Institut für Parallele und Verteilte Systeme
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3316

Polygonmodellrekonstruktion aus Punktwolken

Gregor Rothmaier

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. Paul Levi
Betreuer:	Dipl.-Inf. Daniel Di Marco
begonnen am:	10. April 2012
beendet am:	10. Oktober 2012
CR-Klassifikation:	I.2.10, I.4.5

Kurzfassung

Roboter kommen in vielen Gebieten immer stärker zum Einsatz. Eine Herausforderung der Robotik besteht darin, unbekannte bzw. nicht vollständige Umgebungen mittels externer Sensoren aufzuklären. Deren 3D-Ergebnisse werden üblicherweise in Form von Punktwolken erfasst.

In der vorliegenden Arbeit werden vorhandene Algorithmen zur Polygonmodellrekonstruktion mit Hilfe von Punktwolken aus der RoboEarth Datenbank bewertet. Kriterium ist dabei die möglichst originalgetreue Wiedergabe des abgebildeten Objekts. Die in der RoboEarth Datenbank hinterlegten Punktwolken wurden mit der Microsoft Kinect Kamera erzeugt. Diese Kamera ist für den Consumer Markt konzipiert, d.h., der Preis ist wichtiger als eine ausgeprägte Präzision. Die durch die fehlende Präzision entstandenen Aufnahmefehler werden ebenfalls in der Arbeit berücksichtigt.

Abstract

Robots are increasingly common in many areas. One challenge in robot science is scouting of unknown or rather incomplete environments with external sensors. Their 3D results are usually provided as pointclouds.

In this thesis, existing algorithms reconstructing polygon meshs are evaluated using pointclouds from the RoboEarth database. One main criterion for evaluation is the accuracy of the reconstruction for a given object. The pointclouds taken from the RoboEarth database were generated with the Microsoft Kinect camera. This camera adresses the consumer market and for that reason price is more important than precision. Recording errors due to the lack of high precision are also taken into account in this thesis.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

1 Einleitung

Dreidimensionale Daten (3D-Daten) sind heutzutage aus zahlreichen Indutrieproduktionen nicht mehr wegzudenken. Sie werden ebenso in vielen anderen Fachbereichen, zum Beispiel in der Geologie, Geodäsie, Seismologie und Archäologie verwendet und dienen sogar zur Archivierung von wertvollen Kunstschätzen.

3D-Daten eines Objektes lassen sich durch Laserabtastung des Objektes gewinnen, die zwar genaue Daten liefert, aber durch ihre hohen Kosten hauptsächlich in industriellen Prozessen Anwendung findet. Eine weitere Methode ist die bildgestützte Gewinnung, wobei diese zum einen mit mehreren Kameras durchgeführt werden kann oder zum anderen mit nur einer, jedoch bewegten, Kamera auskommt. In beiden Fällen wird das Objekt aus verschiedenen Blickwinkeln betrachtet und so die 3D-Daten in Form von kartesischen 3D-Koordinaten gewonnen.

Der Vorteil der bildgestützten gegenüber der lasergestützten Gewinnung sind die niedrigeren Kosten, die sogar die Erschließung des Consumer Marktes möglich machte, zum Beispiel die private Erstellung von 3D-Bildern.

Die gewonnenen 3D-Koordinaten repräsentieren Positionen oder Punkte im euklidischen Raum, ihre Gesamtheit nennt man auch Punktwolke. Diese Punktwolke bildet eine Teilmenge der Oberfläche des dreidimensionalen Objektes, sie enthält allerdings nicht mehr alle geometrischen und topologischen Informationen. Zur weiteren Verarbeitung dieser 3D-Daten, bzw. der Punktwolke, wird daraus ein Objektmodell rekonstruiert, das aus Polygonen zusammengesetzt ist.

Es wurden zahlreiche Algorithmen zur Polygonisierung entwickelt, die jeweils Ergebnisse unterschiedlicher Qualität liefern. Ziel der Algorithmen ist es, ein Polygonmodell aus der Punktwolke zu rekonstruieren, das dem Original möglichst nahe kommt und so den Informationsverlust durch die Abtastung wieder ausgleicht. Viele Algorithmen in der Computergrafik arbeiten ausschließlich mit Polygonnetzen. Auch diverse Finite-Elemente-Methoden basieren auf dieser Darstellungsform.

Die Steigerung der Produktivität, aber auch die Realisierung humanerer Arbeitsprozesse hat bewirkt, dass Roboter in vielen Industrieprozessen immer verstärkter zum Einsatz kommen. Die spezielle Herausforderung der Robotik besteht darin, unbekannte bzw. nicht vollständige Umgebungen aufzuklären. Dies erfolgt über externe Sensoren am Roboter. Auch hierzu bietet sich die bildgestützte Gewinnung von Punktwolken an. Dafür wird im Hardwarebereich innerhalb der Informationswissenschaften vermehrt Consumer Hardware verwendet, da sie aufgrund ihres Preises auch in größeren Stückzahlen leicht zu beschaffen ist und die

Weiterentwicklung der Anwendungssoftware sowohl durch die Hersteller als auch einer großen Anwendercommunity weitergetrieben wird.

1.1 Aufgabenstellung

Das EU-geförderte Projekt RoboEarth hat zum Ziel, eine weltweit nutzbare Datenbank zum Wissensaustausch für Roboter zu erstellen. Austauschbar sind Handlungsanweisungen, Umgebungsbeschreibungen und insbesondere auch Objektmodelle. Momentan können Objektmodelle als farbige Punktwolken aufgenommen werden. Für verschiedene Anwendungen (z.B. die Berechnung von Greifpunkten) ist dagegen ein klassisches Polygonmodell erforderlich. In dieser Arbeit sollen Verfahren untersucht werden, welche zu den vorhandenen Punktwolken-Modellen eine Oberflächenrekonstruktion durchführen und so ein aus Polygonen bestehendes 3D-Modell konstruieren können.

Voraussetzung für die Durchführung der Arbeit sind sehr gute Kenntnisse in der Programmiersprache C++ und grundlegende Kenntnisse in dem Bereich der Bildverarbeitung. Die entstehende Software ist so zu gestalten, dass sie leicht wartbar und ohne großen Aufwand weiterzuentwickeln ist.

1.2 Verwandte Arbeiten

Im Bereich der bildgestützten 3D-Datenerfassung mit Consumer Hardware gibt es weitere Ansätze, die gewonnen Daten zu verarbeiten. So wird zum Beispiel von Microsoft Research in Großbritannien am Projekt KinectFusion gearbeitet.

1.2.1 KinectFusion

Mit der auf der Kamera „Kinect“ (siehe Kapitel 2.1) aufsetzenden Software wird es Nutzern ermöglicht, qualitativ hochwertige 3D-Modelle in Echtzeit zu generieren. Eine integrierte „Physik-Engine“ erlaubt es, die eingescannten Objekte realistisch zu manipulieren.

Mit dieser Technik werden ganze Räume, Objekte und Personen dreidimensional erfasst. Mit den Manipulationsmöglichkeiten der KinectFusion ist somit praktisch sehr viel denkbar, wie vom lebensechten Avatar über Objekte, die sich in virtuelle Umgebungen, wie zum Beispiel in Spiele, importieren lassen.

Die Besonderheit an KinectFusion ist, dass die Berechnungen mit einer Standard-Grafikkarte funktionieren. Diese übernimmt sowohl das Tracking der Kamerabewegungen als auch die Bildgenerierung.

KinectFusion ermöglicht einem Benutzer das Halten und Bewegen einer Standard Kinect-Kamera für die rasche Erstellung detaillierter 3D-Rekonstruktionen einer Indoor-Szene. Nur die Tiefeninformationen der Kinect werden verwendet um die 3D Position des Sensors

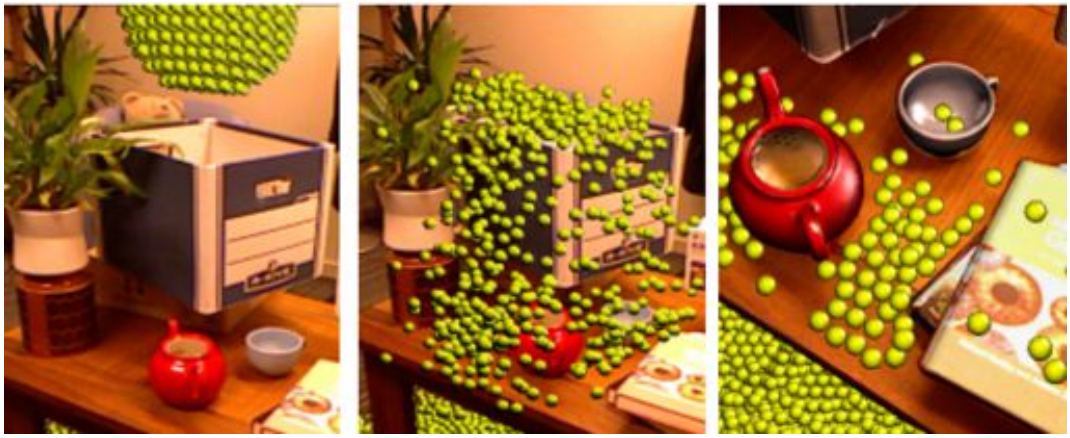

Abbildung 1.1: Interaktive Physiksimulation, direkt auf dem 3D-Modell der Szene. Die Partikel interagieren mit der Szenerie während der Rekonstruktion der Szenerie.[IKH+11]

zu verfolgen und geometrisch korrekte 3D-Modelle der physikalischen Szene in Echtzeit zu rekonstruieren. KinectFusion verwendet eine neuartige GPU-basierte Pipeline um die Menge an Daten schnell und effizient zu verarbeiten, um so dem Echtzeitanspruch der Rekonstruktion gerecht zu werden. [IKH+11]

2 Grundlagen

> Was gesagt werden kann, kann genau gesagt werden. Worüber man nicht sprechen kann, darüber soll man schweigen.
>
> *(Ludwig Wittgenstein)*

In dem Kapitel Grundlagen werden Werkzeuge und Begriffe beschrieben, die für diese Arbeit von Bedeutung sind. Das ist zum einen die Kamera „Kinect“ (Kapitel 2.1), die als Sensor zur bildgestützten Generierung von Punktwolken dient. Zum anderen wird das Robot-Operating-System (ROS) benutzt (Kapitel 2.2), welches die Softwarewerkzeuge zum Umgang mit der Kinect und zum Teil auch die Punktwolken beinhaltet. Des Weiteren liefert RoboEarth (Kapitel 2.3), welches in ROS enthalten ist, die Tools, um aus mehreren Punktwolken eine Gesamtwolke zu bilden und die Ergebnisse in einer Datenbank für Roboter abzulegen. Schließlich wird die Point Cloud Library (PCL) (Kapitel 2.4) beschrieben. Sie liefert verschiedenste Algorithmen, die mit Punktwolken interagieren. Die Eigenschaften von Punktwolken werden in Kapitel 2.5 aufgelistet, in Kapitel 2.6 das Abtasttheorem beschrieben und Kapitel 2.7 das Polygonnetz kurz definiert.

2.1 Kinect

Die Kinect (siehe Abbildung 2.1) wird seit Anfang November 2010 als eine alternative Steuerung von Spielen der Spielekonsole Xbox 360 verkauft. Ihre Besonderheit liegt darin, dass Spieler, anstelle der bisherigen Eingabemethoden über Gamepads, mit ihren Körperbewegungen das Spiel steuern können. Microsoft entwickelte die Kinect in Zusammenarbeit mit der Firma PrimeSense, die den wesentlichen Teil zur 3D-Erfassung lieferte.

Die Kinect setzt sich aus einem Tiefensensor, welcher von PrimeSense stammt, einem 3D-Mikrofon und einer Farbkamera zusammen. Eine Software kombiniert diese Sensordaten und ermöglicht so die Steuerung durch Bewegung und Sprache.

Den wichtigsten Teil, die Erfassung, übernimmt hierbei der Tiefensensor, der die Entfernungsdaten einer Szenerie als Punktwolke liefert. Dabei wird über einen IR-Projektor ein Fleckenbild über die Szene gelegt (siehe Abbildung 2.2) und mit einem IR-Sensor diese Flecken wieder aufgenommen. Die Entfernung wird dabei aus dem Abstand bzw. der Verzerrung der Flecken berechnet. Die Entfernungen werden für jedes Pixel der Szenerie in einer Punktwolke gespeichert. [FSMA08] Diese Verarbeitung wird von einem Chip innerhalb der Kinect durchgeführt, welcher die fertige Punktwolke liefert. Er bietet dabei keine Möglichkeit

Abbildung 2.1: Microsoft Kinect Vorderansicht.[Wika]

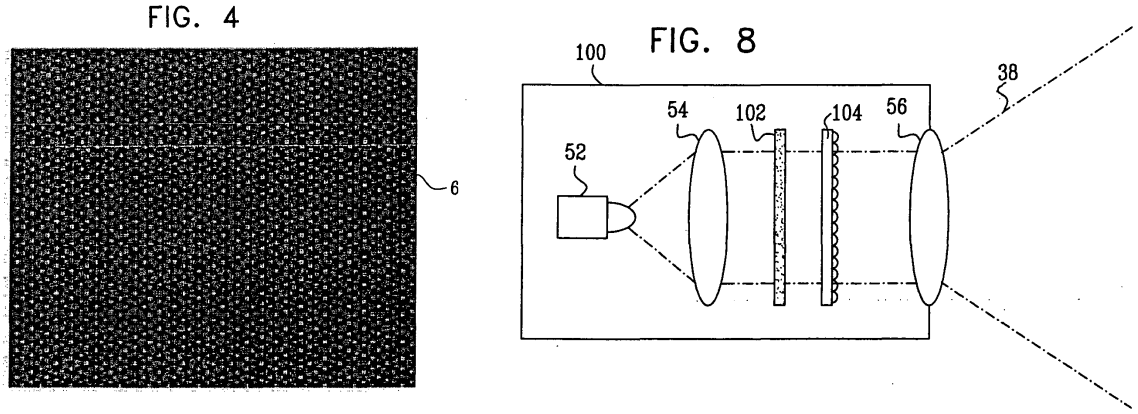

Abbildung 2.2: Projektion des Fleckenbildes laut Patent.[FSMA08]

die Erstellung zu steuern. Das Verfahren orientiert sich am „Structured Light Depth Sensing“, wobei hier anstatt des üblichen Streifenmusters auf ein Punktmuster zurückgegriffen wurde.

Die Kinect verwendet einen speziellen Algorithmus, um aus dieser Punktwolke in Echtzeit Objekte und Gesten zu identifizieren.

2.2 Robot Operating System (ROS)

Das Robot Operating System (ROS) ist ein Betriebssystem für persönliche Roboter. Unter persönlichen Robotern versteht man, im Gegensatz zu Industrierobotern, Roboter, die im

persönlichen Umfeld eingesetzt werden. Ein Beispiel sind die Roboterstaubsauger, die selbstständig einen abgesteckten Bereich von Staub freizuhalten versuchen.

Die Entwicklung von ROS begann 2007 am Stanford Artificial Intelligence Laboratory im Rahmen des Stanford AI Robot (STAIR) Projektes. Heute wird es hauptsächlich am Robotikinstitut Willow Garage weiterentwickelt. Es ist kein Betriebssystem im eigentlichen Sinne, sondern es handelt sich um ein Framework, dessen Bibliotheken auf schon bestehende Betriebssysteme wie Linux, Mac OS X oder Windows aufsetzen.

Die Hauptbestandteile und -aufgaben von ROS sind:

- Hardwareabstraktion, um Geräte einer höheren Softwareebene verfügbar zu machen,
- Gerätetreiber, um die verschiedenen auf dem Markt erhältlichen Geräte betreiben zu können (wie z.B. die Kinect),
- Oft wiederverwendete Funktionalitäten uniform bereitzustellen, um Inkompabilitäten zu vermeiden,
- Nachrichtenaustausch zwischen Programmen bzw. Programmteilen vereinfacht zu ermöglichen,
- Paketverwaltung zur vereinfachten Konfiguration bereitzustellen,
- Programmbibliotheken zum Verwalten und Betreiben der Software auf mehreren Computern bereitzustellen.

Das System ist aufgeteilt in das eigentliche Basissystem „ros“ und „ros-pkg“. Letzteres ist eine Auswahl an Zusatzpaketen, mit denen sich das Basissystem um einzelne Funktionen einfach erweitern lässt.

Im Zuge dieser Arbeit hat sich herausgestellt, dass das Paketsystem zuverlässig und stabil arbeitet, sodass es problemlos verwendet werden konnte. Andererseits weisen einige, v.a. visuelle Zusatzpakete viele meist verschränkte Verknüpfungen zu anderen Paketen auf. Diese Menge an Abhängigkeiten führt unweigerlich zu einer schwer handhabbaren Komplexität, welche, vor allem in Bezug auf den Entwurf eines minimalen Systems, Probleme bereiten würde.

2.3 RoboEarth

RoboEarth ist ein von der Europäischen Union gefördertes Projekt mit dem Ziel, eine große gemeinsame Wissensbasis für Robotersysteme aufzubauen. Diese ermöglicht den Robotern, Objekte zuverlässig zu identifizieren, ihre Handlungen über die Zeit zu verbessern und für unbekannte Umgebungen passende Navigationsstrategien abzurufen, die nicht explizit während der Entwicklung eingebaut wurden.

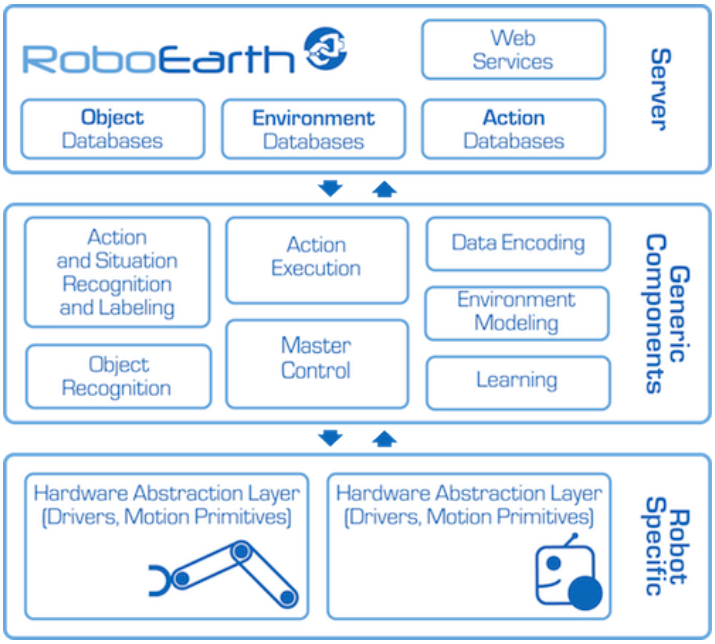

Abbildung 2.3: Schematischer Aufbau von RoboEarth.[Robb]

Die Vernetzung der Roboter über eine Wissensdatenbank beschleunigt den Lern- und Anpassungsprozess für komplexe Aufgaben. Das Ziel ist, diesen als einen autonomen Prozess zu gestalten.

Roboter sind normalerweise nicht in der Lage unstrukturierte Umgebungen, wie z.B. Umgebungen, in denen Menschen arbeiten, zu erfassen und zu bewältigen. Ihre Systeme sind nicht darauf ausgelegt, im Voraus alle Informationen über Besonderheiten jeder denkbaren Situation, auf die sie stoßen könnten, zu besitzen. Jede Reaktion auf eine eintretende Situation muss im Normalfall vorher schon programmiert sein. Ein Robotersystem muss, sobald es sich in einer unbekannten Situation befindet, kontrolliert reagieren. Dies kann entweder eine Standardreaktion, wie z.B. „anhalten“ sein, oder die Anwendung entsprechender Strategien, die es erlauben einen Lösungsweg zu berechnen. Die dafür notwendigen Informationen sind im Normalfall nicht vorhanden. Mit RoboEarth wird die Möglichkeit hierfür geboten. RoboEarth nutzt das Internet, um eine riesige Open-Source-Netzwerk-Datenbank zu schaffen, die von Robotern benutzt und ständig aktualisiert wird.

Das Ziel von RoboEarth ist, Roboter von Erfahrungen anderer Roboter profitieren zu lassen. Damit ist es nicht mehr notwendig, dass ein Roboter alles selbst lernen muss und es bietet sich die Möglichkeit schon erarbeitete Lösungen und Informationen anderen zur Verfügung zu stellen. Somit wird ein schnellerer Fortschritt in der Maschine-Welt-Interaktion sowie der Maschine-Mensch-Interaktion erreicht. [WBC+11]

In Abbildung 2.3 ist der schematische Aufbau von RoboEarth dargestellt. Die untere Ebene abstrahiert die roboterspezifischen Eigenschaften in eine einheitliche Schnittstelle, die von

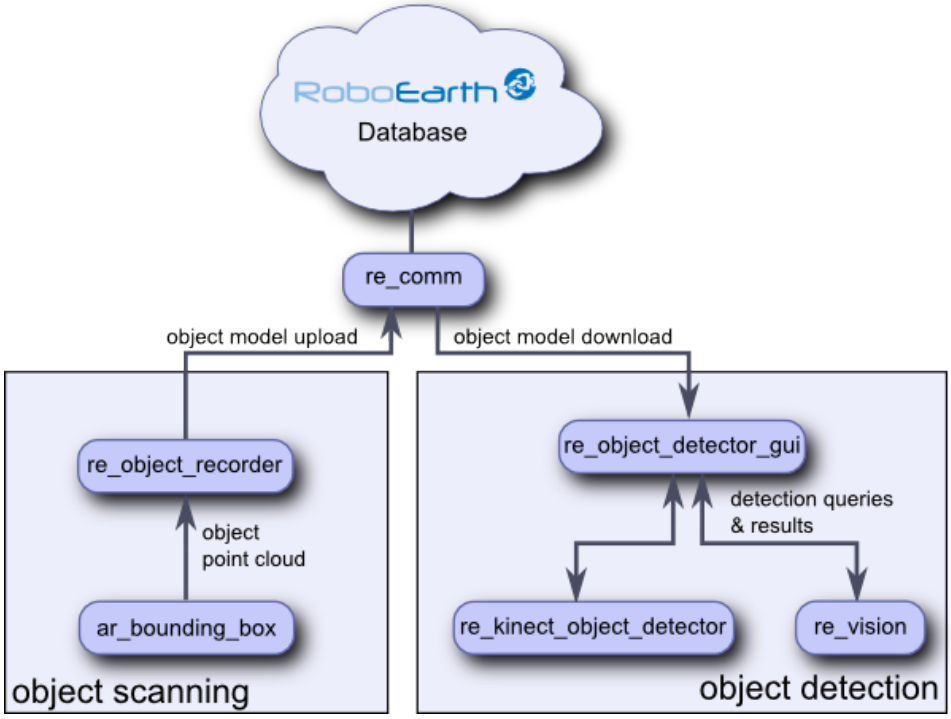

Abbildung 2.4: Kommunikation von Robotern mit der Wissensdatenbank RoboEarth.[Ros]

den generischen Komponenten verarbeitet werden kann. Die oberste Ebene mit dem Server stellt die Informationen in einer Datenbank bereit, die für die oben genannten Lösungsstrategien notwendig sind.

Die Kommunikation der Roboter mit RoboEarth ist in Abbildung 2.4 dargestellt. Sie umfasst im wesentlichen drei Bereiche, die in dem gleichnamigen ROS-Paket realisiert sind. Dies sind die Kommunikationsschnittstelle „re-comm“ selbst, das „object scanning“ und die „object detection“. Das „object scanning“ ermöglicht mit Hilfe der Kinect und einem speziellen Markierungsmuster 360-Grad Punktwolken eines Objektes zu generieren. Die „object detection“ ermöglicht umgekehrt solchermaßen erstellte Objekte aus der Datenbank wiederzuerkennen oder zu identifizieren. Zu diesen Objekten kann der Roboter zusätzlich vorhandene Anwendungs- und Interaktionsmöglichkeiten aus der Datenbank abrufen.

Im Bereich „object scanning“ werden, wie oben erwähnt, die 360-Grad Punktwolken erstellt. Dazu verwendet der „object_recorder“ ein Markierungsmuster wie in Abbildung 2.5 dargestellt. Anhand dieses Markierungsmusters kann der Recorder jede Aufnahme, die z.B. die Kinect liefert, in sein Koordinatensystem eindeutig einsortieren. Dazu benutzt er mehrere der deutlich unterscheidbaren Markierungen. Für eine 360-Grad Aufnahme werden mehrere Kamerapositionen rund um das Objekt benötigt. Ein aufzunehmendes Objekt wird auf das RoboEarth Logo in der Mitte gestellt. In einem späteren Schritt werden die Einzelaufnah-

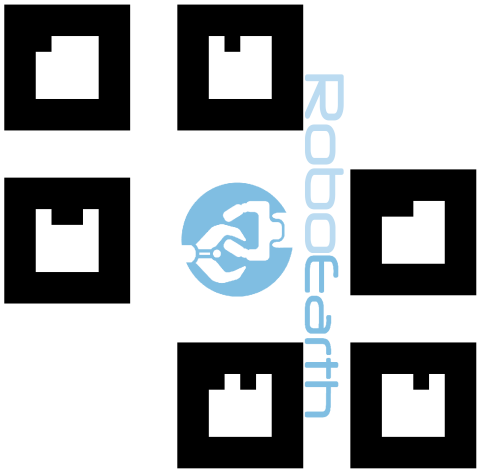

Abbildung 2.5: Markierungsmuster das von RoboEarth verwendet wird.[Roba]

Abbildung 2.6: PCL-Logo.[PCL]

men zu einer Gesamtaufnahme vereint. Das Ergebnis ist im Fall der Kinect eine 360-Grad Punktwolke eines Objektes.

2.4 Point Cloud Library (PCL)

Die Point Cloud Library (PCL) (Abbildung 2.6) ist eine großes, Open Source Projekt zur Verarbeitung von Punktwolken. Das PCL Framework enthält zahlreiche state-of-the-art Algorithmen einschließlich Filterung, Segmentierung, Erkennung von Merkmalen, Oberflächenrekonstruktion und Modellanpassung. Diese Algorithmen werden verwendet, um Oberflächen aus Punktwolken zu erzeugen und zu visualisieren. Ebenso dienen sie dazu, z.B. Ausreißer aus verrauschten Daten zu filtern, 3D-Punktwolken zu vereinen, oder relevante Teile einer Szene zu segmentieren. Weiter können Schlüsselpunkte extrahiert und Deskriptoren berechnet werden, um Objekte in einer Szene anhand ihres geometrischen Erscheinungsbildes zu erkennen.

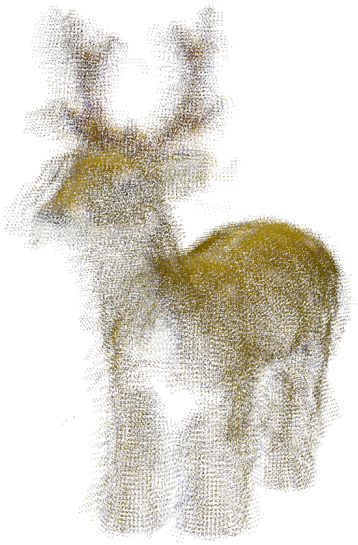

Abbildung 2.7: Punktwolke inklusive Farbinformation eines Plüschrentiers.

PCL ist unter den Bedingungen der Berkeley Software Distribution (BSD) Lizenz veröffentlicht und ist damit Open Source Software. Sie kann ohne Lizenzkosten sowohl für den kommerziellen Gebrauch als auch die Forschung genutzt werden.

Merkmal dieses Open Source Projekts ist die Entwicklung in sogenannten Sprints, die durch eine große Entwicklercommunity erfolgt. Das Projekt bietet daher mit seinen Rekonstruktionsalgorithmen für Punktwolken den optimalen Beitrag für diese Arbeit. Zudem ist die PCL auch in ROS enthalten und kann daher gut in das vorhandene RoboEarth eingebaut werden.

2.5 Punktwolken

Eine Punktwolke ist der Ausgangspunkt für die Rekonstruktion eines Objekts. Daher soll sie zunächst definiert und einige ihrer Eigenheiten beschrieben werden.

Eine Punktwolke P ist eine Menge von Punkten $p_i \in \mathbb{R}^n$ mit $i = 1, \ldots, k$ in einem n-dimensionalen kartesischen Koordinatensystem.

Da zu den Punkten weitere Daten ermittelt werden können, besteht ein Punkt nicht nur aus Koordinateninformationen, sondern beinhaltet zusätzliche Informationen. Ein Beispiel für eine solche Information ist der Farbwert des Punktes. Im dreidimensionalen Raum mit Erfassung der Farbwerte wäre $P = \{(p_i, c_i) : i = 1, \ldots, k; p_i \in \mathbb{R}^3; c_i \in RGB\}$.

Abbildung 2.7 zeigt die Punktwolke eines Plüschrentiers. Die Punkte enthalten auch die jeweilige Farbinformation, die ebenfalls zu sehen ist.

Punktwolken besitzen einige Eigenschaften, auf die hier kurz eingegangen werden soll. Diese Eigenschaften, im Speziellen jener der Kinect-Punktwolken, wirken sich in ihrer späteren Bearbeitung aus.

Diskretheit

Punktwolken, vor allem die in dieser Arbeit verwendeten, stellen Repräsentationen von kontinuierlichen, also nicht diskreten, Objekten dar, die mittels Sensoren aufgezeichnet und durch einen Computer vorverarbeitet wurden. Die Punktmengen sind daher endlich und bestehen aus diskreten Punkten. Die Rekonstruktion macht dabei aus diesen diskreten Punkten, zumindest lokal, kontinuierliche Flächen. [KB00]

Verteilung

Die Verteilung bezieht sich darauf, wie weit die Punkte einer Punktwolke im Raum verstreut sind. Die verwendeten Punktwolken sind lokal stark begrenzt, da es sich um gezielte Objektaufnahmen handelt. Zudem ist die Verteilung nicht uniform, wie zum Beispiel bei Voxelgittern. Dies bringt den Vorteil, dass ein Punkt tatsächlich den abgetasteten Punkt am Originalobjekt repräsentiert und nicht durch einen Voxel approximiert wird. Allerdings bereitet dies beim Verarbeiten der Punktwolken zusätzliche Probleme.

Dichte

Bei der Dichte handelt es sich um die Punkte pro Volumeneinheit, welche sich aus der Verteilung der Punktmenge ableitet. Dabei hängt die Dichte der verwendeten Punktwolken von der Anzahl der Aufnahmen ab, die für die Generierung verwendet wurden. Je mehr Aufnahmen, desto größer ist die Dichte, da die verschiedenen Perspektiven übereinandergelegt werden.

Rauschen

Daten, die mittels physikalischer Sensoren aufgezeichnet werden, beinhalten immer ein Rauschen. Rauschen ist dabei eine Störgröße, welche die Messwerte verfälscht. Bei den Punktwolken, die mit der Kinect erzeugt werden, ist das Rauschen sogar sehr groß. Dies liegt zum einen an der Hardware selbst, da die Kinect nicht auf Präzision ausgelegt ist und zum anderen wird das zu erfassende Objekt zusätzlich noch bewegt. Es wird dabei zwar versucht die Ausrichtung mit Markierungsmustern genau zu ermitteln, dies funktioniert allerdings nicht immer perfekt. [PMG04]

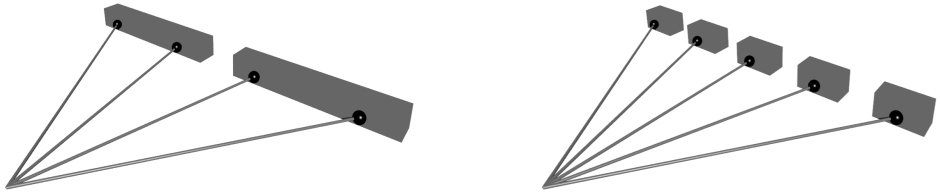

Abbildung 2.8: Abtastung und entstehender Fehler bei diskreten Messwerten.[Hem09]

2.6 Abtasttheorem

Der Aufwand, der bei der Erfassung eines Objektes betrieben wird, ist entscheidend für die Qualität der rekonstruierten Oberfläche. Je mehr Ansichten von einem Objekt vorhanden sind, um so größer ist theoretisch die Dichte, welche die Qualität der Rekonstruktion, insbesondere bei Details verbessert. Der Rechenaufwand steigt hierbei allerdings ebenfalls.

Aus der Signalverarbeitung ist das Abtasttheorem, auch Nyquist-Shannon-Abtasttheorem, bekannt. Es beschreibt, wie gut ein kontinuierliches Signal abgetastet werden muss, um aus dem so entstehenden zeitdiskreten Signal das Ursprungssignal so zu rekonstruieren, dass kein Informationsverlust stattfindet. Die Abtastfrequenz muss laut Theorem mehr als doppelt so groß sein wie die maximale Frequenz im Ursprungssignal.[Sha98]

$$f_{abtast} > 2 \cdot f_{max}$$

Dies lässt sich auch auf das Erfassen bzw. Abtasten eines Objektes mittels eines Sensors übertragen.

Schneidet man das Objekt entlang einer Linie von Abtastpunkten, so erhält man eine zweidimensionale Kurve dieser Oberflächenkontur. Diese Hüllkurve kann man sich aus Abschnitten unterschiedlich langer Periodendauern denken. Gemäß dem Abtasttheorem muss der Abstand zwischen zwei Abtastpunkten kleiner sein als die Hälfte der kleinsten Periodendauer. Oder anders ausgedrückt, um eine detailgetreue Rekonstruktion zu ermöglichen, muss die Auflösung bei der Erfassung mehr als doppelt so groß sein wie das Detail, welches erfasst werden soll.

Durch das Verwenden mehrer Einzelwolken bei der Erfassung wird dies nicht aufgehoben, da ein Detail, welches in den Einzelwolken nicht erfasst ist, nicht zwingend in der Gesamtwolke erscheint, auch wenn dies möglich wäre.

Abbildung 2.8 zeigt die Erfassung von Objekten. Die Punkte sind dabei die erfassten Werte. Die Lücke wird im linken Versuch nicht erfasst. Eine Erhöhung der Genauigkeit bzw. Auflösung, wie im rechten Versuch, würde die Lücke im linken Versuch erfassen. In der Praxis kann die Auflösung allerdings nicht beliebig genau gewählt werden, da sie durch die Hardware begrenzt wird oder durch Anforderungen an die Effizienz der Berechnung

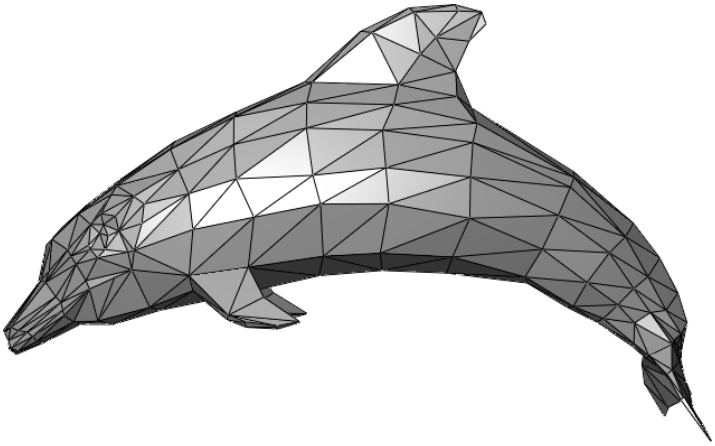

Abbildung 2.9: Polygonnetz eines Delfins.([Wikb])

eingeschränkt werden muss. Mit der dargestellten Auflösung würde bei der Rekonstruktion in beiden Fällen ein durchgehender Balken entstehen.

2.7 Polygonnetze

In der Visualisierung von 3D-Objekten wird häufig auf ein sogenanntes Polygonnetz oder Polygonmodell zurückgegriffen. Dabei handelt es sich um eine endliche Anzahl über Kanten verbundener Punkte bzw. um eine endliche Anzahl miteinander verbundener, planarer Polygone. Ein Polygonnetz approximiert dabei die Oberfläche eines Objektes. So müssen z.B. Krümmungen durch plane Polygone dargestellt werden. Je kleiner dabei die Polygone sind, desto genauer wird die Krümmung approximiert.

In Abbildung 2.9 ist ein Delfin als Polygonnetz dargestellt. Details werden hierbei durch kleinere Flächen dargestellt. Theoretisch kann jede Form von Polygonen für die Erstellung des Netzes verwendet werden. In der Praxis finden allerdings meist Dreiecke und Vierecke Anwendung.

3 Durchführung

> Probleme kann man niemals mit derselben Denkweise lösen, durch die sie entstanden sind.
>
> *(Albert Einstein)*

In diesem Kapitel wird die Durchführung der Aufgabenstellung beschrieben. Dabei wird zunächst auf die Art und Weise der Lösung eingegangen und danach die daraus erhaltenen Ergebnisse vorgestellt. Diese Ergebnisse werden im Weiteren miteinander verglichen und bewertet.

3.1 Lösungsbeschreibung

3.1.1 Punktwolken Kinect

Die Punktwolken, die in dieser Arbeit behandelt werden, stammen von dem ROS Paket „roboearth“ wurden mit der Microsoft Kinect Kamera erzeugt (siehe Abschnitt 2.3). RoboEarth liefert dabei sowohl die Einzelwolken als auch eine Gesamtwolke, welche wiederum mit Hilfe von Markierungsmustern zusammengefügt wird. Dabei werden die Punkte der Einzelwolken, die sich aus verschiedenen Perspektiven ergeben, einfach überlagert.

Der Datensatz der Punktwolke wird im „pcd“-Format gespeichert. Das Format besteht aus zwei Teilen, dem Header und den eigentlichen Nutzdaten. Der Header umfasst zehn obligatorische Felder, die in Tabelle 3.1 aufgelistet und definiert sind. Dieser Header muss zwingend in ASCII abgelegt werden, die eigentlichen Nutzdaten können in ASCII oder binärer Form gespeichert werden. In einer im PCD-Format gespeicherten Datei werden die einzelnen Parameter und Werte mit Leerzeichen separiert. Mit „#“ kann kommentiert werden.

Das Listing 3.1 zeigt eine Beispiel PCD-Datei im ASCII-Format.

3.1.2 Vorverarbeitung

Aus den Punktwolken werden als Endergebnis Polygonmodelle rekonstruiert. Die Qualität dieser Polygonmodelle kann durch eine Vorverarbeitung der ursprünglichen 3D-Daten verbessert werden. Aufgrund der Bauweise und der vorhandenen, begrenzten Auflösung der Kinect muss man davon ausgehen, dass extrem von der Umgebung abweichende Messpunkte eher auf Messfehler als auf tatsächliche Gegebenheiten zurückzuführen sind.

Feld	Inhalt	Beispiel
VERSION	Die Version der PCD Datei. Momentan aktuelle Version: Vo.7	`VERSION .7`
FIELDS	Definiert die Namen der einzelnen Dimensionen, die ein Punkt haben kann.	`FIELDS x y z rgb`
SIZE	Größe jeder Dimension in Byte.	`SIZE 4 4 4 8`
TYPE	Typ jeder Dimension (mögliche Typen: I, U, F).	`TYPE F F F F`
COUNT	Anzahl Elemente einer Dimension. Dies ermöglicht mehrdimensionale Deskriptoren.	`COUNT 1 1 1 1`
WIDTH	Breite einer Punktwolke in Anzahl Punkten. Nur relevant bei sortierten Datensätzen. Ansonsten Gesamtanzahl der Punkte.	`WIDTH 299`
HEIGHT	Höhe einer Punktwolke in Anzahl Punkten. Nur relevant bei sortierten Datensätzen. Ansonsten 1.	`HEIGHT 1`
VIEWPOINT	Definiert eine (Kamera-)Perspektive für den Datensatz. Interessant für das Übertragen in andere Datensätze.	`VIEWPOINT 0 0 0 1 0 0 0`
POINTS	Gesamtanzahl der enthaltenen Punkte.	`POINTS 299`
DATA	Die Art wie die Punkte abgelegt sind (mögliche Optionen: binary, ascii)	`DATA binary`

Tabelle 3.1: Header Felder einer PCD Datei.

Listing 3.1 Beispiel PCD-Datei im ASCII-Format und Farbwerten.

```
# .PCD v.7 - Point Cloud Data file format
VERSION .7
FIELDS x y z rgb
SIZE 4 4 4 4
TYPE F F F F
COUNT 1 1 1 1
WIDTH 7
HEIGHT 1
VIEWPOINT 0 0 0 1 0 0 0
POINTS 7
DATA ascii
0.93773 0.33763 0 4.2108e+06
0.90805 0.35641 0 4.2108e+06
0.81915 0.32 0 4.2108e+06
0.97192 0.278 0 4.2108e+06
0.944 0.29474 0 4.2108e+06
0.98111 0.24247 0 4.2108e+06
0.93655 0.26143 0 4.2108e+06
```

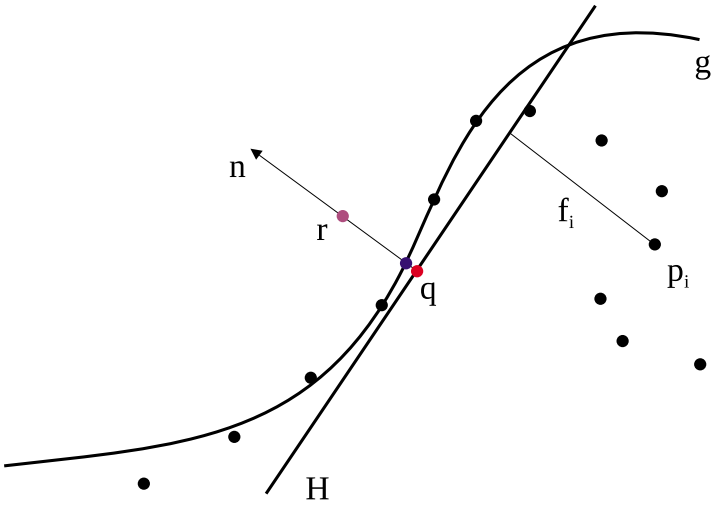

Abbildung 3.1: Projektion eines Punktes r auf die Oberfläche H.[ABCO+01]

Glättung

Die Fehler in den verrauschten Punktwolken werden mittels einer Fehlerminimierung reduziert. Die solchermaßen mit einer quadratischen Fehlermetrik bearbeiteten Wolken erscheinen visuell geglättet und dem Originalobjekt näher. Daher wird die Fehlerminimierung im Folgenden als Glättung bezeichnet.

Zur Glättung der Punktwolken wird das Moving Least Squares (MLS) Verfahren eingesetzt.

Beim MLS Verfahren wird davon ausgegangen, dass eine vorhandene Punktmenge P eine Oberfläche implizit beschreibt. Hierbei wird ein Verfahren verwendet, welches Punkte in der Umgebung der gegebenen Punktmenge auf die implizite Oberfläche der Punktmenge projiziert. Die Menge aller Punkte, die auf diese Art auf sich selbst projiziert wurden, ergibt dann die neue durch MLS geglättete implizite Oberfläche.

Seien die Punkte p_i Punkte einer eingescannten realen Oberfläche S (Siehe Abbildung 3.1). Nun ist es das Ziel, einen Punkt r so zu projizieren, dass die Oberfläche S von der Projektion von r genügend genau approximiert wird. Hierzu wird eine MLS-Projektion verwendet.[Lev01]

Dabei wird folgendermaßen vorgegangen: Zuerst wird eine lokale Referenzebene H für den (violetten) Punkt r erstellt. Die Projektion von r auf H definiert dessen Ursprung q (roter Punkt). Dann wird eine lokale polynomielle Näherungsfunktion g, in Abhängigkeit der Abstände f_i der Punkte p_i zur Ebene H berechnet. In beiden Fällen resultiert die Gewichtung der p_i aus dem Abstand zu q. Das Ergebnis der MLS-Projektion ist die Projektion von r auf die Funktion g.

Die lokale Referenzebene $H = \{x| < n, x > -D = 0, x \in \mathbb{R}^3\}, n \in \mathbb{R}^3, ||n|| = 1$ berechnet sich aus der Minimierung der gewichteten Summe der quadratischen Abstände p_i zu H:

$$\sum_{i=1}^{N}(< n, p_i > -D)^2\theta(||p_i - q||)$$

mit q als Projektion von r auf H und θ als monoton fallende Funktion die nur positive Werte liefert. Eine mögliche Funktion für θ ist nach [Levo1]:

$$\theta(d) = e^{-\frac{d^2}{h^2}}$$

wobei h als fixer Parameter bestimmt, wie stark die benachbarten Punkte eingebunden werden.

Damit lässt sich kontrollieren, wie stark die Punkte geglättet werden. Ein kleiner Wert für h lässt die Gauß-Funktion schneller abfallen und sorgt dafür, dass die Näherung lokaler wird. Ein großer Wert für h sorgt für eine globalere Näherung und für eine größere Glättung. Generell werden Eigenschaften geglättet, deren Größe kleiner ist als h.

Die Referenzebene H für den Punkt r wird dazu benutzt, eine lokale zweidimensionale polynomielle Näherungsfunktion der Oberfläche g in der Umgebung von r zu berechnen. Der Punkt q ist dabei der Ursprung des orthonormalen Koordinatensystems basierend auf H, in dem die Funktion g berechnet wird. Sei nun q_i die Projektion von p_i auf H und f_i der Abstand von p_i zu H, so ist $f_i = n \cdot (p_i - q)$. Die Koeffizienten der polynomiellen Näherungsfunktion g sind dabei so zu wählen, dass die gewichtete Summe der quadratischen Abstände minimal wird:

$$\sum_{i=1}^{N}(g(x_i, y_i) - f_i)^2\theta(||p_i - q||)$$

mit x_i, y_i als Repräsentation von q_i im lokalen Koordinatensystem basierend auf H. θ entspricht der bereits verwendeten Funktion basierend auf dem Abstand der Punkte p_i zu q.[ABCO$^+$01]

Der "Glättungsparameter“ h wird in der verwendeten Software als „Search Radius“ bezeichnet.

Downsampling

Die Kinect tastet ein Objekt gleichmäßig ab, d.h. ohne lokal feine und grobe Objektstrukturen zu berücksichtigen. Daraus folgt, dass Punktmengen generiert werden, die in ihrer Granularität überflüssig sind und somit zur Beschreibung des Objekts keinen wesentlichen Beitrag leisten, bzw. sogar stören können.

Beim Downsampling werden diese Punkte entfernt, um den Datensatz entsprechend zu verkleinern. Dazu wird über die Punktwolke ein Voxelgitter gelegt. Ein Voxel ist, in Analogie

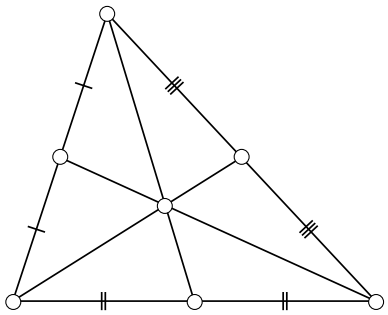

Abbildung 3.2: Berechnung des geometischen Schwerpunkts.[Wikc]

zum Pixel im zweidimensionalen Raum, ein Volumen Pixel. Das Voxel lässt sich hier als Quader betrachten, der eine bestimmte Anzahl an Punkten repräsentiert. Nun wird der geometrische Schwerpunkt bzw. der Volumenschwerpunkt aus diesen im Quader befindlichen Punkten berechnet. Dieser Schwerpunkt ersetzt dann die im Quader enthaltenen Punkte.

Den Grad des Downsampling lässt sich über die Kantenlängen des Quaders einstellen, die bestimmen, wie viele Punkte auf einen Punkt zusammenfallen. In der verwendeten Software heißt diese Kantenlänge „Leafsize“.

3.1.3 Polygonmodell

Das Ergebnis der Polygonmodellrekonstruktion wird als Visualization Toolkit (VTK) Datei gespeichert. Dieses Format verfügt ebenso wie das PCD-Format (siehe Abschnitt 3.1.1) über einen Headerbereich und einen Nutzdatenbereich. Die ersten drei Einträge im Header sind obligatorisch (siehe Tabelle 3.2). Die Nutzdaten können sowohl im ASCII-Format als auch in binärer Form hinterlegt werden. Für diese Arbeit wird hauptsächlich `DATASET POLYDATA` genutzt, um die generierten Polygonmodelle zu speichern. Dabei werden die Daten im Nutzdatenbereich gruppiert. Zunächst werden die Punkte, dann die Vertices und darauf folgend die Polygone abgelegt. Falls Farbinformationen vorhanden sind, folgen diese anschließend. Das Listing 3.2 zeigt eine beispielhaft verkürzte VTK-Datei im ASCII-Format.

Die Rekonstruktion stützt sich auf den von Marton, Rusu und Beetz entworfenen Algorithmus zur schnellen Oberflächenrekonstruktion [MRB09]. Dabei werden zunächst Randpunkte gewählt von denen aus der Algorithmus das Gitter generiert und so lange vergrößert, bis alle möglichen Punkte verbunden sind. Der Algorithmus kann mit unorganisierten Punktwolken umgehen, z.B. jenen, die die Kinect liefert. Die Triangulation wird lokal durchgeführt, d.h., die lokale Umgebung eines Punktes wird anhand der Normalen dieses Punktes erfasst und die in Frage kommenden Punkte werden mit ihm zu Dreiecken verbunden. Die Parameter für diesen Vorgang sind die maximale Anzahl an Nachbarpunkten und die maximale Entfernung zu diesen. Zusätzlich kann der maximale und minimale Winkel für die durch die Triangulation entstehenden Dreiecke als weitere Parameter gewählt werden. Die von der

Feld	Inhalt	Beispiel
Version	Die Version der VTK Datei. Momentan aktuelle Version: 3.0	`# vtk DataFile Version 3.0`
Dateiinfo	Freitext mit maximal 256 Zeichen	`vtk output`
Datenformat	Format der Nutzdaten (mögliche Optionen: binary, ascii).	`BINARY`
DATASET	Definiert die Topologie und Geometrie der Nutzdaten. Es stehen folgende Typen zur Auswahl: STRUCTURED_POINTS, STRUCTURED_GRID, UNSTRUCTURED_GRID, POLYDATA, RECTILINEAR_GRID, FIELD	`DATASET POLYDATA`

Tabelle 3.2: Header Felder einer VTK Datei.

Kinect gelieferten Punktwolken enthalten keine Normalen zu den einzelnen Punkten, diese müssen zuvor berechnet werden.

Als Alternative zu dem oben genannten Algorithmus von Marton, Rusu und Beetz wurde auch der Marching Cubes Algorithmus mit den vorhandenen 3D-Daten getestet. Bei diesem Algorithmus wird der Datensatz in ein 3D-Voxel-Gitter gelegt, da Marching Cubes ursprünglich für Voxeldatensätze entworfen wurde. Das Voxelgitter wird dann in Würfel (cubes) unterteilt. In Abbildung 3.3 sind die fünfzehn reduzierten Grundmuster zu sehen, welche ein Würfel, aufgrund von Symmetrieeffekten, annehmen kann. Die verschiedenen Muster enthalten die daraus resultierende Oberfläche für diesen Würfel. Setzt man alle Würfel zusammen bekommt man die Oberfläche des Datensatzes. [LC87]

Der Marching Cubes Algorithmus lieferte mit den verrauschten Daten der Kinect von extrem ungenügenden bis hin zu gar keinen Ergebnissen, sodass er für diese Arbeit verworfen wurde.

3.1.4 Software „Polyrec“

Parallel zu dieser Arbeit wurde die Software „Polyrec“ entwickelt. Sie umfasst die vorgestellten Vorverarbeitungsschritte und den Algorithmus zur Erstellung von Polygonmodellen aus Punktwolken. Die Software greift dabei auf die Methoden aus der Point Cloud Library zurück und ist in C++ geschrieben.

Sie bietet ein Command Line Interface zur einfachen Bedienung und liefert zusätzliche Methoden, um mit Punktwolken umzugehen, z.B. eine Konvertierung in binärer Form gespeicherter Punktwolken in das ASCII-Format.

Listing 3.2 Gekürzte Beispiel VTK-Datei im ASCII-Format.

```
# vtk DataFile Version 3.0
vtk output
ASCII
DATASET POLYDATA
POINTS 2438 float
0.088154 -0.078626 0.012975
-0.0048638 -0.063108 0.012873
-0.021747 -0.056312 0.013047
-0.018106 -0.058161 0.011913
[...]

VERTICES 2438 4876
1 0
1 1
1 2
1 3
[...]

POLYGONS 4670 18680
3 1 5 72
3 76 5 72
3 143 76 72
3 139 143 72
[...]

POINT_DATA 2438
COLOR_SCALARS scalars 3
0.058824 0.062745 0.1451
0.48235 0.58431 0.78431
0.5451 0.61569 0.86275
0.50196 0.56863 0.84314
[...]
```

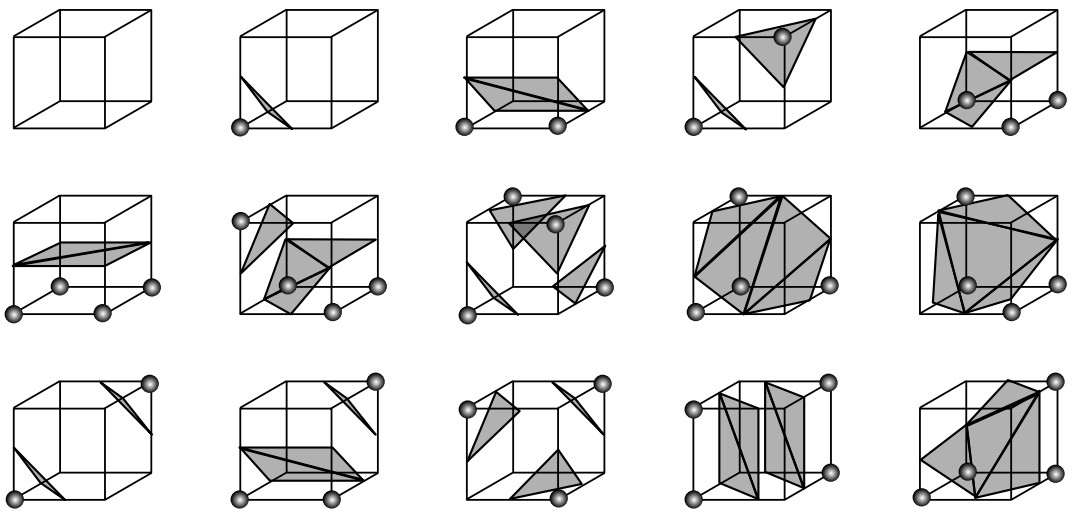

Abbildung 3.3: Die 15 Grundmuster der Würfel des Marching Cubes Algorithmus.[Fav]

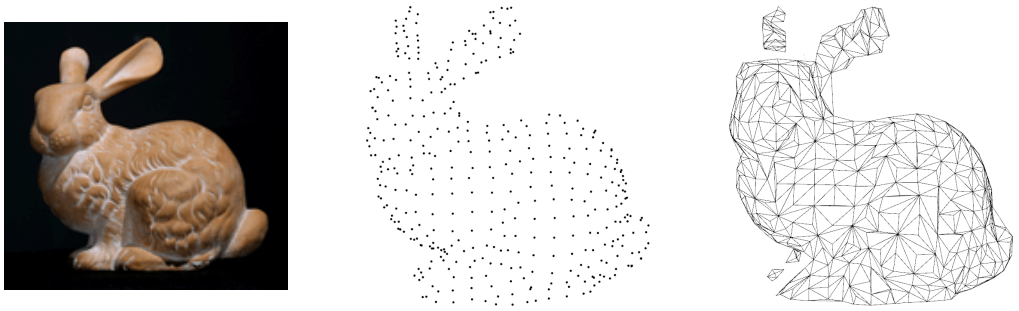

Abbildung 3.4: Rekonstruktion eines Polygons aus einer fehlerfreien Punktwolke anhand des „Stanford Bunny“.[Sta]

3.2 Messergebnisse

Damit eine aussagefähige Auswertung der rekonstruierten Polygonnetze aus Punktwolken, die mit der Kinect erzeugt wurden, gemacht werden kann, ist es wichtig zu wissen, welche Ergebnisse der verwendete Rekonstruktionsalgorithmus im Idealfall liefert. Dies ist aus der Abbildung 3.4 entnehmbar. Sie zeigt den „Stanford Bunny“ im Original im linken Teilbild. In der Mitte ist die fehlerfreie, mit einem Laserscanner erfasste Punktwolke zu sehen. Im rechten Teilbild ist das daraus rekonstruierte Polygonmodell dargestellt, das eine für die Auflösung originalgetreue Approximation ergibt. Dies bedeutet, dass der verwendete Algorithmus gültige Schlussfolgerungen aus den daraus erzeugten Messergebnissen zulässt.

Für die Erzeugung der Polygonmodelle wurden zwei Objekte ausgewählt, das Eistee Tetra-Pak und das Rentier Rudolph (siehe Abbildung 3.5). Beide stammen aus der RoboEarth-Datenbank und sind dort unter den Namen „beverage.icetea“ und „animal.rudolph“ hinterlegt. Der Grund für ihre Auswahl liegt in ihren deutlich unterschiedlichen Oberflächenstrukturen, so kann die des Eistee Tetrapak als „sehr einfach“ und jene des Rentiers Rudolph als „sehr komplex“ eingestuft werden. Man vergleicht somit die zwei Extreme in einem Spektrum möglicher Oberflächenstrukturen. Neben den Objekten muss ebenso die Parametrisierung der Vorverarbeitungen gewählt werden.

Beim Downsampling ist der Wert der Leafsize so festzulegen, dass er für beide Objekte einheitlich ist und zu optimalen Ergebnissen führt. Die durchgeführten Tests lieferten eine Leafsize von 0,002 für die Kantenlänge jeder Achse. Die Abbildungen 3.6 und 3.7 zeigen beispielhaft am komplexen Objekt Rudolph, dass ein noch stärkeres Downsampling mit Leafsize 0,005 zu schlechteren und inakzeptablen Ergebnissen führt. Das linke Teilbild in Abbildung 3.6 zeigt die unbehandelte Punktwolke mit 141440 Punkten. Die Leafsize 0,002 liefert die mittlere Punktwolke mit 22725 Punkten, womit das Original noch gut repräsentiert wird. Im rechten Bild erzeugt die Leafsize 0,005 nur noch 2885 Punkte. Damit sind wesentliche Objektinformationen verloren gegangen.

In Abbildung 3.7 kann man die Auswirkungen des zu starken Downsamplings deutlich sehen. Das resultierende Polygonmodell auf der linken Seite ist stark verrauscht, lässt aber

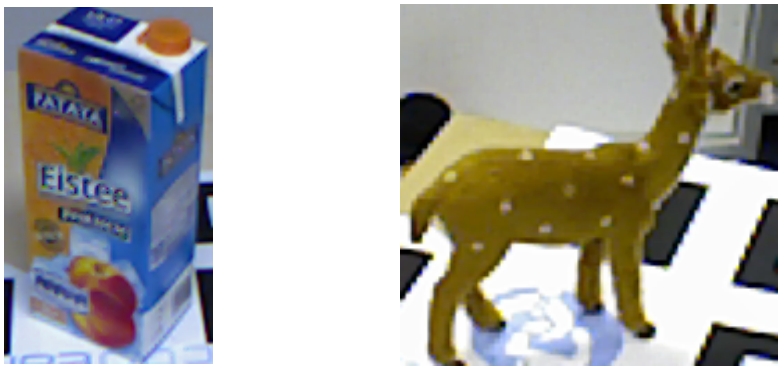

Abbildung 3.5: Für die Auswertung verwendete Objekte: Eistee Tetrapak und Rentier Rudolph.

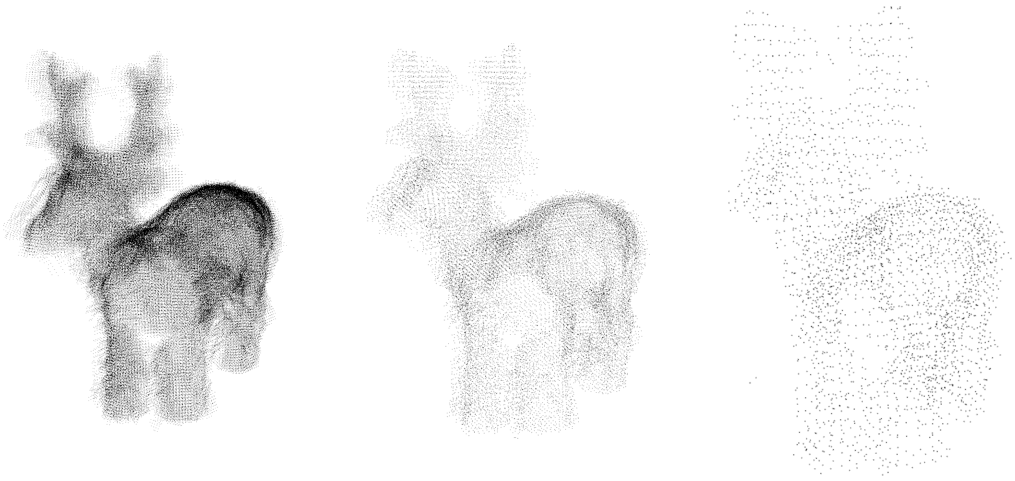

Abbildung 3.6: Downsampling am Objekt Rudolph: Original, Leafsize 0.002, Leafsize 0.005.

schon verlorene Details, vor allem im Kopfbereich erkennen. Eine zusätzliche Glättung dieses Datensatzes, wie im rechten Bild zu sehen, führt dazu, dass das Rentier insgesamt nicht mehr als solches identifiziert werden kann.

Abbildung 3.8 zeigt, dass der gefundene optimale Wert der Leafsize von 0,002 auch für das Objekt Eistee geeignet ist. Im Bild links ist die unbehandelte Punktwolke und rechts jene mit Downsampling dargestellt.

Bei der Glättung der Punktwolke mit dem MLS Verfahren gilt es, den optimalen Wert für den Glättungsparameter zu finden. Dieser experimentell ermittelte Wert ist h = 0,03. In Abbildung 3.9 ist die Glättung mit diesem Wert für beide Objekte dargestellt. Rechts die geglätteten Punktwolken im Vergleich zu den Originalwolken auf der linken Seite. In der oberen Hälfte für Eistee und darunter für Rudolph. Die Glättung liefert klare und feine Konturen, dies ist im Besonderen am Hinterteil von Rudolph zu sehen.

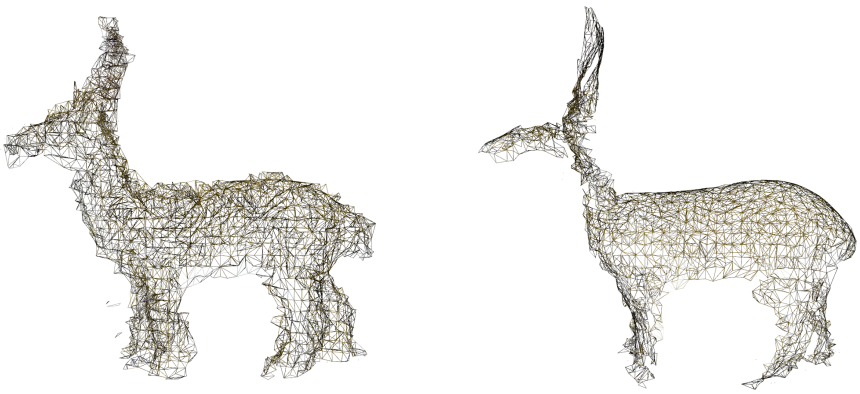

Abbildung 3.7: Vergleich Rudolph: Links Polygonmodell aus Downsampling Leafsize 0.005 und rechts zusätzlich mit MLS geglättet.

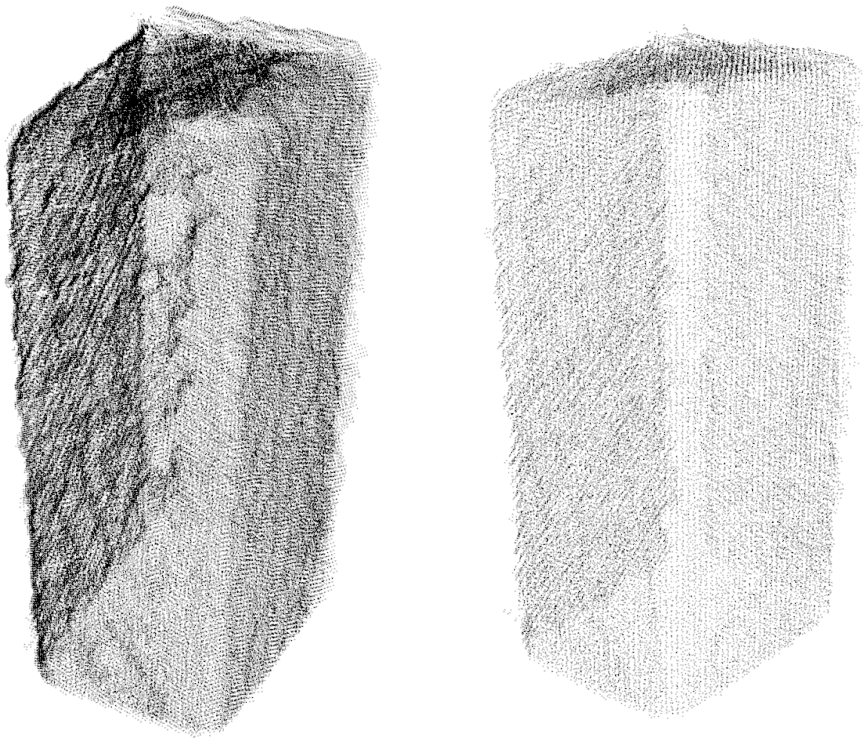

Abbildung 3.8: Vergleich Icetea: Links Originalpunktwolke, rechts mit Leafsize 0.002.

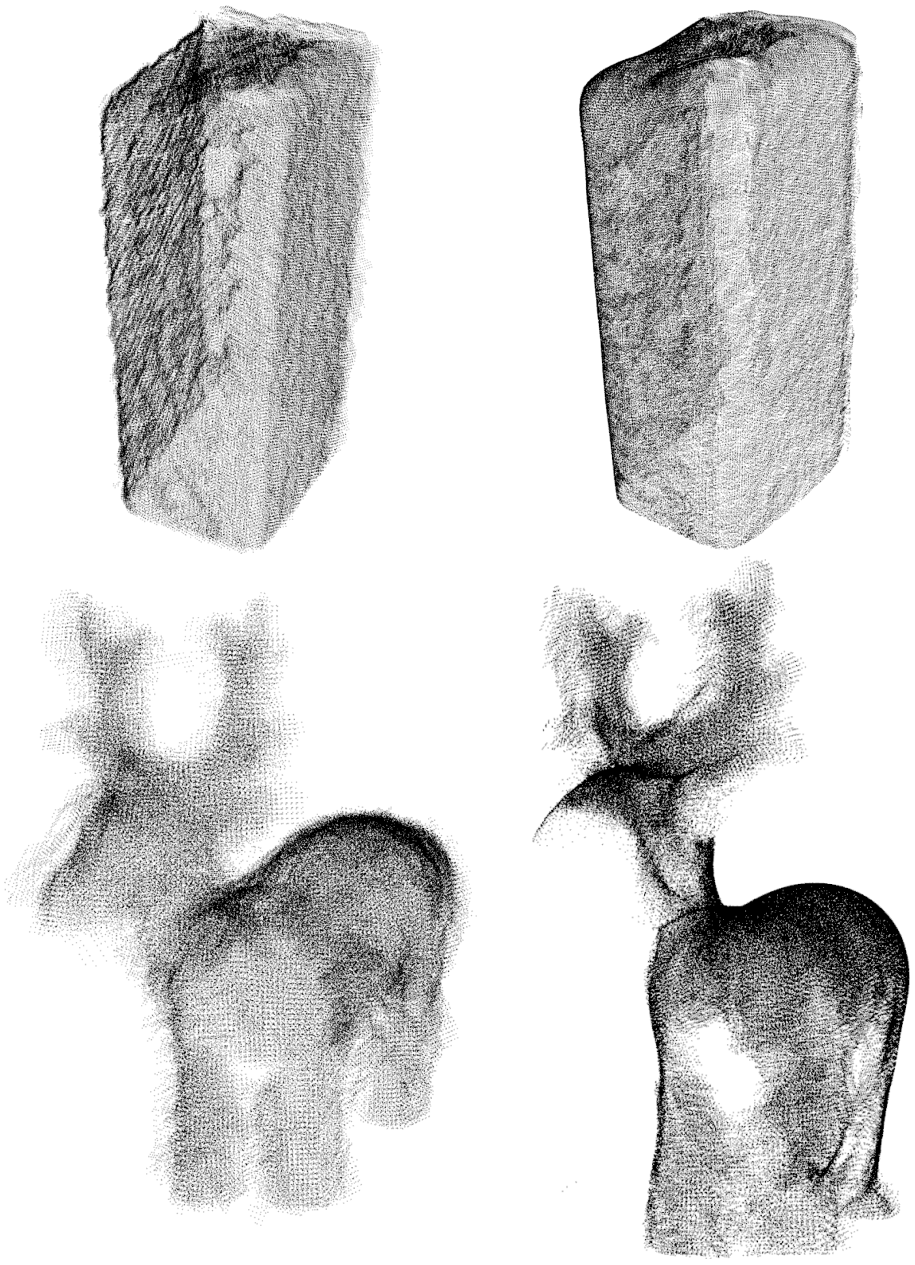

Abbildung 3.9: Vergleich: Links Originalwolken und rechts Glättung mit h = 0.03.

3.3 Vergleich und Auswertung

In diesem Abschnitt werden die unterschiedlich erzeugten Polygonmodelle vorgestellt und ausgewertet. Um die zahlreichen Ergebnisse sinnvoll und übersichtlich vergleichen und auswerten zu können, wurden sie in sogenannte Horizontal- und Vertikalvergleiche unterteilt und in zwei entsprechenden Tabellen aufgelistet.

Die fünf verschiedenen Horizontalvergleiche sind in Tabelle 3.3 enthalten. Sie ergeben sich aus sinnvollen Kombinationen der beiden Vorverarbeitungen auf die Punktwolke. Die Vergleiche sind in den zugeordneten Abbildungen 3.10 - 3.14 dokumentiert. Jede Abbildung ist gleichartig strukturiert, d.h., die beiden linken Teilbilder zeigen die Punktwolken der Objekte Eistee und Rudolph, die beiden rechten Teilbilder die zugehörigen Polygonmodellrekonstruktionen.

Neben diesen Horizontalvergleichen wurden noch sogenannte Vertikalvergleiche angestellt, in dem die Polygonmodellrekonstruktionen aus der rechten Spalte der Tabelle 3.3 „vertikal“ miteinander verglichen werden. Diese vier Vertikalvergleiche werden in Tabelle 3.4 aufgelistet und in den entsprechenden Abbildungen 3.15 - 3.18 dokumentiert.

Horizontal-Vergleich	Objekt	Abbildung	ohne Vorverarbeitung	mit Downsampling	mit Glättung	Polygonmodellrekonstruktion	Bemerkung
H1	Eistee Rudolph	3.10	o, l u, l			o, r u, r	ohne Vorverarbeitung
H2	Eistee Rudolph	3.11		o, l u, l		o, r u, r	nur mit Downsampling
H3	Eistee Rudolph	3.12			o, l u, l	o, r u, r	nur Glättung
H4	Eistee Rudolph	3.13		o, l u, l	o, l u, l	o, r u, r	zuerst mit Downsampling dann Glättung
H5	Eistee Rudolph	3.14		o, l u, l	o, l u, l	o, r u, r	zuerst Glättung dann Downsampling

Tabelle 3.3: Horizontalvergleich der Ergebnisse. „o, u, l, r“ steht für oben, unten, links, rechts und beschreibt die Position des Teilbildes in der Abbildung.

Vertikal-Vergleich	Objekt	Abbildung	Vorverarbeitung je Teilbild		Bemerkung
V1	Eistee	3.15	o, l: DS u, l: DS/MLS	o, r: MLS u, r: MLS/DS	einfaches Objekt
V2	Rudolph	3.16	o, l: DS u, l: DS/MLS	o, r: MLS u, r: MLS/DS	komplexes Objekt
V3	Eistee Rudolph	3.17	o, l: DS/MLS u, l: DS/MLS	o, r: MLS/DS u, r: MLS/DS	nähere Betrachtung der Reihenfolge
V4	Eistee	3.18	l: DS/MLS	r: MLS/DS	Betrachtung bei stark reduzierter Punktanzahl

Tabelle 3.4: Vertikalvergleich der Ergebnisse. „o, u, l, r“ steht für oben, unten, links, rechts und beschreibt die Position des Teilbildes in der Abbildung. „DS“ steht für Downsampling und „MLS“ für die Glättung. Die Reihenfolge der letzten beiden Abkürzungen beschreibt die Reihenfolge der Durchführung.

Auswertung Abbildung 3.10: In dieser Abbildung werden Polygonmodelle mit ihren unbehandelten, ursprünglichen, von der Kinect gelieferten, Punktwolken verglichen. Dabei zeigt sich, dass das Endergebnis nicht mit dem Original übereinstimmt. Die in der unbehandelten Punktwolke enthaltenen Fehler (Messfehler/Rauschen) bewirken ein ungenügendes Ergebnis für beide Objekte.

Auswertung Abbildung 3.11: In dieser Abbildung wird die ursprüngliche Punktwolke einem Downsampling unterzogen. Der Vergleich mit dem daraus resultierenden Polygonmodell zeigt, dass das Downsampling allein nicht ausreicht und die zuvor enthalten Fehler immer noch vorhanden sind. Das Ergebnis ist auch hier für beide Objekte ungenügend.

Auswertung Abbildung 3.12: In dieser Abbildung wird die ursprüngliche Punktwolke einer Glättung unterzogen. Der Vergleich mit dem daraus resultierenden Polygonmodell zeigt ein brauchbares Modell mit einer guten Oberflächenstruktur. Allerdings besteht das Modell für die wiederzugebende Oberfläche immer noch aus zu vielen Teilflächen. Vor allem beim Eistee wird dies deutlich, da dieser eine einfache Grundstruktur besitzt.

Die Glättung der Punktwolke sorgt jedoch für klarere Umrisse, vor allem in den einfachen Bereichen, wie z.B. der Rückenpartie von Rudolph.

Auswertung Abbildung 3.13: In dieser Abbildung werden die ursprünglichen Punktwolken zunächst einem Downsampling und anschließend einer Glättung unterzogen. Dabei erhält man ein akzeptables Ergebnis für beide Objekte. Die sich ergebenden Oberflächen liegen nahe am Original und die dafür verwendete Anzahl an Flächen ist akzeptabel.

Auswertung Abbildung 3.14: In dieser Abbildung werden die ursprünglichen Punktwolken zunächst einer Glättung und anschließend einem Downsampling unterzogen. Der Unterschied zum vorherigen Vergleich ist nur die Reihenfolge der Vorverarbeitungsschritte. Für sich allein betrachtet liefert diese umgekehrte Reihenfolge das gleiche Ergebnis wie der vorherige Vergleich, nämlich eine gute Polygonmodellrekonstruktion der Oberfläche mit einer akzeptablen Anzahl an Flächen.

Auswertung Abbildung 3.15: In dieser Abbildung werden die aus den verschiedenen Kombinationsmöglichkeiten der Vorverarbeitungsschritte resultierenden Polygonmodelle des „einfachen“ Objektes Eistee miteinander verglichen („Vertikalvergleich“). Die beiden oberen Polygonmodelle entstanden mit nur einem einzigen Vorverarbeitungsschritt und weisen unakzeptable Mängel auf. Bei den beiden unteren Modellen wurden sowohl ein Downsampling als auch eine Glättung durchgeführt, mit jeweils akzeptablen Ergebnissen.

Auswertung Abbildung 3.16: In dieser Abbildung werden die aus den verschiedenen Kombinationsmöglichkeiten der Vorverarbeitungsschritte resultierenden Polygonmodelle des „komplexen“ Objektes Rudolph miteinander „vertikal“ verglichen. Die beiden oberen Polygonmodelle entstanden mit nur einem einzigen Vorverarbeitungsschritt und weisen wie beim vorherigen Vergleich unakzeptable Mängel auf. Bei den beiden unteren Modellen wurden sowohl ein Downsampling als auch eine Glättung durchgeführt, mit jeweils akzeptablen Ergebnissen.

Auswertung Abbildung 3.17: In dieser Abbildung werden nun die beiden Polygonmodelle miteinander verglichen, die mit beiden Vorverarbeitungsschritten erzeugt wurden. Sie unterscheiden sich in der Reihenfolge dieser Vorverarbeitungsschritte. Es zeigt sich, dass die zuerst durchgeführte Glättung mit anschließender Reduzierung der Punkte zu einem besseren Ergebnis führt (rechts). Bei den Polygonmodellen, bei denen zuerst ein Downsampling durchgeführt wurde, werden Fehler dadurch stärker in der darauf folgenden Glättung gewichtet. Dies führt zu einer ungewollt unregelmäßigeren Oberflächenstruktur.

Auswertung Abbildung 3.18: In dieser Abbildung ist ein deutlich höheres Downsampling des Eistee zu sehen. Aufgrund seiner einfacheren Struktur ist dies möglich, ohne zu viel Information zu verlieren. Beide Polygonmodelle wurden auf etwa die gleiche Anzahl Punkte reduziert (ca. 2400) um eine noch bessere Vergleichsbasis zu bekommen. Hier ist besonders deutlich zu sehen, dass die Reihenfolge der Vorverarbeitungsschritte einen Einfluss auf das Endergebnis hat und dass die zuerst geglättete Punktwolke zum besten Ergebnis führt.

Auswertung Abbildung 3.19: In dieser Abbildung ist ein weiterer positiver Effekt der Glättung durch MLS dargestellt. Eine nicht korrekt zusammengesetzte ursprüngliche Punktwolke, links zu sehen, enthält zwei überlappende Oberflächen. Eine Glättung, rechtes Bild, behebt diesen Kompositionsfehler. Ein Downsampling bewirkt, wie in der Mitte dargestellt dahingegen nichts.

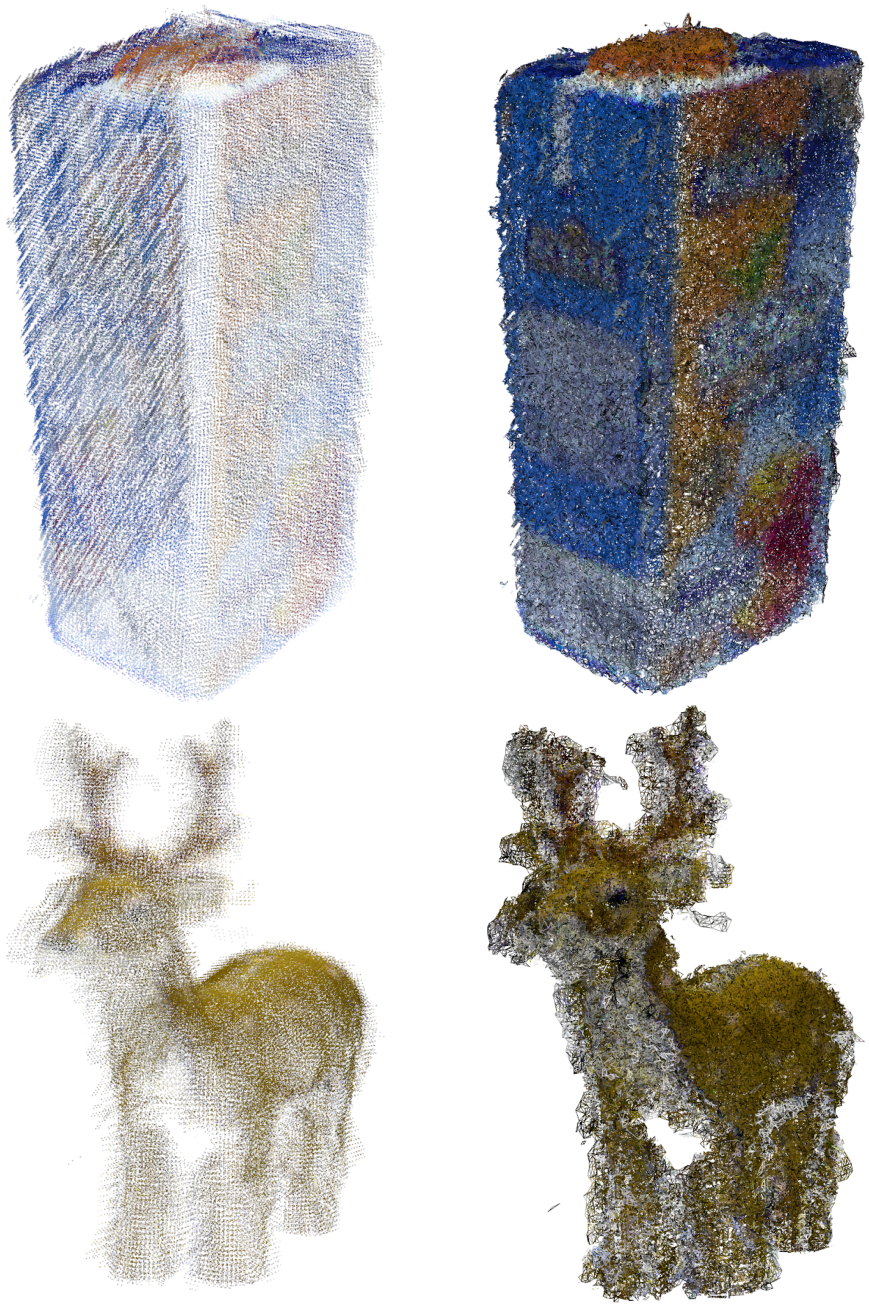

Abbildung 3.10: Direkte Polygonmodellerstellung ohne Vorverarbeitung der Punktwolken.

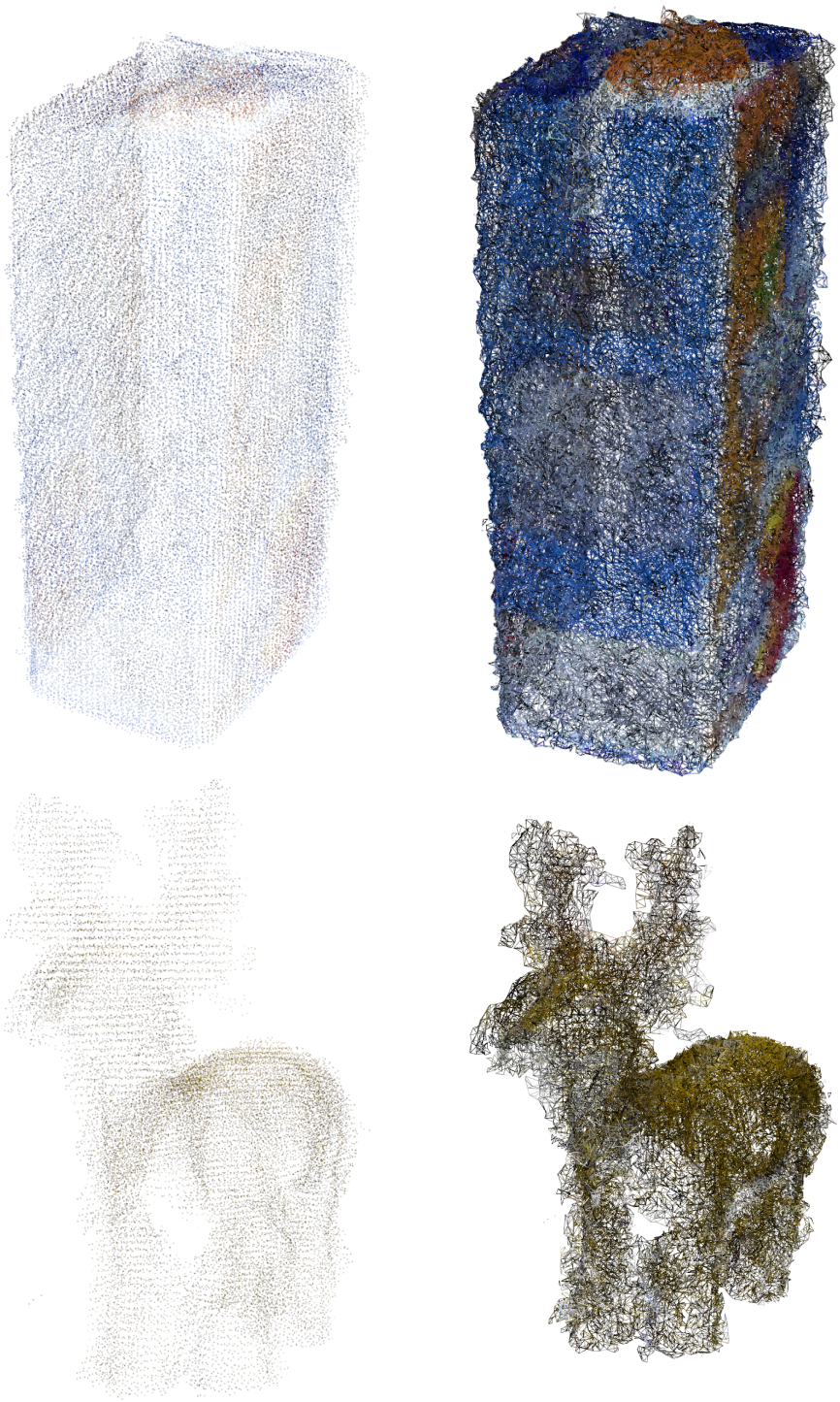

Abbildung 3.11: Polygonmodellerstellung mit Downsampling der Punktwolke.

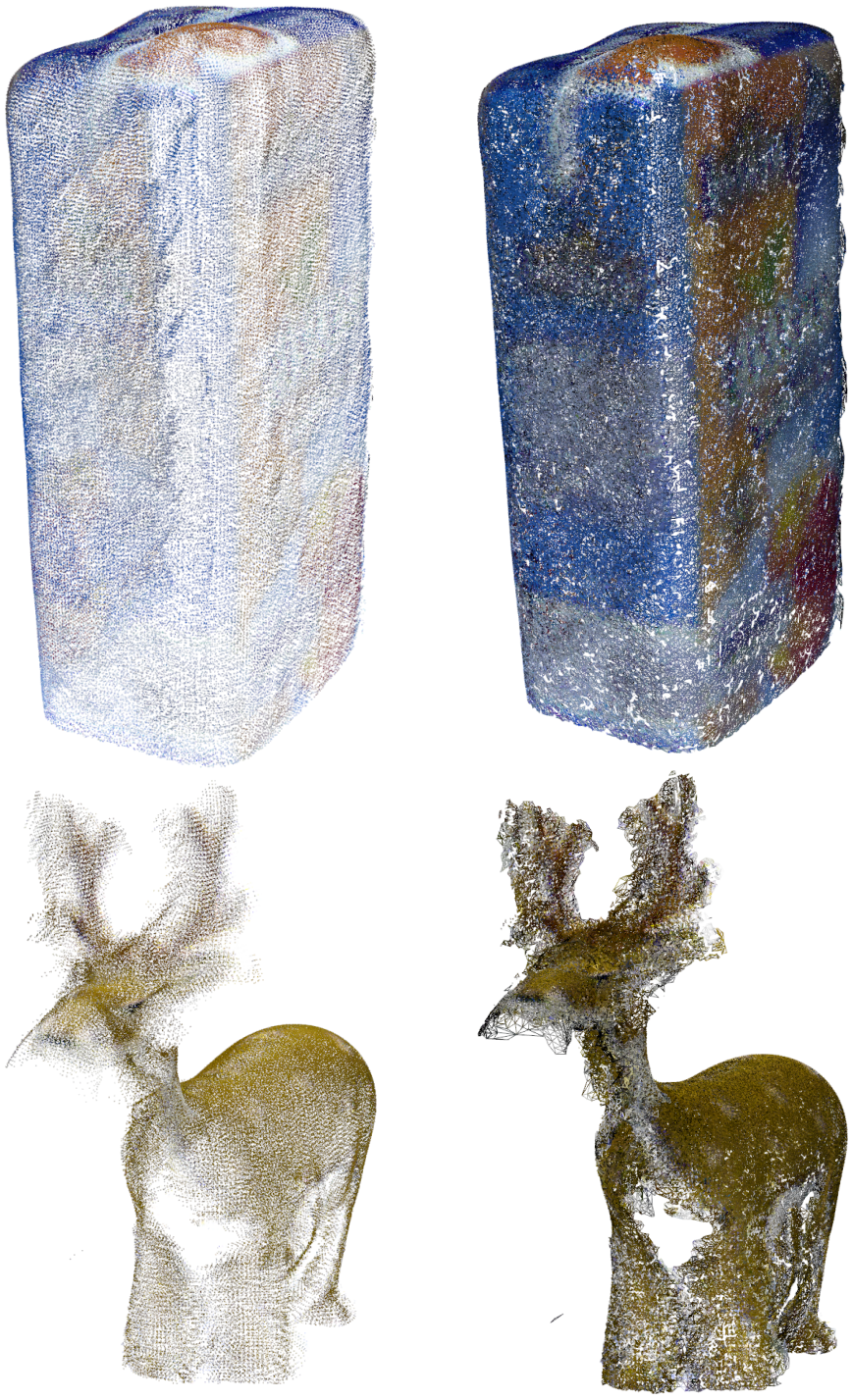

Abbildung 3.12: Polygonmodellerstellung mit Glättung der Punktwolke.

Abbildung 3.13: Polygonmodellerstellung mit Downsampling und anschließender Glättung der Punktwolke.

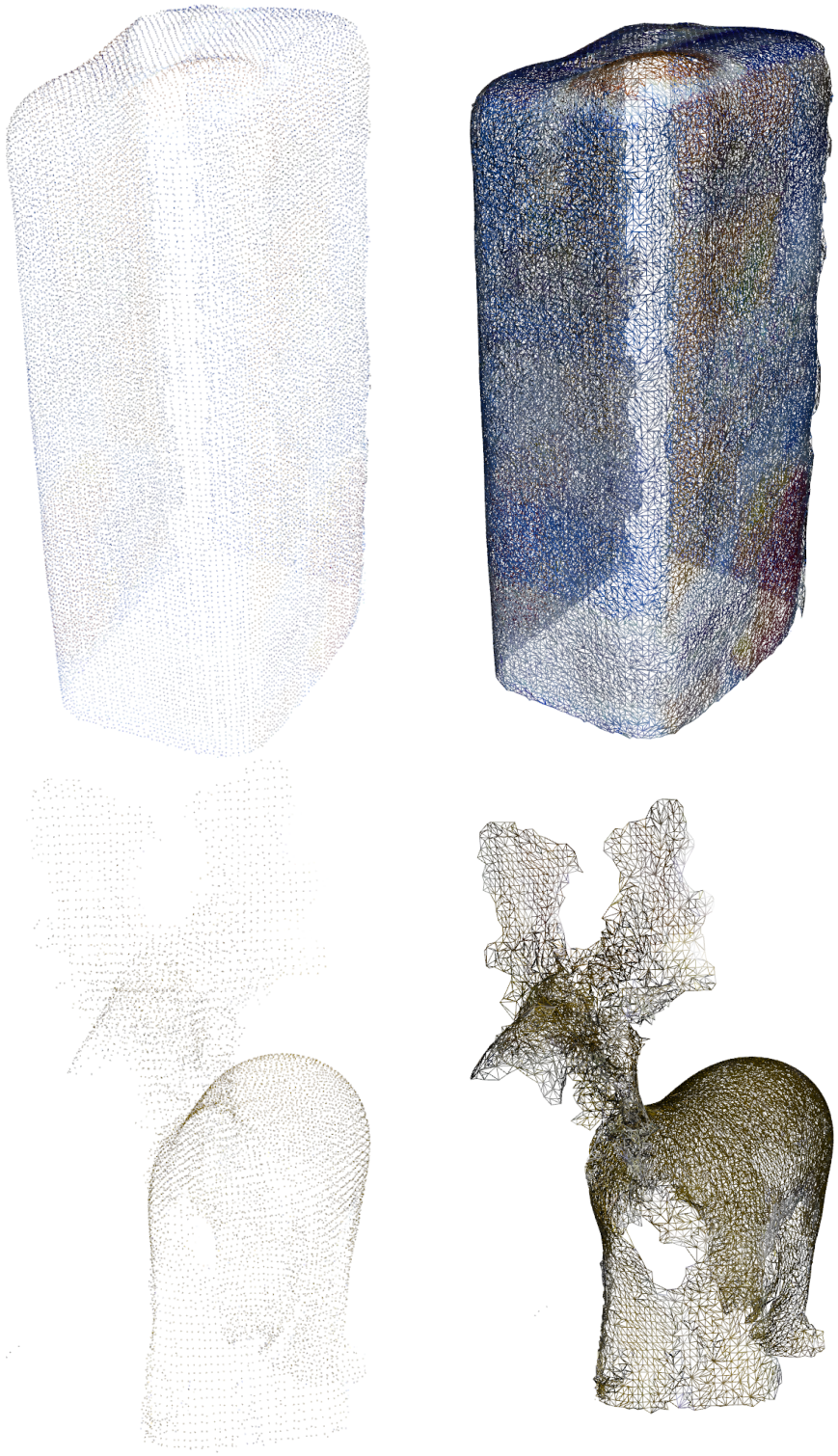

Abbildung 3.14: Polygonmodellerstellung mit Glättung und anschließendem Downsampling der Punktwolke.

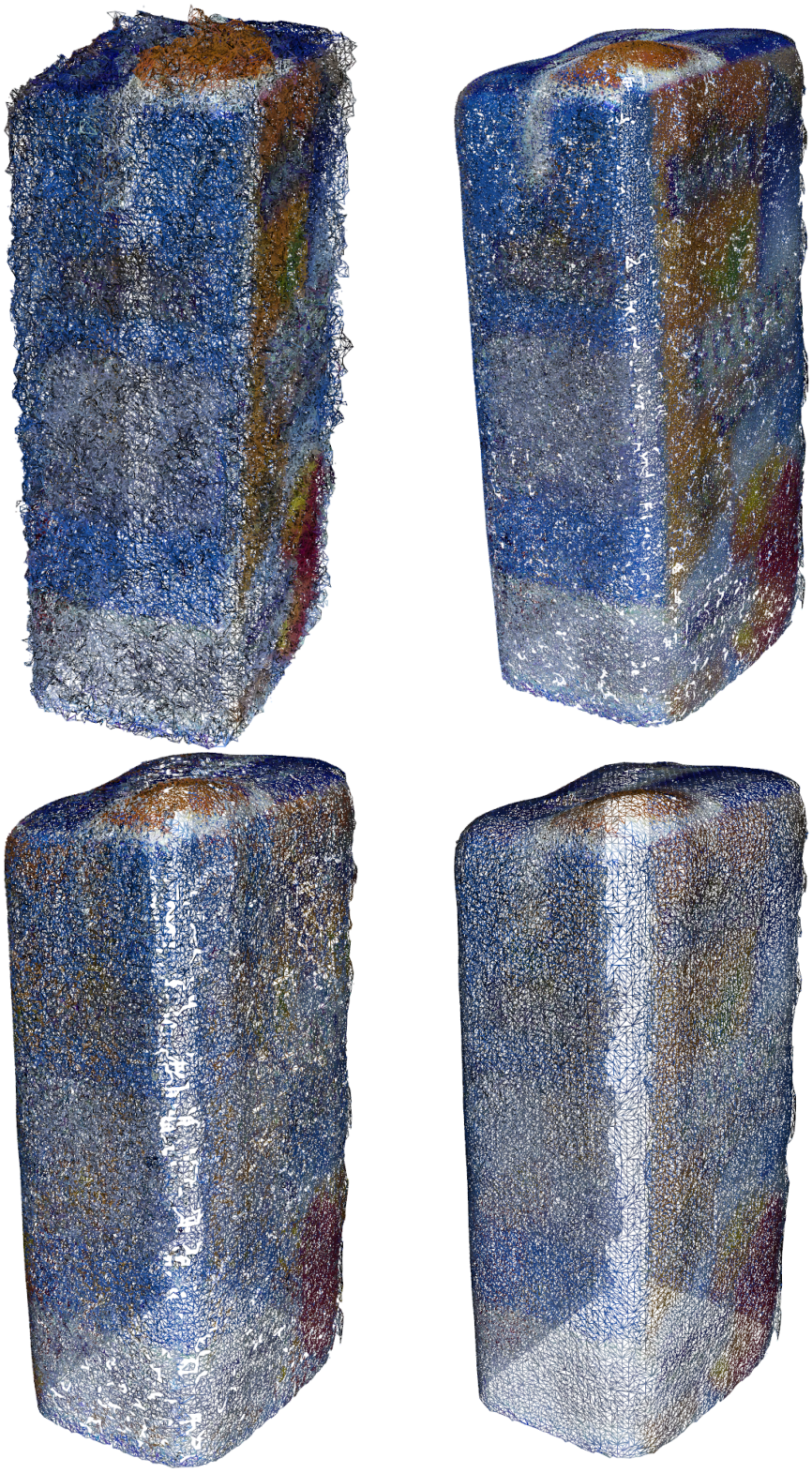

Abbildung 3.15: Eistee Polygonmodelle mit unterschiedlichen Vorverarbeitungen.

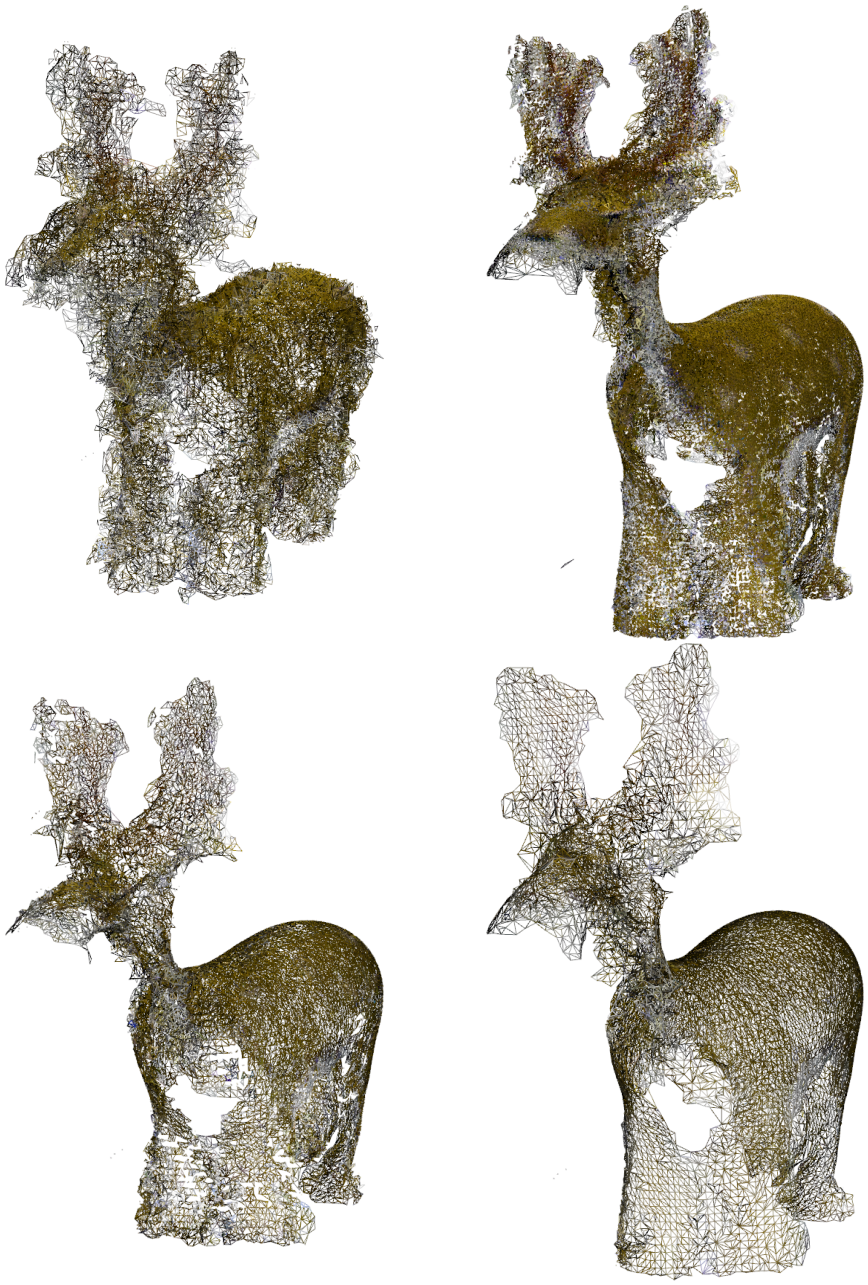

Abbildung 3.16: Rudolph Polygonmodelle mit unterschiedlichen Vorverarbeitungen.

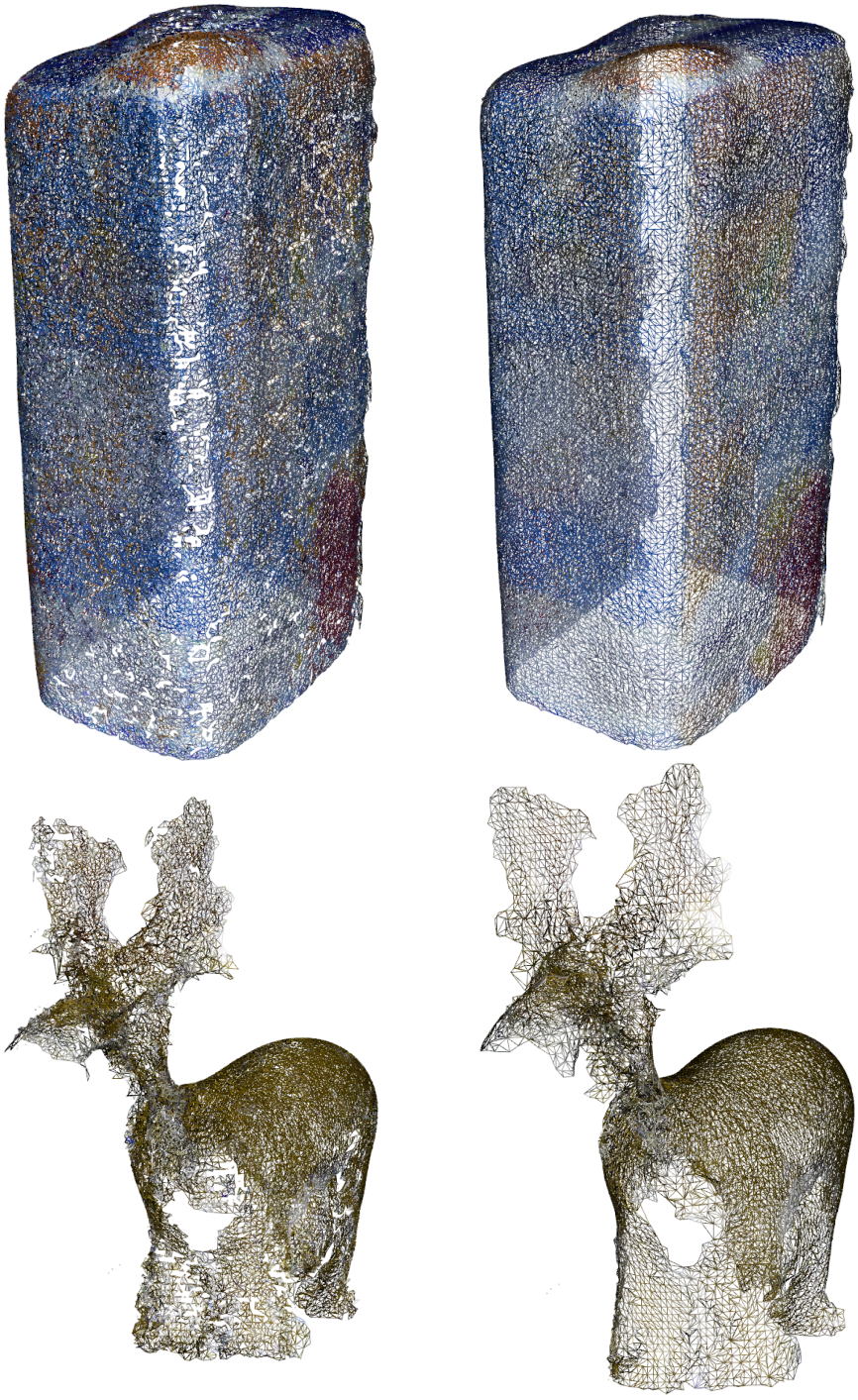

Abbildung 3.17: Polygonmodelle mit Glättung und Downsampling in verschiedener Reihenfolge.

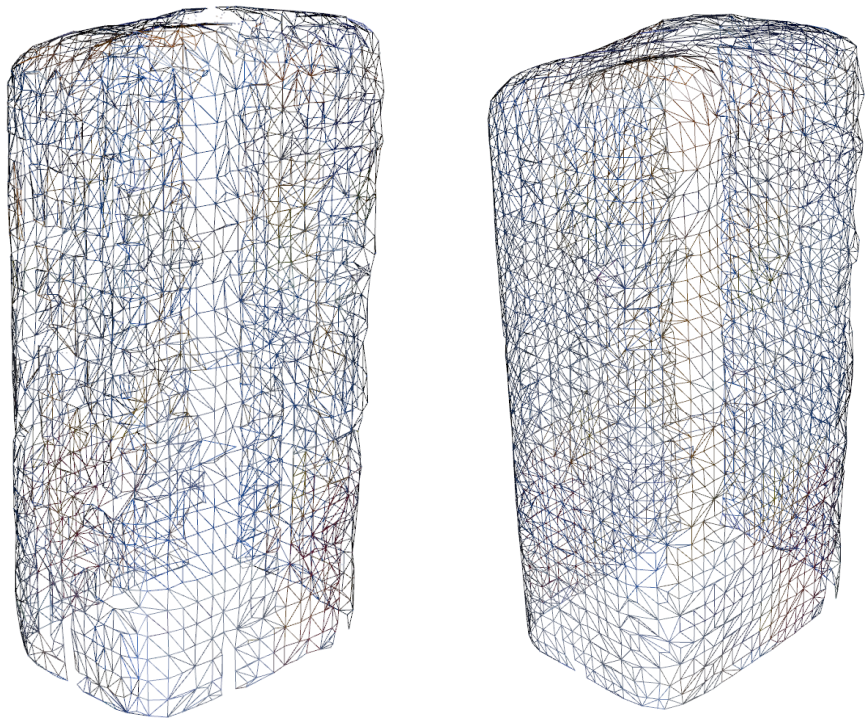

Abbildung 3.18: Eistee Polygonmodelle bei gleicher Punktzahl, mit Glättung und Downsampling in verschiedener Reihenfolge.

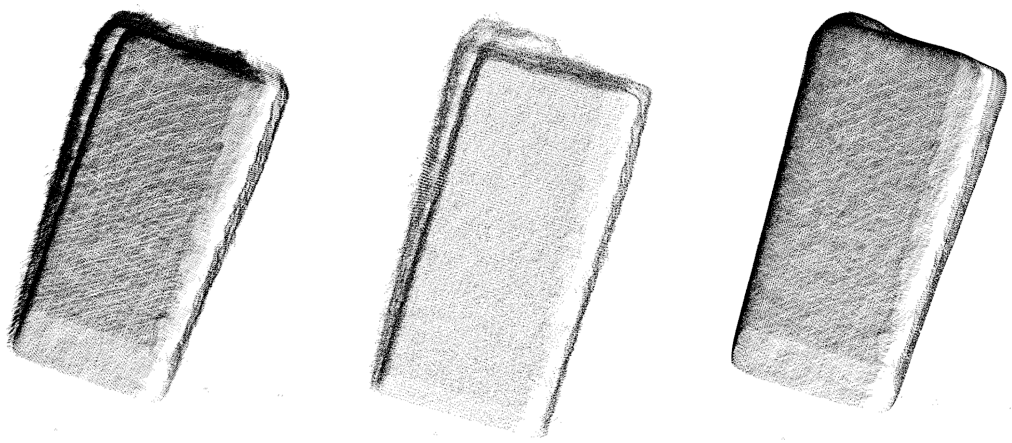

Abbildung 3.19: Behebung eines Fehlers mit MLS.

3.4 Bewertung

Fasst man die Auswertungen aller Vergleiche zusammen, so können die folgenden Bewertungen formuliert werden. Diese beziehen sich auf Punktwolken, die mit der Microsoft Kinect erzeugt wurden unter Einbeziehung der daraus resultierenden Randbedingungen.

Die Kinect liefert besonders verrauschte Punktwolken, die eine direkte Generierung von Polygonmodellen unmöglich macht. Deshalb sind immer vorbearbeitende Schritte notwendig, wobei festgestellt wurde, dass ein Downsampling oder eine Glättung für sich allein nicht ausreicht. Es sollten immer beide angewendet werden, am besten in der Reihenfolge, dass zuerst geglättet und dann reduziert wird. Dabei wird das Rauschen zunächst reduziert und fließt nicht in das Downsampling ein. Umgekehrt sorgt ein vorangehendes Downsampling dafür, dass Rauschen stärker in das Glätten einfließt und damit die Glättung keine so guten Ergebnisse liefern kann. Grund: Das Rauschen bekommt eine ähnliche Gewichtung wie vorhandene Details. Erhöhung des Suchradiuses ist nicht möglich, da hierdurch auch Details verschwinden.

Dem entgegen steht allerdings, dass eine Glättung wesentlich höhere Laufzeiten bei höheren Punktanzahlen hat. So benötigte das Gesamtverfahren bei zuerst angewandtem Glätten und anschließendem Downsampling ca. 15 Minuten, während ein zuerst durchgeführtes Downsampling mit anschließendem Glätten nur etwa 20 Sekunden benötigte. Die Zeiten wurden auf einem Intel Core 2 Duo mit 3.0GHz erfasst, ohne den zweiten Kern für die Berechnung zu benutzen.

Die mit dem MLS-Verfahren geglätteten Punktwolken führen zu visuell verbesserten Zwischenergebnissen. Es wurde zudem untersucht, ob die MLS Glättung Unterschiede bringt, wenn man sie auf die jeweiligen Punktwolken der perspektivischen Einzelaufnahmen oder nur auf die daraus zusammengesetzte Gesamtwolke anwendet. Es hat sich gezeigt, dass die Anwendung der Metrik auf die Einzelaufnahmen keine Verbesserung im Endergebnis im Vergleich zu der Anwendung lediglich auf die Gesamtwolke brachte. Die Metrik ebnet im Wesentlichen die gleichen Extremwerte ein.

In Fällen, in denen die einzelnen Punktwolken nicht hundertprozentig richtig zur Gesamtwolke zusammengesetzt werden und dadurch ein leichter Versatz bestehen bleibt, kann durch Anwendung des MLS-Verfahrens auf die Einzelwolken nicht behoben werden. Im Gegensatz dazu lieferte das Verfahren bei der Gesamtwolke ein akzeptables, brauchbares Ergebnis.

4 Zusammenfassung und Ausblick

In dieser Arbeit wurde eine Prozesskette zur Rekonstruktion von Polygonmodellen aus Punktwolken entwickelt. Sie wurde in der dazu entwickelten Software „Polyrec“ realisiert. Die Software bietet zum einen die Möglichkeit die Prozesskette im Gesamten durchzuführen, zum anderen erlaubt sie die Einzelschritte mit verschiedenen Parametern auf Punktwolken anzuwenden. Sie ist in C++ geschrieben und verwendet Algorithmen, die in der Point Cloud Library enthalten sind.

Die in der Arbeit verwendeten Punktwolken stammen von der Microsoft Kinect Kamera, einer für den Consumer Markt entwickelten 3D-Kamera. Dies macht die Kamera für den Einsatz in der Robotik interessant, da sie kostengünstig zu erhalten ist. Als Folge davon ergibt sich allerdings der Nachteil, dass die Präzision nicht sehr hoch ist. Tatsächlich sind die Punktwolken, die von der Kinect stammen, stark verrauscht. Da sich die Erstellung der Punktwolken nicht manipulieren lässt, ist eine gründliche Vorverarbeitung notwendig, um brauchbare Ergebnisse in Form von guten Polygonmodellen zu erhalten. Die in dieser Arbeit vorgestellte Prozesskette ermöglicht solche Modelle und macht damit die Kinect als Lieferanten für Polygonmodelle von Objekten weiter interessant.

Der verwendete Algorithmus von Marton, Rusu und Beetz zur Polygonmodellrekonstruktion liefert schnell und effizient brauchbare Ergebnisse direkt aus den Punktwolken. Im Gegensatz dazu kommt der Marching Cubes Algorithmus, als ein Standardalgorithmus zur Oberflächenrekonstruktion, nicht oder nur sehr ungenügend mit den Punktwolken der Kinect zurecht, die zunächst durch ein Voxelgitter approximiert werden müssen. Er wurde daher verworfen.

Es wurde gezeigt, dass durch Downsampling und Fehlerminimierung mit dem Moving Least Squares Verfahren, welches sich visuell als Glättung darstellt, brauchbare Polygonmodelle erzeugt werden können. Dabei wurde ebenfalls gezeigt, dass eine zuerst durchgeführte Fehlerminimierung mit anschließendem Downsampling ein besseres Ergebnis liefert, als in umgekehrter Reihenfolge. Allerdings liegt in diesem Falle die Laufzeit wesentlich höher, da das Moving Least Squares Verfahren deutlich sensibler auf die Menge der Eingabedaten reagiert als das robustere Downsampling. Das Moving Least Squares Verfahren eliminiert in beiden Fällen zudem einige Fehler, die bei einer fehlerhaften Komposition der Einzelwolken zu einer 360-Grad Gesamtwolke entstehen können.

Ausblick

Die in dieser Arbeit gewonnen Ergebnisse können als Ausgangspunkt verwendet werden, um weitere Schritte der Prozesskette Objekterkennung anzuwenden. So ist es denkbar, die entstandenen Polygonmodelle einer Eigenschaftsanalyse zu unterziehen, z.B. bei der Bestimmung von Greifpunkten. Ebenso lassen sich die Ergebnisse weiter optimieren. Es ist anzunehmen, dass die Anwendung der Vorverarbeitungsschritte Downsampling und Glättung auf die Punktwolken ein besseres Ergebnis liefern, wenn die Stärke des Downsamplings bzw. der Grad der Glättung lokal angepasst werden kann. Das heißt, in Regionen geringer Komplexität ein hohes Downsampling und eine starke Glättung, in Bereichen hoher Komplexität ein niedriges Downsampling und eine schwache Glättung um Details nicht zu zerstören. Eine Optimierung der Laufzeit der Glättung, die den Löwenanteil der Laufzeit des Verfahrens beansprucht, führt eventuell zu einer deutlichen Steigerung der Performance.

Literaturverzeichnis

[ABCO+01] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, C. Silva. Point set surfaces. In *Visualization, 2001. VIS '01. Proceedings*, S. 21 –29, 537. 2001. doi: 10.1109/VISUAL.2001.964489. (Zitiert auf den Seiten 6, 25 und 26)

[Fav] J.-M. Favreau. URL http://de.wikipedia.org/wiki/Marching_Cubes. (Zitiert auf den Seiten 6 und 29)

[FSMA08] B. Freedman, A. Shpunt, M. Machline, Y. Arieli. Depth mapping using projected patterns. Patent application, 2008. WO 2008/120217 A2. (Zitiert auf den Seiten 6, 13 und 14)

[Hem09] S. Hempel. *Geometrierekonstruktion von LIDAR-Punktwolken zur Echtzeitverarbeitung für autonome Fahrzeuge*. Diplomarbeit, Freie Universität Berlin, Institut für Informatik, 2009. (Zitiert auf den Seiten 6 und 21)

[IKH+11] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, A. Fitzgibbon. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, S. 559–568. ACM, New York, NY, USA, 2011. doi:10.1145/2047196.2047270. URL http://doi.acm.org/10.1145/2047196.2047270. (Zitiert auf den Seiten 6 und 11)

[KB00] L. Kobbelt, M. Botsch. An Interactive Approach to Point Cloud Triangulation. *EUROGRAPHICS*, 19(3), 2000. (Zitiert auf Seite 20)

[LC87] W. E. Lorensen, H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, 1987. doi: 10.1145/37402.37422. URL http://doi.acm.org/10.1145/37402.37422. (Zitiert auf Seite 28)

[Lev01] D. Levin. Mesh-independent surface interpolation. *Geometric Modeling for Scientific Visualization*, 2001. (Zitiert auf den Seiten 25 und 26)

[MRB09] Z. C. Marton, R. B. Rusu, M. Beetz. On fast surface reconstruction methods for large and noisy point clouds. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, S. 3218 –3223. 2009. doi:10.1109/ROBOT.2009.5152628. (Zitiert auf Seite 27)

[PCL] Point Cloud Library Logo. URL http://www.pointclouds.org/downloads/. (Zitiert auf den Seiten 6 und 18)

[PMG04] M. Pauly, N. Mitra, L. Guibas. Uncertainty and Variability in Point Cloud Surface Data. *Eurographics Symposium on Point-Based Graphics*, 2004. (Zitiert auf Seite 20)

[Roba] RoboEarth Markierungsmuster. URL http://www.ros.org/wiki/re_object_recorder. (Zitiert auf den Seiten 6 und 18)

[Robb] RoboEarth. URL http://www.roboearth.org/what-is-roboearth. (Zitiert auf den Seiten 6 und 16)

[Ros] ROS RoboEarth. URL http://www.ros.org/wiki/roboearth. (Zitiert auf den Seiten 6 und 17)

[Sha98] C. Shannon. Communication In The Presence Of Noise. *Proceedings of the IEEE*, 86(2):447 –457, 1998. doi:10.1109/JPROC.1998.659497. (Zitiert auf Seite 21)

[Sta] Stanford Bunny. URL http://graphics.stanford.edu/data/3Dscanrep/. (Zitiert auf den Seiten 6 und 30)

[WBC+11] M. Waibel, M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J. Montiel, A. Perzylo, B. Schiessle, M. Tenorth, O. Zweigle, R. van de Molengraft. RoboEarth. *Robotics Automation Magazine, IEEE*, 18(2):69 –82, 2011. doi:10.1109/MRA.2011.941632. (Zitiert auf Seite 16)

[Wika] Wikipedia. URL http://de.wikipedia.org/wiki/Kinect. (Zitiert auf den Seiten 6 und 14)

[Wikb] Wikipedia. URL http://en.wikipedia.org/wiki/Polygon_mesh. (Zitiert auf den Seiten 6 und 22)

[Wikc] Wikipedia. URL http://en.wikipedia.org/wiki/Centroid. (Zitiert auf den Seiten 6 und 27)

Alle URLs wurden zuletzt am 05. 10. 2012 geprüft.

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

(Gregor Rothmaier)