Hochschule Konstanz
Fakultät Informatik
Institut für Optische Systeme

Eingereicht von
Lorenz Bung
Matrikelnummer 295261

lorenz.bung@htwg-kostanz.de

Rekonstruktion von Meshes für industrielles Bin-Picking und Depalettieren

Konstanz, 30.06.2020

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science (B. Sc.)

an der

Hochschule Konstanz
Technik, Wirtschaft und Gestaltung

Fakultät Informatik
Studiengang Angewandte Informatik

Thema: **Rekonstruktion von Meshes für industrielles Bin-Picking und Depalettieren**

Bachelorkandidat: Lorenz Bung, Banater Str. 9, 78467 Konstanz

1. Prüfer: Prof. Dr. Georg Umlauf
2. Prüfer: Simon Schmeißer

Ausgabedatum: 01.04.2020
Abgabedatum: 30.06.2020

Ehrenwörtliche Erklärung

Hiermit erkläre ich, *Lorenz Bung, geboren am 26.06.1997 in Konstanz*, dass ich

(1) meine Bachelorarbeit mit dem Titel

Rekonstruktion von Meshes für industrielles Bin-Picking und Depalettieren

bei der Isys Vision GmbH unter Anleitung von Prof. Dr. Georg Umlauf selbständig und ohne fremde Hilfe angefertigt und keine anderen als die angeführten Hilfen benutzt habe;

(2) die Übernahme wörtlicher Zitate, von Tabellen, Zeichnungen, Bildern und Programmen aus der Literatur oder anderen Quellen (Internet) sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe;

(3) dass die eingereichten Abgabe-Exemplare in Papierform und im PDF-Format vollständig übereinstimmen.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Konstanz, 30.06.2020

(Unterschrift)

Abstract

Thema:	Rekonstruktion von Meshes für industrielles Bin-Picking und Depalettieren
Bachelorkandidat:	Lorenz Bung
Betreuer:	Prof. Dr. Georg Umlauf Institut für Optische Systeme Simon Schmeißer Isys Vision GmbH
Abgabedatum:	30.06.2020
Schlagworte:	Robotik, ROS, PCL, Punktwolke, Geometrisches Modellieren, Machine Vision, Meshrekonstruktion

Describe the objective and results of this thesis in a few words. Typically one page.

Extended Abstract

Thema:	Rekonstruktion von Meshes für industrielles Bin-Picking und Depalettieren
Bachelorkandidat:	Lorenz Bung
Betreuer:	Prof. Dr. Georg Umlauf Institut für Optische Systeme Simon Schmeißer Isys Vision GmbH
Abgabedatum:	30.06.2020
Schlagworte:	Robotik, ROS, PCL, Punktwolke, Geometrisches Modellieren, Machine Vision, Meshrekonstruktion

Extended Abstract über 2 Seiten. Beispielhafte Texte aus anderen Teamprojekten oder Abschlussarbeiten können aus dem verlinkten Dokument entnommen werden.

Dieser Text soll als Dokumentation des Teamprojekts für den zukünftigen Jahresbericht des Institut für Optische Systeme dienen. Gerne können auch Bilder eingefügt werden. Ebenso wichtig ist es auch die Referenzen aufzulisten wie z.B. [1].

Danksagung

Ich möchte mich herzlich bei Simon Schmeißer von der Firma Isys Vision GmbH für die Betreuung bedanken - ohne ihn wäre diese Bachelorarbeit nicht möglich gewesen.

Auch allen weiteren Mitarbeitern von Isys Vision möchte ich für das Ermöglichen der Arbeit sowie das freundliche Arbeitsumfeld danken.

Weiterhin gilt mein Dank den Mitgliedern des Instituts für Optische Systeme, da sie mich bei meiner Tätigkeit am Institut maßgeblich auf diese Arbeit vorbereitet haben.

Inhaltsverzeichnis

Abkürzungsverzeichnis

PCL Point Cloud Library

ROS Robot Operating System

KinFu Kinect Fusion

ICP Iterative Closest Point

4PCS Four-Points Congruent Sets

1. Einleitung

1.1 Mikado

Isys vision GmbH [2] ist ein Unternehmen aus Freiburg im Breisgau, das sich mit Systemintegration und industrieller Bildverarbeitung beschäftigt. Neben Entwicklungen in der 2D-Bildverarbeitung (zum Beispiel in der Leiterplattenproduktion) spielt auch Machine Vision im 3D-Bereich in Kombination mit Robotik eine große Rolle.

Mikado [3] ist ein Softwarepaket von isys vision, welches zur Robotersteuerung und 3D-Bildverarbeitung in der Industrie eingesetzt wird. Es besteht aus zwei Komponenten: Mikado 3D dient der eigentlichen Bildverarbeitung, während Mikado Adaptive Robot Control (ARC) zusätzlich dazu die Robotersteuerung und Kollisionsplanung beinhaltet. Die Hauptanwendung ist dabei das sogenannte ”Bin-Picking”, also das Greifen von sortenreinen Teilen aus einer unsortierten Kiste. Zur Erfassung der Bilddaten kommen 3D-Kameras von Ensenso [4] zum Einsatz.

1.2 Motivation

Bei der Bestimmung der 6D-Posen von Objekten wird bei Mikado unter Anderem der ”surface based matching”-Algorithmus von MvTec Halcon [5] verwendet. Dafür wird ein CAD-Modell des Objekts benötigt, welches jedoch in vielen Fällen nicht vorhanden ist.

Die Gründe dafür sind vielfältig. Etwa können die existierenden Modelle im aktuellen Fertigungszustand nicht vorhanden sein. Ein weiterer Grund für fehlende Modelle ist, dass diese aus organisatorischen Gründen schwer zu bekommen sind, beispielsweise wenn nur eine Weiterverarbeitung eines zugelieferten Bauteils stattfindet.

Neben fehlenden Modellen sind auch häufig falsche Modelle ein Problem. So kann es vorkommen, dass die existierenden Daten fertigungsbedingt nicht zum tatsächlichen Produkt passen und Teile daher nicht genau erkannt werden können.

Die Generierung eines CAD-Modells aus den Daten der 3D-Kamera ist somit eine Möglichkeit, um für das tatsächlich vorhandene Produkt eine korrekte Repräsentation zu finden, welche im weiteren Bildverarbeitungsprozess genutzt werden kann. Weiterhin dient dies auch der Vereinfachung der Endanwendung von Mikado ARC. Besonders bei oft variierenden Produktkonfigurationen wird der Anwendungsprozess vereinfacht, wenn nicht erst ein entsprechendes CAD-Modell organisiert werden muss.

1.3 Zielsetzung

Für die Rekonstruktion aus der Punktwolke gibt es drei mögliche Ansätze:

1. Es liegt ein stationäres Objekt vor, welches mit einer am Roboter angebrachten 3D-Kamera erfasst wird. Der Roboter bewegt die Kamera um das Objekt herum, um so Informationen aus mehreren Perspektiven zu erhalten. Diese werden danach miteinander kombiniert; der Hintergrund der erfassten Daten wird anschließend eliminiert.
2. Die Kamera ist stationär angebracht, während das zu erkennende Objekt freihändig demonstriert wird. So kann das Objekt aus mehreren Ansichten aufgenommen werden, welche dann zusammengeführt werden.
3. Es liegen mehrere Objekte der selben Art in verschiedenen Orientierungen vor, beispielsweise in einer Kiste. Nach Segmentierung der Objekte wird aus den unterschiedlichen Teilansichten ein repräsentatives Modell generiert.

Ziel der Arbeit ist es, eine geeignete Lösung für den dritten Ansatz zu entwickeln sowie die unterschiedlichen Rekonstruktionsmöglichkeiten zu implementieren.

Weiterhin sollen die Ergebnisse untereinander und mit bereits bestehenden Algorithmen verglichen werden. Dieser Vergleich soll sowohl bezüglich der wichtigsten Eigenschaft der Qualität, als auch auf Basis untergeordneter Faktoren wie Geschwindigkeit der Algorithmen und Nutzungskomfort der Ansätze stattfinden.

2. Grundlagen

Zum Verständnis des Themas der Arbeit ist die Erklärung einiger Grundlagen notwendig. In 2.1 und 2.2 werden zunächst einmal wichtige Grundbegriffe und Datenstrukturen erläutert, die bei der Arbeit mit 3D-Daten auftreten. Anschließend werden die verwendeten Bibliotheken Point Cloud Library (PCL) und Robot Operating System (ROS) erklärt, welche zur Datenverarbeitung bzw. zur Robotersteuerung verwendet wurden. Außerdem sind selbstverständlich die bereits bestehenden Elemente des Mikado-Projekts relevant, da diese Arbeit fundamental darauf aufbaut.

2.1 Aufnahme und Speicherung von 3D-Bildern

Zur Aufnahme von 3D-Bilddaten gibt es mehrere verschiedene Möglichkeiten. Ein LIDAR-System sendet beispielsweise mehrere Lichtstrahlen in verschiedene Richtungen, die anschließend Informationen über die Entfernung zu einem Objekt in diesem Punkt liefern. Eine Stereokamera liefert im Gegensatz dazu zwei Bilder, die anschließend durch spezielle Software zu einem dreidimensionalen Bild zusammengesetzt werden. Eine weitere Möglichkeit besteht darin, ein bestimmtes Muster auf die Umgebung zu projezieren, dieses dann aus einer anderen Perspektive aufzunehmen und aus der räumlichen Verzerrung des Musters die Tiefe zu errechnen.

Die so gewonnenen Informationen lassen sich durch mehrere verschiedene Datenmodelle repräsentieren.

2.1.1 Tiefenbild

Ein Tiefenbild ist eine einfache Möglichkeit, in einem zweidimensionalen Bild zusätzlich Informationen über die Entfernung der Kamera zu Objekten abzuspeichern. Diese Technik ist unter Anderem aus der Computergrafik bekannt, wo sie beim Z-Buffering Anwendung findet [6]. Dabei werden sowohl das aufgenommene Bild, aber zusätzlich auch ein 2D-Array mit der Tiefeninformation des zugehörigen Pixels gespeichert.

So kann das originale 2D-Bild um eine dritte Dimension erweitert werden, wodurch sich beispielsweise dreidimensionale Formen modellieren oder rekonstruieren lassen [7]. Auch in der 3D-Fotografie finden Tiefenbilder Anwendung [8].

Abbildung 2.1: Depth Map eines 2D-Farbbilds. Entnommen aus [9, S.649]

Tiefenbilder haben im Vergleich zu anderen möglichen Datenmodellen den Vorteil, dass 2D-Bilder sehr einfach um zugehörige Tiefeninformationen erweitert werden können. Jedoch gibt es auch einige Nachteile, die nicht umgangen werden können:

- Die 3D-Daten sind ausschließlich aus der Kameraperspektive vorhanden. Um Informationen aus einer anderen Perspektive zu erhalten, muss erst aufwändig umgerechnet werden.
- Da (in den meisten Fällen) nur ein zweidimensionales Array für die Tiefeninformation angelegt wird, können nur die Entfernungen zu den der Kamera nächsten Objekte gespeichert werden. Verdeckte, reflektierende oder durchsichtige Oberflächen können nicht gespeichert werden.
- Das Array ist an die Auflösung der Kamera gebunden. Insbesondere bei großer Entfernung zu Objekten werden diese aufgrund des Bildwinkels sehr schlecht abgetastet, was später zu Aliasing-Effekten führen kann.
- Die Bittiefe der Werte im Array kann - je nach gewünschter Auflösung - zu niedrig sein, bzw. muss erhöht werden. Eine typische Farbtiefe eines Grauwertbilds von 8 Bit repräsentiert beispielsweise nur $2^8 = 256$ Stufen - was schnell zu wenig wird.

Insbesondere bei hoher notwendiger Auflösung im nahen Bereich, aber gleichzeitig vorhandenen weit entfernten Objekten kann dies zum Problem werden.

2.1.2 Voxel Grid

Bei einem als Bitmap vorliegendes 2D-Bild werden die Bilddaten diskretisiert in einem zweidimensionalen Array I gespeichert. Mit einer Bittiefe b, einer Bildhöhe i und einer Bildbreite j ergibt sich damit folgende Matrix:

$$I = \begin{pmatrix} v_{11} & v_{12} & v_{13} & \cdots \\ v_{21} & v_{22} & v_{23} & \cdots \\ v_{31} & v_{32} & v_{33} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}, v_{ij} \in [0; 2^b) \wedge b, i, j \in \mathbb{N}^+$$

Dieses Prinzip lässt sich einfach auf den dreidimensionalen Raum erweitern. Die so erhaltene Datenstruktur nennt sich Voxel Grid. Als Voxel bezeichnet man somit eine einzelne Datenzelle im 3D-Array, also das dreidimensionale Äquivalent zum Pixel.

Voxel Grids finden in vielen Bereichen Anwendung, zum Beispiel in der Medizin. Auch zum Downsampling von Punktwolken werden sie verwendet [10].

Gegenüber einem Tiefenbild hat ein Voxel Grid den inhärenten Nachteil, dass Speicherplatz für jeden Voxel benötigt wird. Dies führt schnell zu großem Speicherbedarf M, da dieser kubisch zur Auflösung r steigt: $M = r_x * r_y * r_z * b$ bei Bittiefe b.

Ein Vorteil im Vergleich zum Tiefenbild ist jedoch, dass die Abhängigkeit von der Kameraperspektive wegfällt. Dadurch können hier auch Objekte modelliert werden, die im Tiefenbild durch eine Verdeckung versteckt wären.

Desweiteren ist der Raum einheitlich diskretisiert. Dies kann je nach Szene und Kameraposition sowohl ein Vor- als auch ein Nachteil sein: Bei einem Tiefenbild nimmt der Abstand der Messpunkte mit der Tiefe ab. Ist die Entfernung zwischen Objekt und Kamera gering, werden diese mit einer deutlich besseren Auflösung abgetastet als sehr weit entfernte Objekte. Im Voxel Grid werden Objekte jedoch überall gleich abgetastet. Dies führt zu verbesserter Auflösung bei entfernten und zu schlechterer Auflösung bei nahen Objekten.

2.1.3 Punktwolken

Ein weiteres häufig verwendetes Datenmodell ist eine Punktwolke. Als Punktwolke bezeichnet man eine Menge $M \subset \mathbb{R}^3$ von Punkten im (mindestens) dreidimensionalen Raum. Zusätzlich zur räumlichen Information können auch noch weitere Daten pro Punkt gespeichert sein, wie RGB-Werte, Genauigkeit oder Objektklasse (falls schon eine Segmentierung vorgenommen wurde). Dadurch gilt:

$$M = \begin{pmatrix} p_x^1 & p_y^1 & p_z^1 & \cdots \\ p_x^2 & p_y^2 & p_z^2 & \cdots \\ p_x^3 & p_y^3 & p_z^3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

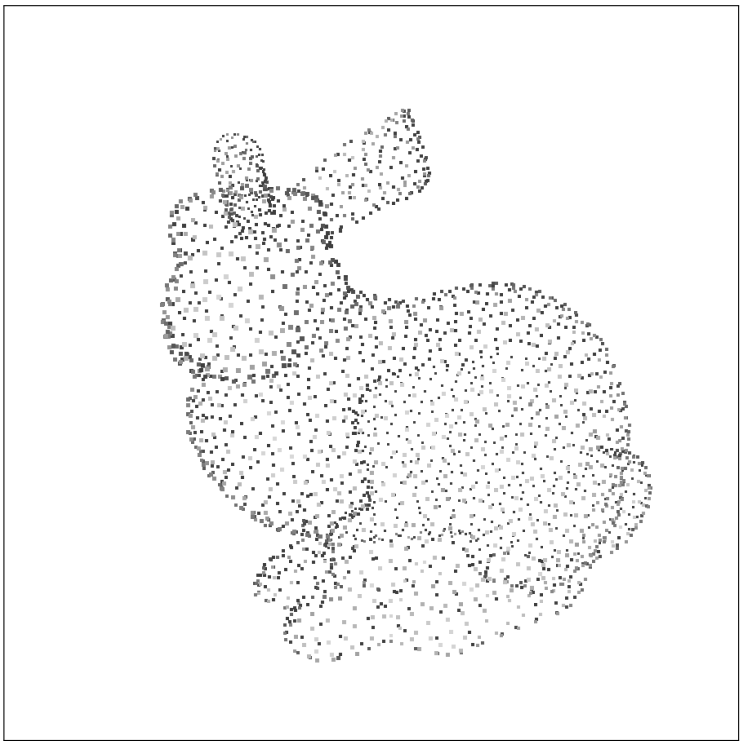

Abbildung 2.2: Punktwolke des Stanford Bunny [11]

Die Nutzung von Punktwolken bringt im Vergleich zu anderen 3D-Datenmodellen einige Vorteile.

- Die zugrundeliegende Datenstruktur ist extrem trivial; es handelt sich um eine einfache Liste. Dies ermöglicht sehr schnelle Operationen, wie zum Beispiel das Hinzufügen eines Punktes. Es lässt sich sehr einfach über die Punkte iterieren.

- Der Speicherbedarf einer Punktwolke wächst dynamisch, und zwar linear mit der Zahl der enthaltenen Punkte. Beim Voxel Grid muss dagegen bereits am Anfang Speicher für jede Zelle reserviert werden, egal ob ein Punkt enthalten ist oder nicht.
- Man ist nicht, wie beim Tiefenbild oder Voxelgrid, auf eine feste Anzahl an Punkten limitiert. Beim Tiefenbild können maximal $n*m$ Punkte (ein Wert pro Pixel) mit einer Tiefeninformation versehen werden, beim Voxel Grid maximal $x*y*z$ Punkte (ein Wert pro Voxel). Eine Kapazitätserweiterung, wie beispielsweise das Hinzufügen neuer Pixelspalten oder -zeilen im Tiefenbild, entfallen bei der Punktwolke.
- Die Auflösung ist nur durch die Gleitkommagenauigkeit der Maschine limitiert. Beim Voxel Grid ist sie im Gegensatz dazu durch die Größe der Voxel limitiert. Beim Tiefenbild ist die Auflösung abhängig von der Tiefe im Bild - bei weiter entfernten Objekten ist die Auflösung auch niedriger.

Es gibt jedoch auch einige Nachteile gegenüber den anderen Repräsentationen:

- Die Rekonstruktion eines Meshs ist nicht eindeutig. Beim Voxel Grid lässt sich, beispielsweise mithilfe des Marching-Cubes-Algorithmus [12], ein eindeutiges Mesh rekonstruieren. Bei einer Punktwolke ist im Gegensatz dazu nicht festgelegt, welche Punkte miteinander verbunden werden sollen.
- Viele Techniken aus der Bildverarbeitung (wie zum Beispiel Segmentierung, Filterung usw.) lassen sich auf Tiefenbilder direkt übertragen. Dies ist bei Punktwolken nicht möglich.

2.2 Meshrepräsentationen

2.3 Point Cloud Library

Bei der PCL [1] handelt es sich um eine Bibliothek, welche die Arbeit mit 3D-Punktwolken immens vereinfacht. Sie bietet Möglichkeiten zur Filterung, Segmentierung, Oberflächenrekonstruktion und Visualisierung von Punktwolken, sowie weitere Module.

Neben den zahlreichen unterstützten Anwendungsgebieten und implementierten Algorithmen bietet die PCL den weiteren entscheidenden Vorteil, dass sie direkt zu ROS

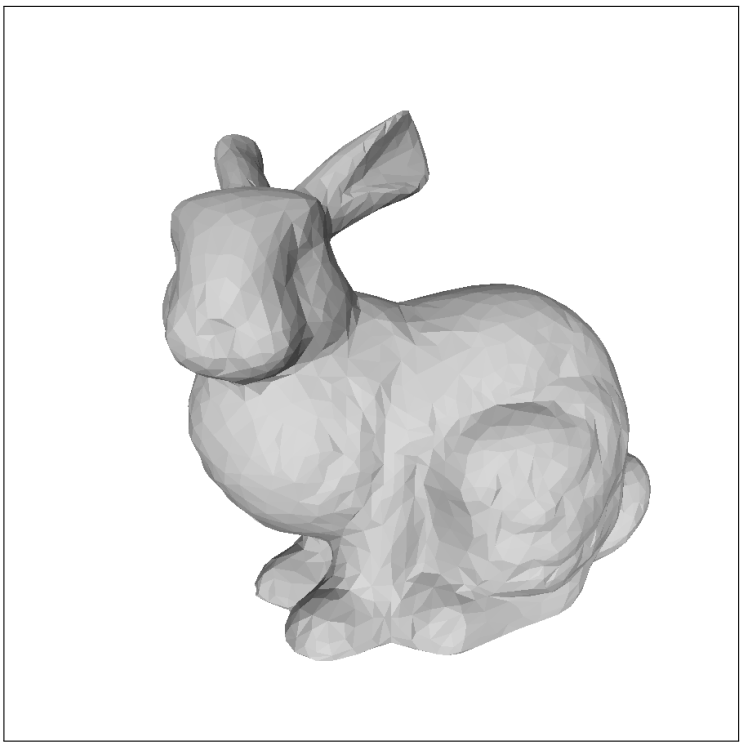

Abbildung 2.3: Mesh des Stanford Bunny [11]

kompatibel ist. Die Kompatibilität wird durch das ROS-Paket `perception_pcl` hergestellt, wodurch Punktwolken, Meshes oder andere Datenstrukturen direkt per Message versendet werden können.

Greedy Projection Reconstruction [13], Poisson Reconstruction [14]

2.4 Robot Operating System

3. Vorhandene Arbeit

In diesem Kapitel wird der aktuelle Stand der Forschung genannt sowie bereits existierende Lösungsansätze erklärt.

3.1 Lokale Registrierung

Bei der lokalen Registrierung von zwei Punktwolken müssen diese bereits grob aneinander ausgerichtet sein. Der Registrierungsalgorithmus verfeinert diese Ausrichtung dann weiter. Einer der bekanntesten Algorithmen zur lokalen Registrierung ist Iterative Closest Point (ICP) und seine vielen verschiedenen Varianten.

3.1.1 Iterative Closest Point

3.1.2 Kinect Fusion

Durch die Veröffentlichung der Microsoft Kinect im Jahr 2010, einer 3D-Kamera für die Nutzung im Entertainment- und Gamingbereich, wurde durch den niedrigen Preis erstmals der breiten Masse der Zugang zu Tiefenkameras ermöglicht [15, 1:55]. Kinect Fusion (KinFu) ist das Ergebnis einer Forschungsarbeit von Microsoft Research [16] und bietet eine Möglichkeit, 3D-Rekonstruktionen in Echtzeit durchzuführen.

YAK? [17]

3.2 Globale Registrierung

Bei der globalen Registrierung müssen sich die beiden Punktwolken nicht nahe der endgültigen Ausrichtung befinden - Translation und Rotation können beliebig sein. Der Nachteil ist die häufig sehr hohe Komplexität der Algorithmen. Außerdem bietet die globale Registrierung oft nur eine Grobregistrierung, weshalb es sich meist anbietet, anschließend noch eine lokale Registrierung zur Verbesserung der Ergebnisse durchzuführen.

3.2.1 Four-Points Congruent Sets

4. Implementierung

5. Auswertung

6. Fazit

Literatur

[1] Radu Bogdan Rusu und Steve Cousins. “3d is here: Point cloud library (pcl)”. In: *2011 IEEE international conference on robotics and automation*. IEEE. 2011, S. 1–4.

[2] *Isys Vision GmbH*. URL: https://www.isys-vision.de/ (besucht am 30. 03. 2020).

[3] *Mikado - 3D Bin Picking System for all applications*. URL: https://www.mikado-robotics.com (besucht am 30. 03. 2020).

[4] *Ensenso - Stereo 3D Cameras*. URL: https://www.ensenso.com (besucht am 30. 03. 2020).

[5] Bertram Heinrich Drost und Markus Ulrich. *Recognition and pose determination of 3D objects in 3D scenes*. US Patent 8,830,229. 2014.

[6] Wolfgang Straßer. “Schnelle Kurven-und Flächendarstellung auf grafischen Sichtgeräten”. Diss. 1974.

[7] Amir Arsalan Soltani u. a. “Synthesizing 3d shapes via modeling multi-view depth maps and silhouettes with deep generative networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, S. 1511–1519.

[8] Andre Redert u. a. “Philips 3D solutions: From content creation to visualization”. In: *Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT’06)*. IEEE. 2006, S. 429–431.

[9] Karsten Muller, Philipp Merkle und Thomas Wiegand. “3-D video representation using depth maps”. In: *Proceedings of the IEEE* 99.4 (2010), S. 643–656.

[10] *Downsampling a PointCloud using a VoxelGrid filter*. Website URL changed in may 2020. URL: http://pointclouds.org/documentation/tutorials/voxel_grid.php (besucht am 06. 04. 2020).

[11] Stanford University Computer Graphics Laboratory. *Stanford Bunny*. URL: http://graphics.stanford.edu/data/3Dscanrep/ (besucht am 30. 04. 2020).

[12] William E Lorensen und Harvey E Cline. “Marching cubes: A high resolution 3D surface construction algorithm”. In: *ACM siggraph computer graphics* 21.4 (1987), S. 163–169.

[13] Zoltan Csaba Marton, Radu Bogdan Rusu und Michael Beetz. “On Fast Surface Reconstruction Methods for Large and Noisy Datasets”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Kobe, Japan, 2009.

[14] Michael Kazhdan, Matthew Bolitho und Hugues Hoppe. “Poisson surface reconstruction”. In: *Proceedings of the fourth Eurographics symposium on Geometry processing*. Bd. 7. 2006.

[15] David Kim. *28c3: KinectFusion*. URL: https://www.youtube.com/watch?v=RvrCAw1IFG0 (besucht am 09. 04. 2020).

[16] Shahram Izadi u. a. “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*. 2011, S. 559–568.

[17] Matthew Klingensmith u. a. “Chisel: Real Time Large Scale 3D Reconstruction Onboard a Mobile Device using Spatially Hashed Signed Distance Fields”. In: *Robotics: science and systems*. Bd. 4. 2015, S. 1.