UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4: Informatik

AGAS
Arbeitsgruppe Aktives Sehen

Schätzung planarer Flächen in verrauschten Tiefenbildern für den RoboCup Rescue Wettbewerb

Studienarbeit
im Studiengang Computervisualistik

vorgelegt von

Sarah Steinmetz

Betreuer:	Prof. Dr.-Ing. Dietrich Paulus, Institut für Computervisualistik, Fachbereich Informatik
Erstgutachter:	Prof. Dr.-Ing. Dietrich Paulus, Institut für Computervisualistik, Fachbereich Informatik
Zweitgutachter:	Dipl.-Inf. Johannes Pellenz, Institut für Computervisualistik, Fachbereich Informatik

Koblenz, im Februar 2006

Erklärung

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

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

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

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

Koblenz, den ..... Unterschrift

Übersicht

In dieser Studienarbeit wird ein Verfahren zur Extraktion eines Oberflächenbegrenzungsmodells aus einem Tiefenbild vorgestellt. Das Modell beschreibt die im Tiefenbild dargestellte Szene durch die Geometrie und die Topologie der planaren Flächen, die in der Szene gefunden werden. Die Geometrie ist gegeben durch die Angabe der Ebenengleichungen der gefundenen Flächen sowie der 3D–Koordinaten der Eckpunkte der Polygone, die diese Flächen beschreiben. Die Informationen über die Topologie der Szene besteht aus einer Nachbarschaftsliste, die für jede Fläche angibt, über welche Kante diese Fläche mit welcher anderen Fläche verbunden ist. Aufbauend auf einem Algorithmus zur Tiefenbildsegmentierung aus PUMA werden die Polygone bestimmt, die die Flächen der Szene beschreiben. Anschließend wird versucht, diese Polygone über Kanten und Eckpunkte zu verbinden, um ein möglichst geschlossenes Modell der Szene zu erhalten.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Problemstellung und Motivation

Die Motivation dieser Studienarbeit war die Unterstützung der Teilnahme der Arbeitsgruppe Aktives Sehen (AGAS) der Universität Koblenz-Landau beim RoboCup 2006 in Bremen (14.-18. Juni 2006). Bei der sogenannten Rescue League sollen Roboter in einem nachgestellten Erdbebenszenario nach Opfern suchen, deren Position bestimmen und diese an die Rettungsmannschaft übermitteln. Um dies zu ermöglichen, muss eine Umgebungskarte der Arena, in der sich der Roboter bewegt, erstellt werden. In Bild 1.1 ist der Roboter Robbie der AGAS beim Einsatz in der Rescue League Arena des RoboCup 2006 zu sehen. Ein möglicher Ansatz zur Erstellung einer 3D–Umgebungskarte ist, die Szene zunächst durch die Extraktion planarer Flächen aus Tiefenbildern zu vereinfachen. Die Approximation der Flächen durch Polygone kann als Grundlage für eine merkmalsbasierte Registrierung aufeinander folgender Tiefenbilder dienen. Solche Tiefenbilder können zum Beispiel mit Hilfe eines Stereokamerasystems erstellt werden. Parallel zu dieser Arbeit wurde daher von Peter Decker die Studienarbeit *Erstellung einer Tiefenkarte aus Stereobildern für den RoboCup Rescue Wettbewerb* bearbeitet.

Die konkrete Aufgabe der vorliegenden Arbeit war also, planare Flächen aus einem einzelnen Tiefenbild zu extrahieren und die dargestellte Szene durch geschlossene, planare Polygone zu approximieren. Da die Ergebnisse der Navigation des Roboters dienen sol-

Bild 1.1: Der Roboter Robbie der AGAS in der Rescue League Arena des RoboCup 2006

len, ist eine Auswertung in Echtzeit erforderlich. Die Programme sollten unter Linux in C++ erstellt werden und auf möglich vorhandene Algorithmen aus der Programmierumgebung für die Musteranalyse (PUMA) zurückgreifen.

1.2 Beitrag der Arbeit

Um möglichst bald Testmaterial zur Verfügung zu haben, wurden im Laufe der Entwicklung der Arbeit Tiefenbilder aus einer 3D–Laserkamera verwendet. Mehr Informationen zu dieser Kamera sind in Kapitel 5 und in [Zin01] zu finden. Neben der Installation der Kamera im Labor und der Einarbeitung in die Bedienung der Kamera, habe ich mich zunächst mit der Literaturrecherche zum Thema und der Untersuchung, welche Algorithmen aus PUMA eventuell verwendet werden können, beschäftigt. Dabei ergab sich, dass die Segmentierung von Tiefenbildern in planare Flächen bereits vorlag. Lediglich ein Teil der vorhandenen Algorithmen musste leicht angepasst werden (siehe Kapitel 3). Der Schwerpunkt dieser Arbeit wurde daher auf die Approximation der Begrenzungen dieser Flächen durch dreidimensionale, planare Polygone gelegt. Dazu werden zunächst die Konturen der segmentierten Regionen bestimmt. Inkrementelles Line Fitting auf den 2D–Konturpunkten in Pixelkoordinaten und die anschließende Übertragung auf die entsprechende 3D–Ebene, liefert lineare, geschlossene Konturen der einzelnen Flächen. Aufgrund von dünn verteilten Tiefenwerten bei Flächen, die stark von der Blickrichtung abgewandt sind, und im Bereich von Objektkanten, driften die eigentlichen Schnittkanten der 3D–Polygone auseinander. Daher werden im Weiteren potentielle Schnittkanten bestimmt und wieder aneinandergefügt, wobei anhand eines Zuverlässigkeitsmaßes entschieden wird, welche der

betroffenen Polyone dazu transliert und rotiert werden müssen. So wird die Orientierung von Polygonen, die aufgrund von dünn verteilten Tiefenwerten und Rauschen verfälscht wird, korrigiert. Die Ergebnisse wurden anhand verschiedener Kriterien experimentell überprüft.

1.3 Aufbau der Arbeit

Der Aufbau der Arbeit ist wie folgt: Im nächsten Kapitel erfolgt zunächst eine einführende Klärung des Begriffs *Tiefenbild*. Im Anschluss daran werden verschiedene Verfahren aus der Fachliteratur vorgestellt, die sich mit der Extraktion planarer Flächen aus Tiefenbildern beschäftigen. In Kapitel 3 wird dann das im Laufe der Studienarbeit entwickelte Verfahren theoretisch dargestellt. Die technische Umsetzung desVerfahrens mit einer Übersicht der implementierten Klassen wird in Kapitel 4 vorgestellt. Hier wird auch die zur Demonstration entwickelte grafische Benutzungsoberfläche erläutert. In Kapitel 5 folgt eine Beschreibung der Experimente und Ergebnisse. Abschließend werden die Ergebnisse der Arbeit nochmals zusammengefasst.

Kapitel 2

Extraktion planarer Flächen aus Tiefenbildern

In diesem Kapitel werden zunächst grundlegende Begriffe geklärt. Im Anschluss daran werden kurz Verfahren aus der Fachliteratur vorgestellt, die sich mit der Segmentierung von Tiefenbildern in planare Regionen befassen. Da im Laufe der Studienarbeit kein eigenes Segmentierungsverfahren entwickelt, sondern auf einem vorhandenen aufgebaut wurde, wird in Abschnitt 2.3 stärker auf die 3D–Modell Extraktion aus Tiefenbildern eingegangen. Abschließend wird der verwendete Segmentierungsalgorithmus aus der PUMA-Bibliothek vorgestellt.

2.1 Tiefenbilder

Für die 3D–Rekonstruktion sind Tiefenbilder von besonderer Bedeutung, da sie direkt Informationen über die dreidimensionale Struktur der abgebildeten Szene liefern. Nach [TV98] und [Bes88] repäsentiert ein einzelner Bildpunkt eines Tiefenbildes den Abstand zwischen einem sichtbaren Punkt in der Szene und einem bekannten Bezugskoordinatensystem. Für die Repräsentation von Tiefenbildern gibt es zwei grundlegende Möglichkeiten: Die Tiefenwerte können zum einen in Form einer Punktewolke vorliegen, das heißt als ungeordnete Liste von 3D–Koordinaten in einem gegebenen Koordinatensystem. Die-

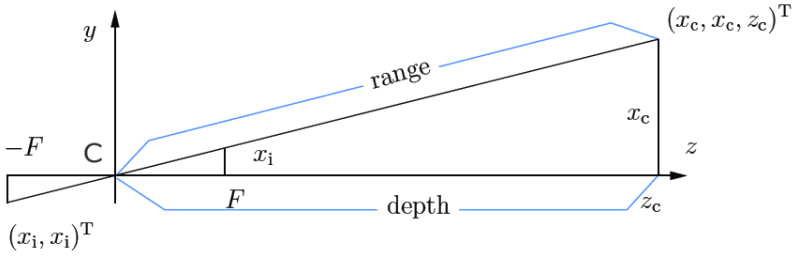

Bild 2.1: Das Lochkameramodell

se Darstellung wird auch als *xyz* Form bezeichnet. Aufgrund der fehlenden räumlichen Ordnung ist eine Verarbeitung dieser Daten schwieriger als bei der zweiten geordneten Darstellungsform: Ein Tiefenbild der Form r_{ij} ist eine Matrix von Tiefenwerten, die für jeden Bildpunkt die Entfernung des aufgenommenen Punktes enthält [TV98]. An dieser Stelle wird nicht genauer beschrieben, zu welchem Bezugspunkt diese Tiefe definiert ist. Bei [Bes88] werden zwei mögliche Unterscheidungen aufgeführt: Bei *orthografischen* r_{ij} Tiefenbildern wird der Abstand eines Punktes im Raum entlang von Strahlen, die orthogonal zur Bildebene verlaufen, gemessen. Diese Definition ist auch bei [JJ90] zu finden. Die meisten aktiven optischen Tiefensensoren (s.u.) liefern dagegen *polare* Tiefenwerte in einem *sphärischen* Koordinatensystem. Ein Wert im Tiefenbild entspricht dabei der Länge des Sichtstrahls zu dem durch einen Pixel abgebildeten Punkt. Der Unterschied wird in Bild 2.1 deutlich. Im dargestellten Lochkameramodell entspricht die Länge des Sichtstrahls der Entfernung (engl. *range*) des betrachteten Punktes zum Kamerazentrum C, also zum Ursprung des Kamerakoordinatensystems. Als Tiefe (engl. *depth*) wird der orthogonale Abstand des Punktes zur *Principal Plane*, also zur xy–Ebene des Kamerakoordinatensystems, bezeichnet. Dies entspricht gerade der Komponente z^{C} des Punktes im Kamerakoordinatensystem.

Die Begriffe *depth* und *range* werden in der Literatur meist synonym verwendet, sodass immer aus dem Kontext geschlossen werden muss, welches Maß gemeint ist. So werden bei [TV98] unter anderem folgende Synonyme für *range image* aufgeführt: *depth image*, *depth map*, *xyz map*, *surface profile*, *2,5–D image*. Weitere mögliche Bezeichnungen sind *range map*, *range picture*, *rangepic*, *3–D image*, *digital terrain map*, *topographic map*,

xyz point list, *contour map* [Bes88]. Wenn in dieser Arbeit von Tiefe oder Entfernung gesprochen wird, ist immer die Entfernung zur Bildebene gemeint, also die karthesische Koordinate z^{C} des Punktes im Kamerakoordinatensystem.

Ein orthografisches Tiefenbild der Form r_{ij} enthält zunächst keine expliziten Informationen über die x– und y–Koordinaten zu einem Tiefenwert. Für die reine Segmentierung des Tiefenbilds in planare Flächen wird oft mit dieser Darstellung gearbeitet, indem die Bildwerte als eine Kurve über der Bildebene betrachtet werden. Die geometrischen Eigenschaften, wie die Planarität einer Fläche, sind auch in dieser Darstellung korrekt wiedergegeben [TV98]. Für die 3D–Rekonstruktion werden aber die Koordinaten im Raum benötigt. Ein Tiefenbild, bei dem für jeden Bildpunkt die karthesischen Koordinaten des abgebildeten Punktes gegeben sind, wird bei [SA02] als eine Menge $\boldsymbol{r} = \{r(i,j), i = 1 \ldots N, j = 1 \ldots M\}$ von 3D–Punkten definiert. Dabei ist $r(i,j)$ der 3D–Punkt, der dem Bildpunkt (i,j) entspricht. Diese Notation wird auch in dieser Arbeit verwendet.

Die Daten polarer Tiefenbilder werden für die Verarbeitung in karthesische Koordinaten umgerechnet [JJ90], [Bes88]. Die für diese Arbeit verwendete 3D–Laserkamera misst die Entfernungen zunächst in Polarkoordinaten. Der Bildprozessor der Kamera rechnet diese anhand der Kalibrierungsinformationen der Kamera in karthesische Koordinaten um. Bild 2.2 zeigt ein Tiefenbild aus der verwendeten Laserkamera in verschiedenen Darstellungen. Im kleinen Bild sind die Tiefenwerte als Grauwerte dargestellt. Punkte, die der Kamera nah sind, sind dunkel, weit entfernte Punkte heller. Im großen Bild sind die Daten des Tiefenbildes als weiße 3D–Punke im Raum zu sehen. Das grüne und das rote Gitter deuten die yz–Ebene bzw. die xz–Ebene an.

Die Tiefenbilder der verwendeten Laserkamera liegen also bereits in den benötigten karthesischen Koordinaten vor. Das zu entwickelnde System sollte aber auch mit Tiefenbildern der Form r_{ij} als Eingabe arbeiten können. Im Folgenden wird daher beschrieben wie aus einem orthografischen Tiefenbild der Form r_{ij} die zugehörigen x und y Koordinaten berechnet werden können.

Gegeben:

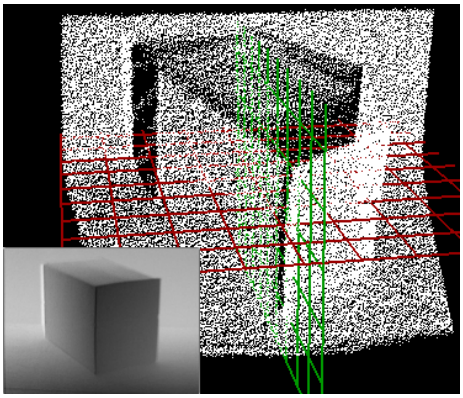

Bild 2.2: Darstellung der Werte eines Tiefenbildes als Grauwerte und 3D–Darstellung des Tiefenbildes

- Kameraparameter[1]: Brennweite F, Hauptpunkt $\boldsymbol{H} = (H_x, H_y)$
- Bildkoordinaten $(x_i^{\mathsf{I}}, y_i^{\mathsf{I}})^{\mathsf{T}}$
- Tiefenwert, also Kamerakoordinate z_i^{C}

Gesucht:

- Kamerakoordinaten x_i^{C} und y_i^{C}

Formel 2.1 beschreibt die zentralperspektivische Abbildung von Kamera- in Bildkoordinaten [HZ00].

$$(x^{\mathsf{C}}, y^{\mathsf{C}}, z^{\mathsf{C}})^{\mathsf{T}} \mapsto (F\frac{x^{\mathsf{C}}}{z^{\mathsf{C}}} + H_x, F\frac{y^{\mathsf{C}}}{z^{\mathsf{C}}} + H_y)^{\mathsf{T}} \tag{2.1}$$

Die gesuchten Kamerakoordinaten berechnen sich also durch

$$x_i^{\mathsf{C}} = \frac{z_i^{\mathsf{C}}}{F}(x_i^{\mathsf{I}} - H_x) \tag{2.2}$$

$$y_i^{\mathsf{C}} = \frac{z_i^{\mathsf{C}}}{F}(y_i^{\mathsf{I}} - H_y) \tag{2.3}$$

[1] Die Kameraparameter sind aus der Kamerakalibrierung bekannt.

Die optischen Sensoren zur Aufnahme von Tiefenbildern können in zwei Klassen unterteilt werden [TV98]: *Aktive* Tiefensensoren üben einen bestimmten aktiven Einfluss auf die Szene aus, um Abstände messen zu können. Die verwendete Laserkamera sendet zum Beipiel Laserimpulse aus und misst wieviel des Signals in einer bestimmten Zeit reflektiert wird, um daraus den Abstand zu berechnen [Zin01]. *Passive* Tiefensensoren rekonstruieren die Tiefeninformation anhand von Intensitätsbildern, zum Beipiel mit Hilfe von Stereobildern.

2.2 Segmentierung von Tiefenbildern

Bei der Segmentierung wird ein Bild in Regionen zusammenhängender Pixel, deren Werte ein bestimmtes Homogenitätskriterium erfüllen, unterteilt. Bei der Tiefenbildsegmentierung sollen alle Punkte innerhalb einer Region auf einer gemeinsamen Oberfläche liegen [HJBJ+96]. Das Ergebnis einer Tiefenbildsegmentierung ist je nach Verfahren eine Beschreibung des Bildes durch die Kanten, die homogene Regionen voneinander trennen, oder eine Liste der Regionen mit ihren zugehörigen Pixeln [CC05]. Meist werden dabei für die Regionen Oberflächenfunktionen berechnet, die die Oberflächen approximieren [Liu93]. Grundsätzlich können die Verfahren danach unterschieden werden, ob das Bild dabei in gekrümmte oder planare Oberflächen, oder auch beide segmentiert wird. In dieser Arbeit sind nur planare Beschreibungen der Oberflächen von Interesse, um eine möglichst einfache Beschreibung der Szene zu erhalten. Gekrümmte Oberflächen können durch mehrere planare Flächen approximiert werden.

Es gibt viele verschiedene Verfahren zur Segmentierung von Tiefenbildern. In [HJBJ+96] ist ein experimenteller Vergleich verschiedener Methoden zur Extraktion planarer Flächen zu finden. Eine aktuellere Übersicht über verschiedene Methoden liefert [CC05]. Hier wird eine Klassifikation in *kantenbasierte, regionenbasierte* und *hybride* Verfahren vorgenommen. Im ersten Fall werden Kanten im Bild gesucht, also Unstetigkeiten in den Tiefenwerten. Meist wird dabei zwischen *Stufenkanten* (*step, jump edges*) und *Dachkanten* (*roof, crease edges*) unterschieden [BS92]. Eine Stufenkante entspricht einem Sprung in den Tiefenwerten, entsteht also, wenn ein Objekt ein anderes oder ein Teil von sich selbst verdeckt. Für die Detektion von Stufenkanten orientiert man sich oft an Kantendetektoren,

die für Intensitätsbilder entwickelt wurden (z.B. Sobel, Prewitt Operator). Dachkanten resultieren aus der unterschiedlichen Orientierung aneinander grenzender Flächen. Hier ist eine Detektion zum Beispiel durch die Berechnung der Normalen für jeden Pixel möglich. Ein Dachkantenpixel liegt vor, wenn der Winkel zwischen der Normalen eines Pixels und der seiner Nachbarn einen Schwellwert übersteigt. Andere Verfahren stellen die Verwendung morphologischer Operatoren zur Detektion und Klassifikation von Kantenpixeln vor. [CC05], [BS92]
Der Nachteil der kantenbasierten Verfahren ist, dass die Methoden meist in zerstückelten Kanten resultieren, die auch bei Verwendung von Konturschließungsverfahren nicht immer vollständig geschlossen werden können [BS92]. In Kapitel 3 wird außerdem gezeigt, dass eine Klassifikation in Sprung- und Dachkanten bei den in der Arbeit verwendeten Tiefenbildern zu Problemen führen könnte, da die Tiefenwerte besonders in den Bereichen von Objektkanten dünn verteilt und stark verrauscht sind.

Regionenbasierte Verfahren sind weiter verbreitet, besonders wenn, wie in dieser Arbeit, Interesse an der Oberflächenrepräsentation der segmentierten Regionen besteht. Einige Verfahren orientieren sich an [BJ88] und führen eine Berechnung der Krümmung in jedem Pixel durch, um die Pixel anhand verschiedener Oberflächentypen (z.B. planar, konvex, konkav) zu klassifizieren. Beim anschließenden Region Growing Verfahren werden iterativ Oberflächenanpassungen benachbarter Punkte der selben Klasse durchgeführt. Verbreitet ist auch der Ansatz durch lokale Ebenenanpassung in der Nachbarschaft, zunächst die Normale für jeden Pixel zu berechnen. Regionen von Pixeln mit ähnlicher lokaler Ebenenorientierung können wie bei [BS92] und [Koc96] durch Clustering–Verfahren bestimmt werden, oder auch durch Region Growing Verfahren, worauf in Abschnitt 2.4 genauer eingegangen wird. Für eine Übersicht alternativer Methoden sei auf [CC05] verwiesen. Regionenbasierte Verfahren haben den Vorteil, dass sie geschlossene Regionen liefern, die allerdings meist ungenaue Grenzlinien besitzen. Daher werden auch oft hybride Verfahren eingesetzt, die beide Ansätze kombinieren.

An dieser Stelle soll nur diese kurze Übersicht über Verfahren zur Segmentierung von Tiefenbildern gegeben werden. Im Rahmen dieser Arbeit wurde keine eigene Segmentierung entwickelt, da bereits ein regionenbasiertes Verfahren in der zu verwendenden Programmierumgebung PUMA vorlag. Das Verfahren, auf dessen Ergebnisse diese Arbeit aufbaut,

wird kurz in Abschnitt 2.4 vorgestellt.

2.3 3D–Modell Extraktion aus Tiefenbildern

Die Segmentierung eines Tiefenbildes liefert zunächst nur eine Einteilung in Regionen, die Flächen in der Szene repräsentieren. Bei regionenbasierten Verfahren werden die Oberflächen mathematisch beschrieben, aber die Grenzenlinien dieser Flächen werden nicht bestimmt. Kantenbasierte Verfahren markieren zwar Kantenpixel im Bild, liefern aber auch keine geschlossene, abstrakte Beschreibung von Flächenkonturen im dreidimensionalen Raum. Für die abstrakte dreidimensionale Beschreibung der Szene, die von einem Tiefenbild dargestellt wird, ist eine Weiterverarbeitung der segmentierten Bilder erforderlich. Alle Verfahren, die im Folgenden beschrieben werden, basieren auf einer regionenbasierten Segementierung des Tiefenbilds. Ihr Ziel ist ein 3D–Modell der Szene zu erstellen, zum Beispiel ein Drahtgittermodell oder ein Flächenbegrenzungsmodell.

Bei [BS92] werden nach der Segmentierung in Regionen planarer Flächen zunächst die Konturpixel der Regionen durch einen Scanline Algorithmus bestimmt. Pixel, die einen Schnitt der Region mit einer Scanline (Bildzeile) darstellen, werden als Konturpixel markiert. Unter Verwendung eines gradientenbasierten Kantenoperators werden die Konturpixel als Stufen- oder Dachkantenpixel klassifiziert. Zur Bestimmung der Dachkanten werden die Schnittlinien benachbarter Flächen berechnet und auf die Bildebene projiziert. Die gesuchte Kante ist das Segment der Schnittlinie, dessen Projektion in direkter Nachbarschaft von zuvor markierten Dachkantenpixeln liegt. Durch den Schnitt von drei Ebenen, deren Regionen benachbart sind, werden die zugehörigen Kantenpunkte korrigiert, sodass sich die Kanten im Schnittpunkt treffen. Stufenkanten werden durch 2D Hough Clustering und Linienverfolgung auf den Stufenkanten–Konturpixeln bestimmt. Zu den Endpunkten der Kante können mit Hilfe der aus der Segmentierung bekannten Ebenengleichung die entsprechenden 3D–Koordinaten berechnet werden. Nahe beieinander liegende Endpunkte von Stufenkanten der selben Region werden durch den Schnitt der Kanten vereint. Abschließend werden in verbleibende Lücken in den Flächenbegrenzungen überbrückende Kanten eingefügt.
Der vorgestellte Ansatz, Objektkanten durch den Schnitt von Flächen zu bestimmen, wird

in einigen Verfahren verwendet. Bei [Liu93] werden zunächst die Regionengrenzlinien durch ein Konturverfolgungsverfahren bestimmt. Ein Teil der Oberflächengrenzlinien wird durch den Schnitt der angrenzenden Flächen (Ebenen oder Quadriken) bestimmt. Für die Beschreibung der übrigen Stufengrenzlinien werden an die übrigen Oberflächenkonturpunkte Kegelschnitte angepasst, sodass die Grenzlinie einer Fläche stückweise durch Parameterformen beschrieben wird. Das Verfahren resultiert in einer symbolischen Beschreibung der dargestellten Szene durch ein Flächenbegrenzungsmodell: Zum einen werden die Relationen von Regionen mit einer Gebietshierarchie beschrieben, der entnommen werden kann, welche Regionen ineinander liegen. Außerdem werden Nachbarschaftsinformationen in einer Matrix abgelegt. Die Objektoberfläche jeder Region wird durch eine Flächengleichung und die Grenzliniensegmente beschrieben. Für jedes Schnittlinien–Segment wird dabei die angrenzende Fläche referenziert.
Auch bei [FEF95] und [Koc96] werden Grenzlinien und Eckpunkte durch den Schnitt von Flächen berechnet.
Das in [HBG94] beschriebene Verfahren resultiert in einem Flächenbegrenzungsmodell. Zunächst wird dazu ein Regionenbeziehungsgraph erstellt, der wiedergibt, welche Regionen über Stufenkanten und Eckpunkte verbunden sind. Dazu werden auch hier die Konturpixel aneinander grenzender Regionen als Stufen- oder Dachkantenpixel klassifiziert: Die 3D–Koordinaten der gegenüberliegenden Konturpixel benachbarter Regionen werden entlang ihres Ortsvektors auf die Ebene, die an die Punkte der jeweiligen Region angepasst wurde, projiziert. Wenn die Abstände dieser Punkte zu der Schnittlinie der zwei Ebenen einen Schwellwert nicht übersteigen, werden die Pixel als Dachkantenpixel markiert. Welche Regionen, genauer Ebenen, über Eckpunkte verbunden sind, wird aus den Kantenbeziehungen abgeleitet. Um die Geometrie der Szene zu rekonstruieren, werden die Ebenen, die über einen Dachkante oder einen Eckpunkt verbunden sind, geschnitten. Dabei werden sogar die Schnittpunkte von mehr als 3 Ebenen bestimmt. Da sich diese Ebenen nicht genau in einem Punkt schneiden, werden sogenannte *Glue Patches* eingesetzt, kleine Polygone, die die beteiligten Kanten verbinden. So wird auch in anderen Problemfällen verfahren, wenn zum Beispiel zwei annähernd parallele Ebenen laut Regionenbeziehungsgraph eine gemeinsame Kante besitzen, die Projektion deren Schnittlinie sich aber weit außerhalb der beiden Regionen befindet.

In Kapitel 3 wird gezeigt, dass die hier angesprochenen Verfahren der Klassifikation von Konturpixeln als Stufen- oder Dachantenpixel, sowie die Bestimmung von Grenzlinien und Eckpunkten durch den Schnitt von Flächen bei den in dieser Arbeit verwendeten Tiefenbildern problematisch ist.

2.4 Verwendete Algorithmen aus PUMA

Bei der Recherche zum Thema und bei der Untersuchung, welche Algorithmen aus PUMA eventuell verwendet werden können, wurde festgestellt, dass die Segmentierung von Tiefenbildern bereits in der Programmierumgebung vorlag. Es handelt sich dabei um die Implementierung aus einer Studienarbeit an der Universität Erlangen zum Thema *Oberflächensegmentierung in Tiefenbildern* [Zin01]. Das Verfahren behandelt zwar auch gekrümmte Flächen, indem diese mit Hilfe von Quadriken angepasst werden, aber durch den modularen Aufbau der Implementation kann diese Option leicht ausgelassen werden, sodass die Szene nur durch planare Flächen approximiert wird. Das Verfahren orientiert sich an [Liu93], ist aber für die Verwendung von Tiefenbildern aus der 3D–Laserkamera, die in dieser Arbeit eingesetzt wurde, angepasst. Nach einer qualitativen Untersuchung der Ergebisse bei verschiedenen Aufnahmen, wurde entschieden, die vorhandenen Klassen einzubinden. Im Folgenden wird das Verfahren nur kurz erläutert, genaue Informationen sind in [Zin01] zu finden.

Zuerst wird das Tiefenbild mit einem kantenerhaltenden Filter geglättet. Beim *Local Plane Fitting* wird für jeden Bildpunkt die Normale berechnet. Dafür wird mit den 3D–Punkten, deren zugehörige Bildpunkte sich im Nachbarschaftsfenster befinden, eine Ebenenanpassung durchgeführt. Darauf folgt ein iteratives Regionenwachstumsverfahren. Als Regionenkern wird dabei der noch nicht zugewiesene Punkt mit geringstem Anpassungsfehler gewählt. Ein benachbarter Punkt wird in die Region aufgenommen, wenn der Abstand zur Ebene der Region nicht zu groß ist und der Winkel zwischen der Normale der Ebene und der Normalen des Punktes unter einem definierten Schwellwert liegt. Die Ebene der vegrößerten Region wird durch eine Ebenenanpassung durch die zugehörigen Punkte aktualisiert. Wenn keine neuen Regionen mehr gebildet werden können, folgt eine Regionenverschmelzung um aneinander grenzende Regionen mit annähernd gleicher Orientierung

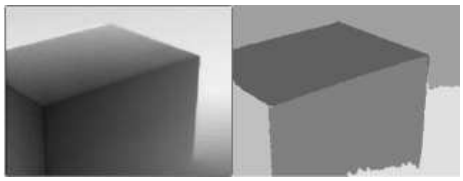

Bild 2.3: Tiefenbild und Regionenbild als Ergebnis der Segmentierung

zu vereinen. Zum Schluss wird eine Ausreißerelimierung durchgeführt, um auch Punkte, die bisher keiner Region zugewiesen werden konnten, einer Region zuzuordnen. Das Ergebnis des Verfahren ist also ein Regionenbild, siehe zum Beispiel Bild 2.3, das die zu einer gemeinsamen Ebene gehörenden Punkte markiert und eine Liste von Ebenen, die die Oberflächen der Regionen annähern.

Kapitel 3

Verfahren zur Extraktion eines B–Rep Modells

In diesem Kapitel wird das im Laufe dieser Studienarbeit entwickelte Verfahren zur Extraktion eines polygonalen 3D–Modells aus einem Tiefenbild erläutert. Bevor auf die neu entwickelten Komponenten eingegangen wird, beschreibt der folgende Abschnitt zunächst, warum und wie der vorhandene Algorithmus zur Segmentierung, genauer die Ausreißereliminierung, angepasst wurde. Abschnitt 3.2 liefert dann eine Übersicht über die benötigten Eingabedaten und die einzelnen Komponenten des Verfahrens. In den folgenden Abschnitten werden diese genauer erläutert.

3.1 Anpassung der vorhandenen Algorithmen

Erst bei der Entwicklung der Konturextraktion (Kapitel 3.4) stellte sich heraus, dass die Ausreißereliminierung des vorhandenen Algorithmus (Kapitel 2.4) angepasst werden musste. In Bild 3.1 sind die 3D–Konturpunkte der segmentierten Regionen in verschiedenen Farben dargestellt. Bei der hellgrünen Kontur der vorderen Fläche des Würfels ist zu sehen, dass die Punkte auf der linken Seite nicht der tatsächlichen Flächengrenze entsprechen, sondern fehlerhafte Messwerte darstellen, die überhaupt keiner Oberfläche angehören. Dies zeigt sich im Vergleich mit der dargestellten Grauwertbilddarstellung der Tiefenwerte. Das Vorgehen der ursprünglichen Ausreißerliminierung führt aber zu dem Ergebnis,

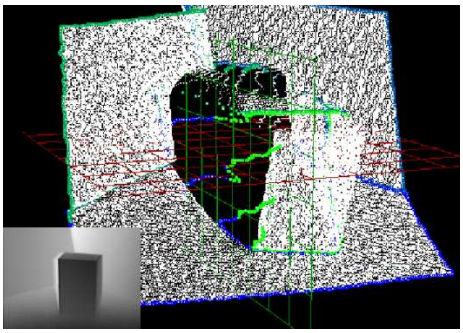

Bild 3.1: Die 3D–Konturpunkte der Regionen nach der ursprünglichen Ausreißereliminierung und die Grauwertbilddarstellung des Tiefenbilds

dass fälschlicherweise alle Bildpunkte einer Region zugeordnet werden.
Das ursprüngliche Verfahren ist in vier Phasen gegliedert, wobei bei Segmentierung in planare Flächen nur die folgenden ersten zwei relevant sind. Zunächst wird die Fläche bestimmt, zu der der Ausreißer den kleinsten Abstand hat. Falls der Bildpunkt mit einem Punkt aus der zugehörigen Region benachbart ist und der Abstand unter einem Schwellwert liegt, wird der Punkt dieser Region zugeordnet. In der zweiten Phase wird der Ausreißer der benachbarten Region zugeordnet, zu deren Fläche er den geringsten Abstand hat. [Zin01]
Das iterative Vorgehen in der zweiten Phase führt zu den in Bild 3.1 dargestellten ”Ausreißer-Ketten”. Das Verfahren wurde daher wie folgt vereinfacht: Ein Ausreißer wird der am wenigsten entfernten Fläche zugeordnet, falls er sich in der 8er–Nachbarschaft eines Punktes der zugehörigen Region befindet. Der Schwellwert für den maximal erlaubten quadratischen Abstand zu der Fläche wurde von 10000 mm auf 2500 mm herabgesetzt. Dieser Wert von maximal 5 cm Abstand zur Ebene entspricht etwa der vom Hersteller angegebenen Fehlertoleranz der Messwerte [Zin01]. Zusätzlich wird überprüft, ob der maximale 3D–Abstand des Ausreißers zu allen Punkten in seiner 8er–Nachbarschaft, die der selben Region angehören, nicht zu groß ist. So werden die genannten ”Ausreißer-Ketten” verhindert. In Bild 3.2 ist ein Vergleich der Ergebnisse der beiden Verfahren, links das ursprüngliche, rechts das neue, zu sehen. Die verbliebenen Ausreißer sind rot dargestellt, die 3D–Konturpunkte der Regionen bunt.

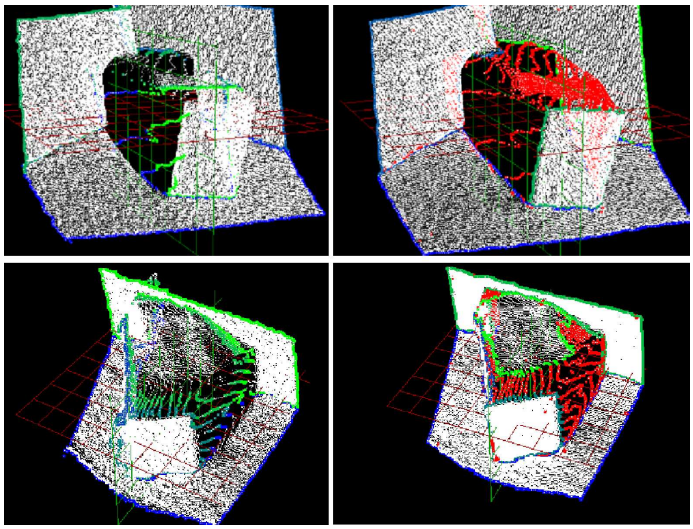

Bild 3.2: Vergleich der Verfahren zur Ausreißereliminierung, links das Ergebnis des ursprünglichen Verfahrens, rechts das Ergebnis des neuen Verfahrens, Ausreißer sind rot, 3D–Konturpunkte der Regionen bunt dargestellt

3.2 Übersicht über das entwickelte Verfahren

Die in Kapitel 2.3 vorgestellten Verfahren haben eine Gemeinsamkeit: In einem frühen Verarbeitungsschritt wird eine Klassifizierung von Konturpunkten in Stufen- oder Dachkantenpunkte durchgeführt, um die Beziehungen zwischen Regionen zu bestimmen. Im Anschluss werden dann je nach Kantenart, die Flächenbegrenzungen berechnet. Ein ähnliches Vorgehen wäre auf Basis der in dieser Arbeit verwendeten Segmentierungsergebnisse unzuverlässig: Bild 3.2 zeigt, dass nach der Ausreißerliminierung noch Punkte vorhanden sind, die fälschlicherweise keiner Fläche zugeordnet wurden. Dies liegt an der dünnen Verteilung der Tiefenwerte im Bereich von Kanten, wie bei der Oberseite der Box im vierten Bild gut zu sehen ist. Hier sind die Tiefenwerte so dünn verteilt, wie in Bereichen, in denen gar keine Fläche im Sichtbereich der Kamera liegt, wie die rechte Seite der Box. Diese Punkte stellen Messfehler dar, sowie auch die Punkte, die in der oberen Fläche bis zur hinteren Wand verlaufen. Besonders in Bereichen von Objektkanten und bei Flächen, deren Normale einen großen Winkel zur Blickrichtung aufweist, sind die Tiefenwerte stark ver-

rauscht. Die vordere Kante der dargestellten Box würde zum Beispiel bei Untersuchung der Nachbarschaft nicht als Dach- sondern als Stufenkante klassifiziert, da die Punkte der oberen Fläche durch die Unterbrechung durch Ausreißerpunkte, zu weit entfernt sind. In dieser Arbeit wird daher keine Klassifizierung von Grenzpunkten vorgenommen. Stattdessen werden hier zuerst die Polygone, die die einzelnen Flächen repräsentieren, bestimmt, und erst anschließend werden Beziehungen zwischen einzelnen Flächen, also gemeinsame Kanten und Eckpunkte, ermittelt.

Das Datenflussdiagramm in Bild 3.3 zeigt den Zusammenhang der einzelnen Komponenten des entwickelten Verfahrens.[1] Das Verfahren erwartet als Eingabe ein *Tiefenbild* und die *Kameraparameter* der Kamera, die zur Berechnung des Tiefenbildes verwendet wurde, oder ein oder mehrere *3D–Bilder* aus der Laserkamera. Das *Tiefenbild* liegt dabei in der r_{ij} Form vor, also als Matrix von Tiefenwerten. Bei der *Bildkonvertierung* werden die zur Verarbeitung benötigten karthesischen 3D–Koordinaten, wie in Kapitel 2.1 beschrieben, berechnet. Das resultierende *3D–Bild* ist ein Tiefenbild $\boldsymbol{r} = \{r(i,j), i = 1 \ldots N, j = 1 \ldots M\}$, siehe Kapitel 2.1. Bilder von der Laserkamera liegen direkt in dieser Form vor, müssen aber u.U. bei der *Vorverarbeitung* gemittelt werden. Die *Segmentierung*[2] liefert ein *Regionenbild* und eine *Oberflächenliste* zurück. Das *Regionenbild* speichert für jeden Pixel die Kennung der Region, der er zugeordnet wurde, oder eine besondere Kennung, falls der Bildpunkt keiner Region zugeordnet wurde. Die *Oberflächenliste* enthält die Ebenengleichungen der Flächen, die an die 3D–Punkte der jeweiligen Region angepasst wurden. Die *Konturextraktion* ermittelt die Konturen der einzelnen Regionen. Die *Konturliste*, sowie die *Oberflächenliste* und das *3D–Bild* werden bei der *3D–Polygon–Extraktion* verwendet, um die 3D–Polygone zu berechnen, welche die Begrenzungen der bekannten Oberflächen beschreiben. Bei der *Polygonverbindung* wird versucht, die freien Polygone aus der *3D–Polygonliste* zu verbinden. Da dabei die Orientierung von Flächen verändert werden kann, werden die Ebenengleichungen in der *Oberflächenliste* entsprechend angepasst. Das Ergebnis des Verfahrens ist das extrahierte Modell als sogennannte *Boundary Representation*. Diese Datenstruktur wird in Kapitel 3.7 genauer erläutert.

[1] Prozesse, die nicht zum entwickelten System gehören sind im Bild grau dargestellt.

[2] Der Prozess ist im Bild grau dargestellt, da das Verfahren aus der PUMA Bibliothek übernommen wurde.

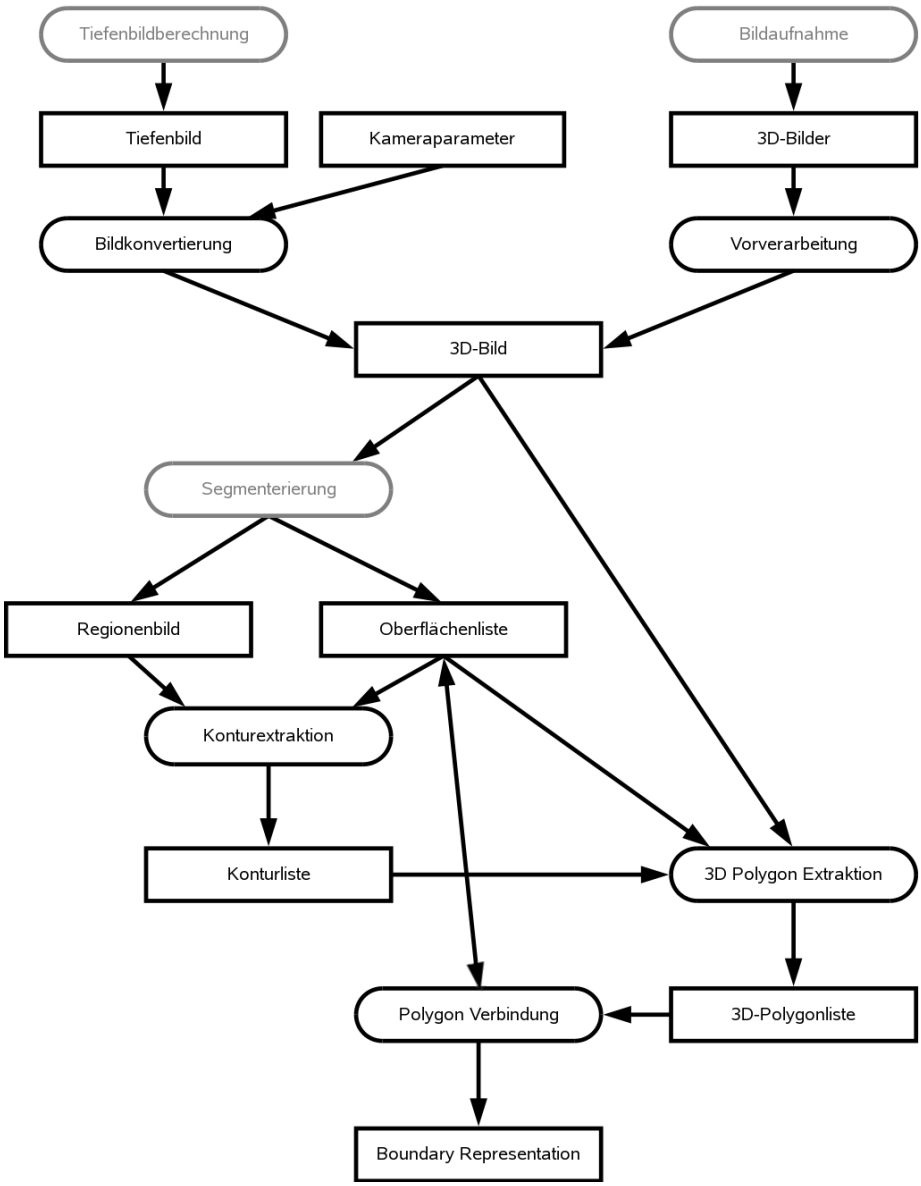

Bild 3.3: Das Datenflussdiagramm zum entwickelten System

3.3 Vorverarbeitung

Ein einzelnes 3D–Bild aus der Laserkamera ist stark verrauscht. Bild 3.4 zeigt die Tiefenwerte eines 3D–Bildes als Grauwertbild und die 3D–Punkteverteilung im Raum. Dabei ist

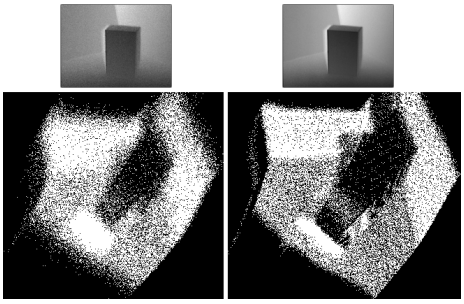

Bild 3.4: Das 3D–Bild vor und nach der Mittelung. Oben: Darstellung der Tiefenwerte als Grauwerte, unten: die 3D–Punkte im Raum

die 3D–Szene so gedreht, dass man sie von oben betrachtet. Die Szene besteht aus einem Karton vor 2 flachen Wänden. Auf der linken Seite ist das starke Rauschen der Tiefenwerte zu erkennen. Eine Segmentierung in planare Flächen ist hier nicht möglich [Zin01]. Auf der rechten Seite ist dieselbe Szene mit 16-facher Mittelung zu sehen. Die Aufnahmesoftware der Laserkamera bietet die Möglicheit, eine Bildsequenz aufzunehmen, die dann gemittelt wird. Diese Funktion konnte, vermutlich aufgrund eines Kabeldefekts, leider nicht genutzt werden. Daher war es notwendig die Mittelung der Tiefenwerte selbst umzusetzen. Für eine Sequenz von Tiefenbildern, die die selbe Szene darstellen, wird der gefilterte Tiefenwert z' für ein Pixel einfach als der Mittelwert der z–Koordinaten an diesem Bildpunkt berechnet. Die zugehörige x– und y–Koordinate (x' und y') wird mit Hilfe der x–, y– und z–Koordinate (x, y und z) eines Tiefenbildes der Sequenz so berechnet, dass sich der neue 3D–Punkt auf demselben Sichtstrahl befindet:

$$x' = x * \frac{z'}{z} \tag{3.1}$$

$$y' = y * \frac{z'}{z} \tag{3.2}$$

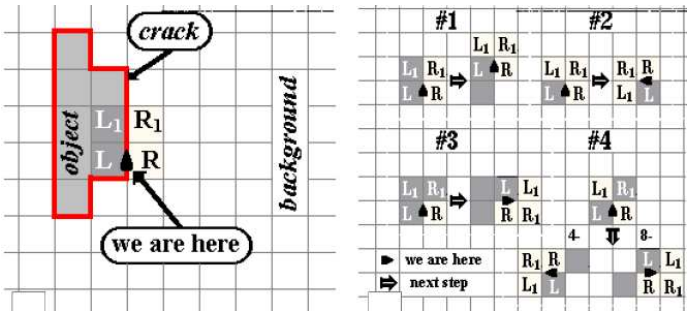

Bild 3.5: Konturverfolgungsalgorithmus Crack–Following, Abb. aus [Kin97]

3.4 Konturextraktion im 2D

Bei der Konturextraktion werden die Konturpixel der segmentierten Regionen und somit die 3D–Oberflächenbegrenzungspunkte der zugehörigen Flächen ermittelt. Für die späteren Verarbeitungsschritte soll eine Kontur als sequentielle Liste von n Bildkoordinaten beschrieben werden, in der jeder Bildpunkt nur einmal abgelegt ist:

$$\boldsymbol{B} = \{(i_0, j_0\}, (i_1, j_1), \ldots, (i_n, j_n)). \tag{3.3}$$

Die 3D–Kontur ergibt sich direkt aus den zu den Bildpunkten gehörenden 3D–Punkten:

$$\boldsymbol{B3d} = \{r(i_0, j_0), r(i_1, j_1), \ldots, r(i_n, j_n)\}. \tag{3.4}$$

Für die Konturverfolgung wurde das *Crack–Following* Verfahren aus [Kin97] implementiert. Ein Konturpunkt ist dabei ein Punkt der Region, in dessen 8–er Nachbarschaft mindestens ein Punkt ist, der nicht in der untersuchten Region liegt. Ausgehend von einem Startpunkt auf der Kontur, wird diese sequentiell gegen den Uhrzeigersinn verfolgt. Eine bestimmte Position auf der Kontur wird dabei durch den Punkt L links der Kontur, innerhalb der Region, und den Punkt R rechts der Kontur, außerhalb der Region, definiert. Anhand der zwei in der jeweiligen Laufrichtung folgenden Punkte L_1 und R_1 wird der neue Konturpunkt L_{new}, wie in Bild 3.5 dargestellt, ermittelt:

- Wenn L_1 innerhalb und R_1 außerhalb: $\mathsf{L}_{\text{new}} = \mathsf{L}_1$, $\mathsf{R}_{\text{new}} = \mathsf{R}_1$

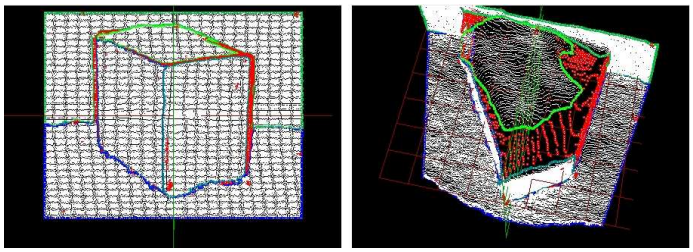

Bild 3.6: Konturpunkte der Flächen, links: orthografische Projektion auf die xy–Ebene, rechts: Punkte im Raum

- Wenn L_1 und R_1 außerhalb: $L_{new} = L$, $R_{new} = L_1$
- Wenn L_1 und R_1 innerhalb: $L_{new} = R_1$, $R_{new} = R$
- Wenn L_1 außerhalb und R_1 innerhalb: $L_{new} = R_1$, $R_{new} = R$

Das Verfahren endet, wenn der Startpunkt wieder erreicht wird.
Für die Bestimmung der Pixelposition des Startpunktes wird die Bounding Box[3] der betrachteten Region genutzt. Von der unteren rechten Ecke der Bounding Box werden die Pixel nach links durchlaufen, bis ein Punkt gefunden wird, der sich in der Region befindet. Dieser wird als Startpunkt verwendet. Dieses Vorgehen berechtigt die Annahme, dass es sich beim Startpunkt der Kontur um einen Eckpunkt der Fläche handelt.

3.5 Extraktion der 3D–Polygone

Die Begrenzung jeder Fläche soll durch ein 3D–Polygon approximiert werden. Gesucht ist also eine Anpassung von zusammenhängenden Liniensegmenten an die 3D–Konturpunkte jeder Fläche. Durch das starke Rauschen der Tiefenwerte und die daraus resultierende ungenaue Segmentierung wird die tatsächliche Form der 3D–Kontur mancher Flächen verfälscht. Die rechte Abbildung von Bild 3.6 zeigt die Begrenzung einer Fläche, die ein Rechteck darstellt, die vier Kanten sind in der Kontur jedoch schwer wiederzufinden. Die

[3]Die Bounding Box einer Region wird bei der Segmentierung ermittelt.

im linken Teilbild dargestellte orthografische Projektion der 3D–Konturpunkte auf die xy–Ebene repäsentiert die tatsächliche Oberflächenkontur besser, da diese Darstellung von den Tiefenwerten unabhängig ist. Es wird daher für jede Fläche zunächst ein 2D–Polygon an die 2D–Kontur

$$\boldsymbol{B2d} = \left\{ (r(i_0, j_0)_{\mathrm{x}}, r(i_0, j_0)_{\mathrm{y}})^{\mathrm{T}}, (r(i_1, j_1)_{\mathrm{x}}, r(i_1, j_1)_{\mathrm{y}})^{\mathrm{T}}, \ldots, (r(i_n, j_n)_{\mathrm{x}}, r(i_n, j_n)_{\mathrm{y}})^{\mathrm{T}} \right\} \quad (3.5)$$

angepasst. Das Vorgehen zur Anpassung von Geraden an die 2D–Konturpunkte ist im folgenden Abschnitt beschrieben. In Kapitel 3.5.2 wird erläutert, wie die Eckpunkte der Polygone festgelegt und schließlich die 3D–Polygone bestimmt werden.

3.5.1 Inkrementelles Line Fitting im 2D

Ein Vergleich verschiedener Algorithmen zur Linienextraktion ist in [NMTS05] zu finden. *Inkrementelles Line Fitting* und *Split and Merge* werden bezüglich Schnelligkeit und Genauigkeit als die besten Verfahren genannt. Durch die sequentielle Ordnung der Konturpunkte sind hier beide Verfahren gut geeignet. Das Inkrementelle Line Fitting ist ein iteratives Verfahren, bei dem die Punkteliste Schritt für Schritt durchlaufen wird. Dabei wird eine Gerade an die bisher passierten Punkte angepasst, bis der Anpassungsfehler zu groß wird. So wird verfahren, bis keine Punkte mehr vorhanden sind. Aufgrund des Vorgehens bei der Konturextraktion kann angenommen werden, dass der Startpunkt einer Kontur einen Eckpunkt des Polygons der Fläche und somit den Startpunkt einer Linie darstellt. Daher eignet sich hier das Inkrementelle Line Fitting zur Approximation der Geraden. Das Standardvorgehen bei diesem Verfahren ist in Bild 3.7 beschrieben, die eigentliche Geradenanpassung wird am Ende des Abschnitts erläutert.
Bei verrauschten Daten hat das Standardverfahren einen Nachteil: Um zu vermeiden, dass eine verrauschte Punkteliste in zu viele kleine Liniensegmente unterteilt wird, muss der Schwellwert für den Anpassungsfehler, $\theta_{\mathrm{maxLineFitErr}}$, hoch gewählt werden. Dies hat zur Folge, dass die Ecken der Kontur nicht genau approximiert werden und dass ein schiefes Gesamtergebnis entsteht. Bild 3.8 zeigt das gewünschte Ergebnis (grünes Rechteck) im Vergleich zum Ergebnis bei zu hohem maximalem Anpassungsfehler (rotes Rechteck). Aus diesem Grund wurde das Verfahren, wie in Bild 3.9 beschrieben, erweitert. Die Idee dabei ist, sich die Position des Punktes in der Liste zu merken, seit dem der Anpassungs-

WHILE noch Punkte vorhanden
Geradenanpassung der nächsten 2 Punkte
Anpassungsfehler fitErr berechnen
WHILE fitErr < $\theta_{\mathrm{maxLineFitErr}}$ AND noch Punkte vorhanden
Geradenanpassung nach Hinzufügen des nächsten Punktes
Anpassungsfehler fitErr berechnen
Letzten Punkt zurücknehmen
Geradenparameter neu berechnen
Gerade zur Liste der gefundenen Geraden hinzufügen

Bild 3.7: Struktogramm zum Algorithmus Inkrementelles Line Fitting

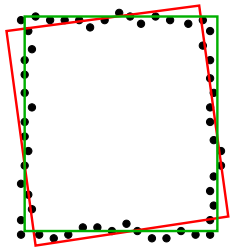

Bild 3.8: Veranschaulichung des Inkrementellen Line Fitting, grünes Rechteck: Ergebnis bei niedrigem Schwellwert, rotes Rechteck: Ergebnis bei hohem Schwellwert für den Anpassungsfehler

fehler mit jedem weiteren Punkt signifikant schlechter wird. Nach Überschreitung des Schwellwerts für den Anpassungsfehler werden dann die Punkte, die nur noch zu einer Verschlechterung beigetragen haben, bei Bestimmung der endgültigen Geraden nicht miteinbezogen. Beim Standardverfahren wird dagegen nur der letzte Punkt verworfen. Über den Schwellwert $\theta_{\mathrm{maxLineFitErrIncr}}$ kann die Strenge des Urteils, ob der aktuelle Punkt zu einer signifikanten Verschlechterung beiträgt, gesteuert werden. Wenn dieser Wert sehr hoch gewählt wird, entspricht das Ergebnis dem des Standardverfahrens. Bild 3.10 zeigt einen Vergleich der beiden Verfahren[4]. In Teilbild (a) sind die orthografisch auf die xy–Ebene

[4]Die Kanten sind zur Veranschaulichung bereits geclippt (siehe Kapitel 3.5.2), das Inkrementelle Line Fitting liefert aber Geraden, keine Segmente.

WHILE noch Punkte vorhanden		
Geradenanpassung der nächsten 2 Punkte		
Anpassungsfehler fitErr berechnen		
WHILE fitErr < $\theta_{\mathrm{maxLineFitErr}}$ AND noch Punkte vorhanden		
	prevFitErr = fitErr	
	Geradenanpassung nach Hinzufügen des nächsten Punktes	
	Anpassungsfehler fitErr berechnen	
	IF	fitErr < (prevFitErr + $\theta_{\mathrm{maxLineFitErrIncr}}$)
	THEN	lastGoodIndex = aktueller Index
Punkte nach lastGoodIndex zurücknehmen		
Geradenparameter neu berechnen		
Gerade zur Liste der gefundenen Geraden hinzufügen		

Bild 3.9: Struktogramm zum erweiterten Algorithmus Inkrementelles Line Fitting

projizierten 2D–Konturpunkte dargestellt. Die oberen Teilbilder zeigen die Ergebnisse des Standardverfahrens mit niedrigem (b) und hohem Schwellwert für den Anpassungsfehler (c). Der niedrige Schwellwert führt dazu, dass die untere verrauschte Kante in zwei Geraden aufgeteilt wird, beim hohen Schwellwert wird die gesamte Anpassung ungenau, das Rechteck ist leicht gedreht. Die unteren Teilbilder zeigen die Ergebnisse des angepassten Verfahrens mit niedrigem (d) und hohem Schwellwert (e). Hier führt ein hoher Schwellwert nicht zum Verlust der Genauigkeit der Anpassung.

Geradenanpassung durch Orthogonal Distance Regression Fitting Bei der zweidimensionalen Geradenanpassung durch Orthogonal Distance Regression Fitting soll für eine Menge von 2D–Punkten eine Gerade bestimmt werden, die die Summe der quadratischen Abstände der Punke zur Geraden minimiert [AV05].

- Gegeben: Menge von n 2D–Punkten $\boldsymbol{p}_i = (x_i, y_i)^{\mathrm{T}}$
- Gesucht: Aufpunkt $\boldsymbol{u}$ und normierter Richtungsvektor $\boldsymbol{v}$ der Geraden

$$\boldsymbol{l} = \boldsymbol{u} + t \cdot \boldsymbol{v}, \tag{3.6}$$

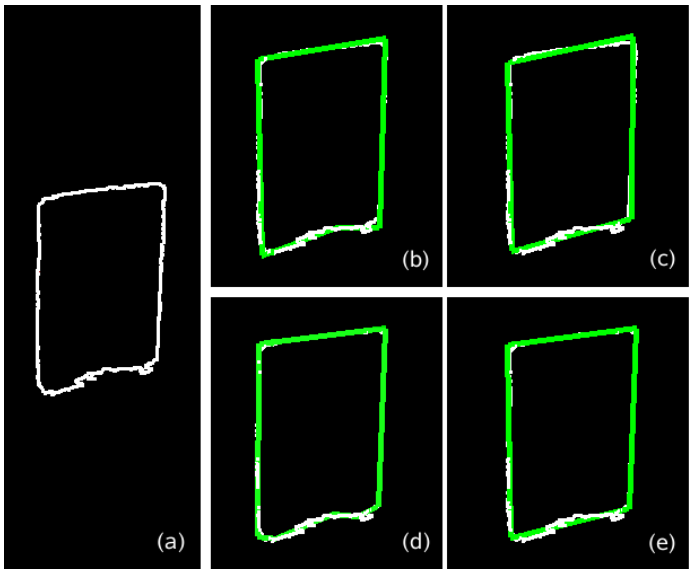

Bild 3.10: Ergebnisse des Inkrementellen Line Fitting, oben: Standardverfahren, unten: erweitertes Verfahren

sodass die Summe der quadratischen Abstände der Punkte zu dieser Geraden minimal ist.

Bild 3.11 veranschaulicht die im Folgenden verwendeten Größen und Formeln nach [AV05]. Sei $\boldsymbol{n}$ ein Einheitsvektor, senkrecht zu $\boldsymbol{v}$, und δ_i der Abstand der Punktes $\boldsymbol{p}_i$ zur Geraden. Für ein Punkt $\boldsymbol{p}_i$ gilt:

$$\boldsymbol{p}_i = \boldsymbol{u} + \zeta_i \cdot \boldsymbol{v} + \delta_i \cdot \boldsymbol{n} \tag{3.7}$$

Mit $\boldsymbol{a}_i = \boldsymbol{p}_i - \boldsymbol{u}$ gilt:

$$\boldsymbol{a}_i - \zeta_i \cdot \boldsymbol{v} = \delta \cdot \boldsymbol{n} \tag{3.8}$$

Gesucht ist das Minimum der Funktion

$$f(\boldsymbol{u}, \boldsymbol{v}) = \sum_{i=1}^{n} \delta_i \tag{3.9}$$

$$= \sum_{i=1}^{n} (\boldsymbol{a}_i - \zeta_i \cdot \boldsymbol{v})^2 \tag{3.10}$$

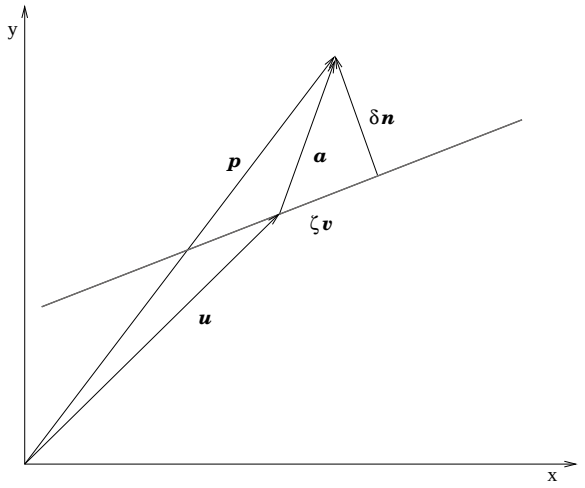

Bild 3.11: Veranschaulichung zum Orthogonal Distance Regression Fitting

Der gesuchte Aufpunkt $\boldsymbol{u}$ ist der Schwerpunkt

$$\boldsymbol{u} = (\bar{x}, \bar{y})^{\mathrm{T}} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{p}_i \tag{3.11}$$

der Punkte $\boldsymbol{p}_i$.

Die Formel 3.10 kann umgeformt werden zu

$$f(\boldsymbol{u}, \boldsymbol{v}) = \sum_{i=1}^{n} (\boldsymbol{a}_i{}^{\mathrm{T}} (\boldsymbol{Id}_2 - \boldsymbol{v}\, \boldsymbol{v}^{\mathrm{T}}) \boldsymbol{a}_i) \tag{3.12}$$

$$= \boldsymbol{v}^{\mathrm{T}} \sum_{i=1}^{n} ((\boldsymbol{a}_i \cdot \boldsymbol{a}_i)\, \boldsymbol{Id}_2 - \boldsymbol{a}_i\, \boldsymbol{a}_i{}^{\mathrm{T}}) \boldsymbol{v} \tag{3.13}$$

$$= \boldsymbol{v}^{\mathrm{T}} \boldsymbol{M} \boldsymbol{v}, \tag{3.14}$$

wobei $\boldsymbol{Id}_2$ die Einheitsmatrix ist und

$$\boldsymbol{M} = \begin{pmatrix} \sigma_{\mathrm{yy}}^2 & \sigma_{\mathrm{xy}}^2 \\ \sigma_{\mathrm{xy}}^2 & \sigma_{\mathrm{xx}}^2 \end{pmatrix}, \tag{3.15}$$

$$\sigma_{\mathrm{xy}}^2 = \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) \tag{3.16}$$

Der Ausdruck 3.14 wird minimal für den Eigenvektor von $\boldsymbol{M}$, der zum kleinsten Eigenwert der Matrix gehört. Der normalisierte Eigenvektor ist der gesuchte Richtungsvekor $\boldsymbol{v}$. Der beim Inkrementellen Line Fitting untersuchte Anpassungsfehler ist der mittlere quadratische Abstand

$$\mu = \frac{1}{n}(\boldsymbol{v}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v}) \tag{3.17}$$

der Punkte zur der angepassten Geraden.

3.5.2 Bestimmung der Eckpunkte der Polygone

Das Ergebnis des Inkrementellen Line Fitting ist zunächst eine Liste von Geraden. Die Schnittpunkte dieser aufeinander folgenden Geraden sind die Eckpunkte des gesuchten 2D–Polygonzugs. Die 3D–Polygone werden bestimmt, indem zu den 2D–Eckpunkten anhand der jeweiligen Ebenenparameter der betrachteten Fläche die zugehörigen z–Kordinaten berechnet werden.
Falls aufeinander folgende Geraden annähernd parallel sind, liegt der Schnittpunkt außerhalb des Polygons. Um dies zu vermeiden wurde der Schwellwert $\theta_{\mathrm{maxMergeAngle3D}}$ eingeführt. Zwei 3D–Geraden, deren eingeschlossener spitzer Winkel kleiner als $\theta_{\mathrm{maxMergeAngle3D}}$ ist, werden verschmolzen, indem an die Menge der 2D–Punkte, die zu den beiden Geraden gehören, eine neue Gerade angepasst wird. In Bild 3.12 ist ein Beispiel dargestellt. Links ist im großen Bild die orthografische Projektion der 3D–Kontur zu sehen. Das kleine Teilbild zeigt den Ausschnitt des Regionenbilds zu der durch einen Rahmen markierten Stelle. Durch Übersegmentierung entsteht hier eine Spitze in der Kontur. Diese resultiert bei der Geradenanpassung in zwei annähernd parallelen Geraden, wie im mittleren Bild zu sehen ist. Rechts ist das Ergebnis nach dem Verschmelzen von annähernd parallelen, aufeinander folgenden Geraden zu sehen.
Das in diesem Kapitel 3.5 beschriebene Verfahren wird auf die Konturen aller Flächen angewendet. Bild 3.13 zeigt das Ergebnis. Im Gegensatz zu den in Kapitel 2.3 genannten Verfahren wird jede Fläche durch ihr begrenzendes Polygon repräsentiert, bevor, wie im folgenden Kapitel beschrieben, die Beziehungen zwischen diesen untersucht werden.

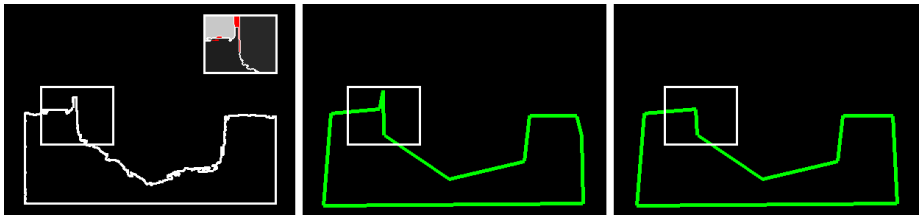

Bild 3.12: Verschmelzung von aufeinander folgenden Geraden, die nach dem Inkrementellen Line Fitting annähernd parallel sind

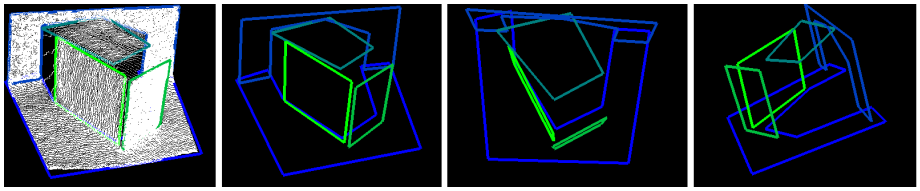

Bild 3.13: Ergebnis der Polygonextraktion, Ansichten der Szene von links oben und rechts, Bild links zeigt die 3D–Punkte des Tiefenbildes

3.6 Verbinden der 3D–Polygone

In Bild 3.13 wird deutlich, dass große Lücken zwischen Polygonen auftreten, die eigentlich verbunden sein sollten. Vor allem die Oberseite der dargestellten Box ist nach hinten ”gewandert”. Dies ist auf das in Kapitel 3.2 angesprochene Problem zurückzuführen, dass Flächen mit dünn verteilten Tiefenwerten nicht vollständig segmentiert werden können. In diesem Kapitel wird beschrieben, wie gemeinsame Kanten und Eckpunkte von Polygonen bestimmt und verbunden werden.

3.6.1 Verbindungskriterien

Einige der Lücken sind so groß, wie solche zwischen Kanten, die nicht zusammengehören. Ein Abstandskriterium im 3D–Raum ist somit nicht zuverlässig und ausreichend um festzustellen welche Polygone, genauer, welche Kanten und Eckpunkte, zu verbin-

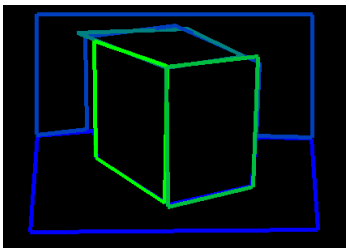

Bild 3.14: Orthografische Projektion des Ergebnis der Polygonextraktion auf die xy–Ebene.

den sind. Daher werden zusätzlich die 2D–Polygone herangezogen. Bild 3.14 zeigt die orthografische Projektion der Polygone auf die xy–Ebene, in dieser Darstellung sind die Zusammenhänge der einzelnen Flächen erkennbar. Die Kriterien für die Entscheidung, ob zwei Kanten verschiedener Polygone verbunden werden, lauten:

1. Die 2D–Kanten sind annähernd parallel.
2. Der Abstand der Mittelpunkte der 2D–Kanten ist klein.
3. Die 2D–Kanten sind annähernd gleich lang.
4. Die 3D–Kanten sind annähernd parallel.

Hierfür werden vier Schwellwerte definiert: Für den Winkel zwischen 2D–Kanten $\theta_{\mathrm{maxAngle2D}}$ und 3D–Kanten $\theta_{\mathrm{maxAngle3D}}$, für den Abstand zwischen den Mittelpunkten der 2D–Kanten $\theta_{\mathrm{maxMidpointDist2D}}$ und für das Verhältnis der kürzeren zur längeren Kante $\theta_{\mathrm{minLengthRatio2D}}$. In Kapitel 3.6.3 wird erläutert, dass das Verbinden zweier Kanten zu einer Transformation der Polygone führen kann, die Überprüfung der Kriterien wird aber immer mit den originalen 2D– und 3D–Polygonen durchgeführt.

3.6.2 Neigungsbasiertes Zuverlässigkeitsmaß

In dem in Kapitel 2.3 vorgestellten Verfahren wird eine gemeinsame Kante von zwei Flächen als ein Segment ihrer Schnittkante bestimmt. In Bild 3.15 ist die Seitenansicht der

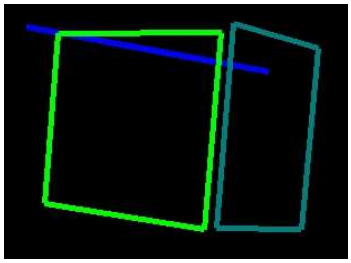

Bild 3.15: Seitenansicht der Box, mit Blick entlang der oberen Fläche

Box aus Bild 3.13 dargestellt. Die Neigung der oberen Fläche ist in Folge der verrauschten Tiefenwerte stark verfälscht[5], sodass ein Schnitt dieser Fläche mit den anderen Flächen zu einem fehlerhaften Ergebnis führen würde. Beim Verbinden von zwei zusammengehörenden Kanten wird daher die Orientierung solcher fehlerhafter Flächen angepasst. Um entscheiden zu können, welche der betrachteten Flächen angepasst wird, wird jeder Fläche zunächst ein Zuverlässigkeitswert zugewiesen. Das verwendete Zuverlässigkeitsmaß ist der Winkel zwischen der Normale der Fläche und der z–Achse, je kleiner der Winkel, umso höher der Zuverlässigkeitswert. Begründet wird dieses Maß durch die Beobachtung, dass die Punkte von Flächen, deren Normale einen großen Winkel zur optischen Achse der Kamera aufweist, stärker verrauscht sind. Möglich wäre auch das Einbeziehen des mittleren Anpassungsfehlers der Ebene, oder der Dichte der Punkte, die zur Fläche gehören.

3.6.3 Algorithmus zum Verbinden der 3D–Polygone

Eine Fläche mit höherer Zuverlässigkeit, im Folgenden *Winner Plane* genannt, kann die Orientierung einer weniger zuverlässigen Fläche, die *Victim Plane*, beim Verbinden der Kanten beeinflussen. Der mögliche Einfluss der *Winner Plane* ist dabei abhängig vom aktuellen Zustand der *Victim Plane*. Diese kann nacheinander folgende Zustände annehmen:

1. FREE Noch keine Kante des Polygons wurde verbunden.
2. JOINED Eine Kante des Polygons wurde bereits verbunden.

[5]Die Seitenflächen der Box stehen eigentlich senkrecht aufeinander.

3. FIXED Zwei Kanten des Polygons wurden bereits verbunden.

Der mögliche Einfluss auf das Polygon nimmt mit diesen Zuständen ab: Ein Polygon im Zustand FREE kann in Orientierung und Position beeinflusst werden. Im Zustand JOINED gibt es nur noch einen Freiheitsgrad: die Rotation um die bereits verbundene Kante. Im Zustand FIXED ist nur die Verschiebung der Endpunkte der noch nicht verbundenen Kanten entlang der Ebene erlaubt. Bild 3.16 zeigt den Algorithmus zum Verbinden der Polygone, die hervorgehobenen Schritte sind im Folgenden erläutert.

Heranziehen des Victim Polygons Wenn die zwei betrachteten Kanten alle Verbindungskriterien erfüllen und sich die Victim Plane im Zustand FREE befindet, wird wie in Bild 3.17 veranschaulicht, wie folgt verfahren. Zunächst wird das Victim Polygon zur Kante des Winner Polygons verschoben. Der Translationsvektor $\boldsymbol{t}$ ist der Vektor zwischen dem Endpunkt A_v der Victim Kante (mit Ortsvektor $\boldsymbol{a}_\mathrm{v}$) und dem Endpunkt B_w der Winner Kante (mit Ortsvektor $\boldsymbol{b}_\mathrm{w}$), die den geringsten Abstand aufweisen.

$$\boldsymbol{t} = \boldsymbol{b}_\mathrm{w} - \boldsymbol{a}_\mathrm{v} \tag{3.18}$$

Dann wird das Victim Polygon so rotiert, dass die Winner Kante in der Ebene des rotierten Victim Polygons liegt. Nach Eulers Theorem kann jede 3D–Rotation als eine Rotation um einen Winkel bezüglich einer Achse, die als Einheitsvektor angegeben ist, ausgedrückt werden [TV98]. Für die Bestimmung von Rotationsachse und Rotationswinkel wird zunächst die Schnittgerade der beiden Ebenen berechnet. Die Rotationsachse ist der normierte Vektor $\boldsymbol{u}$, der senkrecht auf dem Richtungsvektor $\boldsymbol{l}$ der Schnittgeraden und dem Richtungsvektor $\boldsymbol{e}_\mathrm{w}$ der Winner Kante steht.

$$\boldsymbol{u} = \frac{1}{\|\boldsymbol{l} \times \boldsymbol{e}_\mathbf{w}\|} \cdot \boldsymbol{l} \times \boldsymbol{e}_\mathrm{w} \tag{3.19}$$

Der Winkel α' ist der Winkel zwischen diesen zwei Vektoren.

$$\alpha' = \arccos \frac{\boldsymbol{l} \cdot \boldsymbol{e}_\mathrm{w}}{\|\boldsymbol{l}\| \cdot \|\boldsymbol{e}_\mathrm{w}\|} \tag{3.20}$$

Es muss darauf geachtet werden, dass die Richtung der Rotationsachse und der Winkel bzw. das Vorzeichen des Winkels korrekt gewählt sind. Der gesuchte Rotationswinkel α

Sortiere die Ebenen absteigend nach Zuverlässigkeitswert					
Ordne jeder Ebene eine absteigende Liste ihrer potentiellen Victim Planes zu					
WHILE noch Winner Planes vorhanden					
Nimm nächste Winner Plane p_w					
FOR jede Kante e_{w_i} von p_w					
WHILE e_{w_i} noch nicht verbunden AND noch potentielle Victim Planes vorhanden					
Nimm nächste Victim Plane p_v					
WHILE e_{w_i} noch nicht verbunden AND noch Kanten von p_v vorhanden					
Nimm nächste nicht verbundene Kante e_{v_j} von p_v					
IF	e_{w_i} und e_{v_j} Verbindungs–Kriterien erfüllen				
THEN	IF	p_v ist FREE			
	THEN	edgeConfidence = Verhältnis der Kantenlänge von e_{v_j} zum Polygonumfang von p_v			
		IF	**edgeConfidence** $\geq \theta_{minEdgeConfidence}$		
		THEN	**Ziehe** p_v **an** e_{w_i} **heran**		
			Markiere p_v als JOINED		
		ELSE	**Schneide** p_w **und** p_v		
			Markiere p_v als FIXED		
	ELSE	IF	p_v ist JOINED		
		THEN	IF	e_{v_j} **und** e_{w_i} **haben gemeinsamen Endpunkt auf der verbundenen Kante von** p_v	
			THEN	**Drehe** p_v **um verbundenen Kante auf** e_{w_i}	
				Markiere p_v als FIXED	
			ELSE	**Schneide** p_w **und** p_v	
				Markiere p_v als FIXED	
		ELSE	IF	p_v ist FIXED	
			THEN	**Schneide** p_w **und** p_v	
				Markiere p_v als FIXED	

Bild 3.16: Struktogramm des Algorithmus zum Verbinden der Polygone

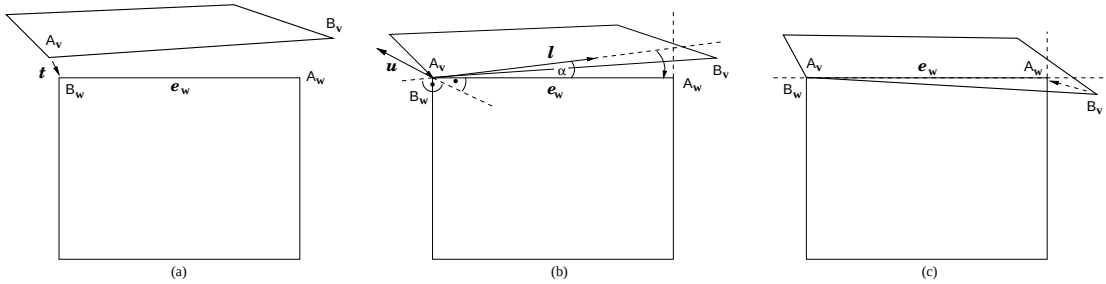

Bild 3.17: Heranziehen des Victim Polygons (a) Translation zur Winner Kante (b) Rotation auf die Winner Kante (c) Ersetzen des Endpunktes der Victim Kante mit Endpunkt der Winner Kante

ist der spitze Winkel zwischen den Vektoren $\boldsymbol{l}$ und $\boldsymbol{e}_{\mathrm{w}}$. Falls der Winkel[6] $\alpha' > \frac{\Pi}{2}$, dann

$$\alpha = -(\Pi - \alpha') \tag{3.21}$$

sonst

$$\alpha = \alpha' \tag{3.22}$$

Durch den Vorzeichenwechsel in Formel 3.21 erfolgt die Anpassung der Rotationsrichtung um die Achse $\boldsymbol{u}$. Die Rotationsmatrix $\boldsymbol{R}$ wird wie folgt berechnet [TV98]:

$$\boldsymbol{R} = \boldsymbol{Id}_3 \cos\alpha + (1 - \cos\alpha) \cdot \boldsymbol{u}\boldsymbol{u}^{\mathrm{T}} + \sin(\alpha) \cdot [\boldsymbol{u}]_{\times}, \tag{3.23}$$

mit Kreuzproduktmatrix

$$[\boldsymbol{u}]_{\times} = \begin{pmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{pmatrix}$$

$\boldsymbol{R}$ in Komponenten:

$$\begin{pmatrix} u_1^2 + (1 - u_1^2)\cos\alpha & u_1u_2(1 - \cos\alpha) - u_3\sin\alpha & u_1u_3(1 - \cos\alpha) + u_2\sin\alpha \\ u_1u_2(1 - \cos\alpha) + u_3\sin\alpha & u_2^2 + (1 - u_2^2)\cos\alpha & u_2u_3(1 - \cos\alpha) - u_1\sin\alpha \\ u_1u_3(1 - \cos\alpha + u_2\sin\alpha & u_2u_3(1 - \cos\alpha) + u_1\sin\alpha & u_3^2 + (1 - u_3^2)\cos\alpha \end{pmatrix}$$

Die zwei Vektoren $\boldsymbol{u}$ und $\boldsymbol{e}_{\mathrm{w}}$ schneiden sich im bereits verbundenen Eckpunkt. Dieser

[6] im Bogenmaß

Punkt des Victim Polygons darf durch die Rotation nicht verändert werden. Vor der Rotation wird daher das Victim Polygon durch den Translationsvektor $\boldsymbol{t}_{\mathrm{o}}$ mit dem bereits verbundenen Eckpunkt in den Ursprung verschoben.

$$\boldsymbol{t}_{\mathrm{o}} = -\boldsymbol{b}_{\mathrm{w}} \tag{3.24}$$

Nach der Rotation wird das Polygon wieder zurück transliert.
Unter Verwendung homogener Koordinaten kann die gesamte Transformation mit einer Transformationsmatrix $\boldsymbol{M} \in \mathbb{R}^{4\times4}$ ausgedrückt werden:

$$\boldsymbol{M} = (-\boldsymbol{T}_{\mathrm{o}})\, \boldsymbol{R}'\, \boldsymbol{T}_{\mathrm{o}}\, \boldsymbol{T} \tag{3.25}$$

Mit den Transformationsmatrizen

$$\boldsymbol{T} = \begin{pmatrix} 1 & 0 & 0 & \\ 0 & 1 & 0 & \boldsymbol{t} \\ 0 & 0 & 1 & \\ 0 & 0 & 0 & 1 \end{pmatrix} \boldsymbol{T}_o = \begin{pmatrix} 1 & 0 & 0 & \\ 0 & 1 & 0 & \boldsymbol{t}_{\mathrm{o}} \\ 0 & 0 & 1 & \\ 0 & 0 & 0 & 1 \end{pmatrix} \boldsymbol{R}' = \begin{pmatrix} & & & 0 \\ & \boldsymbol{R} & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.26}$$

Für die Durchführung der Transformation wird jeder Eckpunkt $\boldsymbol{p}^i, i = 1, \ldots, n$, des Victim Polygons in homogenen Koordinaten ausgedrückt und mit der Matrix multipliziert.

$$\begin{pmatrix} p_1'^i \\ p_2'^i \\ p_3'^i \\ 1 \end{pmatrix} = \boldsymbol{M} \begin{pmatrix} p_1^i \\ p_2^i \\ p_3^i \\ 1 \end{pmatrix} \tag{3.27}$$

Die homogene Koordinate wird dabei nicht beeinflusst, sodass der neue Eckpunkt $\boldsymbol{p}'^i$ durch die ersten drei Komponenten des Ergebnisvektors von 3.27 gegeben ist:

$$\boldsymbol{p}'^i = \left(p_1'^i, p_2'^i, p_3'^i\right)^{\mathrm{T}} \tag{3.28}$$

Nach dieser Transformation liegen die Winner Kante und das Victim Polygon in derselben Ebene. Der zweite Endpunkt B_{v} der Victim Kante wird schließlich durch den entsprechenden Endpunkt A_{w} der Winner Kante ersetzt.

Da dieses Verfahren die Orientierung des Victim Polygons verändert, ist die Operation nur erlaubt, wenn die Zuverlässigkeit der Victim Kante hoch genug ist, das heißt wenn das Verhältnis der Kantenlänge zum Polygonumfang größer ist als der Schwellwert $\theta_{\mathrm{minEdgeConfidence}}$. Falls diese Bedingung nicht gegeben ist, wird wie im letzten Abschnitt (Schnitt von Winner Plane und Victim Plane) beschrieben, verfahren.

Drehen des Victim Polygons Dieser Teil wird ausgeführt, wenn die zu verbindenden Kanten des Winner Polygons und des Victim Polygons bereits einen gemeinsamen Eckpunkt besitzen und dieser ein Endpunkt der bereits verbundenen Kante des Victim Polygons ist. Das Victim Polygon befindet sich also im Zustand JOINED. Die Situation ist in Bild 3.18 (a) dargestellt. Eine Translation ist nicht nötig, da die Winner Kante mit Richtungsvektor $\boldsymbol{e}_{\mathrm{w}}$ und die Victim Kante mit Richtungsvektor $\boldsymbol{e}_{\mathrm{v}}$ bereits im Eckpunkt V_{j} verbunden sind. Durch eine Rotation um die bereits verbundene Kante mit Richtungsvektor $\boldsymbol{e}_{\mathrm{j}}$ wird das Victim Polygon so gedreht, dass die Winner Kante in dessen Ebene liegt. Für die Berechnung des Rotationswinkels α wird ein Vektor $\boldsymbol{l}_{\mathrm{w}}$ in der Winner und ein Vektor $\boldsymbol{l}_{\mathrm{v}}$ in der Victim Ebene benötigt, die einen rechten Winkel zur Rotationsachse besitzen. Diese werden durch das Lot der freien Endpunkte A_{v} und B_{w} der zu verbindenden Kanten auf die Gerade durch V_{j} mit Richtungsvektor $\boldsymbol{e}_{\mathrm{j}}$ berechnet.

$$t = \frac{(\boldsymbol{a}_{\mathrm{v}} - \boldsymbol{v}_{\mathrm{j}}) \cdot \boldsymbol{e}_{\mathrm{j}}}{\|\boldsymbol{e}_{\mathrm{j}}\|^2} \tag{3.29}$$

$$\boldsymbol{f} = \boldsymbol{v}_{\mathrm{j}} + t \cdot \boldsymbol{e}_{\mathrm{j}} \tag{3.30}$$

$$\boldsymbol{l}_{\mathrm{v}} = \boldsymbol{a}_{\mathrm{v}} - \boldsymbol{f} \tag{3.31}$$

Dabei ist $\boldsymbol{f}$ der Fußpunkt des Lots von A_{v} auf die Gerade. Die Berechnung von $\boldsymbol{l}_{\mathrm{w}}$ erfolgt analog. Da die Rotationsrichtung von der Richtung der Rotationsachse $\boldsymbol{u}$ abhängt wird diese mit

$$\boldsymbol{u} = \frac{\boldsymbol{l}_{\mathrm{v}} \times \boldsymbol{l}_{\mathrm{w}}}{\|\boldsymbol{l}_{\mathrm{v}} \times \boldsymbol{l}_{\mathrm{w}}\|} \tag{3.32}$$

berechnet. Die Rotationsmatrix $\boldsymbol{R}$ wird wie in 3.23 berechnet.
Die gesamte Transformation wird mit der Transformationsmatrix $\boldsymbol{M} \in \mathbb{R}^{4\times4}$ ausgedrückt:

$$\boldsymbol{M} = (-\boldsymbol{T}_{\mathrm{o}})\, \boldsymbol{R}'\, \boldsymbol{T}_{\mathrm{o}} \tag{3.33}$$

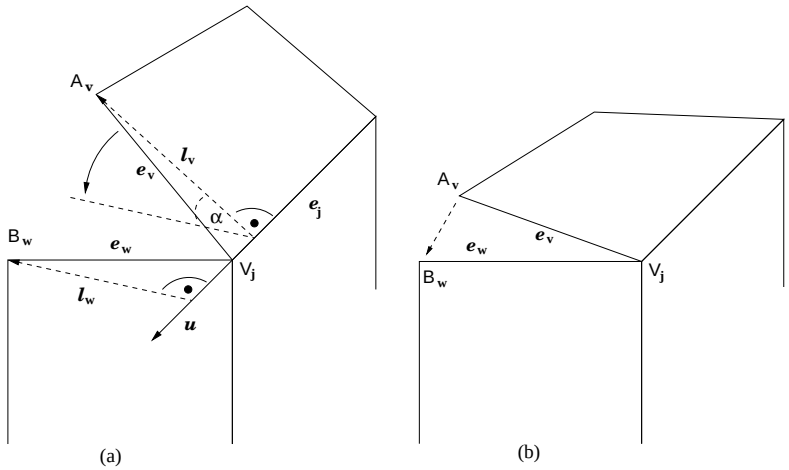

Bild 3.18: Drehen des Victim Polygons um die verbundene Kante $\boldsymbol{e}_{\mathrm{j}}$ (a) Drehen des Polygons auf die Winner Kante (b) Ersetzen des freien Endpunktes der Victim Kante

Die Transformationsmatrizen sind analog zu 3.26, mit Translationsvektor $\boldsymbol{t}_{\mathrm{o}} = (-1) \cdot \boldsymbol{v}_{\mathrm{j}}$ zur Translation des Victim Polygons in den Ursprung.
Nach der Anwendung der Transformation auf jeden Eckpunkt des Victim Polygon wird der zweite, freie Endpunkt A_{v} der Victim Kante durch den entsprechenden Endpunkt B_{w} der Winner Kante ersetzt.

Schnitt von Winner Plane und Victim Plane Wenn sich das Victim Polygon im Zustand FIXED befindet, ist nur noch die Verschiebung der Eckpunkte entlang der Ebene möglich. Die gemeinsame Kante von Winner und Victim Polygon wird in diesem Fall als ein Segment der Schnittgeraden der zwei Ebenen bestimmt. Dieses Segment wird wie folgt ermittelt: Die Vorgängerkante der Victim Kante $\overline{\mathsf{A}_{\mathrm{v}}\mathsf{B}_{\mathrm{v}}}$ sei gegeben durch die Gerade l in Parameterdarstellung. Die Ebene E des Winner Polygons sei in Koordinatenform gegeben.

$$l: \qquad \boldsymbol{p} = \boldsymbol{u} + t \cdot \boldsymbol{v} \tag{3.34}$$

$$E: \qquad ax + by + cz + d = 0 \tag{3.35}$$

Der Endpunkt A mit Ortsvekor $\boldsymbol{a}$ der gesuchten Kante ist der Schnittpunkt der Vorgängerkante der Victim Kante und der Winner Ebene:

$$s = \frac{-(a\boldsymbol{u}_{\mathrm{x}} + b\boldsymbol{u}_{\mathrm{y}} + c\boldsymbol{u}_{\mathrm{z}} + d)}{(a\boldsymbol{u}_{\mathrm{x}} + b\boldsymbol{u}_{\mathrm{y}} + c\boldsymbol{u}_{\mathrm{z}})} \tag{3.36}$$

$$\boldsymbol{a} = \boldsymbol{u} + s \cdot \boldsymbol{v} \tag{3.37}$$

Der zweite Endpunkt B der gesuchten Kante wird analog als Schnittpunkt der Nachfolgerkante der Victim Kante und der Winner Ebene berechnet. Der Startpunkt A_{v} der Victim Kante und der Endpunkt B_{w} der Winner Kante werden durch A ersetzt und der Endpunkt B_{v} der Victim Kante und der Startpunkt A_{w} der Winnerkante durch B.

3.7 Oberflächenbegrenzungsmodell

Für die Repräsentation der extrahierten Polygone eignet sich ein Oberflächenbegrenzungsmodell, wie die sogenannte *Boundary Representation* (B–Rep) [Liu93]. Diese beschreibt die *Geometrie* und die *Topologie* eines Modells. Die topologische Beschreibung gibt durch die Information, welche Kanten und Eckpunkte verbunden sind, die Nachbarschaftsbeziehungen zwischen den Flächen wieder [Hof89]. Diese Beziehungen werden durch das im vorigen Kapitel beschriebene Verfahren ermittelt. Die geometrische Beschreibung liefert die Information über die Lage der die Flächen beschreibenden Kanten und Eckpunkte im Raum. [Hof89]. In dieser Studienarbeit werden außerdem die Ebenengleichungen der Flächen zur Beschreibung der Geometrie des Modells hinzugezogen. Die konkrete Implementierung dieser Datenstruktur ist in Kapitel 4.2 erläutert.

Kapitel 4

Technische Umsetzung

Das im vorigen Kapitel beschriebene Verfahren wurde in C++ mit Hilfe der Programmierumgebung PUMA implementiert. Als Entwicklungsplattform wurde SuSE Linux 9.3 verwendet. Für die Erstellung der in Kapitel 4.3 erläuterten GUI wurde Qt in der Version 3.3.4, sowie der Qt–Designer (Version 3.3.4) verwendet. Für das Einlesen von Tiefenbildern, die als .pgm Datei vorliegen, wurde die OpenCV Bibliothek (Version 0.9.6) eingesetzt. Aus PUMA wurden hauptsächlich die von Timo Zinßer erstellten Klassen zur Segmentierung von Tiefenbildern[1] verwendet und teilweise, wie in Kapitel 3.1 beschrieben, erweitert. Außerdem wurde die Datenstruktur `RangeImage` wie im folgenden Kapitel beschrieben, erweitert. Die im Laufe der Studienarbeit entwickelten Klassen werden in Kapitel 4.2 vorgestellt. In Kapitel 4.3 wird die Bedienung der GUI erläutert. Der gesamte Quelltext liegt dieser Studienarbeit auf CD bei, in Anhang B ist eine Installationsanleitung mit einer Auflistung der Systemvoraussetzungen zu finden.

4.1 Erweiterungen der PUMA Klasse RangeImage

Das im Laufe der Studienarbeit zu entwickelnde System sollte sowohl Tiefenbilder aus der 3D–Kamera, als auch Tiefenbilder, die mit Hilfe eines Stereosystems erstellt wurden,

[1] `SegAlg`, `ThreeDImage`, `FitImage`, `RegionImage`, `SegList`, `SegObject`, `SegPlane`

behandeln. Für die Bilder aus der 3D–Kamera war in PUMA bereits die Datenstruktur `ThreeDImage` gegeben, die auch von der vorhandenen Klasse `SegAlg` als Eingabe zur Segmentierung erwartet wird. Diese Datenstruktur enthält für jeden Bildpunkt die x–, y– und z–Koordinate für den abgebildeten Punkt. Dies ist auch die Datenstruktur, mit der das entwickelte System arbeitet.
Das Verfahren zur Erstellung von Tiefenbildern mit Hilfe von Stereokameras liefert für jeden Bildpunkt einen Tiefenwert, außerdem ist das Stereokamerasystem kalibriert. Gesucht war also eine geeignete Datenstruktur für die Verwaltung dieser Tiefenbilder und die Konvertierung nach `ThreeDImage`. In PUMA existierte bereits eine Datenstruktur `RangeImage` zur Repräsentation von diskreten Tiefenbildern mit quantisierten Tiefenwerten, also als Matrix von `unsigned short` Werten. Diese Klasse ermöglicht durch die Methoden `xcam` und `ycam` auch die Berechnung der x– und y– Koordinate zum jeweiligen Tiefenwert. Dabei werden die Kamerakalibrierungsdaten aus der `CameraGeo3d` Membervariablen verwendet.
Um diese Klasse auch für nicht quantisierte Tiefenwerte in Fließkommazahlen verwenden zu können, wurde im Rahmen der Studienarbeit die Template Klasse `RangeImage` erstellt. Diese besitzt, geerbt von der Klasse `Image` ein `CameraGeo` Objekt zur Berechnung der Koordinaten. Für die spezielle Behandlung von quantisierten Tiefenbildern wurde die Klasse `RangeImageDiscrete` erstellt, die von `RangeImage<unsigned short>` abgeleitet ist und ein `CameraGeo3d` Objekt verwendet. Diese Klasse stellt also die ursprünglich vorhandene Funktion zur Verfügung. Bei der Erstellung des Templates aus der Klasse `RangeImage` wurden die NIHCL Funktionen zur Serialisierung entsprechend angepasst.

4.2 Neu implementierte Klassen

4.2.1 Datenstrukturen

BRep Die Klasse `BRep` implementiert die in Kapitel 3.7 beschriebene Datenstruktur zur Repräsentation eines Oberflächenbegrenzungsmodells. Die Geometrie der Szene ist gegeben durch eine Liste von `PlanePolygon3D`. Die Topolgie wird durch eine Nachbar-

schaftsliste beschrieben. Darin ist für jedes Polygon eine Liste von Nachbarschaftsbeziehungen enthalten. Eine Nachbarschaft wird duch die Struktur `EdgePolygonConnection` ausgedrückt. Diese enthält die Kantennummer einer Kante des betrachteten Polygons und die Id des Polygons, mit dem es über die Kante verbunden ist.

Contour Die Klasse `Contour` beschreibt die Kontur einer bestimmten Region als Liste von Pixelkoordinaten `PixelCoordinate`.

ContourList Die Klasse `ContourList` stellt eine Liste von Konturen, gegeben als `Contour` Objekt, dar. Der Zugriff auf eine Kontur kann über den sequetiellen Index, oder über die Id der zugehörigen Region erfolgen.

PixelCoordinate Die Klasse `PixelCoordinate` repäsentiert eine Pixelkoordinate durch ihre x– und y–Koordinate.

StrLineSeg3D Analog zur PUMA Klasse `StrLineSeg` für zweidimensionale Liniensegmente repräsentiert die Klasse `StrLineSeg3D` Liniensegmente im 3D durch einen Start und einen Endpunkt. Die Länge des Segments kann abgefragt werden, sowie der Winkel zu einer anderen Geraden, die als `StrLineSeg` vorliegt. Das Liniensegment kann durch Rotation und Translation manipuliert werden.

Polygon3D Analog zur PUMA Klasse `Polygon` stellt die Klasse `Polygon3D` eine Datenstruktur zur Repräsentation und Manipultation von 3D–Polygonen zur Verfügung. Ein Polygon wird als Liste von `StrLineSeg3D` beschrieben. Der Umfang und die Anzahl der Eckpunkte des Polygons können abgefragt und Kanten können hinzugefügt und ersetzt werden. Außerdem stehen Methoden zur Translation und Rotation des Polygons zur Verfügung.

PlanePolygon2D Die Klasse `PlanePolygon2D` erweitert die PUMA Klasse `Polygon` um den Parameter `m_RegionId`, um die Zugehörigkeit der Ebene zu einer bestimmten

Region auszudrücken. Außerdem wurden der Copy Konstruktor sowie der Vergleichs– und Zuweisungsoperator hinzugefügt.

PlanePolygon3D Die Klasse `PlanePolygon3D` erweitert die Klasse `Polygon3D` um den Parameter `m_RegionId`, um die Zugehörigkeit der Ebene zu einer bestimmten Region auszudrücken, sowie um das Array `m_PlaneParameters`, das die vier Parameter (a, b, c, d) enthält, die die Ebene beschreiben, in der das Polygon liegt:
$a * x + b * x + c * x + d = 0$.

4.2.2 Algorithmen

In diesem Abschnitt werden die Klassen vorgestellt, die zur Umsetzung der in Kapitel 3 beschriebenen Algorithmen implementiert wurden.

BRepExtractor Mit Hilfe der Klasse `BRepExtractor` kann aus einer Szene, die als `ThreeDImage` oder `RangeImage` gegeben ist, ein Oberflächenbegrenzungsmodell als `BRep` Datenstruktur extrahiert werden. In der Klasse selbst sind keine Algorithmen implementiert, sie dient lediglich als Schnittstelle, um den kompletten Ablauf der Extraktion auszulösen und die benötigten Parameter für die Segmentierung und die Polygon Extraktion zu belegen. Werden die Parameter nicht explizit gesetzt, so werden Default–Werte verwendet. Diese entsprechen den Werten, die in der GUI vorgegeben sind und die für die in Kapitel 5 beschriebenen Experimente eingesetzt wurden.

ContourExtraction Die Klasse `ContourExtraction` berechnet zu einem gegebenen Regionenbild, das als `RegionImage` vorliegt, die Konturen der Regionen und gibt diese als `ContourList` zurück. Die Bestimmung der Konturpixel erfolgt anhand des in Kapitel 3.4 beschriebenen Crack–Following Algorithmus. Jede Kontur stellt daher eine Sequenz von Konturpixelkoordinaten der jeweiligen Region entgegen den Uhrzeigersinn dar, wobei jede Koordinate nur einmal in der Sequenz vorkommt.

LineFitting In der Klasse `LineFitting2D` ist das in Kapitel 3.5.1 beschriebene Verfahren zur Geradenanpassung implementiert. Als Eingabe wird eine Liste von `PointXY` Punkten erwartet. Das Ergebnis ist die an die Punkte angepasste Gerade als `StrLineSeg` Objekt, sowie der Anpassungsfehler, das heißt der mittlere quadratische Abstand der 2D–Punkte von der berechneten Geraden.
Die Funktionalität dieser Klasse wird von der Klasse `IncrementalLineFitting2D` genutzt. Diese erwartet als Eingabe eine Liste von `PointXY` 2D–Koordinaten, die eine geschlossene Kontur darstellen, und berechnet das `Polygon` Objekt, das diese Kontur am besten repräsentiert. Für das Inkrementelle Line Fitting müssen die Parameter `maxLineFitErr`, `maxLineFitErrIncr` und `maxMergeAngle` explizit gesetzt werden. Die Bedeutung der Parameter ist in Kapitel 3.5.1 und Tabelle 5.1 beschrieben.

PolygonExtraction Die Klasse `PolygonExtraction` extrahiert 3D–Polygone aus einem gegebenen `ThreeDImage`. Als Eingabe werden dabei die zugehörige `SegList` und `ContourList` erwartet. Die `ContourList` enthält die Konturen der Regionen, in die das `ThreeDImage` segmentiert wurde, die `SegList` enthält die Beschreibung der zugehörigen Ebenen. Mit der Methode `extractPolygons` wird die in Kapitel 3.5 beschriebene Extraktion der 3D–Polygone ausgelöst. Das Ergebnis ist eine Liste von voneinander unabhängigen 3D–Polyonen `PlanePolygon3D`. Das Verbinden der Polygone aus Kapitel 3.6 ist in der Methode `joinPolygons` implementiert. Das Ergebnis ist ein Oberflächenbegrenzungsmodell der Szene als `BRep`.

4.2.3 Filter

ThreeDImageMean Die Klasse `ThreeDImageMean` stellt einen zeitlichen Mittelwertfilter der Tiefenwerte mehrerer Tiefenbilder der Form `ThreeDImage` zur Verfügung.

ContourFilter Die Klasse `ContourFilter` kann zur Filterung der z–Koordinaten der 3D–Konturpunkte der Regionen eines `ThreeDImage` verwendet werden. Die Klasse wird im entwickelten System nicht genutzt, da die Geradenanpassung auf den 2D–Konturpunkten erfolgt. Die Funktion des Filters wurde zu Demonstrationszwecken aber

nicht aus der GUI entfernt.

4.2.4 Tools

Geometry Die Klasse `Geometry` bietet verschiedene Methoden zur dreidimensionalen Vektorgeometrie:

- Berechnung des Winkels zwischen zwei 3D–Vektoren
- Berechnung des Kreuzpodukts zweier 3D–Vektoren
- Berechnung der Rotationsmatrix zur Rotation um einen bestimmten Winkel um eine gegebene Achse
- Berechnung der Schnittgeraden von zwei Ebenen
- Berechnung des Schnittpunktes einer Geraden und einer Ebene
- Bestimmung der Ebene durch drei Punkte
- Berechnung der Lotgeraden eines 3D–Punktes auf eine 3D–Gerade

ImageConverter Die Klasse `ImageConverter` bietet Methoden zur Konvertierung verschiedener Bildrepräsentationen.

- Umwandlung von `RangeImage` zu `ThreeDImage`
- Umwandlung von `ThreeDImage`, `FitImage` und `RegionImage` in QPixmap zur Darstellung in der GUI

ThreeDImageHandler Die Klasse `ThreeDImageHandler` bietet Methoden, um die Daten, die in einem `ThreeDImage` enthalten sind, auszulesen.

4.3 GUI zur Demonstration der Ergebnisse

Parallel zur Entwicklung des im vorigen Kapitel beschriebenen Systems wurde eine grafische Benutzungsoberfläche erstellt, um die Zwischenergebnisse zu demonstrieren und um möglichst einfach verschiedene Parametereinstellungen vergleichen zu können. Die GUI wurde mit Qt Version 3.3.4 erstellt. Für das Layout der grafischen Oberfläche wurde der Qt Designer verwendet. Dieser erstellt automatisch die Klasse `MainWindow`, die alle Angaben bezüglich der Anordnung der Gui–Elemente enthält und die Slot–Funktionen sowie die Signal–Slot Verbindungen definiert. In der davon abgeleiteten Klasse `MainWindowImpl` wurden die Slot–Funktionen implementiert. Hier finden keine Berechnungen statt, die Gui ist komplett vom eigentlichen System getrennt und ruft lediglich die benötigten Funktionen der oben beschriebene Klassen auf, um die vom Nutzer eingegebenen Parameter zu übergeben und die daraus resultierenden Ergebnisse zu visualisieren. Gleich zu Beginn der Arbeit stellte sich als hilfreich heraus, die Tiefenbilder dreidimensional zu visualisieren. Hierzu wurde die Klasse `RangeImageDisplay` erstellt. Diese von `QGLWidget` abgeleitete Klasse ermöglicht die Darstellung der 3D–Koordinaten des Tiefenbilds mit Hilfe von OpenGL.
Im Folgenden wird die Bedienung der grafischen Oberfläche erläutert. In Bild 4.1 ist die GUI vollständig abgebildet, wobei der Endzustand nach Verbinden der 3D–Polygone dargestellt ist.

3D–Displays Die beiden unteren Fenster dienen der 3D–Visualisierung des Tiefenbilds. Links, im Fenster *3D Range Image*, sind die originalen ungefiltereten 3D–Punkte des Tiefenbilds dargestellt. Rechts, im Fenster *Processed 3D Range Image* werden die jeweiligen Zwischenergebnisse präsentiert. Hält man die linke Maustaste auf einem 3D–Display gedrückt, so kann man durch bewegen der Maus die Szene drehen. Ein Vor- und Zurückbewegen der Maus mit gedrückter rechter Taste ermöglicht das Hinein– und Hinauszoomen. Das Drücken der Taste ”p” bewirkt das Ein– und Ausblenden der weiß dargestellten 3D–Punkte. Im Beispiel Bild 4.1 sind im rechten Display die 3D–Punkte ausgeblendet, um die Ergebnispolygone besser sichtbar zu machen.

Menü Über den Menüpunkt File kann durch Auswahl des Untermenüpunkts *Load x.pmi file* ein Tiefenbild, das mit der 3D–Kamera erstellt wurde, geladen werden. Über einen Dialog kann im Dateisystem eine .pmi Datei gewählt werden. Es wird erwartet, dass der Dateiname mit .x.pmi endet und dass sich im gleichen Ordner die zugehörige *.y.pmi sowie die *.z.pmi Datei befindet. Über den Untermenüpunkt *Load Disparity Map* kann eine .pgm Datei aus dem Dateisystem gewählt werden. Diese Funktion wurde hinzugefügt, um zu testen, wie die Ergebnisse der parallel laufenden Studienarbeit *Erstellung einer Tiefenkarte aus Stereobildern für den RoboCup Rescue Wettbewerb* von Peter Decker verarbeitet werden. Hierbei handelt es sich um Disparitätskarten, vereinfacht wird die Disparität als Tiefe z verwendet und die Pixelkoordinaten werden als x– und y– Koordinaten angenommen. Nach erfolgreichem Laden eines Tiefenbilds wird die Grauwertbilddarstellung im Fenster *2D Range Image* und die 3D–Darstellung im Fenster *3D Range Image* angezeigt.

Image Averaging In diesem Bereich kann über ein Textfeld angegeben werden, wieviele Bilder gemittelt werden sollen. Über den Button *Average Images* wird die Mittelung (Kapitel 3.3) ausgelöst. Es wird ertwartet, dass die zu mittelnden Bilder Dateinamen der Form filename_1.x.pmi, filename_2.x.pmi usw. haben. Die y– und z–Dateien müssen entsprechend benannt sein.

Execute Über den Button *Execute* wird der gesamte Algorithmus ausgeführt. Dabei werden die Parameter verwendet, die in den Bereichen *Segmentation*, *2D Line Fitting* und *Polygon Join* gewählt wurden. Falls dort keine Änderungen vorgenommen wurden, werden die Default-Werte verwendet. Das Endergebnis, die Szene mit den verbundenen extrahierten Polygonen, wird im 3D–Fenster *Processed 3D Range Image* angezeigt. Das Ausführen dieses Buttons entspricht der sequentiellen Ausführung von *Display Normals*, *Display Regions*, *Display Merged Regions*, *Remove Outliers*, *Compute Contours*, *Fit Lines* und *Join Polygons*.

Segmentation Dieser Abschnitt bezieht sich auf die Durchführung und Visualisierung des in PUMA vorhandenen Segmentierungsalgorithmus (Kapitel 2.4) und dessen Erweiterungen (Kapitel 3.1). Über das Textfeld und den Button *Set Value* kann optional der Wert

des Parameters, der in der Drop Down Liste ausgewählt ist, gesetzt werden. Über den Button *Display Normals* wird zunächst die Filterung des Tiefenbildes durchgeführt und im 3D–Fenster werden die geglätteten Werte dargestellt. Außerdem wird die Berechnung der lokalen Normalen ausgelöst. Im Fenster Normals werden diese farbig visualisiert. Das Betätigen des Buttons *Display Regions* löst den Region Growing Algorithmus aus. Das segmentierte Bild wird im Fenster *Regions* in Graustufen dargestellt. Punkte, die keiner Region zugeordnet wurden, sind in diesem und im 3D–Fenster Rot dargestellt. Der Button *Display Merged Regions* führt zur Verschmelzung der Regionen und der Darstellung des segmentierten Bildes im Fenster *Merged Regions*. Durch *Remove Outliers* wird die Ausreißereliminierung durchgeführt. Das Ergebnis wird in *Merged Regions* und im rechten 3D–Fenster angezeigt.

Compute Contours Das Berechnen der Konturen der segmentierten Flächen (Kapitel 3.4) wird über den Button *Compute Contours* ausgelöst. Das Ergebnis wird durch schwarze Konturen im Fenster *Merged Regions* und durch farbige Konturpunkte im 3D–Fenster dargestellt. Die Ausreißer werden ab hier nicht mehr Rot dargestellt.

2D Line Fitting In diesem Bereich können optional die Parameter für das Inkrementelle Line Fitting (Kapitel 3.5) eingestellt werden. Über den Button *Fit Lines* wird der Algorithmus ausgeführt. Das Ergebnis wird im 3D–Fenster angezeigt.

Polygon Join Im diesem Bereich können optional die Parameter für das Verbinden der Polygone (Kapitel 3.6) gesetzt werden. Über den Button *Join Polygons* wird der Algorithmus ausgeführt. Das Ergebnis wird schließlich im 3D–Fenster angezeigt.

Filtering Dieser Bereich dient der separaten Durchführung und Visualisierung der Filterung des Tiefenbildes (Kapitel 2.4). Die ausgelöste Filterung hat keinen Einfluss auf den Ablauf des gesamten Algorithmus durch die erläuterten Funktionen. Außerdem kann über den Button *Filter 3D Contour* eine Fiterung der 3D–Kontur durchgeführt werden. Diese Filterung wird im entwickelten System nicht mehr genutzt, da die Geradenanpassung auf

den 2D–Konturpunkten erfolgt, zu Demonstrationszwecken wurde die Funktion aber nicht aus der GUI entfernt.

Status Information Hier werden Status Informationen des Programms angezeigt, sowie Hinweise für den Nutzer zur Bedienung der Oberfläche oder im Fehlerfall.

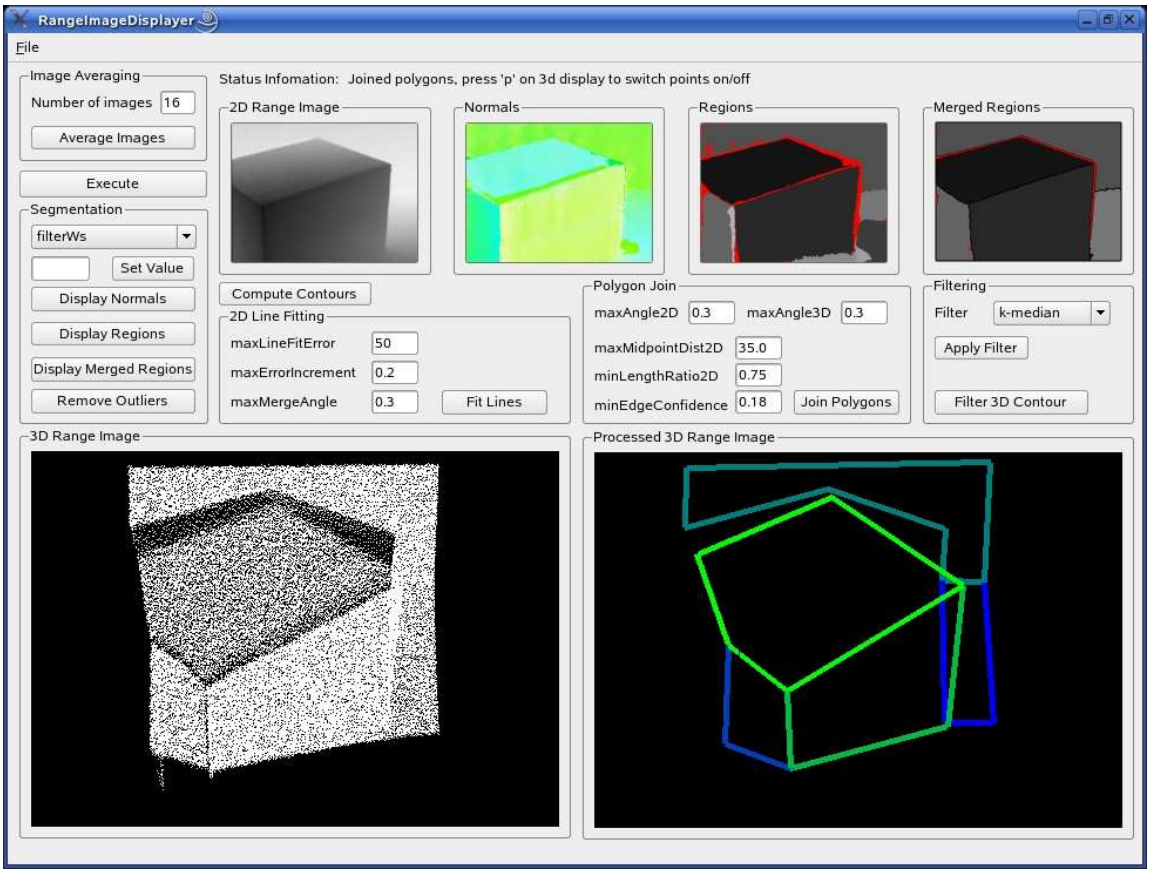

Bild 4.1: Die grafische Benutzungsoberfläche zur Demonstration der Ergebnisse

Kapitel 5

Experimente und Ergebnisse

In diesem Kapitel werden die durchgeführten Experimente beschrieben und die Ergebnisse anhand verschiedener Kriterien ausgewertet.

5.1 Versuchsaufbau

Für die Aufnahme von Testbildern sollte die in der Arbeitsgruppe Aktives Sehen vorhandene 3D–Laserkamera von Daimler-Benz Aerospace verwendet werden. Da es sich dabei um ein älteres Modell handelt, sind zu dieser Kamera im Internet keine Informationen verfügbar, in [Zin01] ist aber eine Beschreibung und eine Evaluation der Kamera zu finden. Diese Arbeit, sowie Dokumentationen des Herstellers befinden sich auf der dieser Studienarbeit beiligenden CD.
Um möglichst einfach Testaufnahmen erstellen zu können, wurde die Kamera zu Beginn der Studienarbeit im Labor der Arbeitsgruppe installiert, siehe Bild 5.1. Die Kamera ist mit einem Kugelkopf an einer Aluprofilstange befestigt, sodass die Kamera beliebig geneigt und gedreht werden kann. Die Aluprofilstange ist an einer weiteren, fest an der Wand montierten Aluprofilstange so befestigt, dass sie in der Höhe verstellt werden kann. Zum Kamerasystem gehören außerdem die Kontrolleinheit der 3D–Kamera und ein Rechner, auf dem die Software zur Aufnahme von Bildern intalliert ist. Informationen zur Bedienung des Kamerasystems sind in [Wic04] zu finden. Bei der Erstellung von Testszenen

Bild 5.1: Die 3D–Laserkamera mit Kontrolleinheit

wurde auf die Ergebnisse aus [Zin01] zurückgegriffen. Da die Tiefenbilder der Kamera bei Aufnahme normaler Gegenstände sehr stark verrauscht sind, wurden für die Testaufnahmen weiße, matte Objekte, z.B. Styroporplatten, verwendet. Mit diesen Gegenständen wurden künstliche Innenraum–Szenen in einem Messbereich von 90 bis 300 cm nachgestellt.

5.2 Versuchsdurchführung

Für die Experimente wurden fünf eigene Testszenen aufgenommen, sowie acht Tiefenbilder aus [Zin01] verwendet, die mit der selben Kamera aufgenommen wurden. Die verwendete Bildauflösung ist 320×240. Für die Verminderung des Rauschens war bei der Aufnahme der Bilder aus [Zin01] die automatische 16fache zeitliche Mittelung aktiviert, bei den eigenen Testaufnahmen wurde diese Mittelung, wie in Kapitel 3.3 beschrieben, vom eigenen Programm durchgeführt. Von den insgesamt 13 Testbildern zeigen acht Bilder ausschließlich Objekte mit planaren Flächen (Bildnummer p01 bis p08), die anderen fünf enthalten auch Objekte mit gekrümmten Oberflächen (Bildnummer m01 bis m05).
Das Ergebnis des entwickelten Verfahrens hängt von mehreren Parametern ab. Auf deren Beudeutung wird in Kapitel 3 an betreffender Stelle genauer eingegangen. Bei der Durchführung der Experimente wurde bei jedem Tiefenbild die selbe Parameterbelegung verwendet. Die Parameterwerte können der Tabelle 5.1 entnommen werden. Diese entsprechen den in der grafischen Benutzungsoberfläche standardmäßig vorgegebenen Werten.
Die Bewertung der Ergebnisse erfolgt anhand zweier Kriterien.

Parameter	Beschreibung	Wert
Inkrementelles Line Fitting		
$\theta_{\text{maxLineFitErr}}$	Der maximale erlaubte mittlere quadratische Abstand der Punkte zur angepassten Geraden (in mm)	50
$\theta_{\text{maxLineFitErrIncr}}$	Die maximal erlaubte Verschlechterung des Anpassungsfehlers (mittlerer quadratischer Abstand) für die Bewertung der Anpassungsgeraden als ”nicht signifikant schlechter werdend”	0.2
$\theta_{\text{maxMergeAngle}}$	Der maximale Wert des spitzen Winkels[a] zwischen zwei 3D–Geraden, der zu einer Verschmelzung der Geraden führt	0.3
Verbinden der Polygone		
$\theta_{\text{maxAngle2D}}$	Der maximale Winkel zwischen zwei 2D–Kanten, der eine Verbindung der Kanten zulässt	0.3
$\theta_{\text{maxAngle3D}}$	Der maximale Winkel zwischen zwei 3D–Kanten, der eine Verbindung der Kanten zulässt	0.3
$\theta_{\text{maxMidpointDist2D}}$	Der maximale Abstand (in mm) zwischen zwei 2D–Kanten, der eine Verbindung der Kanten zulässt	35.0
$\theta_{\text{minLengthRatio2D}}$	Das minimale Längenverhältnis zweier 2D–Kanten, das eine Verbindung der Kanten zulässt	0.75
$\theta_{\text{minEdgeConfidence}}$	Das minimale Verhältnis der Victim Kantenlänge zum Polygonumfang, das eine Änderung der Orientierung des Polygons zulässt	0.18

[a] Winkel sind im Bogenmaß angegeben

Tabelle 5.1: Belegung der Parameter

Winkel zwischen senkrechten Flächen Für alle Szenen, bei denen bekannt ist, dass sie Ebenen enthalten, die senkrecht zueinander stehen, wurden die Winkel zwischen den entsprechenden, durch das implementierte Verfahren rekonstruierten, Flächen berechnet. Der Vergleich zwischen den rekonstruierten Winkeln zum Ideal von 90° dient als Bewertungsmaß für die Güte der Anpassung der Orientierung der Flächen während der Verbindung von Kanten.

Verbindung von Kanten und Eckpunkten Die Korrektheit des Verbindens von Kanten und Eckpunkten[1] wurde untersucht, indem die Anzahl der korrekt und falsch verbundenen Kanten und Eckpunkte ermittelt wurde. Um mögliche Ursachen für fehlende Verbindungen zu finden, wurde auch ermittelt, wie viele der zu verbindenden Kanten und Eckpunkte nach der Segmentierung und der Bestimmung der einzelnen Polygone, potentiell noch verbunden werden können.

5.3 Ergebnisse

Die Ergebnisse der Auswertung der Winkel zwischen senkrechten Flächen sind in Tabelle 5.2 aufgelistet. Für jedes Bild, das senkrechte Flächen enthält ist der durchschnittliche Winkel zwischen den rekonstruierten, senkrechten Ebenen angegeben.
Tabelle 5.3 zeigt die Ergebnisse der Auswertung der Verbindung der Polygone. Hier ist für jedes Bild die Anzahl der zu verschmelzenden Kantenpaare und Eckpunkte angegeben, sowie die Anzahl dieser Kantenpaare und Eckpunkte, die nach Segmentierung und Polygonbestimmung noch vorhanden sind. Bei der Bestimmung der Anzahl der zu verschmelzenden Kantenpaare wurden nur Kanten miteinbezogen, die zu planaren Flächen gehören. In der jeweils dritten Spalte ist die Anzahl der korrekt verbundenen Kantenpaare und Eckpunkte angegeben. Für die Kanten ist auch die Anzahl der fälschlicherweise verbundenen Kantenpaare aufgelistet.

[1] Die Testszenen enthalten nur zu verschmelzende Eckpunkte mit drei zusammentreffenden Kanten.

Bild	Anzahl der senkrechten Ebenen	Durchschittlicher Winkel zwischen den senkrechten Ebenen	Differenz zu $90°$
p01	7	86.94	3.06
p02	7	84.14	5.86
p03	7	83.99	6.01
p04	10	85.54	4.46
p05	7	85.47	4.53
p07	1	83.90	6.10
p08	5	83.32	6.68
m01	10	84.00	6.00
m02	7	83.49	6.51
m03	4	82.65	7.35
m04	3	87.27	2.73
Total	68	84.66	5.34

Tabelle 5.2: Auswertung der Winkel zwischen senkrechten Flächen. Winkel sind in Grad angegeben.

Bild	Zu verbindende Kanten		Korrekt verbunden	Falsch verbunden	Zu verbindende Eckpunkte		Korrekt verbunden
	vor	nach			vor	nach	
	Polygon Bestimmung				Polygon Bestimmung		
p01	7	7	7	0	2	2	2
p02	4	4	4	0	1	1	1
p03	7	5	4	0	1	0	0
p04	7	7	5	0	2	1	1
p05	7	7	7	0	2	2	1
p06	2	2	2	0	0	0	0
p07	7	3	3	0	2	0	0
p08	8	4	4	0	1	0	0
m01	6	5	3	1	1	1	1
m02	6	6	3	0	2	2	1
m03	2	2	0	0	0	0	0
m04	9	6	6	0	2	1	1
m05	6	4	3	0	0	0	0
Total	78	62	52	1	16	10	8

Tabelle 5.3: Auswertung der Verbindung der Polygone.

5.4 Validierung und Verifikation

Die Auswertung der Winkel zwischen senkrechten Flächen (Tabelle 5.2) ergibt im Schnitt eine Differenz von 5.34° zum Optimum von 90°. An dieser Stelle fehlt die Gegenüberstellung zu den Winkeln, die sich ergeben, wenn das Verbinden von Kanten und somit die Korrektur der Orientierung von Ebenen nicht durchgeführt wird. Anhand qualitativer Bewertung der Ergebnisse vor und nach dem Verbinden von Kanten ist insgesamt eine deutliche Verbesserung der Orientierung von Ebenen feststellbar. In Bild 5.2 ist ein Beispiel zum Vergleich dargestellt.

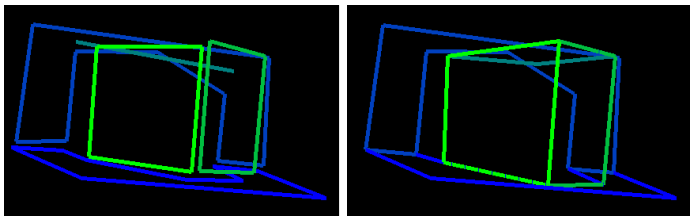

Bild 5.2: Die Szene vor und nach der Verschmelzung von Kanten

Die Auswertung der Verbindung der Polygone (Tabelle 5.3) zeigt, dass nach der Segmentierung und Polygonbestimmung nur noch ca. 79.5% der zu verschmelzenden Kantenpaare und 62.5% der Eckpunkte vorhanden sind. Dies ist vorallem auf das Problem der Untersegmentierung zurückzuführen, da dabei, wie in Bild 5.5 zu sehen, ganze Flächen fälschlicherweise als Ausreißer klassifiziert werden. 83.87% der noch vorhandenen Kantenpaare wurden korrekt verbunden, sowie 80% der Eckpunkte. Falsch verbunden wurde bei den verwendeten Testbildern nur eine von 62 Kanten.
In Anhang A sind die Ergebnisse zu allen Testszenen aufgeführt. Ein paar Beispiele werden im Folgenden genauer untersucht. Alle Bilder zur Darstellung der Ergebnisse zeigen die ursprünglichen Tiefenbilddaten als Grauwertbilddarstellung und als Darstellung der 3D–Punkte im Raum. Außerdem ist das Ergebnis der Segmentierung als Regionenbild dargestellt, sowie im großen Teilbild das Ergebnis der Verbindung der Polygone. In Bild 5.3 konnten alle Kanten und Eckpunkte erfolgreich verbunden werden. In der 3D–Darstellung ist zwischen der oberen und der vorderen Fläche des Quaders eine große Lücke mit nur dünn verteilten Tiefenwerten sichtbar. Die Position und Orientierung der oberen Fläche

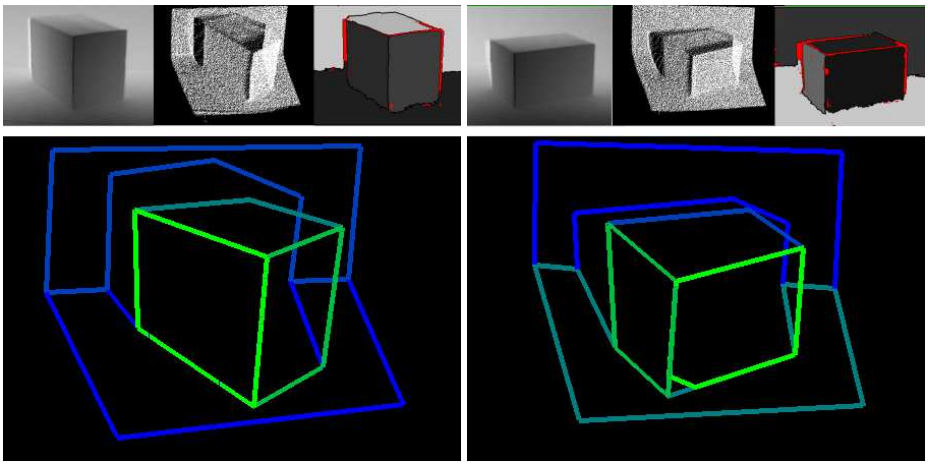

Bild 5.3: Ergebnis zu Testszene p01 Bild 5.4: Ergebnis zu Testszene p05

wurde erfolgreich korrigiert, die Form des Polygons ist allerdings etwas verfälscht, da bei dem in Kapitel 3.6 beschriebenen Prozess der Kantenverschmelzung die Winkel zwischen den Kanten des transformierten Polygons nicht erhalten bleiben. In Bild 5.6 ist ein Artefakt zu sehen, der durch das Vorgehen bei der Bestimmung des Segments der Schnittlinie zwischen zwei Ebenen entstehen kann. Das Problem tritt auf, wenn eine Ecke des Polygons nicht genau angepasst werden kann, sodass die Vor– oder Nachfolgerkante der zu verbindenden Kante ein kleines Kantenstrück darstellt, dessen Richtung stark von der korrekten Kante abweicht. Der Schnitt dieser Kante mit der anderen Ebene, hier im Beispiel die Bodenfläche, führt zu einer Entfernung vom eigentlichen Polygoneckpunkt. Das Problem der Untersegmentierung ist in Bild 5.5 zu sehen. Die obere und eine Seitenfläche des Quaders werden vollständig als Ausreißer klassifiziert, sodass diese im folgenden Ablauf der Polygon-Extraktion nicht miteinbezogen werden können. Bild 5.6 stellt ein Beispiel mit einem Objekt mit einer gekrümmten Flächen dar, diese wird durch mehrere planare Flächen approximiert. Die fehlerhafte Orientierung der oberen Fläche des Quaders ist auf eine fehlerhafte Verschmelzung zweier Kanten zurückzuführen: Die hintere Kante der oberen Fläche und die Kante im Polygon der Rückwand, die die obere ”Schattenkante ” des Quaders darstellt, erfüllen alle Verbindungskriterien. Bild 5.7 zeigt den Zustand nach der Anpassung der Polygone. Die genannten Kanten sind auch im 3D annähernd parallel,

sodass es zu einer Verschmelzung dieser Kanten kommt. Im Anschluss folgt die Verbindung des Polygons mit der oberen Kante der vorderen Fläche des Quaders, sodass sich das Polygon im Zustand FIXED befinden. Die Verschmelzung mit der oberen Kante der rechten Seitenfläche ist somit ein Schnitt der Flächen, wobei die Orientierung der Ebene nicht mehr korrigiert werden kann. Auch wenn eine fehlerhafte Verschmelzung von Kanten bei den verwendeten Testbildern nur einmal vorkommt, ist festzustellen, dass die Verbindungskriterien zu schwach sind. Bei einem Quader, der parallel zur Wand ausgerichtet ist, wir das Problem immer auftreten.

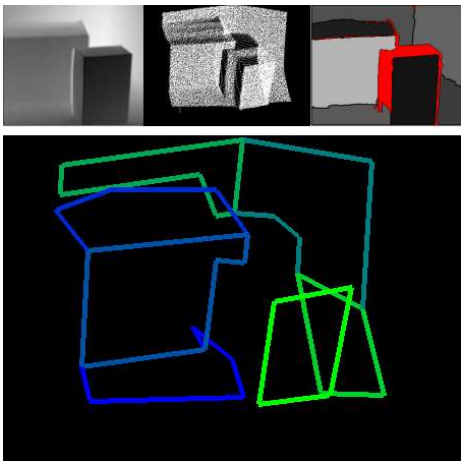

Bild 5.5: Ergebnis zu Testszene p08

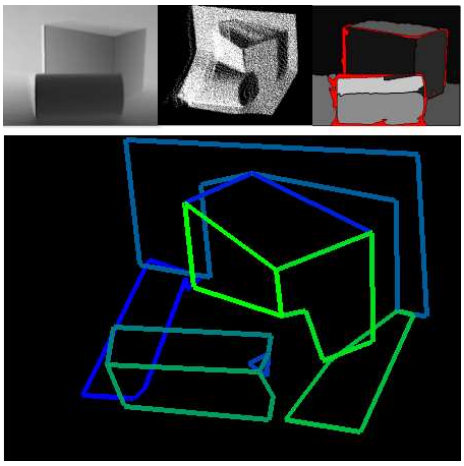

Bild 5.6: Ergebnis zu Testszene m01

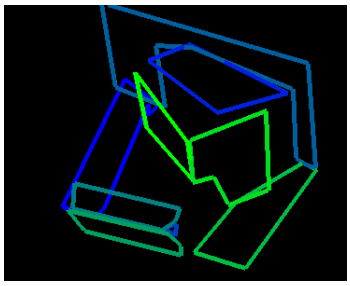

Bild 5.7:

Kapitel 6

Zusammenfassung

In dieser Studienarbeit wurde ein Verfahren zur Extraktion eines Oberflächenbegrenzungsmodells aus einem Tiefenbild vorgestellt. Das Modell beschreibt die im Tiefenbild dargestellte Szene durch die Geometrie und die Topologie der planaren Flächen, die in der Szene gefunden werden. Die Geometrie ist gegeben durch die Angabe der Ebenengleichungen der gefundenen Flächen sowie der 3D–Koordinaten der Eckpunkte der Polygone, die diese Flächen beschreiben. Die Informationen über die Topologie der Szene besteht aus einer Nachbarschaftsliste, die für jede Fläche angibt, über welche Kante diese Fläche mit welcher anderen Fläche verbunden ist.
Die Literaturrecherche zu Beginn der Studienarbeit ergab, dass zu diesem Zeitpunkt in der Programmierumgebung PUMA der Arbeitsgruppe bereits ein Verfahren zur Segmentierung von Tiefenbildern implementiert war [Zin01]. Das Ergebnis dieses Verfahrens ist ein Regionenbild, das jeden Pixel einer der segmentierten Flächen zuordnet, und eine Liste der planaren und gekrümmten Oberflächen mit den sie beschreibenden Ebenengleichungen und Quadriken. Diese Studienarbeit sollte sich auf die Extraktion planarer Flächen beschränken und gekrümmte Flächen durch mehrere planare Flächen annähern. Die Einarbeitung in das vorhandene System zur Segmentierung ergab, dass dieses einfach auch nur zur Segmentierung in planare Flächen genutzt werden kann. Das in dieser Studienarbeit entwickelte System baut auf also auf den Ergebnissen dieses Segmentierungsalgorithmus auf. Ein Teil des Verfahrens, die Zuordnung von als Ausreißer klassifizierten Punkten,

musste im Laufe der Studienarbeit selbst angepasst werden, um zu verhindern, dass auch ungültige Punkte, die Artefakte darstellen, einer Fläche zugewiesen werden.
Der Schwerpunkt dieser Studienarbeit wurde somit auf die Approximation der Begrenzungen der segmentierten Flächen durch dreidimensionale, planare Polygone gelegt, sowie die Verbindung dieser Polygone über Kanten und Eckpunkte. Es folgt ein kurzer Überblick über die Komponenten des entwickelten Verfahrens. Für die Bestimmung der Konturen der Regionen wurde das Crack–Following Verfahren, wie in [Kin97] beschrieben, implementiert. Die Konturpunkte der Regionen liefern, anhand der Daten aus dem Tiefenbild, direkt die 3D–Konturpunkte der Oberflächen. Die Bestimmung des Polygons, das eine Fläche beschreibt, erfolgt durch Inkrementelles Line Fitting auf den 3D–Konturpunkten der Fläche. Für die Geradenanpassung wurde das in [AV05] erläuterte Orthogonal Distance Regression Fitting implementiert. Das starke Rauschen der Tiefenwerte der Oberflächenkonturpunkte erfordert einen hohen Schwellwert für den maximal erlaubten Anpassungsfehler beim Inkrementellen Line Fitting. Dies resultiert beim Standardverfahren ([NMTS05]) in einer ungenauen Anpassung des Polygons an die Kontur. Um auch bei einem hohen Schwellwert eine genaue Anpassung zu erreichen, wurde das Standardverfahren in dieser Studienarbeit erweitert.
Durch die dünn verteilten Tiefenwerte im Bereich von Objektkanten, können besonders vom Betrachter abgeneigte Flächen nicht vollständig segmentiert werden. Dies führt zu großen Lücken zwischen rekonstruierten Flächen, die im realen Modell eigentlich verbunden sind. Nach der Bestimmung der Polygone, die die einzelnen Flächen beschreiben, wird daher versucht, diese über Kanten und Eckpunkte zu verbinden. Anhand verschiedener Kriterien wird für jedes Flächenpaar ermittelt, welche Kanten zu verbinden sind. Die Orientierung von Flächen, die vom Betrachter abgeneigt sind, wird durch Rauschen in den Tiefenwerten verfälscht. Daher wird die Verbindung der Polygone genutzt, um die Orientierung solcher Flächen zu korrigieren. Unter Verwendung eines neigungsbasierten Zuverlässigkeitsmaßes von Flächen wird entschieden, welche Fläche bei der Verbindung von Kanten und Eckpunkten in ihrer Position und Orientierung verändert wird.
Im Laufe der Studienarbeit wurde neben der Entwicklung des eigentlichen Systems, eine grafische Benutzungsoberfläche erstellt. Diese dient der Demonstration der Zwischenergebnisse des Verfahrens und der einfachen Durchführung von Testläufen mit verschiedenen Parametern. Die Daten des Tiefenbildes und die extrahierten Polygone werden dabei

dreidimensional dargestellt. Auch der Ablauf des vorhandenen Segmentierungsalgorithmus wird über diese Oberfläche gesteuert und visualisiert. Die Implementierung des eigentlichen Systems ist dabei völlig unabhängig von der Implementierung der GUI.
Für die Aufnahme von Tiefenbildern wurde die 3D–Laserkamera der Arbeitsgruppe wieder in Betrieb genommen und fest im Labor installiert. Während der Entwicklung der Studienarbeit wurden die Bilder dieser Kamera verwendet. Diese enthalten für jeden Bildpunkt die 3D–Koordinate des abgebildeten Punktes. Parallel zu dieser Arbeit wurde von Peter Decker die Studienarbeit zum Thema *Erstellung einer Tiefenkarte aus Stereobildern für den RoboCup Rescue Wettbewerb* entwickelt. Daher sollte das System auch mit Tiefenbildern arbeiten, die nur die Information über die Tiefe des abgebildeten Punktes enthalten. Zu diesem Zweck wurde die PUMA Klasse RangeImage zu einem Template erweitert, sodass auch Fließkommawerte aufgenommen werden können.
Die Ergebnisse des entwickelten Verfahrens wurden experimentell überprüft und bewertet. Dabei konnte festgestellt werden, dass beim Verbinden der Kanten eine gute Anpassung der Orientierung der Flächen erfolgt. Auch die Verbindung der nach der Segmentierung noch vorhandenen Kanten ist in über 80% der Fälle erfolgreich. Durch die Experimente konnten aber auch noch vorhandene Schwachstellen des Systems entdeckt werden. Die Kriterien für die Verbindung zweier Kanten sind noch zu schwach. Bei den Testdaten wurden zwar nur einmal Kanten verbunden, die nicht zusammen gehören, aber theoretische Überlegungen ergeben, dass dieser Fall häufiger eintreffen kann. Ein weiteres Problem besteht in der Untersegmentierung der Flächenregionen, mit der Folge, dass ganze Flächen als Aureißer klassifiziert werden und somit verloren gehen. An dieser Stelle soll auch darauf hingewiesen werden, dass das System derzeit nicht mit Flächen arbeiten kann, die Löcher aufweisen. Die somit entstehende innere Kontur wird vom System nicht als der Fläche zugehörig ermittelt.
Die Idee zu Beginn der Arbeit war, das zu entwickelnde System in das System des Projektpraktikums Robbie 6 einzubinden, um die Teilnahme am RoboCup 2006 zu unterstützen. Da diese Arbeit und das Projekt, sowie die Studienarbeit von Peter Decker parallel liefen, war eine Integration der Systeme bis zum Zeitpunkt des Wettberwebs nicht möglich. Derzeit wird vom neuen Robbie–Projekt ein 3D–Lasersanner in Betrieb genommen. Wenn die Tiefenbilder, die dieses Gerät liefert nicht zu sehr von den für diese Arbeit verwendeten Tiefenbildern abweichen, ist zu erwarten, dass das System eingesetzt werden kann. Falls

die Tiefenbilder dünner besetzt sind, kann es vermutlich zu Problemen mit der Segmentierung kommen. Das in dieser Studienarbeit entwickelte System ist von der Implementierung des Segmentierungsalgorithmus entkoppelt, sodass in diesem Fall eine Anpassung oder Veränderung der Segmentierung ausreichen sollte.

Anhang A

Ergebnisdokumentation

In diesem Anhang sind die Ergebnisse zu allen Testszenen aufgeführt. Alle Bilder zur Darstellung der Ergebnisse zeigen die ursprünglichen Tiefenbilddaten als Grauwertbilddarstellung und als Darstellung der 3D–Punkte im Raum. Außerdem ist das Ergebnis der Segmentierung als Regionenbild dargestellt, sowie im großen Teilbild das Ergebnis der Verbindung der Polygone.

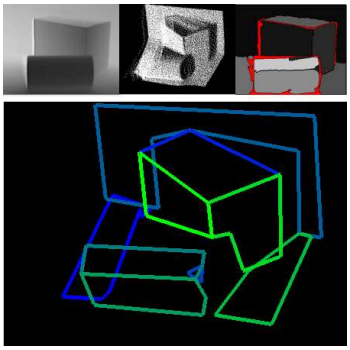

Bild A.1: Ergebnis zu Testszene m01

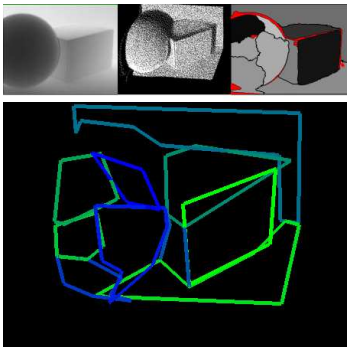

Bild A.2: Ergebnis zu Testszene m02

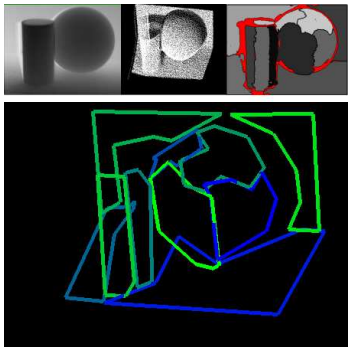

Bild A.3: Ergebnis zu Testszene m03

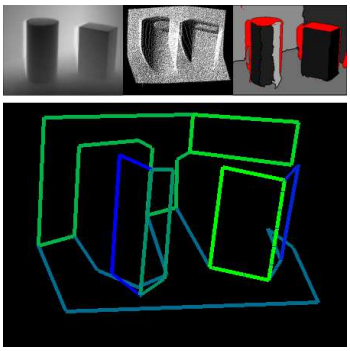

Bild A.4: Ergebnis zu Testszene m04

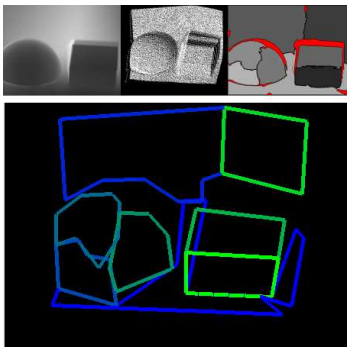

Bild A.5: Ergebnis zu Testszene m05

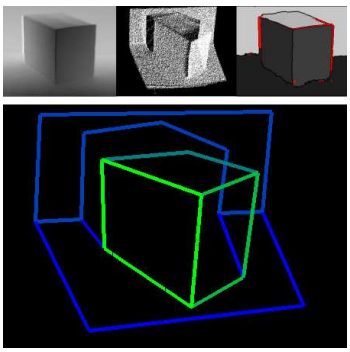

Bild A.6: Ergebnis zu Testszene p01

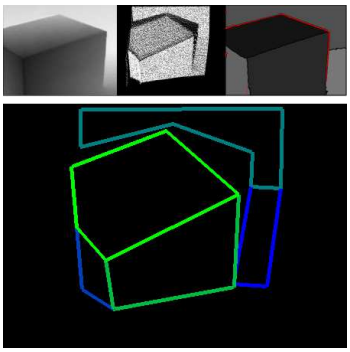

Bild A.7: Ergebnis zu Testszene p02

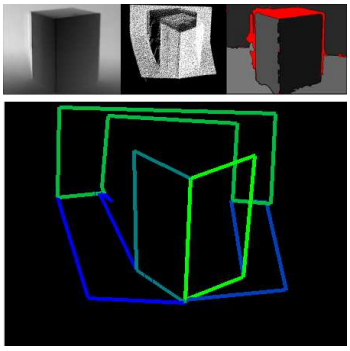

Bild A.8: Ergebnis zu Testszene p03

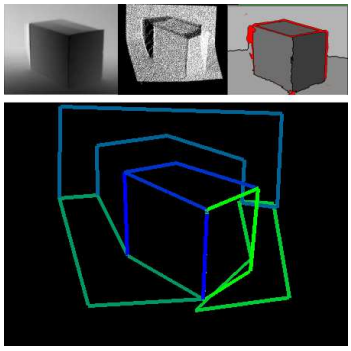

Bild A.9: Ergebnis zu Testszene p04

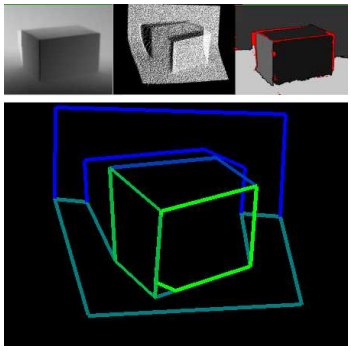

Bild A.10: Ergebnis zu Testszene p05

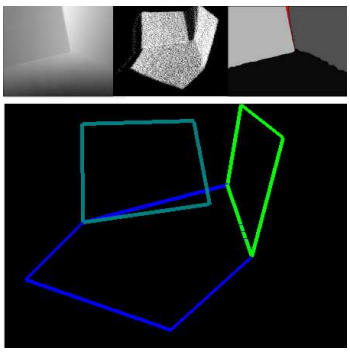

Bild A.11: Ergebnis zu Testszene p06

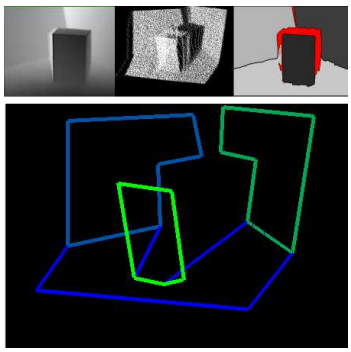

Bild A.12: Ergebnis zu Testszene p07

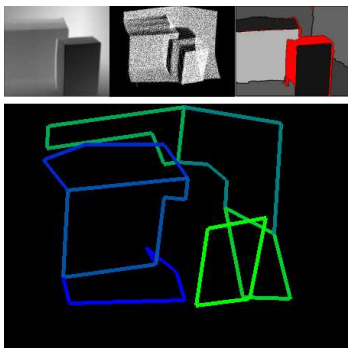

Bild A.13: Ergebnis zu Testszene p08

Anhang B

Installationsanleitung

B.1 Systemvoraussetzungen

Im Folgenden sind die Bibliotheken aufgelistet, die für eine Installation des Programms benötigt werden. Die Versionsangaben beziehen sich auf die Versionen, die bei der Entwicklung des Programms verwendet wurden.

- PUMA (Version development-branch-0-99)
- Qt (Version 3.3.4)
- OpenCV (Version 0.9.6)
- OpenGL (Mesa–Bibliothek Version 6.8.2-30)

Die PUMA Bibliothek muss in der angegeben Version oder in einer neueren Version vorliegen. Es ist darauf zu achten, dass vor der Installation von Puma das Linear Algebra Package (lapack) installiert wird. Dieses wird vom Programm zur Berechnung von Eigenvektoren benötigt. Hierzu müssen die vier Pakete `lapack`, `lapack-manpages`, `blas` und `blasman` und eventuell Fortran-Bibliotheken, z.B. `g2c`, installiert werden. Wenn bei der Konfiguration von Puma die Information

```
+ LAPACK library present (via '-lg2c -llapack -lblas -lg2c
  -lg2c ')
```

ausgegeben wird, wurde die lapack Bibliothek erfolgreich installiert. Die Ausgabe bei der Konfiguration von PUMA liefert auch die Information, ob Qt und OpenCV korrekt installiert sind:

```
+ OPENCV libs found (via -L/usr/local/lib -lcxcore -lcv
  -lhighgui -lcvaux  )
+ QT installation found (via -L/usr/lib/qt3/lib -lqt-mt
  -lSM -lICE  -L/usr/X11R6/lib -lX11 -lXext -lXmu -lXt -lXi)
```

Bei der Installation von PUMA muss bei der Konfiguration die Option

```
--with-ric
```

angegeben werden, damit die Klassen zur Segmentierung von Tiefenbildern eingebunden werden.

Für 3D–Darstellungen in der GUI wird OpenGL benötigt. Hierfür kann unter Linux z.B. die Mesa–Bibliothek (Laufzeitumgebung (libGL/libGLU) für den Betrieb von OpenGL-Programmen) installiert werden. Weitere Informationen zur Verwendung von OpenGL in Qt sind unter `http://doc.trolltech.com/3.3/opengl.html` zu finden.

B.2 Compilieren und Ausführen des Programms

Der gesamte Quellcode der Studienarbeit befindet sich auf der beiligenden CD im Verzeichnis `Quellcode`. Das Unterverzeichnis `GUI` enthält das bereits compilierte und ausführbare Programm `Gui` zur Visualisierung der Studienarbeit. Die grafische Oberfläche wird mit dem Aufruf `./Gui` gestartet. Compiliert wird das Programm durch den Aufruf von `make` in diesem Verzeichnis. Dieses Makefile wird durch den Aufruf von `qmake Gui.pro` automatisch von Qt generiert.

Im Verzeichnis `PumaModifications` befinden sich die Dateien aus Puma, die im Laufe der Studienarbeit angepasst wurden. Diese sind nur zur Dokumentation dort abgelegt und werden nicht zur Übersetzung des Programms benötigt.

Literaturverzeichnis

[AV05] H. Aceves and H. Velazquez. Scalings between physical and their observationally related quantities of merger remnants. *Revista Mexicana de Astronomia y Astrofisica*, (41):523–532, 2005.

[Bes88] P. J. Besl. Active, optical range imaging sensors. *Machine Vision and Applications*, 1(2):127–152, 1988.

[BJ88] Paul J. Besl and Ramesh C. Jain. Segmentation through variable-order surface fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(2):167–192, 3 1988.

[BS92] Suchendra M. Bhandarkar and Andreas Siebert. Integrating edge and surface information for range image segmentation. *Pattern Recognition*, 25(9):947–962, 1992.

[CC05] C. Chao Chen. A survey of techniques on range image segmentation and registration. Ph.D. Second-level Exam Part I, 2005.

[FEF95] A. W. Fitzgibbon, D. W. Eggert, and R. B. Fisher. High-level cad model acquisition from range images. In *University of Edinburgh*, Department of Artificial Intelligence, 1995. University of Edinburgh.

[HBG94] Adam Hoover, Kevin W. Bowyer, and Dimitry Goldgof. Building a valid boundary representation from a segmented range image. Technical report, University of South Florida, Department of Computer Science and Engineering, 1994.

[HJBJ+96] A. Hoover, G. Jean-Baptiste, X. Jiang, P. Flynn, H. Bunke, D. Goldgof, K. Bowyer, D. Eggert, A. Fitzgibbon, and R. Fisher. An experimental comparison of range image segmentation algorithms. *IEEE Trans. on PAMI*, 18(7):673–689, 1996.

[Hof89] Christoph M. Hoffmann. *Geometric and Solid Modeling: An Introduction*. Morgan Kaufmann Publishers, Inc., San Mateo CA, 1989. ISBN 1-55860-067-1.

[HZ00] Richard I. Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000. Uni Koblenz Bib Signatur: INF 2002/7531 + INF 2002/8013.

[JJ90] Ramesh C. Jain and Anil K. Jain. *Analysis and Interpretation of Range Images*. Springer-Verlag, 1990.

[Kin97] V. Kindratenko. *Development and Application of Image Analysis Techniques for Identification and Classification of Microscopic Particles*. PhD thesis, University of Antwerp, Department of Chemistry, 1997.

[Koc96] R. Koch. Surface segmentation and modelling of 3-d polygonal objects from stereoscopic image pairs. In *International Conference on Pattern Recognition*, volume 1, pages 233–237, 1996.

[Liu93] Haibin Liu. *Oberflächenbasierte Segmentierung von Tiefenbildern*, volume 36 of *Dissertationen zur Künstlichen Intelligenz*. Infix Verlag, St. Augustin, Germany, 1993.

[NMTS05] V. Nguyen, A. Martinelli, N. Tomatis, and R. Siegwart. A comparison of line extraction algorithms using 2d laser rangefinder for indoor mobile robotics. *IEEE International Conference on Intelligent Robots and Systems*, 2005.

[SA02] I. Stamos and P. K. Allen. Geometry and texture recovery of scenes of large scale. *Computer Vision and Image Understanding*, 88(2):94–118, 11 2002.

[TV98] E. Trucco and A. Verri. *Introductory Techniques for 3–D Computer Vision*. Prentice Hall, New York, 1998.

[Wic04] Daniel Wickeroth. Einbindung einer 3D-Kamera in das Netzwerk der Universität. Studienarbeit, Universität Koblenz-Landau, 2004.

[Zin01] Timo Zinßer. Oberflächensegmentierung in Tiefenbildern. Studienarbeit, Friedrich-Alexander-Universität Erlangen-Nürnberg, 2001.