierung fehlschlägt.

¹Dass der Wert von σ_1 bei der Ensenso-Kamera höher ist als bei der Kinect, darf nicht zu dem Schluss von höheren Messabweichungen verleiten, da noch ein quadratisches Fehlermodell angewendet wird ($\sigma_z = \sigma_1 \cdot z^2$), in das die Entfernung z eingeht. Die z -Werte unterscheiden sich aufgrund des abweichenden Messbereichs beider Kameras deutlich.

8.1.3 Lenovo Phab 2 Pro

Das Lenovo Phab 2 Pro (Abbildung 8.4a) ist ein Smartphone mit einer integrierten Tiefenkamera, das Googles Projekt-Tango Schnittstelle implementiert. Die Tango-Schnittstelle ist eine für das Android-Betriebssystem vereinheitlichte API, um auf dem Gerät Punktwolken und Positionsdaten verarbeiten zu können. Die Verwendung eines Mobilgeräts für die Rekonstruktion bietet klare Vorteile: Es sind weder Strom- noch Datenkabel nötig, die die Erfassung behindern können, und zudem ist direkt ein Bildschirm verfügbar, auf dem die Rekonstruktion und Nutzerrückmeldungen angezeigt werden können. Zudem besitzt das Smartphone Lagesensorik und ein softwareseitiges Framework zur Posenschätzung des Geräts. Nachteilig ist die geringere Rechenkapazität sowie die Unorganisiertheit und die geringe Auflösung der Punktwolke (zwischen 10 Tausend und 30 Tausend Punkte).

Um das in dieser Arbeit vorgestellte Verfahren auf dem Smartphone ausführen zu können, sind einige Anpassungen nötig, die im Folgenden beschrieben werden.

Die Punktwolke des Geräts ist unorganisiert, das heißt die Punkte liegen als ungeordnete Menge an 3D-Punkten vor. Um organisierte Punktwolken zu erhalten, wie sie in dieser Arbeit benötigt werden, werden die Punkte in einem Vorverarbeitungsschritt geordnet. Dazu wird die intrinsische Kameramatrix K aus dem Gerät ausgelesen und alle Punkte in ein Pixelgitter der Größe 224×172 projiziert. Fallen Punkte außerhalb des Gitters, werden sie verworfen. Punkte, die in ein Pixel fallen, das bereits einen 3D-Punkt enthält, werden ebenfalls verworfen. Dies tritt jedoch nur in sehr seltenen Fällen auf. In Abbildung 8.5 ist das Ergebnis visualisiert. Man erkennt, dass nicht in alle Pixel ein 3D-Punkt projiziert wurde. Durch Mehrfachausführung der Projektion von unterschiedlichsten Punktwolken zeigt sich, dass es stets die identischen Pixel sind, die nicht belegt sind. Die kreisförmige Anordnung um das Bildzentrum lässt Linseneffekte für diesen Umstand vermuten.

Diese Fehlstellen werden gefüllt, indem der z -Wert aus den belegten Pixeln der 8er-Nachbarschaft interpoliert wird. Anschließend wird mithilfe der inversen intrinsischen Kameramatrix ein Strahl durch das fehlende Pixel berechnet und so ein Punkt im Raum bestimmt, der auf dem Strahl liegt und den gemittelten z -Wert besitzt. Durch dieses Vorgehen ist sichergestellt, dass der Punkt auch dem 2D-3D-Zusammenhang, der durch die Kameramatrix beschrieben wird, genügt.

Da stets die selben Pixel nicht belegt sind, kann die Position der Fehlpixel vorab gespeichert werden, sodass keine Suche in der gesamten Punktwolke nötig ist. Dies wäre zudem falsch, denn je nach Szene können auch andere Pixel nicht belegt sein, wenn dort keine Messung zum Beispiel aufgrund von Reflexionen möglich war.



Abbildung 8.4: (a) Lenovo Phab 2 Pro mit integrierter rückwärtiger *Time-of-Flight*-Tiefenkamera. (b) Entwickelte Android-App zur Rekonstruktion. Der Screenshot zeigt ein partielles Modell während der Aufnahme der Sitzbank aus Abbildung 8.7. In grau ist das fusionierte Modell dargestellt, in grün ist das aktuelle Einzelmodell des letzten Frames darübergelegt.

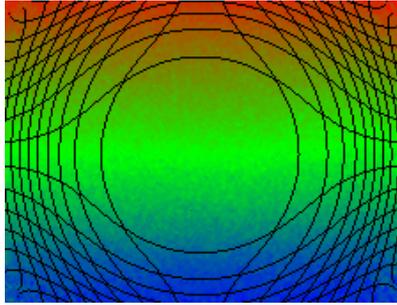
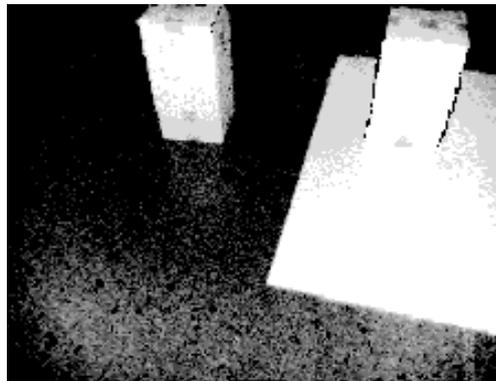


Abbildung 8.5: 2D-Darstellung einer Punktwolke des Lenovo Phab 2 Pro nach Projektion der Punkte in das 2D-Gitter der Größe 224×172 . An farbigen Stellen ist ein Punkt vorhanden, in schwarze Pixel wird nie ein Punkt projiziert.



(a)



(b)

Abbildung 8.6: (a) Erfasste Szene mit zwei hellen Kisten auf schwarzem und weißem Untergrund, (b) Punktwolke des Lenovo Phab 2 Pro. Schwarze Pixel enthalten keinen Punkt, die restlichen Pixel sind anhand der vom Gerät gelieferten Konfidenzwerte eingefärbt (je heller desto mehr konfident).

Das Lenovo Phab 2 Pro Smartphone verwendet einen *Time-of-Flight*-Sensor zur Messung der Tiefeninformationen. Aufgrund von Absorption und Reflexion hat die Farbe von Oberflächen dabei einen großen Einfluss auf die Messabweichungen. Je heller eine Oberfläche, desto kleiner die Messabweichungen. Schwarze Bereiche werden oft gar nicht erkannt. Die Software-Schnittstelle gibt dazu zu jedem 3D-Punkt einen Konfidenzwert im Bereich $[0, 1]$ an. Punkte mit niedrigerem Konfidenzwert sind stärker verrauscht als Punkte mit hohen Werten. In Abbildung 8.6 sind die Konfidenzwerte als Graustufen visualisiert, in schwarzen Pixeln liegt keine Messung vor. Man erkennt deutlich, dass der dunkle Untergrund zum großen Teil fehlt. Der Konfidenzwert eines Punktes kann bei der Rekonstruktion von partiellen B-Reps verwendet werden. An allen Stellen, an denen ein Mittel über Punkte gebildet wird, beispielsweise während des Regionenwachstums zur Bestimmung der Ebenenparameter einer Region, wird hier das mit den Konfidenzwerten gewichtete Mittel verwendet. Bei der Festlegung des Gewichtes einer Fläche, die für die Fusion wichtig ist, wird anstelle der Anzahl der Punkte eines Segments die Summe der Konfidenzwerte der Punkte verwendet. Sind alle Konfidenzwerte gleich Eins, so ist die Vorgehensweise identisch, ansonsten werden Flächen mit vielen unzuverlässigen Punkten weniger stark gewichtet.

Das Lenovo Phab 2 Pro Smartphone besitzt eine Fischauge-Kamera, die zusammen mit den Lagesensoren ausschließlich dazu genutzt wird, die Lage des Geräts im Raum zu bestimmen.

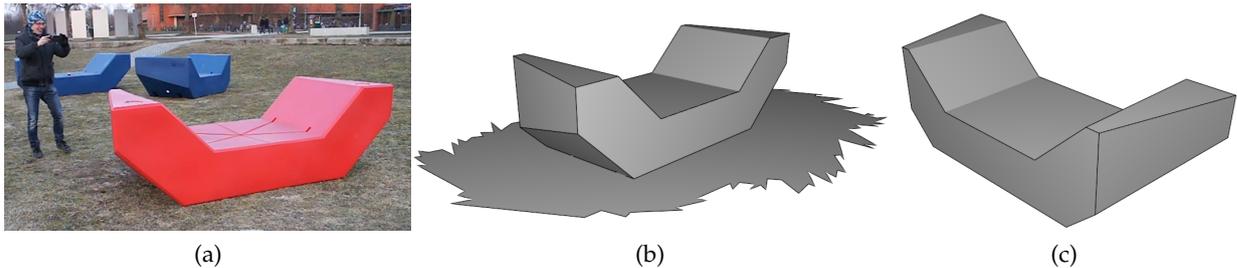


Abbildung 8.7: (a) Rekonstruktion einer Sitzbank mit dem Lenovo Phab 2 Pro, (b) Rekonstruiertes B-Rep-Modell, (c) Modell nach manueller Entfernung der Bodens und Schließen aller Löcher inklusive der nicht sichtbaren Bodenfläche

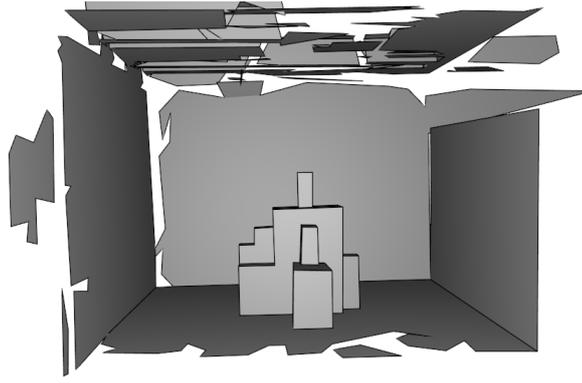
Die Registrierung von partiellen B-Reps wird deshalb für das Mobilgerät wie folgt angepasst: Zur Berechnung der Pose zwischen zwei B-Reps werden nicht die Merkmale und Deskriptoren aus Kapitel 6 verwendet, sondern die Pose, die das Gerät zur Verfügung stellt. Diese Pose wird dann genutzt, um Flächenkorrespondenzen zu ermitteln. Mithilfe der Flächenkorrespondenzen wird dann die Pose nochmals verfeinert, indem die Posenoptimierung aus Kapitel 6.2.6 angewandt wird. Dadurch wird erreicht, dass wegen der zusätzlich verfügbaren Sensorik Symmetrien und zu wenige geometrische Merkmale kein Problem mehr darstellen. Durch die Posenoptimierung wird die Pose dann noch verbessert, indem die Geometrie der beiden Modelle berücksichtigt wird.

Insgesamt wurden zwei Vorgehensweisen zur Rekonstruktion mit dem Lenovo Phab 2 Pro realisiert. Einerseits wurde das beschriebene Verfahren mithilfe des nativen Android-Development-Kit für das Android-Betriebssystem cross-kompiliert, sodass allein mit dem Smartphone rekonstruiert werden kann (Abbildung 8.4b). Andererseits wurde eine Remote-Variante realisiert, in der die Punktwolken des Geräts per kabelloser Netzwerkverbindung zu einem Desktop-Rechner gesendet werden, auf dem die Rekonstruktion ausgeführt wird. Anschließend werden die partiellen B-Reps zurückgeschickt und auf dem Mobilgerät angezeigt. Durch die geringere Rechenkapazität des Mobilgeräts kann auf dem Gerät nur eine Framerate von ca. 1 bis 1.5 Rekonstruktionen pro Sekunde erreicht werden, während bei der Netzwerkvariante trotz des zusätzlichen Sendeaufwandes circa 6 Rekonstruktionen pro Sekunde möglich sind.

In Abbildung 8.7 ist die Rekonstruktion einer Sitzbank mit einer Strukturgröße von 8 cm abgebildet. Durch die freie Beweglichkeit und direkte Rückmeldung über das Display ist eine vollständige Rekonstruktion leicht möglich. Zum Vergleich mit der Kinect wird der Innenraum aus Abbildung 8.2e nochmals mit dem Lenovo Phab 2 Pro aufgenommen. Da die Einschränkungen der Registrierung aufgrund der Positionssensorik nun wegfallen und zudem kein Kabel nötig ist, kann der gesamte Innenraum nun rekonstruiert werden (Abbildung 8.8).



(a)



(b)

Abbildung 8.8: (a) Rekonstruktion des Raumes mit mehreren gestapelten Kisten (vergleiche Abbildung 8.2e), (b) mit dem Lenovo Phab 2 Pro rekonstruiertes B-Rep Modell

8.2 Anwendungen

In diesem Abschnitt werden Anwendungen aus dem Bereich der Robotik vorgestellt, in denen das vorgestellte System integriert wurde.

8.2.1 ENACT

Das Framework ENACT ist ein am Lehrstuhl des Autors entwickeltes *entity-actor* Framework zur Modellierung des Weltzustandes eines Roboters. Der Fokus liegt dabei auf einer effizienten Speicherung von Daten und einer einfachen Zusammenarbeit von unterschiedlichen, getrennten Softwarekomponenten, wie beispielsweise Perzeption, Aufgabenplanung, Bahnplanung und Greifplanung. Objekte (z.B. zu manipulierende Werkstücke) mit all ihren Eigenschaften werden dabei als Entitäten modelliert, Aktoren sind Software-Komponenten, die Entitäten modifizieren. Das hier vorgestellte Rekonstruktionssystem kann als Aktor in das Robotersystem integriert werden, um Objekte als B-Rep Modelle zu erzeugen und diese dem Gesamtsystem zur Verfügung zu stellen. Ausführlichere Informationen dazu sind in [Werner16] zu finden.

8.2.2 SIMERO

Das Projekt SIMERO befasst sich mit Methoden für eine sichere Mensch-Roboter Koexistenz und Kooperation. Durch ein Multikamerasystem wird der Arbeitsraum eines stationären Industrieroboters überwacht und in Echtzeit die Position eines Menschen im Arbeitsraum verfolgt. Dies ermöglicht eine adaptive Bahnplanung des Roboters unter Berücksichtigung des Menschen. Für das System müssen dabei alle statischen Hindernisse im Arbeitsraum bekannt sein, um diese einerseits von dynamischen Hindernissen (wie zum Beispiel Menschen) zu unterscheiden und andererseits um Aussagen über Verdeckungen treffen zu können. Je nach Einsatzgebiet können sich die statischen Hindernisse (Tische, Förderbänder, Regale, etc.) bei wechselnden Aufgaben von Zeit zu Zeit ändern.

Das in dieser Arbeit vorgestellte Scan-System ermöglicht hier eine einfache Erfassung der statischen Hindernisse. In dem in [Werner18] vorgestellten System werden dazu mit dem Lenovo Phab 2 Pro die statischen Hindernisse als B-Rep rekonstruiert. Anschließend wird das Modell auf den Roboter ausgerichtet und in das SIMERO-Überwachungssystem geladen. Nun kann das Überwachungssystem dynamische Hindernisse (Menschen) sicher erfassen (Abbildung 8.9).

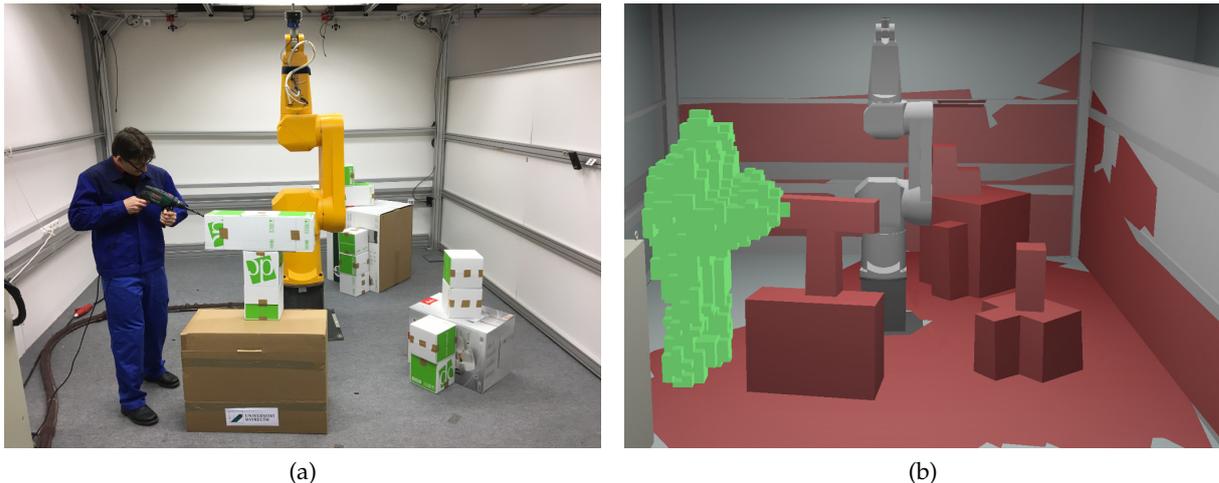


Abbildung 8.9: (a) In einer Arbeitszelle mit stationärem Industrieroboter und statischen Hindernissen interagieren Mensch und Maschine gleichzeitig. (b) Das Multikamerasystem verwendet die mit dem hier vorgestellten System rekonstruierten statischen Hindernisse (rot), um den Menschen (grün) sicher erkennen zu können.

8.3 Zusammenfassung

In diesem Kapitel wurde die Realisierung des vorgestellten 3D-Scan-Systems beschrieben. Die prototypische Umsetzung unterstützt dabei drei unterschiedliche Kameras als Eingabegerät. Die Sensoren unterscheiden sich deutlich hinsichtlich Messgenauigkeit, Messbereich und Auflösung. Für alle drei konnten Szenen und Objekte gut rekonstruiert werden, jedoch wurden unterschiedliche Vor- und Nachteile der einzelnen Varianten deutlich: Kabelgebundene handgehaltene Kameras schränken insbesondere bei der Erfassung von Innenräumen ein, sodass nicht alle Stellen erfasst werden können. Eine beliebige Verlängerung des Kabels ist in der Praxis oft nicht möglich. Die bei der Registrierung erläuterten Probleme von symmetrischen Objekten und linear abhängigen Flächen können auch hier beobachtet werden. Dies äußert sich darin, dass symmetrische Objekte nur zur Hälfte gescannt werden können oder die Wände eines Innenraums nur dann rekonstruiert werden können, wenn eine Raumecke sichtbar ist. Dies ist aber aufgrund des Messbereichs oder Verdeckungen nicht immer möglich. Die Kombination aus Sensor, Bildschirm und Computer im Lenovo Phab 2 Pro Smartphone besitzt diese Nachteile nicht, da hier die Positionssensorik des Geräts ausgenutzt werden kann. Insgesamt eignet sich das Smartphone sehr gut für das 3D-Scan-System, da keine weiteren Geräte oder Kabel nötig sind und die Visualisierung inklusive Nutzerrückmeldungen direkt auf dem Bildschirm angezeigt werden können. Aufgrund der geringen Auflösung ist das Gerät allerdings nicht für kleine Objekte oder feine Details geeignet. Anhand zweier ausgewählten Anwendungen wurde zudem gezeigt, wie das vorgestellte System im Anwendungsgebiet der Robotik eingesetzt und integriert werden kann.

Kapitel 9

Fazit

Inhalt

9.1 Zusammenfassung	184
9.2 Ausblick	187

In diesem Kapitel wird die gesamte Arbeit zuerst kurz zusammengefasst (Abschnitt 9.1). Dabei werden nochmals die wissenschaftlichen Fragestellungen aus Kapitel 1.4.2 betrachtet und diskutiert. Den zweiten Teil bildet ein Ausblick auf potentielle zukünftige Arbeiten für weitere Schritte in Richtung der Vision eines intelligenten 3D-Scanners (Abschnitt 9.2).

9.1 Zusammenfassung

In vielen Anwendungsbereichen, wie der Computergrafik, der Robotik oder beim computer-gestützten Entwerfen spielen 3D-Modelle eine große Rolle. Die sensorgestützte Erzeugung von 3D-Modellen ist dabei eine einfache Möglichkeit, Modelle von realen Objekten oder Räumen zu erstellen. Im Gegensatz zur manuellen Erzeugung ist dabei kein Expertenwissen im Umgang mit 3D-Grafikprogrammen vonnöten. Um manuelle Modellierung so weit wie möglich zu reduzieren, ist die Vision, einen allumfassenden intelligenten 3D-Scanner zu besitzen, der alle benötigten Informationen schnell und auf einfache Art und Weise erfasst.

Um dieser Vision einen Schritt näher zu kommen, wurden die Grenzen bestehender Systeme untersucht. Dabei wurde eine Zweiteilung deutlich: Auf der einen Seite existieren Verfahren mit Fokus auf Erfassung und Verarbeitung, die einfach, robust und in Echtzeit 3D-Modelle mit handgehaltenen Kameras rekonstruieren können. Sie besitzen allerdings ein Defizit beim Informationsgehalt im Ausgabemodell, da höchstens Oberflächenmodelle in Form von Dreiecksnetzen oder eine Volumendarstellungen in Form von diskretisierten Voxelräumen erstellt werden. Auf der anderen Seite stehen Verfahren, die viele und hochwertige Informationen extrahieren können und so ein kontinuierliches Volumenmodell erzeugen, aber deutliche Nachteile in der Erfassung und Verarbeitung besitzen, da globale und wenig verrauschte Rohdaten als Eingabe erwartet werden.

In dieser Arbeit wurde ein neuartiger Ansatz präsentiert, der diese Lücke erstmals überwindet und hochwertige Ausgabemodelle mit einer online-fähigen, einfachen Erfassung verbindet. Dadurch können *Boundary-Representation* (B-Rep) Modelle mit einer handgehaltenen Tiefenkamera inkrementell erzeugt werden. Das Vorhandensein von partiellen Modellen zu jedem

Zeitpunkt während der Aufnahme ermöglicht zudem eine parallele Nutzung der geometrischen Modelle für andere Anwendungen (zum Beispiel die Greifplanung eines Roboters) oder für das Scan-System selbst, indem die Modelle für die Registrierung oder für die Erzeugung von Nutzerrückmeldungen verwendet werden. Die Ausgabe beschränkt sich dabei auf planare Modelle.

Die eingangs genannten wissenschaftlichen Fragestellungen (Kapitel 1.4.2) werden nun nochmals beleuchtet und beantwortet. In der ersten Fragestellung wurde die Repräsentation von unvollständigen Modellen adressiert, die für eine inkrementelle Vorgehensweise nötig sind:

F1 In welcher Form können Modelle repräsentiert werden, die noch kein vollständiges Volumen umschließen, aber dennoch die gleichen hochwertigen Eigenschaften besitzen wie kontinuierliche Volumenmodelle?

Dazu wurden in dieser Arbeit partielle B-Rep Modelle vorgestellt und definiert (Kapitel 3.2). Die Verwendung der *half-edge* Datenstruktur ermöglicht durch den Verzicht auf ein zwingendes Vorhandensein von Zwillingskanten die Modellierung von unvollständigen Modellen. Ansonsten besitzen die partiellen Modelle dieselben Eigenschaften wie vollständige B-Rep Modelle.

Die zweite Fragestellung beschäftigte sich anschließend mit der Rekonstruktion bei bekannter Sensorpose:

F2 Inwieweit ist es bei bekannter Sensorpose möglich, inkrementell und online solche Modelle zu erzeugen?

Die inkrementelle und online-fähige Erzeugung wird erreicht, indem eine verschränkte und modellbasierte Kombination von SLAM und DSR eingesetzt wurde (Kapitel 3.1). Das heißt, dass ein globales Modell in Form eines partiellen B-Rep Modells verwendet wird. Aus einer einzelnen Punktwolke wird dazu direkt ein partielles B-Rep rekonstruiert, das anschließend in das globale Modell fusioniert wird. Durch die direkte Abstraktion der Rohdaten auf höherwertige geometrische Informationen wird die Datenmenge zudem stark verringert, sodass Speicher- und Laufzeitaufwand handhabbar sind und eine Online-Ausführung möglich ist. Die Rekonstruktion (Kapitel 4) basiert dabei auf dem Segmentierungsansatz von Holz et al. [Holz14], der um ein Konzept zur tiefenkorrekten Filterung der Segmentierung erweitert wird. Anschließend werden im Polygonalisierungsschritt durch Konturverfolgung und Kantenanpassung 3D-Polygone erzeugt, die abschließend zur einem partiellen B-Rep zusammengesetzt werden. Insgesamt wird dabei die organisierte Struktur der Punktwolke ausgenutzt, um die Bestimmung von Nachbarschaftsbeziehungen zwischen Punkten und Segmenten zu beschleunigen.

Bei der Fusion (Kapitel 5) wird das Problem einer Vervollständigung und Verbesserung des Modells bei gleichzeitiger Konsistenthaltung der B-Rep-Datenstruktur gelöst. Durch eine flächenweise Verschmelzung kann das Problem auf den 2D-Raum reduziert werden, sodass die Anwendung eines Plane-Sweep-Algorithmus möglich ist, der die Datenstruktur bis auf Zwillingskantenbeziehungen konsistent hält. Diese werden anschließend in einem Kantenverknüpfungsschritt wieder korrigiert. Durch die Verwendung von Flächengewichten und einer gewichteten Mittelung wird dabei sichergestellt, dass das resultierende Modell in Bereichen der Überlappung genauer und in den anderen Bereichen vollständiger ist als die Ausgangsmodelle.

Die durchgeführte Evaluation zeigte dabei die Leistungsfähigkeit und die Grenzen des Ansatzes: Die Laufzeit weist eine quadratische Abhängigkeit von der Breite der organisierten Eingabepunktwolke auf, was bei den drei verwendeten Sensoren zu einer ungefähren Rekonstruktionsrate von 1 Hz (Lenovo Phab 2 Pro mit Rekonstruktion auf dem Gerät) über 2 Hz

(Microsoft Kinect) bis 6 Hz (Lenovo Phab 2 Pro mit Rekonstruktion auf Desktop-Rechner) führt. Eine Online-Ausführung ist somit möglich. Die Güte des Modells ist abhängig von der Größe der Messunsicherheiten, durch die die Wahl der Strukturgröße nach unten beschränkt wird. Bei den verwendeten Sensoren ist bei der IDS Ensensio die Rekonstruktion mit einer Strukturgröße im Bereich von 5 mm möglich, bei der Microsoft Kinect und dem Lenovo Phab 2 Pro liegt die untere Grenze im Bereich 30-50 mm. Höheres Rauschen führt einerseits zu Abweichungen in den geometrischen Elementen (z.B. Ebenengleichungen), die aber durch die Fusion gut herausgemittelt werden können. Andererseits fehlen häufiger Ecken und Kanten in den Einzelmodellen, die durch mehr Einzelaufnahmen ausgeglichen werden müssen, um ein vollständiges Modell zu erhalten. Da bei der Fusion keine topologischen Änderungen außer der Verschmelzung von Flächen mehr durchgeführt werden, ist eine Limitierung des Ansatzes, dass ein topologischer Fehler in einer Einzelrekonstruktion in das fusionierte Modell übertragen wird und dort auch nicht mehr korrigiert wird.

Durch die Einschränkung auf planare B-Reps werden gekrümmte Oberflächen durch planare Stücke angenähert. Ist diese Approximation in den Einzelmodellen unterschiedlich, so kann eine Fusion dieser Teile fehlschlagen, da die Topologie zu unterschiedlich ist. Dies kann dazu führen, dass gekrümmte Oberflächen nicht vollständig rekonstruiert werden können.

Bisher wurde angenommen, dass die Aufnahmepose bekannt ist. Dies ist beispielsweise bei Geräten mit integrierter Positionssensorik der Fall oder bei Kameras, die an einem Roboterarm montiert sind. Bei handgehaltenen Kameras muss die Pose jedoch erst bestimmt werden, was zur dritten Fragestellung führt:

F3 Inwieweit kann die Pose automatisch bestimmt werden, sodass ein *Simultaneous-Localization-And-Mapping* (SLAM)-System realisiert wird?

Es wurde gezeigt, dass eine Registrierung auf Basis der partiellen B-Rep Modelle möglich ist. Dazu wurde in merkmalsbasierter Ansatz vorgestellt (Kapitel 6). Das Problem von verschiedenen Topologien in beiden Modellen wird durch eine allgemeine Definition von Merkmalen, bestehend aus drei Flächen, gelöst, die keine Adjazenzen voraussetzen, diese aber trotzdem ausnutzen, falls sie vorhanden sind. Da partielle Modelle oft wenige aber dafür sehr informationsreiche Merkmale aufweisen, genügt eine einzige Korrespondenz an Merkmalen zur Erzeugung einer Posenhypothese. Durch ein Bewertungsverfahren werden falsche Hypothesen aussortiert und die beste Hypothese ermittelt.

Insgesamt wird so ein SLAM-System realisiert, das gleichzeitig die Pose schätzt und ein B-Rep Modell rekonstruiert. Die Evaluation zeigte, dass in mehr als 90% der Fälle eine korrekte Pose ermittelt werden kann. Das Verwerfen von falschen Hypothesen mittels eines Konflikttests weist dabei eine hohe Trefferquote (83-98 %) bei einer Genauigkeit von 55-84% auf. Dieses Verhalten eignet sich somit für die Verarbeitung eines Stroms an Punktwolken, bei dem das Auslassen von Frames wesentlich weniger schwerwiegend ist als eine Fusion mit einer falschen Pose, die das globale Modell verfälscht. Die Grenzen des vorgestellten Registrierungsverfahrens liegen einerseits in der Notwendigkeit von drei linear unabhängigen Flächen zur Erzeugung von Merkmalen. Sind alle in einem Einzelbild sichtbaren Flächen linear abhängig, so kann das Modell nicht registriert werden. Andererseits besitzen symmetrische Objekte oder Szenen mit symmetrischen Teilen die identischen Merkmale, so dass hier die Registrierung falsche Posen ergeben kann. Beide Einschränkungen resultieren aus der Tatsache, dass nur die reine Geometrie betrachtet wird, anhand der eine Entscheidung ohne zusätzliches Wissen, wie beispielsweise Farbe, prinzipiell nicht möglich ist.

Als letztes wurde die Frage betrachtet, wie eine einfache Handhabung des Systems ermöglicht wird:

F4 Inwieweit und in welcher Weise kann das System den Anwender bei der Erfassung unterstützen?

Dazu wurden in dieser Arbeit drei Ansätze vorgestellt: die einfache Parametrisierung des Gesamtsystems durch die Strukturgröße als Universalparameter, eine automatische Vervollständigung von Modellen und Nutzerhinweise als Einblendung in die Live-Visualisierung. Durch den Bezug aller algorithmischen Parameter auf die Strukturgröße (Kapitel 3.3) wird erreicht, dass der Nutzer nur noch einen Parameter angeben muss. Für die Strukturgröße als Schwelle für zu rekonstruierende geometrische Merkmale ist zudem eine intuitive Interpretation vorhanden, die auch Personen ohne Kenntnis der Funktionsweise des Systems zugänglich ist. Hardware-abhängige Parameter, wie die Kameramatrix, müssen nur einmalig bestimmt werden, da sie für einen Sensor konstant bleiben.

Um dem Nutzer die Erstellung eines vollständigen Modells zu erleichtern, wurde das Konzept von Löchern in partiellen Modellen eingeführt (Kapitel 7). Während der Rekonstruktion kann durch Bestimmung von spezifischen Eigenschaften die passende Strategie für die Behandlung von Unvollständigkeiten gewählt werden. Einerseits werden kleine Löcher automatisiert geschlossen, andererseits werden Nutzerhinweise in Form von 3D-Pfeilen in die Live-Visualisierung eingeblendet, durch die der Nutzer auf Löcher im Modell hingewiesen wird. So wird eine Unterbrechung der Erfassung zur Inspektion des Modells vermieden. Da keine Kollisionstest ausgeführt werden, ist es möglich, dass sich Pfeile mit dem Modell oder mit anderen Pfeilen schneiden.

Das beschriebene 3D-Scan-System wurde zudem mit drei verschiedenen Sensoren prototypisch umgesetzt (Kapitel 8.1). Dabei wurde deutlich, dass bei der Erfassung von Innenräumen ein Gerät, das Sensor, Bildschirm und Rechner in Einem vereint, deutlich angenehmer zu handhaben ist. Aufgrund der geringen Auflösung eignen sich solche heute verfügbaren Geräte allerdings (noch) nicht für die Rekonstruktion von kleinen Objekten. Die Integration des vorgestellten Verfahrens in die Robotiksysteme ENACT und SIMERO zeigte zudem zwei exemplarische Anwendungsfälle auf (Kapitel 8.2).

9.2 Ausblick

Zum Abschluss wird nun ein Ausblick auf potentielle zukünftige Arbeiten gegeben. Zunächst werden mögliche Ansatzpunkte beschrieben, die das vorgestellte System direkt betreffen und dieses verbessern oder dessen Einschränkungen reduzieren.

Ist keine externe Poseninformation verfügbar, so ist eine Limitierung des Registrierungsansatzes die Notwendigkeit von linear unabhängigen Flächen und die Behandlung von Symmetrien. Eine möglicher Ansatzpunkt wäre hier die ergänzende Verwendung von Farbinformationen, die viele Tiefenkameras zusätzlich zur Verfügung stellen. Sind ausschließlich linear abhängige Flächen sichtbar, so können Farbmerkmale zur Fixierung der restlichen Freiheitsgrade bei der Bestimmung einer Pose genutzt werden. Bei symmetrischen Objekten, bei denen mehr als eine valide Posenhypothese erzeugt wird, kann anhand der Farbmerkmale eine Entscheidung getroffen werden. Grundsätzlich bietet die Verwendung von Farbinformationen zudem die Möglichkeit, texturierte B-Rep Modelle zu erzeugen, die somit noch mehr Informationsgehalt als die hier verwendeten Modelle besitzen.

Die Beschränkung auf planare Flächen sorgt dafür, dass gekrümmte Oberflächen durch planare Stücke approximiert werden. Eine vollständige Loch-freie Rekonstruktion gekrümmter Oberflächen ist dabei meist nicht möglich, da die Approximation nicht eindeutig ist und zu unterschiedlichen Topologien in den Einzelmodellen führt, bei denen eine Fusion nicht immer möglich ist. Ein Ansatzpunkt ist hier, eine Strategie für die Fusion sich ändernder Topologien

zu entwickeln. Dies vermeidet zudem auch den Fehlerfall, dass einzelne, topologisch falsche Einzelrekonstruktionen einen großen Einfluss auf das Endmodell besitzen. Zudem wäre dies auch ein Schritt in die Richtung, mit dynamischen Szenen umzugehen, da sich bewegende Objekte meist auch eine Änderung der Topologie zur Folge haben. Der zweite Ansatzpunkt ist eine grundsätzliche Erweiterung des Vorgehens auf Modelle mit höherem Oberflächengrad, um gekrümmte Oberflächen abbilden zu können. Dabei sind geometrische Primitive, wie Kugeln oder Zylinder, als Flächenelemente denkbar oder auch allgemeinere Repräsentationen wie Spline- oder NURBS-Oberflächen.

Anhand der Laufzeitanalyse ist erkennbar, dass die meiste Zeit für Operationen verwendet wird, die direkt auf Pixelebene arbeiten. Hier ist eine Implementierung der Segmentierung auf der GPU denkbar, um die Rekonstruktion zu beschleunigen.

Im Bereich der intuitiven Benutzbarkeit des Systems wurden in dieser Arbeit Nutzerhinweise in Form von 3D-Pfeilen vorgeschlagen. Durch Kollisionstests könnte vermieden werden, dass sich ein Pfeil mit einem anderen oder dem Objekt schneidet. Zudem ist denkbar, die Anzahl der Nutzerhinweise einzuschränken, indem beispielsweise nur auf das Loch, das am nächsten zum Nutzer positioniert ist, hingewiesen wird. Der Weg des Nutzers könnte so gelenkt werden. Grundsätzlich ist hier eine Benutzerstudie hilfreich, um herauszufinden, wie viel die Nutzerhinweise einem Anwender helfen und welche Verbesserungen überhaupt sinnvoll sind.

Mit der Vision eines intelligenten 3D-Scanners wird im Folgenden ein allgemeinerer Ausblick gegeben. In dieser Arbeit wurde erstmals ein System präsentiert, das die Lücke zwischen bestehenden Systemen aus dem SLAM- und DSR-Bereich überwindet und inkrementell und online kontinuierliche Volumenmodelle in Form von planaren B-Reps erzeugt. Einzelne Systeme aus einem der beiden Bereiche sind dennoch aufgrund der Spezifität in ihrer Domäne leistungsfähiger. Der logisch nächste Schritt in Richtung der Vision ist deshalb, diese Fähigkeiten auch hier im vorliegenden System zu beherrschen. Im Bereich der SLAM-Systeme sind dies insbesondere eine hohe Framerate (20-60 Bilder pro Sekunde) und der Umgang mit dynamischen Szenen. Im Bereich der DSR können weitere geometrische Merkmale im Modell erkannt werden wie Abrundungen, Parallelitäten oder Symmetrien.

Um noch informationsreichere Modelle zu erhalten, können wiederkehrende Teilmodelle identifiziert werden, sodass ein Baugruppenmodell entsteht. Die automatische Erkennung von Bezeichnungen, Funktionen oder anderem zusätzlichen Wissen ist anschließend ein Schritt in Richtung semantischer Modelle.

Abbildungsverzeichnis

1.1	Eigenschaften eines 3D-Scanners	12
1.2	Ausgabespezifische Eigenschaften eines 3D-Scan-Systems	13
1.3	Einordnung verschiedener Arten von 3D-Modellen	14
1.4	Verarbeitungsspezifische Eigenschaften eines 3D-Scan-Systems	15
1.5	Sensorspezifische Eigenschaften eines 3D-Scan-Systems	16
1.6	Aufnahmespezifische Eigenschaften eines 3D-Scan-Systems	17
1.7	Schematische Darstellung der Leistungsfähigkeit von 3D-Scan-Systemen	19
2.1	Arten von Segmentierungsmethoden	34
3.1	Schematische Darstellung der vier Kombinationsmöglichkeiten von SLAM-Systemen mit hochwertigen kontinuierlichen Volumenmodellen als Ausgabe	39
3.2	CSG-Datenstruktur	42
3.3	Repräsentationen von 3D-Modellen	43
3.4	Schematische Darstellung der <i>half-edge</i> und <i>winged-edge</i> Datenstruktur	44
3.5	3D-Modell in <i>half-edge</i> Repräsentation	45
4.1	Beispiele verschiedener Hüllen	61
4.2	Arten von Konturverfolgungsalgorithmen	61
4.3	Eingabepunktwolke zur Veranschaulichung der Rekonstruktionsschritte	64
4.4	Verschiedene Arten von Vernetzungen aus [Holz14]	65
4.5	Adaptives Netz der Punktwolke aus Abbildung 4.3	66
4.6	Veranschaulichung des Regionenwachstums	67
4.7	Segmentierung mit erweitertem Regionenwachstum	68
4.8	Beispiel tiefenkorrekter, strukturgrößenabhängiger Filtermasken	70
4.9	Rangordnungsfiler	71
4.10	Morphologische Filterung	72
4.11	Eigenschaften von Polygonen	72
4.12	Verschiedene Nachbarschaften bei der Konturverfolgung	74
4.13	Erzeugung eines 3D-Polygons anhand der Kontur	76
4.14	Binärbilder zur Konturverfolgung	77
4.15	Erzeugung von 3D-Polygonen	77
4.16	Ergebnis der Polygonalisierung	78
4.17	Ergebnis der Anpassung von inzidenten Kanten	79
4.18	Anpassung von inzidenten Kanten	79
4.19	Anpassung von pseudo-inzidenten Kanten	81
4.20	Ergebnis der Anpassung von pseudo-inzidenten Kanten	82
4.21	Ergebnis der Anpassung virtueller Kanten	82
4.22	Zwei Ansichten des erzeugten partiellen B-Rep Modells	83
4.23	Optimierung der Ecken eines B-Reps	84
4.24	Datensätze zur Evaluation	87

4.25	Beispiel für die statistische Auswertung einer Rekonstruktion	88
4.26	Flächenerkennungsrate des Ashtray-Datensatzes	90
4.27	Flächenerkennungsrate bei $\sigma = 0.001$ in Abhängigkeit der Strukturgröße	91
4.28	Abhängigkeit der Laufzeit von der Auflösung und der Strukturgröße	93
4.29	Verteilung der Laufzeit auf die einzelnen Rekonstruktionsschritte	94
5.1	Flächenkorrespondenzen zwischen zwei B-Reps	97
5.2	Berechnung einer gewichteten mittleren Ebene	100
5.3	Abstand zwischen zwei Flächen	101
5.4	2D-Flächen-Vereinigung	102
5.5	Beispiele von Zwillingskantenbeziehungen nach der 2D-Flächenvereinigung	104
5.6	Korrektur der Zwillingskantenbeziehungen (1)	105
5.7	Korrektur der Zwillingskantenbeziehungen (2)	107
5.8	B-Reps nach der Fusion aller partiellen B-Reps	110
5.9	Kanten- und Eckenerkennungsrate des <i>anvil</i> -Datensatzes in Abhängigkeit des Levels an Rauschen	111
5.10	Resultierende Modelle des <i>anvil</i> -Datensatzes nach der Fusion aller 11 Einzelmodelle bei unterschiedlichem Rauschen	112
5.11	Kanten- und Eckenerkennungsrate im <i>anvil</i> -Datensatzes in Abhängigkeit der Anzahl an fusionierten Einzelmodellen	114
5.12	Resultierende Modelle des <i>anvil</i> -Datensatzes nach der Fusion einer unterschiedlichen Anzahl an stark verrauschten Einzelmodellen	114
5.13	Fehlerfall bei der Fusion durch falsche Topologie	115
6.1	Registrierung von partiellen B-Reps	119
6.2	Vorgehensweise zur Registrierung zweier B-Reps	122
6.3	Verschiedene Arten von Kanten	124
6.4	Mehrdeutigkeit bei der Bestimmung des Innenwinkels	125
6.5	Beispiele falscher Posenhypothesen	129
6.6	Beispiel unterschiedlicher Gütemaße bei der Registrierung	130
6.7	Veranschaulichung des Konflikttests	132
6.8	Zur Evaluation verwendeter Roboterarm	135
6.9	Die vier zur Evaluation der Registrierung verwendeten Test-Datensätze	136
6.10	Häufigkeit einer bestimmten Anzahl an Hypothesen und Berechnungszeit des approximativen relativen Gütemaßes	138
6.11	Quantil-Verteilung des Fehlers δ_{error} und t_{error}	140
6.12	Drehsymmetrisches Testobjekt <i>BigBen</i>	141
6.13	Quantil-Verteilung des Fehlers δ_{error} und t_{error} für den <i>BigBen</i> -Datensatz	142
7.1	Beispiel eines Loches nach Definition 7.1	147
7.2	Beispiele für Löcher mit unterschiedlichen Kardinalitäten k	148
7.3	Beispiele unterschiedlicher Orientierung von Löchern mit Kardinalität $k = 1$	148
7.4	Beispiele von Löchern mit Kardinalität $k = 2$	150
7.5	Beispiele von Löchern mit Kardinalität $k = 3$	151
7.6	Veranschaulichung von Transitionshalbkanten	152
7.7	Veranschaulichung der verwendeten Bezeichner bei Löchern mit $k = 2$	156
7.8	Flächen mit gleichzeitig konvexer und reflexiver Kante	156
7.9	Veranschaulichung der verwendeten Bezeichner bei Löchern mit $k = 3$	158
7.10	Ein Loch gebildet durch drei linear abhängige Flächen	158
7.11	Konvexe und reflexive Kanten	159
7.12	Bestimmung der Konvexität zwischen den Flächen f_i und f_j	159
7.13	Fallunterscheidung der Lage von Transitionshalbkanten	160

7.14	Beispiel undefinierter Fälle	160
7.15	Schätzung der Konvexität	161
7.16	Schließen von inneren Löchern	162
7.17	Nutzerhinweise für innere Löcher bis $k = 3$	163
7.18	Nutzerhinweise für äußere Löcher bis $k = 3$	164
7.19	Ein Loch der Kardinalität $k = 6$	165
7.20	Nutzerhinweise für Löcher der Kardinalität $k = 1$	167
7.21	Nutzerhinweise für innere Löcher der Kardinalität $k = 2$	168
7.22	Nutzerhinweise für äußere Löcher der Kardinalität $k = 2$	168
7.23	Nutzerhinweise für äußere Löcher der Kardinalität $k = 3$	169
7.24	Nutzerhinweise für innere Löcher der Kardinalität $k = 3$	170
7.25	Spezialfälle von Löchern	170
7.26	Testszene mit einer Microsoft Kinect	171
7.27	Testszene mit dem Lenovo Phab 2 Pro	172
8.1	Übersicht über die Komponenten und Daten des entwickelten 3D-Scan-Systems	175
8.2	Innenraumrekonstruktionen mit der Microsoft Kinect for XBOX 360	176
8.3	Rekonstruktion eines Modells mit der IDS Ensensio N10	177
8.4	Das Lenovo Phab 2 Pro Smartphone	178
8.5	Organisieren der Punktwolken des Lenovo Phab 2 Pro	179
8.6	Beispiel für Punktkonfidenzen des Lenovo Phab 2 Pro	179
8.7	Rekonstruktion eines Objektes mit dem Lenovo Phab 2 Pro	180
8.8	Rekonstruktion eines Innenraumes mit dem Lenovo Phab 2 Pro	181
8.9	Projekt SIMERO	182

Tabellenverzeichnis

1.1	Beispiele von Sensoren	17
1.2	Beispiele von unterschiedlichen Erfassungsweisen von 3D-Scannern	18
1.3	Eigenschaften des in dieser Arbeit zu entwickelnden Scan-Systems	20
2.1	Einteilung bestehender Systeme	36
4.1	Fälle bei der Konturverfolgung	74
4.2	Statistische und quantitative Ergebnisse der Rekonstruktion	89
4.3	Statistische und quantitative Ergebnisse bei einem Rauschlevel von $\sigma = 0.001$	92
4.4	Laufzeit der Rekonstruktion	94
5.1	Statistische und quantitative Ergebnisse nach Fusion der partiellen B-Reps pro Datensatz bei einem Rauschlevel von $\sigma = 0.001$	109
5.2	Statistische und quantitative Ergebnisse nach Fusion der 11 partiellen B-Reps des <i>anvil</i> -Datensatzes mit einer Strukturgröße von $\delta_s = 50mm$	111
5.3	Statistische und quantitative Ergebnisse nach Fusion von einer unterschiedlichen Anzahl an Einzelmodellen des <i>anvil</i> -Datensatzes	113
6.1	Kombinationen aus physikalischen Elementen (Fläche, Kante, Ecke) eines B-Reps	123
6.2	Fallunterscheidung zur Berechnung des Winkels α_{ij}	125
6.3	Übersicht über direkte (<i>closed form</i>) Optimierungsverfahren zur Bestimmung einer Starrkörpertransformation aus unterschiedlichen Arten von Korrespondenzen	134
6.4	Statistik zu den Datensätzen	136
6.5	Mittelwert μ , Standardabweichung σ , Median Q_2 und oberes Quartil Q_3 der Fehlmaße δ_{error} und t_{error} über alle Datensätze	138
6.6	Zeit zur Berechnung des Gütemaßes	138
6.7	Statistische Auswertung des Konflikttests	139
7.1	Mögliche Ausprägungen der verschiedenen Möglichkeiten der Behandlung von Löchern	166
7.2	Anzahl der möglichen Fälle bei der Behandlung von Löchern der Kardinalität $k \leq 3$	167

Quellenverzeichnis

- [Agathos07] A. Agathos, I. Pratikakis, S. Perantonis, N. Sapidis & P. Azariadis. *3D Mesh Segmentation Methodologies for CAD Applications*. Computer-Aided Design and Applications, Band 4, Nr. 6, Seiten 827–841, 2007. Zitiert auf Seite 33.
- [Arbeiter14] G. Arbeiter, S. Fuchs, J. Hampp & R. Bormann. *Efficient Segmentation and Surface Classification of Range Images*. In 2014 IEEE International Conference on Robotics and Automation (ICRA), Seiten 5502–5509, Mai 2014. Zitiert auf Seite 59, 60, 62 und 70.
- [Arun87] K. S. Arun, T. S. Huang & S. D. Blostein. *Least-Squares Fitting of Two 3-D Point Sets*. IEEE Transactions on Pattern Analysis and Machine Intelligence, Band PAMI-9, Nr. 5, Seiten 698–700, September 1987. Zitiert auf Seite 133 und 134.
- [Asaeedi13] S. Asaeedi, F. Didehvar & A. Mohades. *Alpha-Concave Hull, a Generalization of Convex Hull*. arXiv:1309.7829 [cs], September 2013. Zitiert auf Seite 60 und 61.
- [Ataer-Cansizoglu13] E. Ataer-Cansizoglu, Y. Taguchi, S. Ramalingam & T. Garaas. *Tracking an RGB-D Camera Using Points and Planes*. In The IEEE International Conference on Computer Vision (ICCV) Workshops, Juni 2013. Zitiert auf Seite 29 und 121.
- [Attene06] M. Attene, B. Falcidieno & M. Spagnuolo. *Hierarchical Mesh Segmentation Based on Fitting Primitives*. The Visual Computer, Band 22, Nr. 3, Seiten 181–193, März 2006. Zitiert auf Seite 34.
- [Babu16] B. W. Babu, S. Kim, Z. Yan & L. Ren. *σ -DVO: Sensor Noise Model Meets Dense Visual Odometry*. In 2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), Seiten 18–26, September 2016. Zitiert auf Seite 27.
- [Barequet97] G. Barequet & S. Kumar. *Repairing CAD Models*. In Visualization '97., Proceedings, Seiten 363–370, Oktober 1997. Zitiert auf Seite 146.
- [Barki15] H. Barki, G. Guennebaud & S. Foufou. *Exact, Robust, and Efficient Regularized Booleans on General 3D Meshes*. Computers & Mathematics with Applications, Band 70, Nr. 6, Seiten 1235–1254, September 2015. Zitiert auf Seite 98.
- [Bartoli03] A. Bartoli, R. I. Hartley & F. Kahl. *Motion from 3D Line Correspondences: Linear and Nonlinear Solutions*. In 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings., Band 1, Seiten 477–484, Juni 2003. Zitiert auf Seite 133 und 134.
- [Baumgart72] B. G. Baumgart. *Winged Edge Polyhedron Representation*. Technical report, Stanford University, Stanford, CA, USA, 1972. Zitiert auf Seite 43.
- [Baumgartl09] J. Baumgartl. *Modellierung von Objekten Im Dreidimensionalen Mittels Eines Kalibrierten Kameranetzwerks*. Bachelorarbeit, Universität Bayreuth, Bayreuth, Februar 2009. Zitiert auf Seite 73.

- [Bénière13] R. Bénière, G. Subsol, G. Gesquière, F. L. Breton & W. Puech. *A Comprehensive Process of Reverse Engineering from 3D Meshes to CAD Models*. Computer-Aided Design, Band 45, Nr. 11, Seiten 1382 – 1393, 2013. Zitiert auf Seite 32.
- [Benkő01] P. Benkő, R. R. Martin & T. Várady. *Algorithms for Reverse Engineering Boundary Representation Models*. Computer-Aided Design, Band 33, Nr. 11, Seiten 839–851, September 2001. Zitiert auf Seite 18 und 32.
- [Besl88] P. J. Besl & R. C. Jain. *Segmentation through Variable-Order Surface Fitting*. IEEE Transactions on Pattern Analysis and Machine Intelligence, Band 10, Nr. 2, Seiten 167–192, März 1988. Zitiert auf Seite 33.
- [Besl92] P. J. Besl & N. D. McKay. *A Method for Registration of 3-D Shapes*. IEEE Transactions on Pattern Analysis and Machine Intelligence, Band 14, Nr. 2, Seiten 239–256, Februar 1992. Zitiert auf Seite 119.
- [Biber03] P. Biber & W. Strasser. *The Normal Distributions Transform: A New Approach to Laser Scan Matching*. In Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003), Band 3, Seiten 2743–2748, Oktober 2003. Zitiert auf Seite 120.
- [Biber04] P. Biber, S. Fleck & W. Strasser. *A Probabilistic Framework for Robust and Accurate Matching of Point Clouds*. In Pattern Recognition, Lecture Notes in Computer Science, Seiten 480–487. Springer, Berlin, Heidelberg, August 2004. Zitiert auf Seite 120.
- [Bischoff05] S. Bischoff & L. Kobbelt. *Structure Preserving CAD Model Repair*. Computer Graphics Forum, Band 24, Nr. 3, Seiten 527–536, September 2005. Zitiert auf Seite 146.
- [Borrmann11] D. Borrmann, J. Elseberg, K. Lingemann & A. Nüchter. *The 3D Hough Transform for Plane Detection in Point Clouds: A Review and a New Accumulator Design*. 3D Research, Band 2, Nr. 2, Seite 3, Juni 2011. Zitiert auf Seite 59.
- [Bosché10] F. Bosché. *Automated Recognition of 3D CAD Model Objects in Laser Scans and Calculation of As-Built Dimensions for Dimensional Compliance Control in Construction*. Advanced Engineering Informatics, Band 24, Nr. 1, Seiten 107–118, 2010. Zitiert auf Seite 121.
- [Botsch07] M. Botsch, M. Pauly, L. Kobbelt, P. Alliez, B. Lévy, S. Bischoff & C. Rössl. *Geometric Modeling Based on Polygonal Meshes*. In ACM SIGGRAPH 2007 Courses, SIGGRAPH '07, New York, NY, USA, 2007. ACM. Zitiert auf Seite 98.
- [Breitbach15] T. Breitbach. *Effizienzsteigerung Der Zellenmodellierung Für Industrieroboter Durch Hybride Simulationsmodelle*. Dissertation, RWTH Aachen, 2015. Zitiert auf Seite 145 und 146.
- [Bruderlin91] B. Bruderlin. *Robust Regularized Set Operations on Polyhedra*. In Proceedings of the Twenty-Fourth Annual Hawaii International Conference on System Sciences, Band 1, Seiten 691–700, Januar 1991. Zitiert auf Seite 98.
- [Buchholz15] D. Buchholz. *Bin-Picking – New Approaches for a Classical Problem*. Dissertation, Technische Universität Braunschweig, 2015. Zitiert auf Seite 121.
- [Bülow13] H. Bülow & A. Birk. *Spectral 6DOF Registration of Noisy 3D Range Data with Partial Overlap*. IEEE Transactions on Pattern Analysis and Machine Intelligence, Band 35, Nr. 4, Seiten 954–969, 2013. Zitiert auf Seite 25.
- [Cahier12] L. K. Cahier, T. Ogata & H. G. Okuno. *Incremental Probabilistic Geometry Estimation for Robot Scene Understanding*. In 2012 IEEE International Conference on Robotics and Automation, Seiten 3625–3630, Mai 2012. Zitiert auf Seite 98.

- [Campbell15] D. Campbell & L. Petersson. *An Adaptive Data Representation for Robust Point-Set Registration and Merging*. In 2015 IEEE International Conference on Computer Vision (ICCV), Seiten 4292–4300, Dezember 2015. Zitiert auf Seite 120.
- [Campbell16] D. Campbell & L. Petersson. *GOGMA: Globally-Optimal Gaussian Mixture Alignment*. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Seiten 5685–5694, Juni 2016. Zitiert auf Seite 120.
- [Chang11] K.-H. Chang & C. Chen. *3D Shape Engineering and Design Parameterization*. Computer-Aided Design and Applications, Band 8, Nr. 5, Seiten 681–692, 2011. Zitiert auf Seite 31.
- [Cohen-Steiner04] D. Cohen-Steiner, P. Alliez & M. Desbrun. *Variational Shape Approximation*. In ACM SIGGRAPH 2004 Papers, SIGGRAPH '04, Seiten 905–914, New York, NY, USA, 2004. ACM. Zitiert auf Seite 34.
- [Cupec15] R. Cupec, E. K. Nyarko, D. Filko, A. Kitanov & I. Petrović. *Place Recognition Based on Matching of Planar Surfaces and Line Segments*. The International Journal of Robotics Research, Band 34, Nr. 4-5, Seiten 674–704, April 2015. Zitiert auf Seite 121 und 122.
- [Curless96] B. Curless & M. Levoy. *A Volumetric Method for Building Complex Models from Range Images*. In Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '96, Seiten 303–312, New York, NY, USA, 1996. ACM. Zitiert auf Seite 27.
- [Danielsson81] P.-E. Danielsson. *An Improvement of Kruse's Segmentation Algorithm*. Computer Graphics and Image Processing, Band 17, Nr. 4, Seiten 394–396, Dezember 1981. Zitiert auf Seite 61.
- [de Berg08] M. de Berg, O. Cheong, M. van Kreveld & M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Santa Clara, CA, USA, 3rd edition, 2008. Zitiert auf Seite 102 und 103.
- [Denker13] K. Denker, D. Hagel, J. Raible, G. Umlauf & B. Hamann. *On-Line Reconstruction of CAD Geometry*. In 3DV-Conference, 2013 International Conference On, Seiten 151–158, 2013. Zitiert auf Seite 33.
- [Deschaud10] J.-E. Deschaud & F. Goulette. *A Fast and Accurate Plane Detection Algorithm for Large Noisy Point Clouds Using Filtered Normals and Voxel Growing*. In International Conference On Three-Dimensional Data Processing, Visualization, and Transmission 2010, Paris, France, Mai 2010. Zitiert auf Seite 59.
- [Ding01] D. Ding & S. Wang. *Computation of 3-D Form-Closure Grasps*. IEEE Transactions on Robotics and Automation, Band 17, Nr. 4, Seiten 515–522, August 2001. Zitiert auf Seite 18.
- [Domínguez13] S. Domínguez, E. Zalama, J. García-Bermejo, R. Worst & S. Behnke. *Fast 6D Odometry Based on Visual Features and Depth*. In S. Lee, H. Cho, K.-J. Yoon & J. Lee, Editoren, Intelligent Autonomous Systems 12, Band 193 of *Advances in Intelligent Systems and Computing*, Seiten 245–256. Springer Berlin Heidelberg, 2013. Zitiert auf Seite 25.
- [Douglas73] D. H. Douglas & T. K. Peucker. *Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or Its Caricature*. Cartographica: The International Journal for Geographic Information and Geovisualization, Band 10, Nr. 2, Seiten 112–122, 1973. Zitiert auf Seite 81.

- [Dube11] D. Dube & A. Zell. *Real-Time Plane Extraction from Depth Images with the Randomized Hough Transform*. In 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), Seiten 1084–1091, November 2011. Zitiert auf Seite 59.
- [Duckham08] M. Duckham, L. Kulik, M. Worboys & A. Galton. *Efficient Generation of Simple Polygons for Characterizing the Shape of a Set of Points in the Plane*. *Pattern Recognition*, Band 41, Nr. 10, Seiten 3224–3236, Oktober 2008. Zitiert auf Seite 60 und 61.
- [Duda00] R. O. Duda, P. E. Hart & D. G. Stork. *Pattern Classification (2Nd Edition)*. Wiley-Interscience, 2000. Zitiert auf Seite 34.
- [Edelsbrunner03] Edelsbrunner, Harer & Zomorodian. *Hierarchical Morse—Smale Complexes for Piecewise Linear 2-Manifolds*. *Discrete & Computational Geometry*, Band 30, Nr. 1, Seiten 87–107, 2003. Zitiert auf Seite 32.
- [Edelsbrunner83] H. Edelsbrunner, D. Kirkpatrick & R. Seidel. *On the Shape of a Set of Points in the Plane*. *IEEE Transactions on Information Theory*, Band 29, Nr. 4, Seiten 551–559, Juli 1983. Zitiert auf Seite 60 und 61.
- [Endres12] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers & W. Burgard. *An Evaluation of the RGB-D SLAM System*. In *Robotics and Automation (ICRA)*, 2012 IEEE International Conference On, Seiten 1691–1696, Mai 2012. Zitiert auf Seite 24 und 26.
- [Fischler81] M. A. Fischler & R. C. Bolles. *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*. *Commun. ACM*, Band 24, Nr. 6, Seiten 381–395, Juni 1981. Zitiert auf Seite 24, 35 und 59.
- [Foley96] J. Foley. *Computer Graphics: Principles and Practice*. Addison-Wesley systems programming series. Addison-Wesley, 1996. Zitiert auf Seite 42.
- [Gallo11] O. Gallo, R. Manduchi & A. Rafii. *CC-RANSAC: Fitting Planes in the Presence of Multiple Surfaces in Range Data*. *Pattern Recognition Letters*, Band 32, Nr. 3, Seiten 403–410, Februar 2011. Zitiert auf Seite 59.
- [Galton06] A. Galton & M. Duckham. *What Is the Region Occupied by a Set of Points?* In *Geographic Information Science, Lecture Notes in Computer Science*, Seiten 81–98. Springer, Berlin, Heidelberg, September 2006. Zitiert auf Seite 60.
- [Galvez-Lopez11] D. Galvez-Lopez & J. Tardos. *Real-Time Loop Detection with Bags of Binary Words*. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* 2011, Seiten 51–58, 2011. Zitiert auf Seite 25.
- [Garland01] M. Garland, A. Willmott & P. S. Heckbert. *Hierarchical Face Clustering on Polygonal Surfaces*. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics, I3D '01*, Seiten 49–58, New York, NY, USA, 2001. ACM. Zitiert auf Seite 34.
- [Gelfand04] N. Gelfand & L. J. Guibas. *Shape Segmentation Using Local Slippage Analysis*. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, SGP '04*, Seiten 214–223, New York, NY, USA, 2004. ACM. Zitiert auf Seite 34.
- [Georgiev11] K. Georgiev, R. T. Creed & R. Lakaemper. *Fast Plane Extraction in 3D Range Data Based on Line Segments*. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Seiten 3808–3815, September 2011. Zitiert auf Seite 60.
- [Gotardo03] P. F. U. Gotardo, O. R. P. Bellon & L. Silva. *Range Image Segmentation by Surface Extraction Using an Improved Robust Estimator*. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, Band 2, Seiten 33–38, Juni 2003. Zitiert auf Seite 59.

- [Grant81] G. Grant & A. F. Reid. *An Efficient Algorithm for Boundary Tracing and Feature Extraction*. Computer Graphics and Image Processing, Band 17, Nr. 3, Seiten 225–237, November 1981. Zitiert auf Seite 61.
- [Grimson90] W. E. L. Grimson & D. P. Huttenlocher. *On the Sensitivity of the Hough Transform for Object Recognition*. IEEE Transactions on Pattern Analysis and Machine Intelligence, Band 12, Nr. 3, Seiten 255–274, März 1990. Zitiert auf Seite 128.
- [Grisetti09] G. Grisetti, C. Stachniss & W. Burgard. *Nonlinear Constraint Network Optimization for Efficient Map Learning*. IEEE Transactions on Intelligent Transportation Systems, Band 10, Nr. 3, Seiten 428–439, 2009. Zitiert auf Seite 26.
- [Gschwandtner11] M. Gschwandtner, R. Kwitt, A. Uhl & W. Pree. *BlenSor: Blender Sensor Simulation Toolbox*. In G. Bebis, R. Boyle, B. Parvin, D. Koracin, S. Wang, K. Kyunghnam, B. Benes, K. Moreland, C. Borst, S. DiVerdi, C. Yi-Jen & J. Ming, Editoren, *Advances in Visual Computing*, Band 6939 of *Lecture Notes in Computer Science*, Seiten 199–208. Springer Berlin Heidelberg, 2011. Zitiert auf Seite 85 und 93.
- [Guo18] X. Guo, J. Xiao & Y. Wang. *A Survey on Algorithms of Hole Filling in 3D Surface Reconstruction*. The Visual Computer, Band 34, Nr. 1, Seiten 93–103, Januar 2018. Zitiert auf Seite 146.
- [Hachenberger07] P. Hachenberger, L. Kettner & K. Mehlhorn. *Boolean Operations on 3D Selective Nef Complexes: Data Structure, Algorithms, Optimized Implementation and Experiments*. Computational Geometry, Band 38, Nr. 1, Seiten 64–99, September 2007. Zitiert auf Seite 98.
- [Hänel12] M. Hänel, L. Grüne, D. Henrich, S. Kuhn & J. Pannek. *Optimal Camera Placement to Measure Distances Regarding Static and Dynamic Obstacles*. International Journal of Sensor Networks, Band 12, Nr. 1, Seiten 25–36, 2012. Zitiert auf Seite 18.
- [Harati07a] A. Harati, S. Gächter & R. Siegwart. *Fast Range Image Segmentation for Indoor 3D-SLAM*. IFAC Proceedings Volumes, Band 40, Nr. 15, Seiten 475–480, Januar 2007. Zitiert auf Seite 60.
- [Harati07b] A. Harati & R. Siegwart. *Orthogonal 3D-SLAM for Indoor Environments Using Right Angle Corners*. In Proceedings of the 3rd European Conference on Mobile Robots ECMR 2007, Seiten 144–149, 2007. Zitiert auf Seite 120.
- [Hoffmann01] C. M. Hoffmann. *Robustness in Geometric Computations*. Journal of Computing and Information Science in Engineering, Band 1, Nr. 2, Seiten 143–155, März 2001. Zitiert auf Seite 98.
- [Hoffmann89] C. M. Hoffmann. *Geometric and Solid Modeling: An Introduction*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989. Zitiert auf Seite 14.
- [Holz11] D. Holz, S. Holzer, R. B. Rusu & S. Behnke. *Real-Time Plane Segmentation Using RGB-D Cameras*. In RoboCup Symposium, 2011. Zitiert auf Seite 59.
- [Holz14] D. Holz & S. Behnke. *Approximate Triangulation and Region Growing for Efficient Segmentation and Smoothing of Range Images*. Robotics and Autonomous Systems, Band 62, Nr. 9, Seiten 1282–1293, September 2014. Zitiert auf Seite 59, 62, 63, 64, 65, 67, 94, 185 und 190.
- [Holzer12] S. Holzer, R. B. Rusu, M. Dixon, S. Gedikli & N. Navab. *Adaptive Neighborhood Selection for Real-Time Surface Normal Estimation from Organized Point Cloud Data Using Integral Images*. In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Seiten 2684–2689, Oktober 2012. Zitiert auf Seite 55, 56, 86 und 175.

- [Hoover96] A. Hoover, G. Jean-Baptiste, X. Jiang, P. J. Flynn, H. Bunke, D. B. Goldgof, K. Bowyer, D. W. Eggert, A. Fitzgibbon & R. B. Fisher. *An Experimental Comparison of Range Image Segmentation Algorithms*. IEEE Transactions on Pattern Analysis and Machine Intelligence, Band 18, Nr. 7, Seiten 673–689, Juli 1996. Zitiert auf Seite 85 und 87.
- [Hsu16] M.-C. Hsu, C. Wang, F. Xu, A. J. Herrema & A. Krishnamurthy. *Direct Immersogeometric Fluid Flow Analysis Using B-Rep CAD Models*. Computer Aided Geometric Design, Band 43, Seiten 143 – 158, 2016. Zitiert auf Seite 18.
- [Huang01] J. Huang & C.-H. Menq. *Automatic Data Segmentation for Geometric Feature Extraction from Unorganized 3-D Coordinate Points*. IEEE Transactions on Robotics and Automation, Band 17, Nr. 3, Seiten 268 –279, Juni 2001. Zitiert auf Seite 32.
- [Huang02] J. Huang & C.-H. Menq. *Identification and Characterization of Regular Surfaces from Unorganized Points by Normal Sensitivity Analysis*. Journal of Computing and Information Science in Engineering, Band 2, Nr. 2, Seiten 115–124, September 2002. Zitiert auf Seite 32.
- [Huang03] J. Huang & C.-H. Menq. *Automatic CAD Model Reconstruction from Multiple Point Clouds for Reverse Engineering*. Journal of Computing and Information Science in Engineering, Band 2, Nr. 3, Seiten 160–170, Januar 2003. Zitiert auf Seite 32.
- [Inoue01] K. Inoue, T. Itoh, A. Yamada, T. Furuhashi & K. Shimada. *Face Clustering of a Large-Scale CAD Model for Surface Mesh Generation*. Computer-Aided Design, Band 33, Nr. 3, Seiten 251–261, März 2001. Zitiert auf Seite 34.
- [Izadi11] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison & A. Fitzgibbon. *KinectFusion: Real-Time 3D Reconstruction and Interaction Using a Moving Depth Camera*. In Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, UIST '11, Seiten 559–568, New York, NY, USA, 2011. ACM. Zitiert auf Seite 27 und 145.
- [Jian11] B. Jian & B. C. Vemuri. *Robust Point Set Registration Using Gaussian Mixture Models*. IEEE Transactions on Pattern Analysis and Machine Intelligence, Band 33, Nr. 8, Seiten 1633–1645, August 2011. Zitiert auf Seite 120.
- [Ju09] T. Ju. *Fixing Geometric Errors on Polygonal Models: A Survey*. Journal of Computer Science and Technology, Band 24, Nr. 1, Seiten 19–29, Januar 2009. Zitiert auf Seite 146.
- [Kaess08] M. Kaess, A. Ranganathan & F. Dellaert. *iSAM: Incremental Smoothing and Mapping*. Robotics, IEEE Transactions on, Band 24, Nr. 6, Seiten 1365–1378, 2008. Zitiert auf Seite 26.
- [Kähler15] O. Kähler, V. A. Prisacariu, C. Y. Ren, X. Sun, P. Torr & D. Murray. *Very High Frame Rate Volumetric Integration of Depth Images on Mobile Devices*. IEEE Transactions on Visualization and Computer Graphics, Band 21, Nr. 11, Seiten 1241–1250, November 2015. Zitiert auf Seite 28.
- [Kähler16] O. Kähler, V. A. Prisacariu & D. W. Murray. *Real-Time Large-Scale Dense 3D Reconstruction with Loop Closure*. In Computer Vision – ECCV 2016, Lecture Notes in Computer Science, Seiten 500–516. Springer, Cham, Oktober 2016. Zitiert auf Seite 28 und 98.
- [Katz03] S. Katz & A. Tal. *Hierarchical Mesh Decomposition Using Fuzzy Clustering and Cuts*. ACM Trans. Graph., Band 22, Nr. 3, Seiten 954–961, Juli 2003. Zitiert auf Seite 34.
- [Kaushik10] R. Kaushik, J. Xiao, S. L. Joseph & W. Morris. *Fast Planar Clustering and Polygon Extraction from Noisy Range Images Acquired in Indoor Environments*. In 2010 IEEE International Conference on Mechatronics and Automation, Seiten 483–488, August 2010. Zitiert auf Seite 60.

- [Keller13] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich & A. Kolb. *Real-Time 3D Reconstruction in Dynamic Scenes Using Point-Based Fusion*. In Proceedings of the 2013 International Conference on 3D Vision, 3DV '13, Seiten 1–8, Washington, DC, USA, 2013. Zitiert auf Seite 28 und 98.
- [Kerl13a] C. Kerl, J. Sturm & D. Cremers. *Dense Visual SLAM for RGB-D Cameras*. In 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Seiten 2100–2106, November 2013. Zitiert auf Seite 27.
- [Kerl13b] C. Kerl, J. Sturm & D. Cremers. *Robust Odometry Estimation for RGB-D Cameras*. In Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), Mai 2013. Zitiert auf Seite 24.
- [Keyser99a] J. Keyser, S. Krishnan & D. Manocha. *Efficient and Accurate B-Rep Generation of Low Degree Sculptured Solids Using Exact Arithmetic: II—computation*. Computer Aided Geometric Design, Band 16, Nr. 9, Seiten 861–882, Oktober 1999. Zitiert auf Seite 98.
- [Keyser99b] J. Keyser, S. Krishnan & D. Manocha. *Efficient and Accurate B-Rep Generation of Low Degree Sculptured Solids Using Exact Arithmetic: I—representations*. Computer Aided Geometric Design, Band 16, Nr. 9, Seiten 841–859, Oktober 1999. Zitiert auf Seite 98.
- [Khoshelham12] K. Khoshelham & S. O. Elberink. *Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications*. Sensors, Band 12, Nr. 2, Seiten 1437–1454, 2012. Zitiert auf Seite 86.
- [Khoshelham16] K. Khoshelham. *Closed-Form Solutions for Estimating a Rigid Motion from Plane Correspondences Extracted from Point Clouds*. ISPRS Journal of Photogrammetry and Remote Sensing, Band 114, Seiten 78 – 91, 2016. Zitiert auf Seite 133 und 134.
- [Kohlhepp06] P. Kohlhepp, G. Bretthauer, M. Walther & R. Dillmann. *Using Orthogonal Surface Directions for Autonomous 3D-Exploration of Indoor Environments*. In 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Seiten 3086–3092, Oktober 2006. Zitiert auf Seite 120.
- [Korkalo16] O. Korkalo & S. Kahn. *Real-Time Depth Camera Tracking with CAD Models and ICP*. Journal of Virtual Reality and Broadcasting, Band 13, Nr. 1, August 2016. Zitiert auf Seite 121.
- [Kummerle11] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige & W. Burgard. *G2o: A General Framework for Graph Optimization*. In IEEE International Conference on Robotics and Automation (ICRA), 2011, Seiten 3607–3613, 2011. Zitiert auf Seite 26.
- [Kyriazis13] I. Kyriazis & I. Fudos. *Building Editable Free-Form Models from Unstructured Point Clouds*. Computer-Aided Design and Applications, Band 10, Nr. 6, Seiten 877–888, 2013. Zitiert auf Seite 33.
- [Labbé13] M. Labbé & F. Michaud. *Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation*. IEEE Transactions on Robotics, Band 29, Nr. 3, Seiten 734–745, 2013. Zitiert auf Seite 25 und 27.
- [Laidlaw86] D. H. Laidlaw, W. B. Trumbore & J. F. Hughes. *Constructive Solid Geometry for Polyhedral Objects*. In Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '86, Seiten 161–170, New York, NY, USA, 1986. ACM. Zitiert auf Seite 98.
- [Lavoué05] G. Lavoué, F. Dupont & A. Baskurt. *A New CAD Mesh Segmentation Method, Based on Curvature Tensor Analysis*. Computer-Aided Design, Band 37, Nr. 10, Seiten 975–987, September 2005. Zitiert auf Seite 33.

- [Le17] T. Le & Y. Duan. *A Primitive-Based 3D Segmentation Algorithm for Mechanical CAD Models*. Computer Aided Geometric Design, Band 52–53, Seiten 231 – 246, 2017. Zitiert auf Seite 35.
- [Lee12] T.-k. Lee, S. Lim, S. Lee, S. An & S.-y. Oh. *Indoor Mapping Using Planes Extracted from Noisy RGB-D Sensors*. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2012, Seiten 1727–1733, 2012. Zitiert auf Seite 25.
- [Lee99] K. Lee. *Principles of CAD/CAM/CAE Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999. Zitiert auf Seite 13, 42 und 98.
- [Li11] Y. Li, X. Wu, Y. Chrysathou, A. Sharf, D. Cohen-Or & N. J. Mitra. *GlobFit: Consistently Fitting Primitives by Discovering Global Relations*. ACM Trans. Graph., Band 30, Nr. 4, Seiten 52:1–52:12, Juli 2011. Zitiert auf Seite 35.
- [Lloyd82] S. Lloyd. *Least Squares Quantization in PCM*. IEEE Transactions on Information Theory, Band 28, Nr. 2, Seiten 129–137, März 1982. Zitiert auf Seite 34.
- [Lorensen87] W. E. Lorensen & H. E. Cline. *Marching Cubes: A High Resolution 3D Surface Construction Algorithm*. SIGGRAPH Comput. Graph., Band 21, Nr. 4, Seiten 163–169, August 1987. Zitiert auf Seite 27.
- [Magnusson09] M. Magnusson. *The Three-Dimensional Normal-Distributions Transform : An Efficient Representation for Registration, Surface Analysis, and Loop Detection*. Dissertation, Örebro University, School of Science and Technology, 2009. Zitiert auf Seite 120.
- [Mäntylä87] M. Mäntylä. *An Introduction to Solid Modeling*. Computer Science Press, Inc., New York, NY, USA, 1987. Zitiert auf Seite 43.
- [Marks05] P. Marks. *Capturing a Competitive Edge Through Digital Shape Sampling & Processing (DSSP)*. SME Blue Book Series. Society of Manufacturing Engineers, 2005. Zitiert auf Seite 30.
- [Miandarhoie17] A. Miandarhoie, K. Khalili & H. Mohammadinejad. *CAD Mesh Models Segmentation into Swept Surfaces*. The International Journal of Advanced Manufacturing Technology, Band 92, Nr. 9-12, Seiten 3659–3671, Oktober 2017. Zitiert auf Seite 34.
- [Miyatake97] T. Miyatake, H. Matsushima & M. Ejiri. *Contour Representation of Binary Images Using Run-Type Direction Codes*. Machine Vision and Applications, Band 9, Nr. 4, Seiten 193–200, Februar 1997. Zitiert auf Seite 61, 63 und 73.
- [Montello98] D. Montello. *Kartenverstehen: Die Sicht Der Kognitionspsychologie*. Zeitschrift für Semiotik, Stauffenburg Verlag, Band 20, Nr. 1-2, 1998. Zitiert auf Seite 146.
- [Moreira07] A. J. C. Moreira & M. Y. Santos. *Concave Hull: A k-Nearest Neighbours Approach for the Computation of the Region Occupied by a Set of Points*. In GRAPP 2007, Proceedings of the Second International Conference on Computer Graphics Theory and Applications, Seiten 61–68, Barcelona, Spain, 2007. Zitiert auf Seite 60 und 61.
- [Naylor90] B. Naylor, J. Amanatides & W. Thibault. *Merging BSP Trees Yields Polyhedral Set Operations*. In Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '90, Seiten 115–124, New York, NY, USA, 1990. ACM. Zitiert auf Seite 98.
- [Naylor92] B. F. Naylor. *Interactive Solid Geometry via Partitioning Trees*. In Proceedings of the Conference on Graphics Interface '92, Seiten 11–18, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc. Zitiert auf Seite 98.

- [Newcombe11] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges & A. Fitzgibbon. *KinectFusion: Real-Time Dense Surface Mapping and Tracking*. In 2011 10th IEEE International Symposium on Mixed and Augmented Reality, Seiten 127–136, Oktober 2011. Zitiert auf Seite 27, 98 und 121.
- [Nguyen12] C. Nguyen, S. Izadi & D. Lovell. *Modeling Kinect Sensor Noise for Improved 3D Reconstruction and Tracking*. In 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference On, Seiten 524–530, Oktober 2012. Zitiert auf Seite 55.
- [Nießner13] M. Nießner, M. Zollhöfer, S. Izadi & M. Stamminger. *Real-Time 3D Reconstruction at Scale Using Voxel Hashing*. ACM Trans. Graph., Band 32, Nr. 6, Seiten 169:1–169:11, November 2013. Zitiert auf Seite 28 und 98.
- [Park12] J.-S. Park & S.-J. Oh. *A New Concave Hull Algorithm and Concaveness Measure for N-Dimensional Datasets*. Journal of Information Science and Engineering, Band 28, Seiten 587–600, Mai 2012. Zitiert auf Seite 60.
- [Pathak09] K. Pathak, N. Vaskevicius, J. Poppinga, M. Pfingsthorn, S. Schwertfeger & A. Birk. *Fast 3D Mapping by Matching Planes Extracted from Range Sensor Point-Clouds*. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2009, Seiten 1150–1155, Oktober 2009. Zitiert auf Seite 25.
- [Pathak10] K. Pathak, A. Birk, N. Vaskevicius & J. Poppinga. *Fast Registration Based on Noisy Planes with Unknown Correspondences for 3D Mapping*. IEEE Transactions on Robotics, Band 26, Seiten 424–441, 2010. Zitiert auf Seite 120.
- [Pavlidis82] T. Pavlidis. *Bilevel Pictures*. In Algorithms for Graphics and Image Processing, Seiten 129–165. Springer, Berlin, Heidelberg, 1982. Zitiert auf Seite 61.
- [Pfister00] H. Pfister, M. Zwicker, J. van Baar & M. Gross. *Surfels: Surface Elements As Rendering Primitives*. In Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '00, Seiten 335–342, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co. Zitiert auf Seite 28.
- [Pomerleau13] F. Pomerleau, F. Colas, R. Siegwart & S. Magnenat. *Comparing ICP Variants on Real-World Data Sets*. Autonomous Robots, Band 34, Nr. 3, Seiten 133–148, April 2013. Zitiert auf Seite 119.
- [Poppinga08] J. Poppinga, N. Vaskevicius, A. Birk & K. Pathak. *Fast Plane Detection and Polygonalization in Noisy 3D Range Images*. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2008, Seiten 3378–3383, September 2008. Zitiert auf Seite 59.
- [Poutrain01] K. Poutrain & M. Contensin. *Dual B-Rep-CSG Collision Detection for General Polyhedra*. In Proceedings Ninth Pacific Conference on Computer Graphics and Applications. Pacific Graphics 2001, Seiten 124–133, 2001. Zitiert auf Seite 18.
- [Qian14] X. Qian & C. Ye. *NCC-RANSAC: A Fast Plane Extraction Method for 3-D Range Data Segmentation*. IEEE Transactions on Cybernetics, Band 44, Nr. 12, Seiten 2771–2783, Dezember 2014. Zitiert auf Seite 59.
- [Requicha85] A. Requicha & H. Voelcker. *Boolean Operations in Solid Modeling: Boundary Evaluation and Merging Algorithms*. Proceedings of the IEEE, Band 73, Nr. 1, Seiten 30–44, Januar 1985. Zitiert auf Seite 98.
- [Rossignac07] J. Rossignac. *Solid and Physical Modeling*. Technical report, Georgia Institute of Technology, Atlanta, 2007. Zitiert auf Seite 98.

- [Roth12] H. Roth & M. Vona. *Moving Volume KinectFusion*. In Proceedings of the British Machine Vision Conference, Seiten 112.1–112.11. BMVA Press, 2012. Zitiert auf Seite 28.
- [Rusinkiewicz01] S. Rusinkiewicz & M. Levoy. *Efficient Variants of the ICP Algorithm*. In Proceedings Third International Conference on 3-D Digital Imaging and Modeling, Seiten 145–152, 2001. Zitiert auf Seite 27 und 119.
- [Salas-Moreno14] R. F. Salas-Moreno, B. Glocken, P. H. J. Kelly & A. J. Davison. *Dense Planar SLAM*. In 2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), Seiten 157–164, September 2014. Zitiert auf Seite 28.
- [Sapidis95] N. S. Sapidis & P. J. Besl. *Direct Construction of Polynomial Surfaces from Dense Range Images Through Region Growing*. ACM Trans. Graph., Band 14, Nr. 2, Seiten 171–200, April 1995. Zitiert auf Seite 33.
- [Schnabel07] R. Schnabel & R. Wahl. *Efficient RANSAC for Point-Cloud Shape Detection*. Computer Graphics Forum, Band 26, Nr. 2, Seiten 214 – 226, 2007. Zitiert auf Seite 35 und 59.
- [Schneider03] P. Schneider & D. H. Eberly. *Geometric Tools for Computer Graphics*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003. Zitiert auf Seite 131.
- [Schweizer71] H.-J. Schweizer. *Experimentelle Untersuchungen über den Auge-Hand-Regelkreis bei der Durchführung einer Balancieraufgabe unter stroboskopischer Beleuchtung*. Kybernetik, Band 9, Nr. 5, Seiten 182–189, November 1971. Zitiert auf Seite 146.
- [Segal90] M. Segal. *Using Tolerances to Guarantee Valid Polyhedral Modeling Results*. In Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '90, Seiten 105–114, New York, NY, USA, 1990. ACM. Zitiert auf Seite 98.
- [Seo16] J. Seo, S. Chae, J. Shim, D. Kim, C. Cheong & T.-D. Han. *Fast Contour-Tracing Algorithm Based on a Pixel-Following Method for Image Sensors*. Sensors (Basel, Switzerland), Band 16, Nr. 3, März 2016. Zitiert auf Seite 61.
- [Shamir08] A. Shamir. *A Survey on Mesh Segmentation Techniques*. Computer Graphics Forum, Band 27, Nr. 6, Seiten 1539–1556, 2008. Zitiert auf Seite 33.
- [Siegwart04] R. Siegwart & I. R. Nourbakhsh. *Introduction to Autonomous Mobile Robots*. Bradford Company, Scituate, MA, USA, 2004. Zitiert auf Seite 18.
- [Silva02] L. Silva, O. R. P. Bellon & P. F. U. Gotardo. *A Global-to-Local Approach for Robust Range Image Segmentation*. In Proceedings. International Conference on Image Processing, Band 1, Seiten 773–776, 2002. Zitiert auf Seite 59.
- [Stamati10] V. Stamati & I. Fudos. *Building Editable B-Rep Models from Unorganized Point Clouds*. Technical report Tech. Rep. TR-2010-04, Computer Science Department, University of Ioannina, Ioannina, Greece, Juli 2010. Zitiert auf Seite 33.
- [Steinbruecker11] F. Steinbruecker, J. Sturm & D. Cremers. *Real-Time Visual Odometry from Dense RGB-D Images*. In Workshop on Live Dense Reconstruction with Moving Cameras at the Intl. Conf. on Computer Vision (ICCV), 2011. Zitiert auf Seite 24.
- [Steinbruecker13] F. Steinbruecker, J. Sturm & D. Cremers. *Large-Scale Multi-Resolution Surface Reconstruction from RGB-D Sequences*. In IEEE International Conference on Computer Vision (ICCV), Sydney, Australia, 2013. Zitiert auf Seite 28.
- [Stockman01] G. Stockman & L. G. Shapiro. *Computer Vision*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2001. Zitiert auf Seite 70.

- [Stückler14] J. Stückler & S. Behnke. *Multi-Resolution Surfel Maps for Efficient Dense 3D Modeling and Tracking*. Journal of Visual Communication and Image Representation, Band 25, Nr. 1, Seiten 137–147, Januar 2014. Zitiert auf Seite 28.
- [Suzuki85] S. Suzuki & K. Abe. *Topological Structural Analysis of Digitized Binary Images by Border Following*. Computer Vision, Graphics, and Image Processing, Band 30, Nr. 1, Seiten 32 – 46, 1985. Zitiert auf Seite 61.
- [Taguchi13] Y. Taguchi, Y.-D. Jian, S. Ramalingam & C. Feng. *Point-Plane SLAM for Hand-Held 3D Sensors*. In IEEE International Conference on Robotics and Automation (ICRA), 2013, Seiten 5182–5189, 2013. Zitiert auf Seite 29, 121 und 134.
- [Taketomi17] T. Taketomi, H. Uchiyama & S. Ikeda. *Visual SLAM Algorithms: A Survey from 2010 to 2016*. IPSJ Transactions on Computer Vision and Applications, Band 9, Nr. 1, Seite 16, Dezember 2017. Zitiert auf Seite 23.
- [Tam13] G. K. L. Tam, Z. Q. Cheng, Y. K. Lai, F. C. Langbein, Y. Liu, D. Marshall, R. R. Martin, X. F. Sun & P. L. Rosin. *Registration of 3D Point Clouds and Meshes: A Survey from Rigid to Nonrigid*. IEEE Transactions on Visualization and Computer Graphics, Band 19, Nr. 7, Seiten 1199–1217, Juli 2013. Zitiert auf Seite 120.
- [Tangelder04] J. Tangelder & R. Veltkamp. *A Survey of Content Based 3D Shape Retrieval Methods*. In Shape Modeling Applications, 2004. Proceedings, Seiten 145–156, 2004. Zitiert auf Seite 120.
- [Tatavarti17] A. Tatavarti, J. Papadakis & A. R. Willis. *Towards Real-Time Segmentation of 3D Point Cloud Data into Local Planar Regions*. In SoutheastCon 2017, Seiten 1–6, März 2017. Zitiert auf Seite 60 und 62.
- [Taylor03] G. R. Taylor & L. Kleeman. *Robust Range Data Segmentation Using Geometric Primitives for Robotic Applications*. In Signal and Image Processing (SIP 2003), Proceedings of the IASTED International Conference, August 13-15, 2003, Honolulu, HI, USA, Seiten 467–472, 2003. Zitiert auf Seite 34.
- [Theologou15] P. Theologou, I. Pratikakis & T. Theoharis. *A Comprehensive Overview of Methodologies and Performance Evaluation Frameworks in 3D Mesh Segmentation*. Computer Vision and Image Understanding, Band 135, Seiten 49–82, Juni 2015. Zitiert auf Seite 33.
- [Thibault87] W. C. Thibault & B. F. Naylor. *Set Operations on Polyhedra Using Binary Space Partitioning Trees*. In Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '87, Seiten 153–162, New York, NY, USA, 1987. ACM. Zitiert auf Seite 98.
- [Toussaint83] G. Toussaint. *Solving Geometric Problems with the Rotating Calipers*. In Proceedings of Mediterranean Electrotechnical Conference (IEEE MELECON) '83, Seiten A10.02/1–4, 1983. Zitiert auf Seite 131.
- [Trevor12] A. J. B. Trevor, J. G. Rogers & H. I. Christensen. *Planar Surface SLAM with 3D and 2D Sensors*. In 2012 IEEE International Conference on Robotics and Automation, Seiten 3041–3048, Mai 2012. Zitiert auf Seite 59.
- [Trevor13] A. Trevor, S. Gedikli, R. Rusu & H. I. Christensen. *Efficient Organized Point Cloud Segmentation with Connected Components*. In 3rd Workshop on Semantic Perception, Mapping and Exploration (SPME), Mai 2013. Zitiert auf Seite 60.

- [Turk94] G. Turk & M. Levoy. *Zippered Polygon Meshes from Range Images*. In Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '94, Seiten 311–318, New York, NY, USA, 1994. ACM. Zitiert auf Seite 98.
- [van Kaick11] O. van Kaick, H. Zhang, G. Hamarneh & D. Cohen-Or. *A Survey on Shape Correspondence*. Computer Graphics Forum, Band 30, Nr. 6, Seiten 1681–1707, 2011. Zitiert auf Seite 120.
- [Várady06] T. Várady, M. A. Facello & Z. Terék. *Automatic Extraction of Surface Structures in Digital Shape Reconstruction*. In Proceedings of 4th International Conference on Geometric Modeling and Processing (GMP 2006), Seiten 1–16. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. Zitiert auf Seite 32 und 98.
- [Várady08] T. Várady. *Automatic Procedures to Create CAD Models from Measured Data*. Computer-Aided Design and Applications, Band 5, Nr. 5, Seiten 577–588, 2008. Zitiert auf Seite 30 und 31.
- [Várady97] T. Várady, R. R. Martin & J. Cox. *Reverse Engineering of Geometric Models—an Introduction*. Computer-Aided Design, Band 29, Nr. 4, Seiten 255–268, April 1997. Zitiert auf Seite 12.
- [Vieira05] M. Vieira & K. Shimada. *Surface Mesh Segmentation and Smooth Surface Extraction through Region Growing*. Computer Aided Geometric Design, Band 22, Nr. 8, Seiten 771–792, November 2005. Zitiert auf Seite 34.
- [Vo15] A.-V. Vo, L. Truong-Hong, D. F. Laefer & M. Bertolotto. *Octree-Based Region Growing for Point Cloud Segmentation*. ISPRS Journal of Photogrammetry and Remote Sensing, Band 104, Seiten 88–100, Juni 2015. Zitiert auf Seite 59.
- [Wang11] C. C. L. Wang. *Approximate Boolean Operations on Large Polyhedral Solids with Partial Mesh Reconstruction*. IEEE Transactions on Visualization and Computer Graphics, Band 17, Nr. 6, Seiten 836–849, Juni 2011. Zitiert auf Seite 98.
- [Wang17] R. Wang, M. Schwörer & D. Cremers. *Stereo DSO: Large-Scale Direct Sparse Visual Odometry with Stereo Cameras*. In International Conference on Computer Vision (ICCV), Venice, Italy, Oktober 2017. Zitiert auf Seite 25.
- [Whelan12] T. Whelan, J. B. McDonald, M. Kaess, M. F. Fallon, H. Johannsson & J. J. Leonard. *Kintinuuous: Spatially Extended KinectFusion*. In RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras, Sydney, Australia, Juli 2012. Zitiert auf Seite 28.
- [Whelan13] T. Whelan, M. Kaess, J. J. Leonard & J. McDonald. *Deformation-Based Loop Closure for Large Scale Dense RGB-D SLAM*. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013, Seiten 548–555, November 2013. Zitiert auf Seite 28.
- [Whelan15a] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker & A. J. Davison. *ElasticFusion: Dense SLAM Without A Pose Graph*. In Robotics: Science and Systems (RSS), Rome, Italy, Juli 2015. Zitiert auf Seite 28.
- [Whelan15b] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard & J. McDonald. *Real-Time Large-Scale Dense RGB-D SLAM with Volumetric Fusion*. The International Journal of Robotics Research, Band 34, Nr. 4-5, Seiten 598–626, April 2015. Zitiert auf Seite 28 und 98.
- [Whelan16] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison & S. Leutenegger. *ElasticFusion: Real-Time Dense SLAM and Light Source Estimation*. The International Journal of Robotics Research, Band 35, Nr. 14, Seiten 1697–1716, Dezember 2016. Zitiert auf Seite 28 und 98.

- [Winkelbach06] S. Winkelbach. *Das 3d-Puzzle-Problem: Effiziente Methoden Zum Paarweisen Zusammensetzen von Dreidimensionalen Fragmenten*. Dissertation, University of Braunschweig - Institute of Technology, 2006. Zitiert auf Seite 121, 122 und 128.
- [Wu05] J. Wu & L. Kobbelt. *Structure Recovery via Hybrid Variational Surface Approximation*. Computer Graphics Forum (Proc. of Eurographics), Seiten 277–284, 2005. Zitiert auf Seite 34.
- [Wuttke12] S. Wuttke, D. Perpeet & W. Middelmann. *Quality Preserving Fusion of 3D Triangle Meshes*. In 2012 15th International Conference on Information Fusion, Seiten 1476–1481, Juli 2012. Zitiert auf Seite 98.
- [Xiao11] J. Xiao, J. Zhang, J. Zhang, H. Zhang & H. P. Hildre. *Fast Plane Detection for SLAM from Noisy Range Images in Both Structured and Unstructured Environments*. In 2011 IEEE International Conference on Mechatronics and Automation, Seiten 1768–1773, August 2011. Zitiert auf Seite 60.
- [Xiao13] J. Xiao, B. Adler, J. Zhang & H. Zhang. *Planar Segment Based Three-Dimensional Point Cloud Registration in Outdoor Environments*. Journal of Field Robotics, Band 30, Nr. 4, Seiten 552–582, 2013. Zitiert auf Seite 59 und 120.
- [Yan06] D.-M. Yan, Y. Liu & W. Wang. *Quadric Surface Extraction by Variational Shape Approximation*. In Proceedings of the 4th International Conference on Geometric Modeling and Processing, GMP'06, Seiten 73–86, Berlin, Heidelberg, 2006. Springer-Verlag. Zitiert auf Seite 34.
- [Yan12] D.-M. Yan, W. Wang, Y. Liu & Z. Yang. *Variational Mesh Segmentation via Quadric Surface Fitting*. Computer-Aided Design, Band 44, Nr. 11, Seiten 1072–1082, November 2012. Zitiert auf Seite 34.
- [Yan17] Z. Yan, M. Ye & L. Ren. *Dense Visual SLAM with Probabilistic Surfel Map*. IEEE Transactions on Visualization and Computer Graphics, Band 23, Nr. 11, Seiten 2389–2398, November 2017. Zitiert auf Seite 29.
- [Yang16] J. Yang, H. Li, D. Campbell & Y. Jia. *Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration*. IEEE Transactions on Pattern Analysis and Machine Intelligence, Band 38, Nr. 11, Seiten 2241–2254, November 2016. Zitiert auf Seite 120.
- [Younes16] G. Younes, D. C. Asmar & E. A. Shamma. *A Survey on Non-Filter-Based Monocular Visual SLAM Systems*. arXiv:1607.00470v1 [cs.CV], 2016. Zitiert auf Seite 23.

Eigene Publikationen

- [Sand14] M. Sand & D. Henrich. *Intuitive Erzeugung von 3D-Modellen Mit Handgehaltenen Sensoren*. In Mensch & Computer 2014 - Tagungsband, 14. Fachübergreifende Konferenz Für Interaktive Und Kooperative Medien - Interaktiv Unterwegs - Freiräume Gestalten, Band 14, Seiten 85–94, München, 2014. de Gruyter Oldenbourg. Zitiert auf Seite 22 und 145.
- [Sand16] M. Sand & D. Henrich. *Incremental Reconstruction of Planar B-Rep Models from Multiple Point Clouds*. The Visual Computer, Band 32, Nr. 6, Seiten 945–954, 2016. Zitiert auf Seite 58 und 97.
- [Sand17] M. Sand & D. Henrich. *Matching and Pose Estimation of Noisy, Partial and Planar B-Rep Models*. In Proceedings of the Computer Graphics International Conference, CGI '17, Seiten 30:1–30:6, Yokohama, Japan, 2017. ACM. Zitiert auf Seite 119.
- [Werner16] T. Werner, M. Gradmann, E. M. Orendt, M. Sand, M. Spangenberg & D. Henrich. *ENACT: An Efficient and Extensible Entity-Actor Framework for Modular Robotics Software Components*. 47th International Symposium on Robotics (ISR), 2016. Zitiert auf Seite 181.
- [Werner18] T. Werner, M. Sand & D. Henrich. *Sparse and Precise Reconstruction of Static Obstacles for Real-Time Path Planning in Human-Robot Workspaces*. 50th International Symposium on Robotics (ISR), 2018. Zitiert auf Seite 181.

Sonstige eigene Arbeiten

- [Sand13] M. Sand. *Erstellung Eines Unterstützenden Systems Zur Dreidimensionalen Raumerfassung Mit Einer Handgehaltenen Farb-/Tiefenkamera*. Masterarbeit, Universität Bayreuth, Bayreuth, 2013. http://www.ai3.uni-bayreuth.de/resypub/?mode=pub_show&pub_ref=sand213a. Zitiert auf Seite 22.

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die von mir angegebenen Quellen und Hilfsmittel verwendet habe.

Weiterhin erkläre ich, dass ich die Hilfe von gewerblichen Promotionsberatern bzw. –vermittlern oder ähnlichen Dienstleistern weder bisher in Anspruch genommen habe, noch künftig in Anspruch nehmen werde.

Zusätzlich erkläre ich hiermit, dass ich keinerlei frühere Promotionsversuche unternommen habe.

Bayreuth, den 26. Juli 2018

Maximilian Sand