

**Optimierung von Augmented Reality  
Anwendungen durch die Berücksichtigung  
von Tiefeninformationen mit Googles Project  
Tango**

**MASTERTHESIS**

ausgearbeitet von  
**Steffen Tröster**

vorgelegt an der  
**TECHNISCHEN HOCHSCHULE KÖLN**  
**INGENIEURWISSENSCHAFTLICHES ZENTRUM**  
**FAKULTÄT FÜR INFORMATIONS-,**  
**MEDIEN- UND ELEKTROTECHNIK**

im Studiengang  
**TECHNISCHE INFORMATIK (MASTER)**

in Kooperation mit der  
**inovex GmbH**

Erster Prüfer: Prof. Dr. Hubert Randerath  
Technische Hochschule Köln

Zweiter Prüfer: Prof. Dr. Martin Eisemann  
Technische Hochschule Köln

Köln, im April 2016



**Adressen:**

Steffen Tröster  
Ennest 30  
57392 Schmallenberg  
[steffen.troester@googlemail.com](mailto:steffen.troester@googlemail.com)

Prof. Dr. Hubert Randerath  
Technische Hochschule Köln  
Institut für Nachrichtentechnik  
Betzdorfer Straße 2  
50679 Köln  
[hubert.randerath@th-koeln.de](mailto:hubert.randerath@th-koeln.de)

Prof. Dr. Martin Eisemann  
Technische Hochschule Köln  
Institut für Informatik  
Steinmüllerallee 1  
51643 Gummersbach  
[martin.eisemann@th-koeln.de](mailto:martin.eisemann@th-koeln.de)

Christian Meder  
inovex GmbH  
Ludwig-Erhard-Allee 6  
76131 Karlsruhe  
[christian.meder@inovex.de](mailto:christian.meder@inovex.de)

**Fakultät für Informations-,  
Medien- und Elektrotechnik**

## **Masterarbeit**

Thema:

**Optimierung von Augmented Reality Anwendungen  
durch die Berücksichtigung von Tiefeninformationen mit  
Googles Project Tango**

Name, Vorname: Tröster, Steffen

Anschrift: Münchener Straße 29, 51103 Köln

Matrikelnummer: 11075591

Studiengang: Technische Informatik (Master)

Erstprüfer: Prof. Dr. Hubert Randerath

Zweitprüfer: Prof. Dr. Martin Eisemann

Anfertigungszeitraum: 10.11.2015

Fertigstellung/Abgabedatum: 11.04.2016

## **Erklärung der Urheberschaft**

Ich erkläre an Eides statt, dass ich die vorgelegte Abschlussarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Ort, Datum

Unterschrift

## Zusammenfassung

Project Tango ist eine neue mobile Plattform des Google Advanced Technology and Project (ATAP) Teams, die in der Lage ist, Bewegungsverfolgung, Tiefenwahrnehmung und Umgebungswiedererkennung auf Smartphones und Tablets anbieten zu können. Durch die kontinuierliche Bestimmung der relativen Gerätelage eignet sich die Plattform besonders für dreidimensionale Augmented Reality (AR) Anwendungen. Die Illusion dieser AR Anwendungen wird besonders dann gestört, wenn sich reale Objekte in einer Szene räumlich vor virtuellen Objekten befindet und diese virtuellen Objekte nicht entsprechend ausgespart werden.

Diese Arbeit vergleicht drei Überdeckungsverfahren, mit denen diese Überlagerung der virtuellen Objekte mit Hilfe der Tiefenwahrnehmung von Project Tango und des Z-Buffer Algorithmus realisiert werden kann. Die Tiefeninformationen für den Z-Buffer werden hierfür zum einen direkt aus den Sensordaten und alternativ mit einer TSDF Rekonstruktion und einer selbst zusammengestellten Ebenenrekonstruktion bestimmt. Außerdem wird auf einen zusätzlichen Ansatz eingegangen, der zur Verbesserung dieser Tiefeninformationen die Bildinformationen der Farbkamera durch den Guided Filter berücksichtigt. Diese Mechanismen werden im Laufe der Arbeit prototypisch umgesetzt und gegenübergestellt.

## **Abstract**

Project Tango is a new mobile platform by Google's Advanced Technology and Projects (ATAP) Teams, which brings motion tracking, depth perception, and area learning to smartphones and tablets. With its motion tracking technology, Project Tango is suitable for precise three dimensional augmented reality (AR) applications. The illusion of the model projection in these AR applications is often disrupted when real objects in the scene are located in front of virtual projections, which are not getting occluded.

This presented thesis is comparing three occlusion mechanisms, which can solve the virtual object occlusion with Project Tangos depth perception by applying the Z-Buffer algorithm. The Z-Buffer is filled either by the direct sensor data, by a TSDF reconstruction method or by a self combined and implemented plane based reconstruction. Additionally a guided image filtering approach is applied to the depth map to interpolate according to the edges of the RGB image frame. The proposed mechanisms are prototypically implemented and comparatively analyzed in detail.

# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>1</b>
1.1 Motivation . . . . .	2
1.2 Zielsetzung und Vorgehen . . . . .	3
<b>2 Thematische Vorbemerkung</b>	<b>5</b>
2.1 Augmented Reality . . . . .	5
2.1.1 Technische Anforderungen . . . . .	6
2.1.1.1 Display Technologie . . . . .	6
2.1.1.2 Tracking Technologien . . . . .	8
2.1.1.3 Interaktions Technologien . . . . .	9
2.1.2 Anwendungsbereiche . . . . .	10
2.1.3 Einschränkungen und Probleme . . . . .	12
2.1.4 Realisierung von Überdeckungen in Augmented Reality	12
2.2 Project Tango . . . . .	14
2.2.1 Geräte und Hardware . . . . .	14
2.2.2 Konzepte und Schnittstellen . . . . .	15
2.2.2.1 Motion Tracking . . . . .	16
2.2.2.2 Depth Perception . . . . .	17
2.2.2.3 Area Learning . . . . .	18
2.2.3 Einordnung von Project Tango im Kontext der Augmented Reality . . . . .	19
<b>3 Theoretische Grundlagen</b>	<b>20</b>
3.1 Z-Buffer Algorithmus . . . . .	20
3.2 Random Sample Consensus . . . . .	21
3.3 Ermittlung der konvexen Hülle . . . . .	22
3.4 Octree Datenstruktur . . . . .	22
3.5 Interaktion durch Raypicking . . . . .	24

<b>4 Verfahren zur Realisierung von Überdeckungen in Augmented Reality durch Tiefeninformationen</b>	<b>26</b>
4.1 Verdeckung durch Pointcloud Projektion . . . . .	27
4.2 Planare Rekonstruktion . . . . .	29
4.2.1 RANSAC zur Ebenendetektion . . . . .	30
4.2.2 Bestimmung der Ebenenausbreitung . . . . .	32
4.2.3 Clustering der aufgenommenen Punkte . . . . .	34
4.3 Echtzeit Polygon Rekonstruktion . . . . .	35
4.3.1 Truncated Signed Distance Function . . . . .	37
4.3.2 Spatial Hashing . . . . .	38
4.3.3 Marching Cubes . . . . .	39
4.3.4 Chisel mit Space Carving . . . . .	40
4.4 Tiefenanpassungen durch Farbbilder . . . . .	41
<b>5 Implementierung</b>	<b>46</b>
5.1 Finaler Prototyp . . . . .	46
5.2 Technische Umsetzung und Struktur . . . . .	47
5.3 Umsetzung der Verfahren . . . . .	50
5.4 Technische Problemstellungen . . . . .	53
<b>6 Tests</b>	<b>55</b>
6.1 Statische Testszenen . . . . .	55
6.2 Durchführung der Tests . . . . .	57
6.3 Auswertung der Ergebnisse . . . . .	58
<b>7 Fazit</b>	<b>60</b>
7.1 Evaluation . . . . .	60
7.2 Einsatz der Verfahren . . . . .	65
7.3 Ausblick . . . . .	66
<b>A Ergebnisaufnahmen</b>	<b>72</b>
<b>B Differenz OpenCV Python Skript</b>	<b>76</b>
<b>Literaturverzeichnis</b>	<b>76</b>

# Kapitel 1

## Einleitung

Project Tango ist eine neue mobile Plattform des Google Advanced Technology and Projects (ATAP) Teams, welche Bewegungsverfolgung, Tiefenwahrnehmung und Umgebungswiedererkennung auf mobilen Endgeräten realisiert.

„Project Tango combines 3D motion tracking with depth sensing to give your mobile device the ability to know where it is and how it moves through space.“ (Google, 2015d)

Diese Verfügbarkeit dieser Echtzeitdaten ermöglicht viele verschiedene neue Einsatzmöglichkeiten auf mobilen Endgeräten wie Smartphones und Tablets. Typische Einsatzszenarien dieser Plattform sind die Indoor Navigation, die Vermessung der Umgebung sowie andere Anwendungen im Bereich Virtual und Augmented Reality. Der Fokus dieser Forschungsarbeit liegt hier in dem Anwendungsbereich der dreidimensionalen Augmented Reality (AR).

Die Anwendungsgebiete für Augmented Reality (dt. Erweiterte Realität) sind sehr vielseitig und liegen in der Medizin, der Unterhaltungsindustrie, der Bildung und in vielen weiteren Industriezweigen. Eine barrierefreie Navigationshilfe, Einblendungen für eine persönliche Assistenz, kontextsensitive Projektionen und Computerspiele sind Beispiele für typische Anwendungen, die durch AR umgesetzt werden können.

Für die erfolgreiche Umsetzen einer Augmented Reality Anwendung, müssen die Kameraeigenschaften, wie Brennweite, Verzerrung und die Position

der Kamera zu jeder Zeit und idealerweise in Echtzeit bekannt sein. Sensoren wie Kompass, INS (Trägheitsnavigationssystem) oder GPS können zwar eine grobe Lokalisierung ohne bekannte Merkmale im Raum ermöglichen, führen aber langfristig zu Fehlern, wenn keine optischen Referenzen gegeben sind. Mit Hilfe der Bewegungsverfolgung durch Project Tango kann diese Lokalisierung der Kamera und somit die korrekte Positionierung von virtuellen Objekten im Raum deutlich zuverlässiger, in Echtzeit und ohne vordefinierte Merkmale im Raum realisiert werden. Project Tango eignet sich daher sehr gut für die Umsetzung und den Einsatz von AR Anwendungen.

## 1.1 Motivation

Um eine für den Betrachter effektive und optimierte Augmented Reality Anwendung umsetzen zu können, benötigt man laut Azuma u. a. (2001) die Möglichkeit mehr Informationen über relevante Objekte im realen Raum ermitteln zu können. Durch diese Informationen könnte dem Nutzer zum Beispiel eine Interaktion mit realen Objekten ermöglicht werden oder anhand optischer und semantischer Einordnung der Umgebung passende Funktionen angeboten werden.

Hinsichtlich der zuletzt erwähnten Kontextsensitivität existieren viele Ansätze, basierend auf optischen Merkmalen der Umgebung. So kann zum Beispiel ein optisches Tracking von realen Objekten, wie von Lee u. Hollerer (2008) beschrieben, umgesetzt werden. Project Tango nutzt bereits optische Merkmale, um eine Positionsverfolgung oder das Lernen der Umgebung umzusetzen. Wären diese Merkmale für den Entwickler als Schnittstelle verfügbar, könnte man mit diesen Informationen solche kontextsensitiven Anwendungen umsetzen. Der Fokus soll in dieser Arbeit jedoch nicht auf den optischen Merkmalen sondern auf den Tiefeninformationen, die Project Tango durch den eingebauten Tiefensensor in Form einer Pointcloud liefern kann, liegen.

Ein sinnvoller Einsatz von Augmented Reality Anwendungen besteht darin, virtuelle Objekte in eine echte Szene zu projizieren. Dabei überlagert die Projektion des virtuellen Objekts das aktuelle Kamerabild oder den aktuellen

Sichtbereich und erwirkt dadurch den Anschein, als ob sich das virtuelle Objekt wirklich in der Szene befindet. Dieser Effekt funktioniert solange erfolgreich, bis ein reales Objekt sich räumlich vor das virtuelle Objekt bewegt und die zu erwartende Überlagerung des virtuellen Objekts nicht erfolgt. Diese fehlerhafte Darstellung durch eine fehlende Überdeckung ist in Abbildung 1.1 zu erkennen.



Abbildung 1.1: AR Projektion mit Project Tango - Links: Erfolgreiche Projektion. Rechts: Fehlerhafte Darstellung ohne Überdeckung.

Die Verfügbarkeit der Tiefeninformationen bei Project Tango könnte die Interaktionen oder Darstellungen in einer Augmented Reality Anwendung präziser an die echten räumlichen Gegebenheiten anpassen. Es existieren zum Beispiel prototypische Anwendungen, in denen virtuelle Markierungen passend an echten Objekten im virtuellen Raum positioniert werden können, indem sie auf die aktuellen Tiefeninformationen des Sichtbereichs zurückgreifen. Eine weitere Idee ist es, Überlagerungen virtueller Objekte ermitteln zu können, an denen sich reale Objekte im Vordergrund befinden.

## 1.2 Zielsetzung und Vorgehen

Diese Arbeit will die Fragestellung beantworten, durch welche Verfahren mithilfe der Tiefeninformationen von Project Tango automatisch und in Echtzeit die Überdeckung virtueller Objekte mit realen Objekten in einer Augmented Reality Szene realisiert werden kann. Dabei soll Project Tango als autonomes System betrachtet werden, welches diese Problemstellung selbstständig und mit den eingeschränkten Ressourcen dieser mobilen Plattform lösen soll.

Hierzu sollen zunächst bestehende Verfahren zur Bestimmung einer Augmented Reality Überdeckung durch eine Literaturrecherche gefunden werden. Diese Verfahren sollen dabei auf Ihre Anwendbarkeit mit der Project Tango Hardware überprüft werden. Sollten sich aus der Recherche weitere Ideen ergeben, wie speziell auf der Project Tango Hardware eine Überdeckung umgesetzt oder verbessert werden kann, sollen diese mit in die Arbeit eingebunden werden. Eine Idee könnte zum Beispiel sein, auch das Farbbild der normalen Kamera von Project Tango in die Optimierung der virtuellen Überlagerung einfließen zu lassen. Die identifizierten Verfahren sollen hiernach entsprechend implementiert werden, um sie darauf folgend in einer Testumgebung gegenüber zu stellen.

Strukturell wird in dieser Arbeit in Kapitel 2 auf die thematischen Grundlagen zu Augmented Reality und Project Tango eingegangen. Hier werden auch die existierenden Verfahren zur AR Überdeckung angesprochen. Unter Kapitel 3 sind theoretische Grundlagen zu finden, die bei der späteren Umsetzung verschiedener Verfahren angewendet werden. Kapitel 4 beinhaltet die Argumentation und Beschreibung der gewählten Verfahren, welche unter Kapitel 5 auf der Project Tango Hardware umgesetzt werden. In Kapitel 6 werden die vorliegenden Umsetzungen in einem Testszenario gegenübergestellt, um eine Aussage treffen zu können, welcher Ansatz auf der Hardware oder für einen bestimmten Einsatz gut funktionieren könnte.

# Kapitel 2

## Thematische Vorbemerkung

Im ersten Teil dieses Kapitels werden die Grundlagen zu Augmented Reality beschrieben, wie diese Technologie einzuordnen ist, welche technischen Anforderungen ein Augmented Reality System hat und wo typische Einsatzszenarien liegen. Außerdem wird hier auf den aktuellen Stand der Forschung bezüglich der Bestimmung von Überdeckungen in einem Augmented Reality System eingegangen. Hiernach wird Googles Project Tango Technologie vorgestellt, welche Konzepte angewendet werden und wie diese Technologie im Bereich Augmented Reality einzuordnen ist.

### 2.1 Augmented Reality

Augmented Reality (AR) ist eine Klasse aus dem Realitäts-Virtualitäts-Kontinuum von Milgram u. a. (1995), welches in Abbildung 2.1 abgebildet ist. Diese Klasse beschreibt die Darstellungen von realen und virtuellen Informationen in einer Repräsentationsform, wobei hier reale und virtuelle Objekte in einer realen Umgebung kombiniert dargestellt werden können. Diese virtuellen Objekte sind in der realen Umgebung idealerweise fest lokalisiert und fügen sich somit in das reale Erscheinungsbild ein. Typischerweise sind AR Anwendungen interaktiv und stellen die virtuellen Objekte in Echtzeit und dreidimensional in der realen Welt dar. Für die Definition von AR Anwendungen gibt es zudem keine Limitierung für die Darstellungstechnologie, wie zum Beispiel das Project Tango Tablet oder ein Head-Mounted-Display. AR

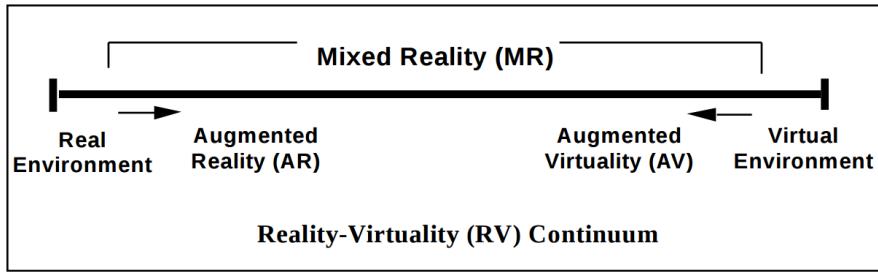


Abbildung 2.1: Vereinfachte Darstellung des Realitäts-Virtualitäts-Kontinuums von Milgram, Takemura, Utsumi, u. Kishino (1995)

beschränkt sich zudem nicht auf den angesprochenen Sinn - so sind zum Beispiel AR Anwendungen mit visueller, taktiler oder sogar olfaktorischer Umsetzung möglich.

Virtual Reality (VR) oder auch Virtual Environment hingegen kapselt sich von der realen Umgebung ab und bietet Interaktionen in reinen virtuellen Umgebungen. Diese rein virtuelle Darstellung konnte sich im Gegensatz zu Augmented Reality deutlich schneller entwickeln, da die technologischen Anforderungen an VR deutlich geringer sind. (Van Krevelen u. Poelman, 2010)

### 2.1.1 Technische Anforderungen

Dieser Abschnitt widmet sich den technischen Anforderungen an Augmented Reality, indem die potentiellen Display Technologien beschrieben werden, mögliche Trackingverfahren zur Ermittlung der Betrachtungsposition erläutert werden und auch die Systeme behandelt werden, mit denen ein Nutzer mit den virtuellen Darstellungen interagieren kann.

#### 2.1.1.1 Display Technologie

Der erste wichtige Teil der technologischen Anforderungen an AR sind visuelle Anzeigen (visual displays), die neben der Möglichkeit eines dreidimensionalen Renderings, welches hier aus der Virtual Reality vorausgesetzt werden soll, weitere Charakteristika mit sich bringen. Nach Van Krevelen u. Poelman

(2010) lassen sich diese Technologien zunächst in je drei Arten der Darstellung und Positionierung unterteilen.

Die einfachste und günstigste Art der visuellen Darstellung in AR ist „video see-through“, wodurch die reale Umgebung durch eine Video Aufnahme ersetzt wird und die virtuellen Objekte digital in die Videoaufnahme gerendert werden. Das bietet die Möglichkeit Objekte aus der realen Umgebung zu entfernen oder zu ändern oder, anhand der Luminanz Information vom Video, das Rendering der virtuellen Objekte entsprechend an die Realität anzupassen. Anwendung findet diese Technologie typischerweise in Tablets, Smartphones oder Head-Mounted-Displays.

Die nächste Möglichkeit zur Darstellung ist „optical see-through“. Hier werden die virtuellen Objekte durch transparente Spiegel in das Sichtfeld des Betrachters gebracht. Anders als bei „video see-through“ bleibt die reale Auflösung für die visuelle Aufnahme des Betrachters gleich und es können zudem nur Latenzprobleme bei dem Rendering der virtuellen Objekte und nicht bei der Darstellung der realen Umgebung auftreten. Auf der anderen Seite besteht bei dieser Technologie das Problem, dass die Darstellung von virtuellen Objekten nicht kräftig genug ist, um die reale Umgebung auf Grund von der transparenten Darstellungsoberfläche komplett auszublenden. Typische Geräte dieser Technologie sind Headmounted Displays wie Google Glass<sup>1</sup> oder stationäre Geräte wie der HoloDesk<sup>2</sup>.

Die dritte Möglichkeit ist die projizierte Darstellung, in der die Augmented Reality Überlagerung auf die realen Objekte projiziert werden. Diese Darstellung ermöglicht die Abdeckung vom gesamten Sichtfeld des Betrachters, benötigt aber eine entsprechende Kalibrierung oder eine Strukturwahrnehmung bei Umgebungsänderungen.

Neben der Art der Darstellung können die Display Technologien laut Azuma u. a. (2001) anhand ihrer Positionierung klassifiziert werden. Man unterscheidet zwischen am Kopf befestigten Displays (head-mounted), tragbaren Displays (hand-held) und räumlich positionierten Displays. Zu jeder dieser Displayarten gibt es wiederum unterschiedliche technische Umsetzungen mit ihren spezifischen Vor- und Nachteilen bezüglich ihrer Anwendungsszenarien.

---

<sup>1</sup>Googel Glass - <https://developers.google.com/glass/> (23.02.2016)

<sup>2</sup>HoloDesk - <http://research.microsoft.com/en-us/projects/holodesk/> (23.02.2016)

### 2.1.1.2 Tracking Technologien

Um eine virtuelle Projektion im realen Raum auf nicht stationären Displaytechnologien zu realisieren, müssen die Position und gegebenenfalls relative Positionsänderungen des Displays bestimmt werden, auch „augmented reality registration“ genannt. Man spricht dabei üblicherweise von den „six degrees of freedom (6DOF)“, der Position im Raum (x, y, z) und der Orientierung (yaw, pitch, roll). Da bei vielen Displaytechnologien auch die verwendete Kamera direkt mitgeführt wird, entsprechen diese Informationen meist auch den extrinsischen Kameraeigenschaften der AR Kamera.

Frühe Techniken für die Registrierung benötigten üblicherweise eine speziell vorbereitete Räumlichkeit, denn sie basierten auf mechanischen, magnetischen oder Ultraschallsensoren um die Position zu bestimmen. Diese Sensoren sind zwar noch im Einsatz und bilden auch den Grundstein für die AR und VR Forschung, sind aber praktisch gesehen zu komplex und aufwändig für die meisten Anwendungsfälle. (Van Krevelen u. Poelman, 2010)

Für ein grobes Positions-Tracking, vor allem auch außerhalb von Gebäuden wird GPS genutzt. Für großräumige Anwendung ist GPS, mit einer Varianz von 10-15 Metern und in Kombination mit einem Kompass, durchaus praktikabel. Als Beispiel reicht diese Genauigkeit aus, um sichtbare Flugzeuge oder Sterne visuell aufzubereiten. Innerhalb von Gebäuden basiert die grobe Positionierung laut Van Krevelen u. Poelman (2010) oft auf verfügbaren Wifi Access Points oder RFID Markern. LaMarca u. a. (2005) demonstrieren hierzu auch die Möglichkeit dieser Idee für grobe Lokalisation außerhalb von Gebäuden einzusetzen.

Optische Tracking Verfahren, basierend auf Bildverarbeitung, bieten laut Van Krevelen u. Poelman (2010) deutlich genauere Resultate als die zuvor beschriebenen Verfahren. Es gibt hier viele verschiedene sensorische Ansätze, ein optisches Tracking zu realisieren. Frühe Verfahren, wie die von Dunston u. a. (2008) oder Narzt u. a. (2006), nutzten Passmarker (fiducial marker) oder Licht emittierende Dioden (LED) in einem vordefinierten Modell, um zwischen aufgenommenen Bildern die Marker oder LEDs zu detektieren und zusammengehörige zwischen den Bildern zu finden, um daraus eine Kamera transformation zu berechnen. Neue Verfahren ohne Marker, wie das sogenannte

„visual odometry“ von Nistér u. a. (2004), nutzen Techniken zur Feature Detection und Matching, um gemeinsame Punkte zwischen aufgenommenen Bildern zu bestimmen und somit die Bewegungen zu berechnen.

Viele kommerzielle und erfolgreiche Tracking Verfahren beruhen jedoch auf hybriden Ansätzen, in denen die Informationen mehrerer Sensoren kombiniert werden, um potentielle Messfehler eines Sensors oder einer Methodik auszuschließen. So werden zum Beispiel Neigungssensor, Kompass und Gyroskop mit einem optischen Verfahren kombiniert, um ein Tracking der sechs Freiheitsgrade zu optimieren. Diese Erweiterung des optischen Verfahrens wird auch „visual-inertial odometry“ genannt. (Van Krevelen u. Poelman, 2010)

Azuma u. a. (2001) erwähnt an dieser Stelle auch die Kalibrierung der Sensoren, die für ein präzises Registrieren nötig ist. So müssen zum Beispiel die Linseneigenschaften der Kamera für optisches Tracking bekannt sein, damit die Verfahren mit Krümmungen, Verzerrungen und den perspektivischen Eigenschaften der Kamera umgehen können. Diese Informationen sind auch bei video see-through Displays für ein korrektes Projizieren der 3D Objekte wichtig. Zudem wird erwähnt, dass man Messfehlern oder Drifts der Position zum Beispiel unter Zuhilfenahme von Gyroscop Informationen entgegenwirken kann, indem man Ereignisse wie einen Schritt des Nutzers einfließen lässt. (Azuma u. a., 2001)

### 2.1.1.3 Interaktions Technologien

Neben den Display und Tracking Technologien ist es notwendig dem Nutzer andere angemessene Interaktionsmöglichkeiten anzubieten, da in der Regel das klassische zweidimensionale WIMP Paradigma (Windows, Icons, Menus and Pointer) im dreidimensionalen Kontext von AR keine ausreichende Gebrauchstauglichkeit bietet. Dennoch müssen die Interaktionstechnologien in Augmented Reality die üblichen Interaktionen, welche unter anderem aus WIMP bekannt sind, unterstützen. Dazu gehören zum Beispiel das Auswählen, Positionieren und Drehen von virtuellen Objekten, das Zeichnen von Pfaden oder Flugbahnen, sowie die Eingabe von quantitativen Werten oder Texten. (Van Krevelen u. Poelman, 2010)

Frühe Augmented Reality Systeme nutzten einfache Trackballs, Trackpads, Touchscreens oder Gyroskopmäuse für eine zweidimensionale Interaktion mit dem System. Später wurden dreidimensionale Äquivalente eingeführt, wie 3D Mäuse oder Stifte, die eine dreidimensionale Interaktion ermöglichen. Diese greifbaren Schnittstellen werden auch TUIs genannt (Tangible User Interface) und ermöglichen eine unidirektionale Interaktion mit dem System. Zudem wurden auch TUIs mit haptischen Feedback eingeführt, wie zum Beispiel die 3D Maus PHANTOM. (Van Krevelen u. Poelman, 2010)

Eine weitere Art der TUIs sind laut Azuma u. a. (2001) Gegenstände, mit denen der Nutzer natürlich interagieren kann und die vom System optisch erfasst werden, um die Positionsänderung der Objekte anhand von Markern oder anderen optischen Merkmalen zu bestimmen. Somit kann ein Nutzer, zum Beispiel, für die virtuelle Einrichtung eines Raums, die virtuellen Möbel mit Hilfe eines echten Gegenstands im Raum verschieben.

Nicht taktile Systeme verwenden meist optische Aufnahmen, um Gesten der Hände, des gesamten Körpers oder die Blickrichtung des Nutzers erkennen zu können. Für die Realisierung werden Kameras am Körper oder im Raum verwendet. Außerdem ist es möglich, Spracherkennung in die Interaktion mit einfließen zu lassen, um eine möglichst authentische Interaktion zu bieten. Wie auch bei den Tracking Technologien existieren hierbei hybride Systeme, die verschiedene Interaktions Technologien kombinieren. (Van Krevelen u. Poelman, 2010)

### 2.1.2 Anwendungsbereiche

Über die Jahre haben Wissenschaftler immer mehr Bereiche identifiziert, die von der Anwendung von Augmented Reality profitieren können. Van Krevelen u. Poelman (2010) nennt dazu als erstes Einsatzgebiet die persönliche Assistenz, in der AR Systeme eingesetzt werden können, um zum Beispiel mit Hilfe von Brillen (etwa der Google Glass) Namen der sichtbaren Personen anzuzeigen, die Navigation in unbekannten Regionen einzublenden oder beim Sightseeing kontextrelevante Informationen im Sichtfeld anzuzeigen.

Neben der persönlichen Assistenz können auch Anwendungen in der Industrie laut Van Krevelen u. Poelman (2010) von AR profitieren. Es lassen sich zum Beispiel virtuelle Designumgebungen umsetzen, die es ermöglichen, ein Auto in Lebensgröße zu gestalten. Auch bei der Fertigung und Konstruktion können den Arbeitern unterstützende Informationen angezeigt werden. So werden zum Beispiel zu erledigende Schweißstellen hervorgehoben oder der Plan zur Konstruktion entsprechend eingeblendet. Eine weitere Möglichkeit wäre es, für die Instandhaltung komplexer Maschinen dem Nutzer, über ein AR System eine Art Röntgenblick mit Hinweisen auf potentielle Schwachstellen bereitzustellen. Auch in der Rüstungsindustrie existieren Anwendungsgebiete für Augmented Reality Systeme. So können zum Beispiel Gefechte für eine Kampfausbildung besser simuliert werden. (Azuma u. a., 2001)

Für Anwendungsbereiche in der Medizin ist ein sehr genaues Tracking der Freiheitsgrade erforderlich, da AR in der Chirurgie und Behandlung von Patienten Anwendung findet. Erstellte Röntgenbilder oder Ultraschallbilder können hierdurch, anstatt auf einem separaten Monitor, direkt auf die entsprechende Körperstelle projiziert werden, wodurch gegebenenfalls eine genauere Untersuchung oder Behandlung möglich ist. (Van Krevelen u. Poelman, 2010)

Augmented Reality wird auch im Entertainment Sektor eingesetzt. Videoübertragungen von Sportereignissen werden heutzutage oft durch zusätzliche Informationen angereichert. So erhalten zum Beispiel American Football Spiele dynamische Spielfeldbegrenzungen. Auch die Werbeeinblendungen am Rand des Spielfelds können entsprechend dem Gebiet der Ausstrahlung ausgetauscht werden. (Azuma u. a., 2001)

Ein weiteres großes Anwendungsgebiet für Augmented Reality sind laut Azuma u. a. (2001) Computerspiele, in denen es möglich ist, in einer beliebigen Umgebung Objekte eines Spiels im Raum zu platzieren und mit ihnen entsprechend zu interagieren. Die natürlichere Interaktion gegenüber herkömmlichen Spielplattformen und die Nutzung in einer persönlichen Spielumgebung führt zu einem intensiveren Spielerlebnis.

Auch in pädagogischen Bereichen, in Schulen oder Museen können AR Anwendungen eingesetzt werden. Zur Vermittlung von geometrischen oder mathematischen Grundlagen gibt es die Möglichkeit der kollaborativen und

interaktiven Visualisierung von Körpern, an denen etwa Parameter manipuliert werden, um danach ihre Eigenschaften besser beobachten und nachvollziehen zu können. (Van Krevelen u. Poelman, 2010)

### **2.1.3 Einschränkungen und Probleme**

Die frühen Augmented Reality Systeme sind auf Grund ihrer Größe sehr unhandlich und mobil daher nur mit großem Aufwand anwendbar. Durch die Verfügbarkeit mobiler und performanter Endgeräte ist ein mobiler Einsatz wiederum ermöglicht worden. Jedoch besitzen die aktuellen Geräte wie Smartphones oder Tablets nicht die entsprechende Sensorik für ein präzises Tracking der sechs Freiheitsgrade. Van Krevelen u. Poelman (2010) weisen zudem darauf hin, dass die Registrierung der Tiefe für die Anwendung von Überdeckungen oder korrekter Positionierung bei einer Interaktion ein komplexes Problem sei. Wie in Kapitel 2.2 zu finden, geht Project Tango dieses Problem der Sensorik entsprechend an und versucht Schnittstellen zu bieten, um sowohl das Tracking zu ermöglichen als auch Tiefeninformationen über die aktuelle Szene zu liefern.

Eine weitere erwähnenswerte Problematik ist neben den sensorischen Themen von Augmented Reality die Herangehensweise zur Gestaltung der Nutzeroberflächen für AR. Denn die UI Konzeption gestaltet sich, laut Azuma u. a. (2001), schwierig. Die Anreicherungen durch Augmented Reality führt schnell dazu, dass das Sichtfeld überladen wirkt. Jedoch sollen dem Nutzer immer die Informationen zur Verfügung gestellt werden, die gegebenenfalls kontextsensitiv und relevant sind. Diese angesprochenen Faktoren führen aktuell bei Augmented Reality Anwendungsgebieten noch zu einer geringen Akzeptanz der Endverbraucher.

### **2.1.4 Realisierung von Überdeckungen in Augmented Reality**

Es gibt einige Verfahren und Ansätze, um eine Überlagerung in Augmented Reality Szenen zu realisieren. Wloka u. Anderson (1995) bilden hierfür den Grundstein für die verschiedenen existierenden Methoden. Sie stellen in

Ihrer Arbeit ein Verfahren vor, welches mit Bildern aus einer Stereokamera ein Stereomatching durchführt und dadurch ein Tiefenbild generiert. Dieses Tiefenbild führt in dem Renderingprozess mit Hilfe des Z-Buffer Algorithmus (beschrieben in Kapitel 3.1) zum Ausschluss von Teilen der virtuellen Objekte, die von realen Objekten überlagert werden. Das Ergebnis des Stereomatchings ist in ihrer Arbeit mit gewissen Ungenauigkeiten behaftet und generiert unerwünschte Lücken in der Projektion des virtuellen Objekts. Arbeiten wie von Seo u. Lee (2013) mit neuen Tiefensensoren, wie der Microsoft Kinect<sup>3</sup>, erhalten durch den selben Mechanismus deutlich bessere Ergebnisse.

Die Arbeit von Breen u. a. (1996) nahmen diesen Ansatz von Wloka u. Anderson (1995) auf und stellten die Idee vor, neben einer deutlich genaueren Überdeckung, auch eine Interaktion mit realen Objekten zu realisieren. Hierfür werden virtuelle Modelle der realen Objekte in der Szene passend an der echten Umgebung und der Position der realen Objekte ausgerichtet. Dieses Vorgehen setzt jedoch voraus, dass die entsprechenden virtuellen Modelle für die realen Objekte bereits vorliegen. Nach dieser Ausrichtung wird die Tiefe der virtuellen Objekte bestimmt, um daraus, mit dem Verfahren von Wloka u. Anderson (1995), eine Überlagerung durch den Ausschluss der weiteren virtuellen Objekte der Szene zu bewirken.

Neben den modellbasierten Verfahren existieren auch kantenbasierte Verfahren, wie das von Berger (1997), in dem Objektkanten auf optischer Basis mit Filtern ermittelt werden. Diese Kanten werden über mehrere Bilder verfolgt, um die Tiefeninformation der Kanten durch Epipolargeometrie und Heuristiken zu bestimmen. Berger (1997) gewinnt darauf folgend eine Tiefenmaske, indem er annimmt, dass Konturen, die unter einer gewissen Distanz von einander entfernt sind, zu einem Objekt gehören. Klein u. Drummond (2004) erreichen mit Bergers Verfahren und einer auf mobiler Hardware umgesetzten Umgebung sehr überzeugende Ergebnisse in einer vordefinierten Umgebung. Zwar verspricht dieses Vorgehen eine kantengenaue Überdeckung virtueller Objekte, führt aber bei komplexeren Szenen, in denen die Kanten nicht mehr erfolgreich verfolgt werden oder nicht zu einem Objekt zugeordnet werden können, zu Fehldarstellungen. Außerdem werden Ausbreitungen innerhalb des Objekts, welche nicht als Kante erkannt werden können, nicht berücksichtigt.

---

<sup>3</sup>Microsoft Kinect - <https://dev.windows.com/en-us/kinect> (04.03.16)

Die letzte Variante wurde von Breen u. a. (1996) bereits erwähnt und ist die Ermittlung der Überdeckung durch eine Rekonstruktion der Szene. Dieser Ansatz verschiebt durch das Verfahren von Wloka u. Anderson (1995) die Problemstellung der AR Überdeckung in den Bereich der Echtzeit Rekonstruktionsproblematik. Bekannte Verfahren hierfür sind zum Beispiel KinectFusion von Newcombe u. a. (2011) oder die Echtzeitrekonstruktion von Nießner u. a. (2013). Diese sehr komplexen Verfahren sind meist auf der Grafikhardware von Desktopsystemen umgesetzt und generieren eine detaillierte Rekonstruktion aus den Tiefeninformationen in Echtzeit. Diese Rekonstruktionen können laut Newcombe u. a. (2011) für eine Echtzeitüberdeckung in Augmented Reality Systemen dienen. Vorteilhaft bei rekonstruktionsbasierter Überdeckung ist zudem, dass auch Interaktionen mit echten Objekten, wie bei Breen u. a. (1996), erfolgreich umgesetzt werden können.

## 2.2 Project Tango

Project Tango ist eine Technologie Plattform für Android Tablets und Smartphones von Google's Advanced Technology and Projects Group (ATAP). Das Ziel dieser Plattform ist es, Motion Tracking (Positionierung), Depth Perception (Tiefeninformation/Pointcloud) und Area Learning (Lokalisierung) auf mobile Endgeräte zu bringen, um verschiedenste Anwendungs-Szenarien abzudecken. Typische Szenarien sind Indoor Navigation, Virtual Reality Anwendungen, Vermessungs- und Rekonstruktionssoftware und Augmented Reality Anwendungen.

Das System ermöglicht in erster Linie ein Tracking von Positionsänderungen des Geräts im Raum und bietet somit eine genaue relative Lokalisierung. Mit Hilfe dieser Lokalisierung und der Hinzunahme von visuellen Merkmalen im Raum ist das Gerät in der Lage, seine Umgebung kennenzulernen und gegebenenfalls die Lokalisierung zu korrigieren oder aber diese in einer bereits erlernten Umgebung zu bestimmen. Zusätzlich bietet Project Tango die Möglichkeit, mit Hilfe eines Tiefensensors, eine Pointcloud der Tiefeninformation pro Bildausschnitt zu ermitteln, um Anwendungen auch räumliche Informationen bereitzustellen. (Google, 2015d)

### 2.2.1 Geräte und Hardware

Da das Project Tango zum Zeitpunkt der Verfassung dieser Thesis noch unter Entwicklung steht, gibt es von Google nur erste Entwickler Prototypen. Das erste Gerät „Peanut Phone“ im Smartphone Format, welches in Abbildung 2.2 rechts unten zu erkennen ist, wurde Anfang 2014 veröffentlicht und ein halbes Jahr später bereits durch eine neue Generation, dem „Yellowstone Tablet“ ersetzt. Dieses 7“Tablet, zu sehen rechts oben im Bild 2.2, verfügt, wie in der Abbildung links zu erkennen, über einen Infrarot Laser Projektor, eine Fisheye Kamera und eine normale 4 Megapixel Kamera auf der Rückseite. Zudem sind, wie in aktuellen Smartphones und Tablets üblich, ein Beschleunigungssensor, Umgebungslichtsensor, Barometer, Kompass, GPS und ein Gyroskop verbaut. Das Gerät wird von einem NVIDIA Tegra K1 Prozessor betrieben und verfügt über 4GB Arbeitsspeicher. (Google, 2015d) Mit diesem Gerät wurden die später beschriebenen Techniken umgesetzt und evaluiert.

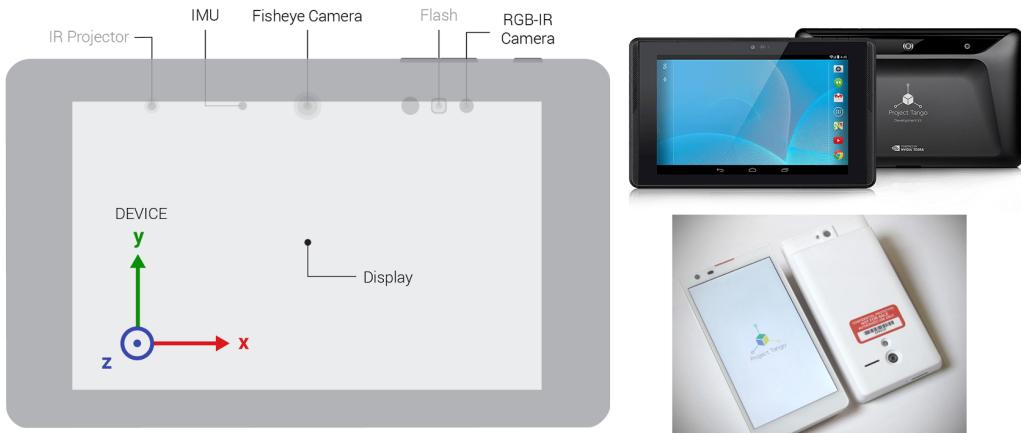


Abbildung 2.2: Links: schematischer Aufbau der Google Project Tango Hardware. Rechts: Das aktuelle Entwickler Gerät im Tablet Format (oben) und das alte Entwickler Gerät im Smartphone Format (unten). Übernommen von Google (2015c)

### 2.2.2 Konzepte und Schnittstellen

Generell betrachtet ist das Project Tango eine Plattform, die Computer Vision nutzt, um dem Gerät die Möglichkeit zu bieten seine relative Positionierung in der umgebenden Szene in Echtzeit zu bestimmen. Auf den Geräten

kommt Googles Android Betriebssystem zum Einsatz, weshalb zu beachten ist, dass es sich bei der Plattform nur bedingt um eine Echtzeitumgebung handelt. Das liegt daran, dass der Linux Kernel keine Garantien für die zeitlich präzise Ausführung von Instruktionen auf Grund von Scheduling geben kann. Google weist daher darauf hin, dass das System als „soft-realtime“ betrachtet werden sollte. Daher sollten Messergebnisse verschiedener Sensoren unter Berücksichtigung ihrer Aufnahmepunkte verwendet werden. (Google, 2015e)

#### **2.2.2.1 Motion Tracking**

Um die relative Bewegung vom Start des Project Tango Systems bestimmen zu können, nutzt es „visual-inertial odometry“. (Google, 2015e) Dabei handelt es sich um eine erweiterte Variante von Visual Odometry. Das von Nistér u. a. (2004) veröffentlichte Verfahren Visual Odometry ist in der Lage aus einfachen Videoinhalten in Echtzeit die Bewegung der Kamera zu bestimmen. Hierzu werden zunächst übergreifende Features, zum Beispiel Punkte aus der FAST Kantenerkennung von Trajković u. Hedley (1998), aus mehreren Bildern bestimmt. Um daraus eine Transformation zwischen den Bildern ermitteln zu können, wird der 5-point Algorithmus von Nistér (2004) angewendet. Dieser Algorithmus ist in der Lage das Problem zu lösen, eine relative Transformation zwischen zwei Bildern mit gegebenen 5 Punktübereinstimmungen zu ermitteln. Außerdem wird erwähnt, dass mit Hilfe des Schätzverfahrens RANSAC (beschrieben in Absatz 4.2.1) bei einer Überbestimmung des Modells, ein potentieller Fehler deutlich minimiert werden kann.

Project Tango lässt an dieser Stelle die internen Sensoren zur Rotation, Orientierung und Bewegung mit in die Bestimmung der Kameratransformation einfließen, um so ein präziseres Ergebnis erzielen zu können. Außerdem wird versucht mit Hilfe des Kalman Filters, nach dem gleichnamigen Autor Kalman (1960), die Fehler der Sensoren bei dieser Echtzeitmessung zu reduzieren. Über eine längere Messzeit oder eine größere Entfernung vom Ursprung kann es jedoch zu kleinen Abweichungen kommen. Außerdem existiert zum aktuellen

Zeitpunkt noch ein „Drift“ Problem<sup>4</sup>, was zu großen Messfehlern führen kann. Es wird jedoch versucht diese Probleme mit dem Konzept „Area Learning“, beschrieben in Kapitel 2.2.2.3, zu lösen. (Google, 2015e)

Wie genau das Verfahren aussieht, welche Techniken zur Feature Detection oder Feature Matching genutzt werden und welche Features hierfür erkannt werden, ist nicht bekannt. Klingensmith u. a. (2015), als Mitglieder von Googles Advanced Technologies and Projects Abteilung ATAP, erwähnen jedoch, dass nähere Informationen über das Verfahren von Kottas u. a. (2013) und Mourikis u. a. (2007) beschrieben werden. Sie erläutern in ihren Arbeiten, welche Mechanismen eingesetzt werden können, um eine Migration aller Sensorinformationen für ein zuverlässiges hybrides optisches Tracking zu realisieren.

### 2.2.2.2 Depth Perception

Zur Tiefenmessung ist die Project Tango Hardware mit einem kalibrierten Infrarot Laserprojektor ausgestattet. Dieser streut Infrarot Punkte mit einer Auflösung von 320 x 180 Punkten in den Raum, um mithilfe von Aufnahmen der RGB Kamera eine Punktewolke der Tiefeninformation zu bestimmen. Aufgrund einer ausgewogenen Konfiguration zwischen Messbereich, Messfehlern und dem Energieverbrauch, liegt der Messbereich der Sensorkombination, laut Google (2015e), zwischen einem halben und vier Metern.

Dadurch, dass diese Technologie auf der Aufnahme von projiziertem Infrarotlicht basiert, ist ein Einsatz der Tiefenmessung außerhalb geschlossener Räume nicht möglich (Google, 2015e). Außerdem entstehen Messfehler durch reflektierende, lichtabsorbierende oder zu komplex strukturierte Oberflächen, wie zum Beispiel Metalle, LCD Monitore oder Hochflor Teppiche.

Die zuvor erwähnten Punktewolken werden in dem eigens definierten XYZij Format von der Entwicklungsschnittstelle zurückgegeben. Dabei wird jeder Punkt mit den  $X$ ,  $Y$  und  $Z$  Koordinaten im Weltkoordinatensystem und den beiden Indizes  $i$  und  $j$  für die Spalte und Zeile der projizierten

---

<sup>4</sup>Das Drift Problem tritt auf, wenn zu wenige übereinstimmende Punkte im Raum zwischen den Bildaufnahmen der Fisheye Kamera gefunden werden. Das führt in der Regel zu relativen Bewegungen der virtuellen Kamera obwohl keine reale Bewegung statt findet.

Punkte auf der Bildebene angegeben (Google, 2015e). Man spricht dabei von einer organisierten Punktewolke, da durch die  $i$  und  $j$  Koordinaten die direkten Nachbarn, ausgehend von dem Aufnahmeblickwinkel, eines Punktes bestimmt werden können. Hieraus ist es möglich Tiefenbilder, die sogenannten „Depth Maps“, zu bestimmen, für die es viele verschiedene Computer Vision Verfahren zur Bestimmung von Objekten, Strukturen und Fluchtpunkten gibt. Die Schnittstellen liefern jedoch, laut Google (2015a), zum aktuellen Entwicklungsstand die beschriebenen Informationen über die Spalten  $i$  und Zeilen  $j$  noch nicht.

Google (2015e) weist darauf hin, dass das Generieren von polygonbasierten Rekonstruktionen noch nicht in den Schnittstellen enthalten ist. Es gibt jedoch freie Dritt-Bibliotheken und -Systeme, wie das Robot Operating System (ROS)<sup>5</sup> oder die Point Cloud Library (PCL)<sup>6</sup>, die für eine weitere Verarbeitung genutzt werden können.

### 2.2.2.3 Area Learning

Area Learning bezeichnet den Prozess, in dem Project Tango Geräte in der Lage sind, durch visuelle Hinweise die umgebende Welt kennenzulernen und auf die Position des Gerätes zu schließen. Es ermöglicht somit eine Unterstützung für Motion Tracking und löst das Problem, das Gerät in einer bereits bekannten Umgebung zu lokalisieren, wie in Abbildung 2.3 links zu erkennen ist. Project Tango bietet außerdem die Möglichkeit diese visuellen Hinweise und ihre Position im Raum in sogenannten „Area Description Files“ zu speichern und wiederzuverwenden. (Google, 2015e)

Wie bereits erwähnt entstehen bei Motion Tracking über eine längere Strecke Messfehler. Während diese Strecke mit einem Project Tango Gerät abgelaufen wird, ermittelt es fortlaufend die Position und den Pfad, den der Nutzer im Raum gegangen ist. Erkennt es während der Strecke visuelle Merkmale mittels Area Learning, wird der Pfad anhand der Positionen der Merkmale entsprechend angepasst. Project Tango unterscheidet hier zwischen zwei Manipulationen, „loop closures“, zur Zusammenführung des Pfads wenn

---

<sup>5</sup>Robot Operating System - <http://ros.org/> (23.02.2016)

<sup>6</sup>Pointcloud Library - <http://pointclouds.org/> (23.02.2016)

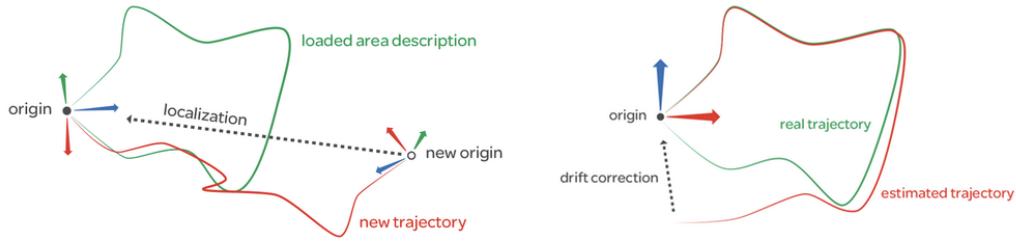


Abbildung 2.3: Links: Lokalisierungsprozess durch Area Learning. Rechts: Korrektur von Motion Tracking anhand gelernter Merkmale. Übernommen von Google (2015c)

ein Kreis gelaufen wurde, und „drift corrections“, um den erwähnten Drift-Effekt bei zu wenigen optischen Features im visual-inertial odometry zu korrigieren. Die drift correction ist in Abbildung 2.3 rechts zu erkennen. (Google, 2015e)

Auch bei diesem Prozess werden die genauen Details nicht näher erläutert und es ist nicht bekannt, wie die Area Descriptions definiert sind oder was sie enthalten. Google (2015e) weist jedoch darauf hin, dass, auch wenn die Area Descriptions Files keine direkten Bilder enthalten, es möglich sei, Rückschlüsse auf die gelernte Umgebung ziehen zu können.

### 2.2.3 Einordnung von Project Tango im Kontext der Augmented Reality

Da sowohl die Grundlagen aus dem Bereich Augmented Reality als auch die technische Basis von Project Tango bekannt sind, kann die Project Tango Hardware bezüglich Augmented Reality näher eingeordnet werden. Bei der Hardware handelt es sich um ein hand-held Gerät, welches mit einer video see-through Display Technologie Augmented Reality Anwendungen ermöglicht. Hierfür kann wahlweise die normale RGB Kamera oder die Graustufen Fish-Eye Kamera verwendet werden. Für beide Kameras können die intrinsischen Kameraparameter ausgelesen werden, wodurch die Eigenschaften der realen Kamera durch die virtuelle Kamera übernommen werden können. Das führt zu einer parallax freien Überblendung und zu einer guten Tiefenwahrnehmung.

Als Tracking Technologie wird hier eine hybride optische Variante angewendet. Dabei wird die Visual Odometry mit der Fish-Eye Kamera durch interne Sensoren für die Rotation, Orientierung und Bewegung kombiniert. Außerdem kann das Verfahren gegebenenfalls durch Googles Area Learning Mechanismen angereichert werden, um Messfehlern entgegenzuwirken. Die Eingabe erfolgt durch den Touchscreen des Tablets. Eine Interaktion mithilfe von optischer Gestenerkennung oder anhand der Tiefeninformationen ist zudem auch denkbar.

# Kapitel 3

## Theoretische Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen und Algorithmen näher erläutert, die zur Realisierung der Überlagerungsverfahren im nachfolgenden Kapitel 4 angewendet werden.

### 3.1 Z-Buffer Algorithmus

Der Z-Buffer Algorithmus, welcher fast zur selben Zeit von Straßer (1974) und Catmull (1974) vorgestellt wurde, ermöglicht in der Computergrafik eine einfache Berechnung der Überdeckungen von gerenderten Objekten auf der Bildebene. Hierzu wird neben der Bildebene noch ein sogenannter „Z-Buffer“ eingeführt, der für jedes gerenderte Pixel auf der Bildebene eine Tiefeninformation festhält. Initial enthält der gesamte Z-Buffer die Tiefeninformation die der Backclipping-Ebene entspricht. Somit lässt sich für jedes Pixel einer bestimmten Position bestimmen, ob dieser einen kleineren Tiefenwert besitzt und somit auf die Bildebene gerendert wird.

Heutzutage ist dieser leicht zu implementierende Mechanismus aus Performancegründen in nahezu jeder Grafikkarte in Hardware implementiert. Das Problem bei dieser einfachen Umsetzung ist jedoch, dass jedes Objekt der Szene den gesamten Renderingprozess durchlaufen muss, auch wenn es später auf der Bildebene nicht erscheint. Abhilfe für dieses Problem bietet der Hierarchical Z-Buffer von Greene u. a. (1993), welcher nicht sichtbare

Objekte zuvor aussortiert. Für die Anwendung für die Optimierungsverfahren in Kapitel 4 ist jedoch nur die Idee des einfachen Z-Buffers relevant.

## 3.2 Random Sample Consensus

Der „RAndom SAmples Consensus“ Algorithmus (RANSAC), vorgestellt von Fischler u. Bolles (1981), ist in der Lage, aus einer Menge von Daten mit vielen Ausreißern, die Parameter für ein passendes Modell zu schätzen, welches von den meisten Messergebnissen unterstützt wird. Anders als andere Schätzverfahren wie „Least-Median“ oder „M-Schätzer“, welche aus der Statistik Literatur entnommen und entsprechend angepasst wurden, wurde RANSAC speziell für die Anwendung in der Computergraphik entwickelt. Der Kern dieses Algorithmus ist das wiederholte Bestimmen eines Modells aus zufälligen und für die Ermittlung des Modells ausreichenden Stichproben. Listing 3.1 zeigt den Verlauf des RANSAC Algorithmus. Die Anzahl der Iterationen  $N$  hängt dabei allein von dem Anteil der Ausreißer in den Messwerten ab. Daher sollte die Anzahl entsprechend gewählt werden, um die Wahrscheinlichkeit zu verringern, dass Ausreißer in den Stichproben enthalten sind. (Derpanis, 2010)

**Eingabe:** Messwerte  $P$ , Modelltoleranz  $e$ , maximale Iterationen  $N$   
**Ausgabe:** Modell  $m$ , Unterstützende Messwerte  $P_m$

1. Wähle zufällig so viele Stichproben aus den Messwerten  $P$ , wie nötig sind, um das Modell zu bestimmen
2. Bestimme aus den gewählten Stichproben das Modell  $m$
3. Ermittle die Anzahl der Messwerte  $P$ , die mit einer entsprechenden Toleranz  $e$  das ermittelte Modell  $m$  unterstützen
4. Wenn prozentual genügend Messwerte aus  $P$  das Modell  $m$  unterstützen dann terminiere und gehe zu 7
5. Wenn  $|P_m| < |P|$  dann  $|P_m| = |P|$
6. Wiederhole die Schritte 1-4  $N$  mal
7. Ermittle aus den unterstützenden Messwerten  $P_m$  erneut das finale Modell  $m$

---

Listing 3.1: Der RANSAC Algorithmus

Die in Schritt sieben beschriebene erneute Modellermittlung kann laut Fischler u. Bolles (1981) durch ein Regressionsverfahren wie der Methode der

kleinsten Quadrate bestimmt werden. Anwendung findet dieser Algorithmus in der Ebenendetektion aus Kapitel 4.2.1.

### 3.3 Ermittlung der konvexen Hülle

Als eine konvexe Hülle wird eine Teilmenge einer euklidischen Gesamtmenge an Punkten bezeichnet, für die folgende Bedingung gilt: Wählt man aus der Gesamtmenge zwei beliebige Punkte, so ist die Strecke zwischen ihnen immer innerhalb der konvexen Hülle. Für die Bestimmung der konvexen Hülle von Punkten  $P$  in  $\mathbb{R}^2$  existieren verschiedenste Algorithmen, wie zum Beispiel der Monotone Chain von Andrew (1979), QuickHull nach Eddy (1977) oder der Divide-and-Conquer Algorithmus von Preparata u. Shamos (1985). Da alle diese Algorithmen ein ähnliches Laufzeitverhalten aufweisen, wird für die spätere Anwendung hier exemplarisch der Graham Scan nach Graham (1972) näher erläutert. Dies ist einer der populärsten Algorithmen für die Berechnung der konvexen Hülle und besitzt eine Komplexität von  $O(n \log n)$ .

Gestartet wird der Algorithmus mit der Menge aller Punkte  $P$  und mit einem Startpunkt  $P_0$ , welcher Bestandteil der konvexen Hülle ist. Hierzu wird meist der Punkt mit dem niedrigsten  $y$  Faktor gewählt ( $P_0 = P_{\min(y)}$ ). Listing 3.2 zeigt den Verlauf des Algorithmus des Graham Scans. Außerdem wird in Abbildung 3.1 zusätzlich die erste Sortierung und das Unterscheidungskriterium für die Sortierung und Aussortierung der Punkte verdeutlicht. (Sunday, 2001)

Eingabe: Menge der Punkte  $P$ , außen liegender Punkt  $P_0$   
Ausgabe: Punkte der konvexen Hülle in  $P$

```
i = 0
sortiere P nach dem Winkel zu P0
solange i <= |P|
    wenn  $\angle P_{i-1}P_i > \angle P_{i-1}P_{i+1}$ , also  $P_i$  rechts von  $\vec{P}_{i-1}P_{i+1}$  liegt
        inkrementiere i
    ansonsten
        entferne  $P_i$  aus P
        dekrementiere i
```

---

Listing 3.2: Graham Scan Algorithmus

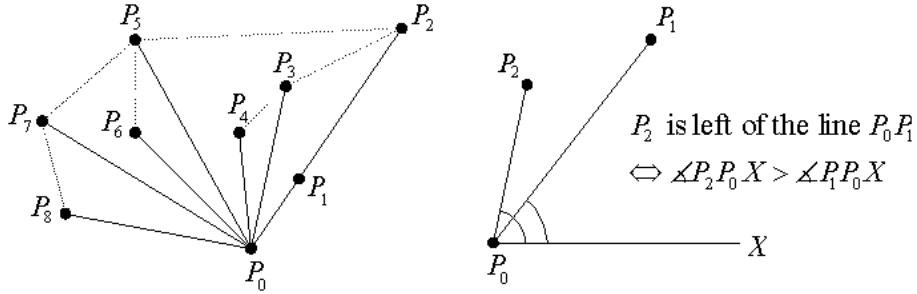


Abbildung 3.1: Sortierung der Punkte nach Winkel zum Startpunkt (links). Das Unterscheidungskriterium für die Sortierung (rechts). Übernommen von Sunday (2001)

### 3.4 Octree Datenstruktur

Ein Octree ist in erster Linie eine Datenstruktur, die wie ein Baum mit beliebiger Tiefe aufgebaut ist und pro Knoten acht Kinder besitzt. Die Funktion ist dabei in  $\mathbb{R}^3$  gleich zu einem Binärbaum in  $\mathbb{R}$  oder einem Quadtree in  $\mathbb{R}^2$ . Ein Knoten repräsentiert im Octree einen Würfel, der durch seine Kinder in acht Kind-Würfel aufgeteilt wird. Diese Aufteilung ist in Abbildung 3.2 exemplarisch dargestellt.

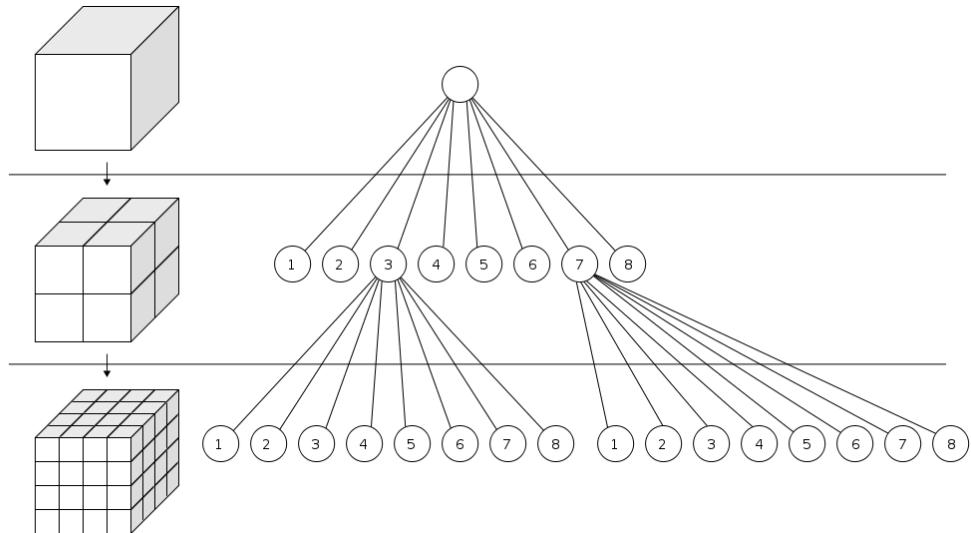


Abbildung 3.2: Schematische Darstellung eines Octrees mit zwei Untergliederungen. Übernommen von Dumusc u. a. (2011)

Durch diese räumliche Aufteilung ergeben sich verschiedene Vorteile gegenüber linearen Datenstrukturen. So müssen Bereiche zum Festhalten

räumlicher Informationen im Octree nur dann allokiert werden, wenn diese Bereiche auch verwendet werden. Außerdem kann man mithilfe von Octrees ein einfaches Downsampling von Daten ermöglichen. Speichert man Punkte in den untersten Knoten eines Octrees kann man durch eine Tiefenbegrenzung beim Zugriff auf den Baum ein sehr effektives Downsampling der Punkte vornehmen. Zudem kann ein Octree zum einfachen räumlichen Clustern von dreidimensionalen Objekten dienen.

## 3.5 Interaktion durch Raypicking

Wie in Kapitel 2.1.1.3 beschrieben, bedarf es bei der Umsetzung von Augmented Reality Systemen eines anderen Interaktionsparadigmas. Auch wenn die Entwicklung der neuen Tablet und Smartphone Geräte mit Touchscreens eine neue Interaktionsform eingeführt haben, ist sie in den meisten Fällen auf eine zweidimensionale Ebene beschränkt. In der Entwicklung von Virtual Reality oder voll virtuellen Anwendungen wie Spielen wird oft für die Auswahlgeste der Raypicking Mechanismus verwendet, um eine zweidimensionale Interaktion im dreidimensionalen Raum zu ermöglichen. Dabei wird anhand der Kameraeigenschaften ein Strahlensatz bestimmt, der von dem ausgewählten Punkt auf der Bildebene in die Szene reicht und mit den darin enthaltenen Objekten auf Schnittpunkte überprüft wird. Darüber hinaus gibt es verbesserte semantische Interaktionsformen basierend auf einer zweidimensionalen Toucheingabe, wie von Elmquist u. Fekete (2008) beschrieben.

Hier soll aber zunächst eine Raypicking Variante für Augmented Reality Anwendungen umgesetzt werden, die nicht von einem kompletten Modell in Form von Polygonen oder anderen Primitiven der realen Umgebung ausgeht. Diese AR Interaktion ermöglicht, anhand der Tiefeninformationen, das passende Positionieren von virtuellen Objekten im realen Raum und lässt sich auf weitere Interaktionen erweitern. Voraussetzung für die folgende Umsetzung ist die entsprechende Kalibrierung und Gleichstellung der intrinsischen Kameraparameter und der Verfügbarkeit der extrinsischen Parameter der realen Kamera.

Als Erstes wird ein Strahl erzeugt, der durch die Position der virtuellen Kamera und durch den jeweils ausgewählten Punkt auf der Viewingplane

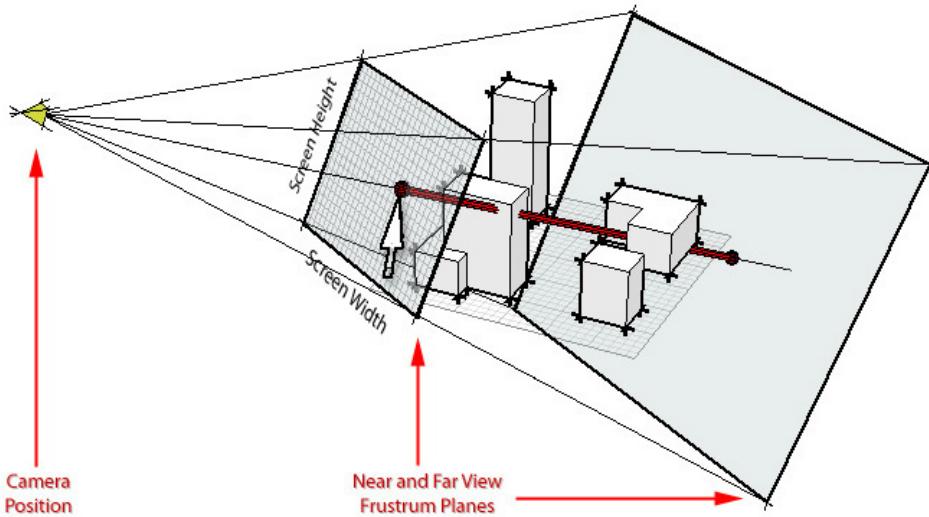


Abbildung 3.3: Raypicking Visualisierung. Übernommen von Marsh (2011)

läuft. Den Ursprung der virtuellen Kamera bestimmt dabei Google Tangos „Motion Tracking“. Der gewählte beziehungsweise berührte Punkt auf dem Touchscreen wird dabei zunächst von Pixeln in das Verhältnis  $[-1, 1]$  des Punktes umgerechnet. Hiernach wird die Projektion auf die Viewingplane durch eine Multiplikation mit  $T$  aus Gleichung 3.1 rückgängig gemacht. Die Gerade aus den beiden Punkten kann danach genutzt werden, um den Schnitt von Objekten vor der Kamera zu ermitteln. (OpenGL, 2016)

$$T = MV_{ModelView}^{-1} * P_{Projection}^{-1} \quad (3.1)$$

Angewendet auf die Tiefeninformation aus Tangos „Depth Perception“ wird die Punktewolke, wie in Abbildung 3.3, anstelle der Objekte, vor die Kamera projiziert. In den projizierten Punkten wird danach der entsprechende Punkt gesucht, welcher sich am nächsten am zuvor bestimmten Strahl befindet. Durch diese beschriebenen Schritte kann der Nutzer mit einer zweidimensionalen Geste einen Punkt in der Tiefe bestimmen. Diese Methode kann zudem um die Ermittlung einer Ebenennormalen erweitert werden. Hierzu werden um den selektierten Punkt Nachbarn gefunden, mit denen durch RANSAC eine Ebene ermittelt wird. Durch die ermittelte Ebenennormale können virtuelle Objekte dann nicht nur an die ausgewählte Stelle positioniert werden, sondern auch an der realen Oberflächenorientierung ausgerichtet werden.

# Kapitel 4

## Verfahren zur Realisierung von Überdeckungen in Augmented Reality durch Tiefeninformationen

Die folgenden Abschnitte widmen sich den Verfahren zur möglichen Realisierung von Überlagerungen durch Tiefen- und Bildinformationen. Nach der Recherche zu möglichen Verfahren soll erst einmal der grundlegende Ansatz von Wloka u. Anderson (1995) zur einfachen Überlagerung durch Tiefeninformationen mithilfe der Projektion der von Project Tango gelieferten Pointcloud realisiert werden, da dieses Ausschlussverfahren das grundlegende Vorgehensmodell zur Überlagerung darstellt.

Die kanten- und modellbasierten Verfahren zur AR Überdeckung aus Kapitel 2.1.4 werden hier nicht weiter berücksichtigt, da sie offensichtliche Nachteile gegenüber anderen Ansätzen bergen. So muss beim modellbasierten Verfahren bereits ein Modell der echten Umgebung existieren und das kantenbasierte Verfahren schränkt den Einsatz auf eine weniger komplexe Szene ein. Außerdem sind beide Verfahrensarten so konzipiert, dass sie keine direkten Tiefeninformationen benötigen, die aber von Project Tango generiert werden können und hier auch ausgehend von der anfangs beschriebenen Zielsetzung genutzt werden. Aus diesem Grund widmen sich die darauf folgend beschriebenen Umsetzungen den rekonstruktionsbasierten Verfahren.

Im Laufe dieser Arbeit wurde zunächst ein eigenes Echtzeit Rekonstruktionsverfahren basierend auf einer Ebenenerkennung zusammengestellt und entwickelt, welches auch auf der mobilen Project Tango Hardware realisierbar ist. Daraufhin wurde nach weiteren Recherchen ein neues Rekonstruktionsverfahren gefunden, welches im Zusammenhang mit Project Tango veröffentlicht, und somit für den Einsatz auf mobiler Hardware konzipiert wurde. Auch dieser Ansatz wird hier näher beschrieben, um ihn später zu implementieren und zu testen. Zuletzt soll näher auf die Möglichkeit eingegangen werden, die resultierenden Tiefeninformationen aus der Pointcloud oder aus dem Rendering der Rekonstruktion mit Hilfe der Bildaufnahmen der Farbkamera, zu verbessern.

## 4.1 Verdeckung durch Pointcloud Projektion

Die erste und weniger aufwändige Idee eine Überlagerung in Augmented Reality zu realisieren, ist die Überführung der Pointcloud in eine Depthmap, die wiederum in den Renderingprozess mit eingebracht wird. Das Verfahren von Kanbara u. a. (2000) verfolgt einen ähnlichen Weg mit einer Stereokamera und einer video see-through Displaytechnologie in Form eines Head-Mounted Display. Wie in Abbildung 4.1 zu erkennen, bestimmen sie mit Hilfe der Stereokamera Tiefeninformationen, die das gerenderte virtuelle Objekt an den Positionen ausspart, an denen Tiefeninformationen im Vordergrund vorliegen. Aufgrund von weiteren Forschungsergebnissen aus dem Bereich des Stereomatchings, erreichen sie bessere Ergebnisse als Wloka u. Anderson (1995), die das Vorgehen als Erste vorgestellt haben.

Anders als im Verfahren von Kanbara u. a. (2000) wird hier eine Depthmap anhand der vorhandenen Pointcloud aus dem Infrarotsensor von Project Tango gewonnen. Dadurch, dass die intrinsischen Kameraeigenschaften der Farbkamera zur Verfügung stehen, welche auch die Aufnahmequelle der Infrarotpunkte ist, kann ein Punkt  $P = [X, Y, Z]$  der Pointcloud mit der Gleichung 4.1 auf die Bildebene überführt werden. Hier stehen die Variablen  $f_{x/y}$  für die Brennweite und  $c_{x/y}$  für den Bildmittelpunkt auf der Bildebene. (Google)

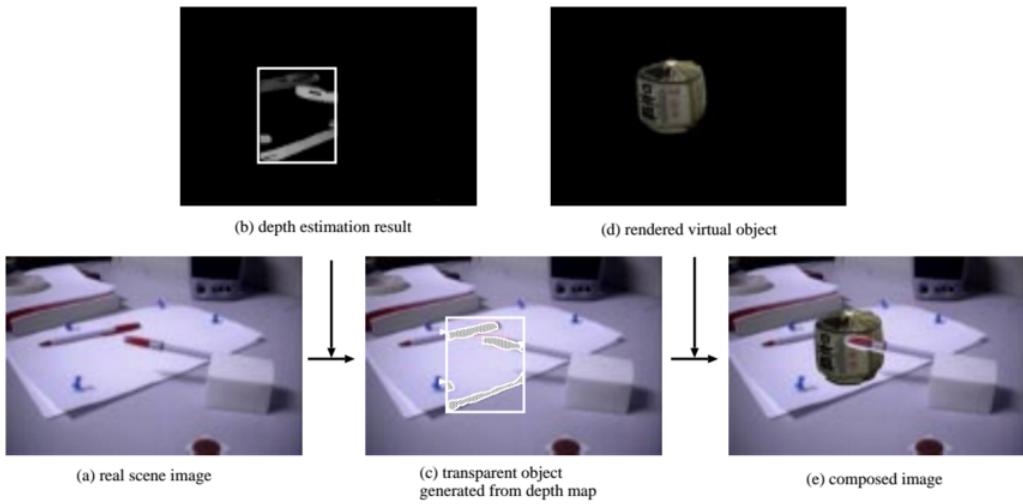


Abbildung 4.1: Visualisierung der Methode zur Vedeckung durch Depth Maps.  
Übernommen von Kanbara u. a. (2000)

$$x = \frac{X * f_x * \frac{r_d}{r_u}}{Z} + c_x \quad y = \frac{Y * f_y * \frac{r_d}{r_u}}{Z} + c_y \quad (4.1)$$

Da die Linsen einer Kamera nie perfekte Brechungseigenschaften besitzen, muss an dieser Stelle auch die Verzerrung der Linse mit berücksichtigt werden. In den Gleichungen aus 4.1 ist diese Verzerrung in den Parametern  $\frac{r_d}{r_u}$  enthalten. Zur Kalibrierung der Linse wird bei Project Tango für die normale Farbkamera und somit auch für die Aufnahme der Infrarotpunkte die Technik von Tsai (1987) verwendet (Google). Das hier beschriebene Verzerrungsmodell basiert dabei auf den drei Parametern  $k_1 \downarrow k_3$ , welche die radiale Verzerrung ausgehend vom Linsenmittelpunkt annähernd modellieren können. Hierzu wird für jeden Punkt, wie in Gleichung 4.2 gezeigt, die radiale Distanz  $r_u$  zum Linsenmittelpunkt und die durch die Linseneigenschaft verzerrte radiale Distanz  $r_d$  ermittelt. Dieses Verhältnis streckt oder staucht die Position radial auf der Bildebene.

$$r_u = \sqrt{\frac{X^2 + Y^2}{Z^2}} \quad r_d = r_u + k_1 * r_u^3 + k_2 * r_u^5 + k_3 * r_u^7 \quad (4.2)$$

An dieser berechneten Position auf der Bildebene bzw. Depthmap wird nun ein Punkt mit einem Graustufenwert, entsprechend der Entfernung  $|\overrightarrow{PO_{cam}}|$  vom Punkt  $P$  zum Kameraursprung  $O_{cam}$  gezeichnet. Der Farbwert richtet sich

dabei nach der Konvention des Renderingframeworks und den Informationen über die vordere und hintere Clippingebene.

Die Auflösung des Tiefensensors der Project Tango Hardware ist mit  $320 \times 180$  Pixeln gegenüber der Auflösung der Farbkamera mit  $1280 \times 720$  vier mal kleiner. Zusätzlich ist die Dichte der Pointcloud zum eigentlichen Sensor geringer als ihre eigene Auflösung. So würde man bei einer Auflösung von  $320 \times 180 = 57600$  Tiefenpunkte bei idealen Verhältnissen erwarten. Project Tango liefert jedoch unter guten Bedingungen durchschnittlich 17000 Tiefenpunkte. Aufgrund der kleineren Auflösung und der geringeren Informationsdichte werden die gezeichneten Punkte auf der Depthmap hier mit einem Radius von 4 Bildpunkten gezeichnet.

Nachdem die Depthmap generiert wurde, kann zum Ausschluss der Pixel der virtuellen Objekte, welches sich hinter einem realen Objekt befinden, der Z-Buffer Algorithmus aus Kapitel 3.1 wie von Wloka u. Anderson (1995) beschrieben, angewendet werden. Hierzu wird vor dem virtuellen Rendering der Z-Buffer mit den generierten Informationen aus der Depthmap gefüllt. Pixel der virtuellen Objekte werden somit nicht gerendert, wenn an dieser Position eine geringere Distanz zu realen Objekten in den Tiefeninformationen vorliegen.

## 4.2 Planare Rekonstruktion

Dieses Kapitel widmet sich der Idee, eine Rekonstruktion für ein Überlagerungsverfahren durch eine Ebenendetektion zu realisieren. Yang u. Förstner (2010) erwähnen hierzu, dass Ebenen in fast allen künstlichen Umgebungen zu finden sind und aufgrund ihrer vorteilhaften geometrischen Eigenschaften in verschiedensten Computer Vision Verfahren verwendet werden. Daher gibt es viele Forschungsarbeiten, Methoden und Algorithmen, um aus verschiedenen Informationsquellen ein Ebenenmodell zu extrahieren.

Das „Simultaneous Localization and Mapping“ (SLAM) Verfahren von Trevor u. a. (2012) detektiert Ebenen mit dem RANSAC Algorithmus. RANSAC bietet gegenüber anderen Algorithmen zur Ebenendetektion den Vorteil, ein

Modell auch bei vielen Ausreißern performant ermitteln zu können. Agglomeratives Clustering und Region Growing wie von Feng u. a. (2014) beschrieben, eignet sich aufgrund des Ausgabeformats von Project Tango nicht direkt, da es keine organisierte Point Cloud ausgibt und die Daten durch Reflektionen und Löcher stärker mit Fehlern behaftet sind.

Das selbst zusammengestellte Verfahren zur Ebenendetektion besteht daher aus folgenden Komponenten: Wie in dem Ansatz von Yang u. Förstner (2010) wird der RANSAC Algorithmus auf Würfeln ausgeführt, die eine Menge gesammelter Punkte aus der Pointcloud beinhalten. Dabei entsprechen die Würfel den untersten Knoten eines Octrees, welcher hier zur Speicherung und Aufnahme aller Punkte verwendet wird. Diese untersten Knoten des Octrees werden folgend auch Cluster genannt. Eine gefundene Ebene in einem Würfel wird wie im SLAM Verfahren von Trevor u. a. (2012)persistiert. Die Repräsentation der Ebene  $P$  wird dort wie in Gleichung 4.3 festgehalten. Dabei handelt es sich um den Normalenvektor  $\vec{n}$  und der Distanz zum Ursprung  $d$  der Hesse Normalform einer Ebene, sowie der Punkte der konvexen Hülle  $H$ . Trevor u. a. (2012) erläutern, dass die konvexe Hülle in der Repräsentation festgehalten wird, um eine sukzessive Verbesserung einer Ebene nach mehreren Messdurchläufen zu ermöglichen. So werden die Punkte der konvexen Hülle pro Messvorgang kombiniert, damit die Ebenenausbreitung auch außerhalb des Sichtfeldes beibehalten werden kann.

$$P = [\vec{n}, d, H] \quad H = \vec{h}_1, \vec{h}_2, \dots, \vec{h}_n \quad (4.3)$$

Die einzelnen Schritte des Vorgehens werden in den folgenden Absätzen näher erläutert. Ein grober Ablauf des Vorgehens wird aber bereits in Listing 4.1 zusammengefasst und in Pseudocode beschrieben.

### 4.2.1 RANSAC zur Ebenendetektion

Um mit dem RANSAC Algorithmus, beschrieben in Kapitel 3.2, Ebenen in einer Punktewolke bestimmen zu können, werden pro Iteration drei Stichproben  $\vec{A}$ ,  $\vec{B}$  und  $\vec{C}$  gewählt, die zur Bestimmung einer Ebene ausreichen. Das Ebenenmodell, hier in der Hesse Normalform mit dem Normalenvektor

Eingabe: Octree  $O$ , Anzahl der zu suchenden Ebene in Clustern  $N$   
Ausgabe: Polygonpunkte  $T_{Gesamt}$

```

für jedes Cluster  $C = [C_{Punkte}, C_{Ebenen}]$  aus  $O$ 
 führe  $N$  mal aus
    bestimme Ebene  $[\vec{n}, d, P]$  mit RANSAC aus  $C_{Punkte}$ 
    wenn keine Ebene mit genügend  $P$  gefunden wurde
        nächstes Cluster (continue)
    wenn Ebene mit  $[\vec{n}, d, H_{alt}]$  in  $C_{Ebenen}$  existiert
        füge die konvexe Hülle  $H_{alt}$  zu  $P$  hinzu
    bestimme die konvexe Hülle  $H_{neu}$ 
    trianguliere  $H_{neu}$  zu  $T_{Ebene}$ 
     $T_{Gesamt} += T_{Ebene}$ 
     $C_{Ebenen} += [\vec{n}, d, H_{neu}]$ 
     $C_{Punkte} - P$ 

```

---

Listing 4.1: Planare Echtzeitrekonstruktion

$\vec{n}$  und dem Abstand zum Koordinatenursprung  $d$ , lässt sich dabei durch die Gleichung 4.4 bestimmen.

$$\vec{n} = \left\| \vec{AB} \times \vec{AC} \right\| \quad d = \vec{A} \cdot \vec{n} \quad (4.4)$$

Um zu ermitteln, ob ein weiterer Punkt  $\vec{P}$  aus der Pointcloud eines Clusters die gefundene Ebene  $[\vec{n}, d]$  unterstützt, wird die kürzeste Distanz  $d_P$  zwischen Punkt und Ebene wie in Gleichung 4.5 ermittelt. Ein entsprechender Toleranzwert für die Distanz  $d_{min}$ , im gezeigten RANSAC Algorithmus  $e$  genannt, wird später bei der Umsetzung abhängig von der Ungenauigkeit des Tiefensensors gewählt.

$$d_P = \vec{n} \cdot \vec{P} - d \quad support_{d_P} = d_P < d_{min} \quad (4.5)$$

Um das finale Modell der Ebene zu bestimmen, wie im ursprünglichen RANSAC Algorithmus in Punkt 7. aus Listing 4.1 beschrieben, und somit die Varianz des Abstands der Punkte zur Ebene zu minimieren, wird mithilfe der unterstützenden Punkte  $P_s = [x, y, z]$  eine lineare Regression durchgeführt. Diese mittelt ein Ebenenmodell  $E = [\vec{n}, d]$  aus den zuvor ermittelten Punkten mithilfe der Methode der kleinsten Quadrate. Hoppe u. a. (1992) nutzen dazu für eine ähnliche Problemstellung die Eigenwert-Dekomposition der

Kovarianzmatrix der Punkte  $P_s$  zum Zentrum  $\vec{p}_c$ . In Gleichung 4.6 wird das Zentrum aus den unterstützenden Punkten  $P_s$  bestimmt. Gleichung 4.7 zeigt die Bestimmung der Kovarianzmatrix  $CV$  im Bezug zum Zentroid, in der  $\otimes$  für das dyadische Produkt<sup>1</sup> steht.

$$\vec{p}_c = \frac{\sum_{n=0}^{|P|} \vec{p}_n}{|P|} \quad (4.6)$$

$$CV = \sum_{n=0}^{|P|} (\vec{p}_n - \vec{p}_c) \otimes (\vec{p}_n - \vec{p}_c) \quad (4.7)$$

Wendet man nun auf der Kovarianzmatrix  $CV$  die Eigenwert-Dekomposition an, erhält man die Normale  $\vec{n}$  aus dem Eigenvektor  $\|\vec{v}_i\|$  mit dem kleinsten Eigenwert  $\lambda_i$ . Somit würde bei  $\lambda_1 \geq \lambda_2 \geq \lambda_3$  die Zuweisung  $\vec{n} = \|\vec{v}_3\|$  folgen. Die Distanz zum Ursprung  $d$  entspricht dem Kreuzprodukt aus dem Zentroiden  $\vec{p}_c$  und der neu gewonnen Normalen  $\vec{n}$ . (Hoppe u. a., 1992)

#### 4.2.2 Bestimmung der Ebenenausbreitung

Nachdem die Ebene und die korrespondierenden Punkte zur Ebene gefunden wurden, muss noch die Ausbreitung der Fläche bestimmt werden, da die Ebene in Hesse Normalform lediglich die Position  $\vec{n} * d$  und Ausrichtung  $\vec{n}$  festhält. Trevor u. a. (2011) nutzt hierfür die konvexe Hülle der korrespondierenden Punkte und trianguliert diese. Wie diese Triangulation genau bestimmt wird, wurde von Trevor u. a. (2011) nicht beschrieben.

Um diese Bestimmung performant umsetzen zu können, kann man sich die Eigenschaft der Ebene zunutze machen und die dreidimensionalen Punkte durch Parallelprojektion als zweidimensionale Punkte auf die gefundene Ebene projizieren. Denn die Triangulation ist nach dem Erhalten der zweidimensionalen konvexen Hülle, wie im Listing 4.2 beschrieben, direkt bestimbar. Nach der Triangulation können die Ecken der gefundenen Polygone jeweils zurück projiziert werden. Die Gleichungen 4.8 und 4.9 bilden die Projektion

---

<sup>1</sup>Wenn  $\vec{a}$  und  $\vec{b}$  die Komponenten  $a_i$  und  $b_j$  beinhalten, resultiert aus  $\vec{a} \otimes \vec{b}$  eine Matrix mit den Komponenten  $a_i b_j$  an der  $ij$  Position. (Hoppe u. a., 1992)

der Punkte wobei  $R_{\vec{n} \rightarrow \vec{z}}$  der Rotationsmatrix zwischen dem Normalenvektor  $\vec{n}$  und der Z-Achse  $\vec{z}$  entspricht.

$$p_{2d} = (p_{3d} - (\vec{n} * d)) * R_{\vec{n} \rightarrow \vec{z}} \quad (4.8)$$

$$p_{3d} = (p_{2d} * R_{\vec{n} \rightarrow \vec{z}}^{-1}) + (\vec{n} * d) \quad (4.9)$$

**Eingabe:** Unterstützende Ebenenpunkte aus RANSAC  $P$

Transformation  $R_{\vec{n} \rightarrow \vec{z}}$

**Ausgabe:** Polygone  $T_{Ebene}$

```

Projiziere alle Punkte aus  $P_s$  mit  $R_{\vec{n} \rightarrow \vec{z}}$  zu  $P_{2d}$ 
Bestimme die konvexe Hülle  $H$  aus  $P_{2d}$  mit Graham Scan
starte mit leerer Menge  $P_{2dmesh}$ 
für  $i$  von 0 bis  $|H| - 2$ 
    füge  $H_0$  zu  $P_{2dmesh}$  hinzu
    füge  $H_{i+1}$  zu  $P_{2dmesh}$  hinzu
    füge  $H_{i+2}$  zu  $P_{2dmesh}$  hinzu
Projiziere alle Punkte aus  $P_{2dmesh}$  mit  $R_{\vec{n} \rightarrow \vec{z}}^{-1}$  zu  $P_{3dmesh}$ 
```

---

Listing 4.2: Bestimmung der Ebenenausbreitung und Triangulation

In Abbildung 4.2 ist der Gesamtprozess für die Bestimmung der Ebenenausbreitung veranschaulicht. Nachdem eine Ebene mit RANSAC detektiert wurde, werden die korrespondierenden Punkte auf diese zweidimensionale Ebene projiziert. Daraufhin wird die konvexe Hülle gebildet und die Triangulation der Hülle bestimmt. Diese Polygon Eckpunkte werden danach wieder zurück in den dreidimensionalen Raum projiziert.

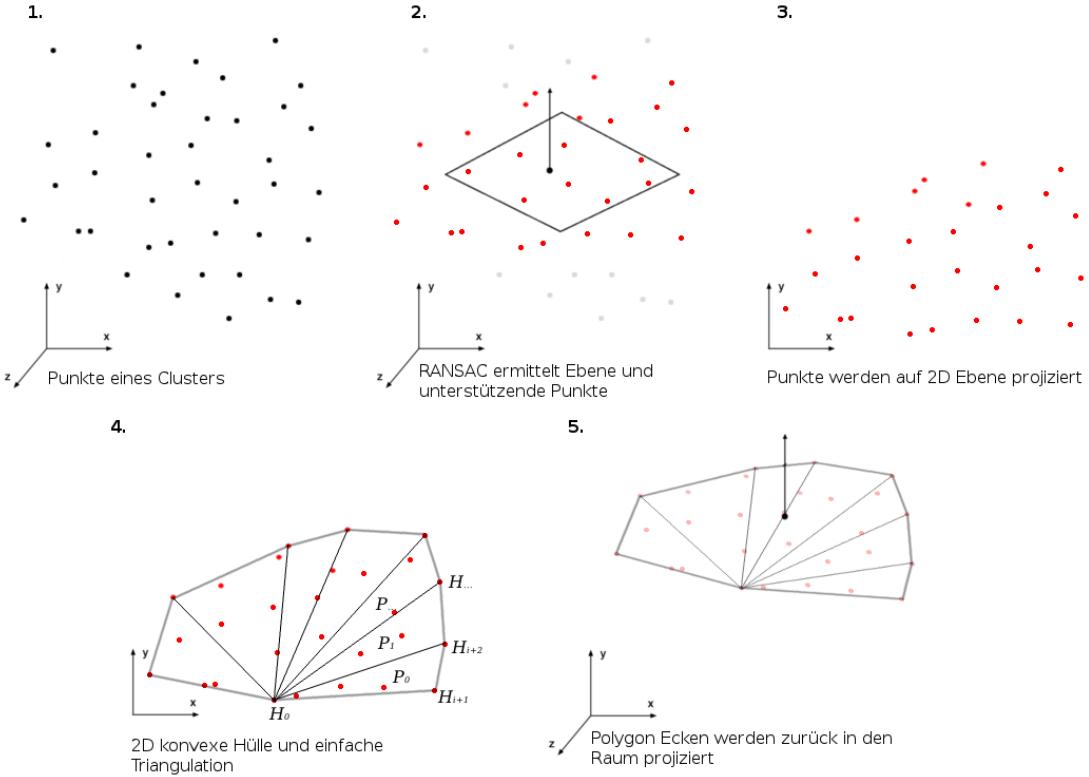


Abbildung 4.2: Veranschaulichung des Gesamtprozesses zur Bestimmung der Ebenenausbreitung durch RANSAC und der konvexen Hülle.

### 4.2.3 Clustering der aufgenommenen Punkte

Wie im Listing 4.1 zu erkennen, wird das zuvor beschriebene Vorgehen für die planare Rekonstruktion immer pro Cluster eines Cluster-Pools durchgeführt. Dadurch werden pro Durchgang des Algorithmus nur ein Bruchteil der gesammelten Punkte rekonstruiert, was wiederum eine Rekonstruktion in Echtzeit möglich macht. Außerdem verhindert das Clustering das Bilden von konvexen Hüllen über Ebenen, die in Zwischenbereichen nicht mit genügend Punkten unterstützt werden. Dieses Problem ist in Abbildung 4.3 links zu sehen, in welcher eine blaue Ebene rekonstruiert wird, die sich über einen Durchgang ohne vorhandene Punkte streckt.

Getestet wurden hier das K-Mean Clustering, Agglomeratives Clustering und einfaches räumliches Clustern mithilfe eines Octrees. Das K-Mean Clustering hat, wie in Abbildung 4.3 rechts zu erkennen, gute Ergebnisse für die

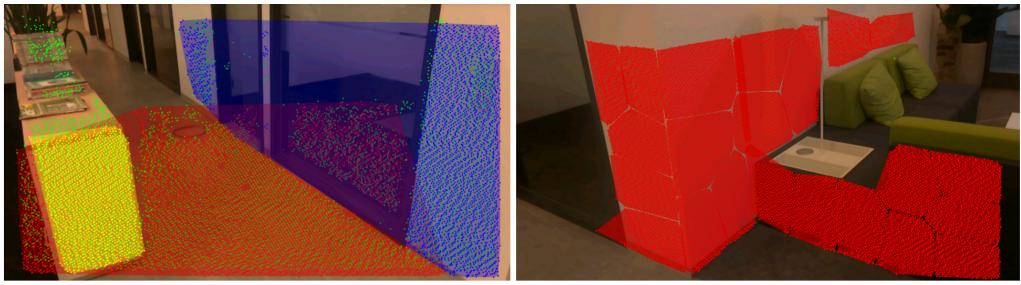


Abbildung 4.3: Links: Ebenenrekonstruktion ohne Clustering. Rechts: Rekonstruktion mit K-Mean Clustering.

Aufteilung einer Ebenen liefert, benötigt aber zuvor eine feste Anzahl von Clustern. Agglomeratives Clustering, getestet mit dem euklidischen Distanzmaß, würde zwar die Anzahl der Cluster dynamisch bestimmen, ist jedoch zu aufwändig für eine Echtzeitanwendung, wenn dieses auf die gesamten Messergebnisse durchgeführt wird. Ein Einsatz vom K-Mean oder Agglomerativen Clustering ist mangels Skalierbarkeit also nicht möglich.

Gute Ergebnisse liefert wiederum ein einfaches räumliches Clustern mit einem Octree, welcher die aufgenommenen Punkte direkt in Knoten des Baums zuweist. Das bietet zudem den Vorteil, dass diese Datenstruktur direkt als Speicherort der aufgenommenen Punkte und Ebenen dienen kann. Außerdem entspricht dies dem Vorgehen für die Anwendung von RANSAC auf Würfeln, welches von Yang u. Förstner (2010) beschrieben wurde.

### 4.3 Echtzeit Polygon Rekonstruktion

Das zweite rekonstruktionsbasierte Überlagerungsverfahren bezieht sich in diesem Kapitel auf den aktuellen Forschungsstand der Echtzeitrekonstruktion und soll in den folgenden Absätzen näher beschrieben werden. Die Echtzeit 3D Rekonstruktion ist bereits ein etabliertes Forschungsgebiet in der Computergrafik und gewinnt, aufgrund von kostengünstigen Consumer Tiefensensoren, wie die Microsoft Kinect<sup>2</sup>, Xtion<sup>3</sup> oder Structure<sup>4</sup>, zunehmend an Bedeutung. Dabei liegt der Fokus zunehmend auf der Echtzeitrekonstruktion, da diese

<sup>2</sup>Microsoft Kinect - <https://dev.windows.com/en-us/kinect> (13.03.16)

<sup>3</sup>Ausus Xtion - [https://www.asus.com/de/3D-Sensor/Xtion\\_PRO\\_LIVE/](https://www.asus.com/de/3D-Sensor/Xtion_PRO_LIVE/) (13.03.16)

<sup>4</sup>Occipital Structure - <http://structure.io/> (13.03.16)

Geräte in der Lage sind, Tiefeninformationen, zwar mit leichten Messfehlern, aber in Echtzeit, zu liefern. Nießner u. a. (2013) erwähnen an dieser Stelle zudem den möglichen Einsatz für Augmented Reality:

„The ability to obtain reconstructions in real-time opens up various interactive applications including: augmented reality (AR) where real-world geometry can be fused with 3D graphics and rendered live to the user; ...“ (Nießner u. a., 2013)

Die Herausforderung in der Echtzeitrekonstruktion liegt dabei in der möglichst performanten Fusion von mehreren überlagernden Tiefenbildern, welche aus verschiedenen Betrachtungswinkeln aufgenommen werden. Hieraus soll eine möglichst detaillierte Repräsentation der echten Umgebung generiert werden, welche sich im Idealfall stetig verbessert. Diese Problemstellung unterscheidet sich dabei von herkömmlichen Rekonstruktionsverfahren wie dem von Hoppe u. a. (1992) und der Poisson Rekonstruktion von Kazhdan u. a. (2006). Aktuelle Verfahren nutzen verschiedenste optimierte Datenstrukturen, welche zudem durch den Einsatz von entsprechenden GPU Implementierungen beschleunigt und in Echtzeit angewendet werden können. Hier spielt die Gegenüberstellung von Detailgrad, der Skalierung und Geschwindigkeit stets eine große Rolle. (Nießner u. a., 2013)

Bekannte Verfahren wie KinectFusion (Newcombe u. a., 2011), ein SLAM Verfahren von Bylow u. a. (2013) oder DynamicFusion (Newcombe u. a., 2015) nutzen die „Truncated Signed Distance Function“, kurz TSDF, zur Speicherung und Migration der Oberflächeninformation mehrerer Depth Maps. Das Verfahren von Nießner u. a. (2013) erweitert diesen Ansatz mit einem effizienten räumlichen Hashingverfahren, um die Zugriffszeiten und Speicherverbrauch zu minimieren. Darüber hinaus nimmt das Verfahren Chisel von (Klingensmith u. a., 2015) diese Vorteile auf und kombiniert TSDF mit „visual-inertial odometry“, der Trackingtechnologie von Project Tango. In den folgenden Absätzen werden die Mechanismen hinter TSDF, dem räumlichen Hashing und den Vorteilen von Chisel näher erläutert. Außerdem soll auch noch auf das Rendering der TSDF Oberfläche durch Marching Cubes eingegangen werden, welches in Chisel verwendet wird.

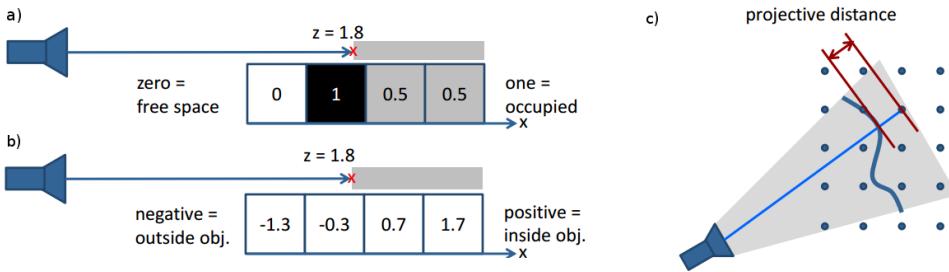


Abbildung 4.4: a) Beispielhafte Voxel Füllung von Occupancy Maps; b) Beispielhafte Voxel Füllung durch TSDF; c) Exemplarische 2D Darstellung der Oberfläche mit entsprechenden Strahlensatz für die TSDF. Übernommen von Sturm (2014)

#### 4.3.1 Truncated Signed Distance Function

Bei der von Curless u. Levoy (1996) vorgestellten räumlichen Repräsentation von Oberflächen, Truncated Signed Distance Function (TSDF), wird der Raum in Voxel einer gewünschten Auflösung unterteilt. Anders als bei Occupancy Maps, in denen die Voxel als sichtbar oder unsichtbar binär markiert werden, werden bei TSDF in den Voxeln die jeweiligen Entferungen zur nächstgelegenen Oberfläche angegeben. Wichtig dabei ist das Vorzeichen, welches angibt, ob sich ein Voxel innerhalb oder außerhalb eines Objektes befindet. Abbildung 4.4 zeigt unter a) die Ergebnisse mit Occupancy Maps und in b) die Voxel von TSDF. (Curless u. Levoy, 1996)

Gefüllt wird die Repräsentation durch die Depth Maps und die entsprechende Kameraposition, die im Fall von Project Tango durch Motion Tracking bereits gegeben ist. So wird für jede Tiefeninformation ein Strahl ausgehend von der Kameraposition generiert, der die durchgeschnittenen Voxel aktualisiert. Der Strahl ist dabei von der Länge begrenzt, um die, zu aktualisierenden Voxel klein zu halten und zudem keine Oberflächen zu aktualisieren, die sich weiter hinter der gefundenen Oberfläche befinden. Dieses Vorgehen ist in Abbildung 4.4 c) zu erkennen. (Sturm, 2014)

Der Vorteil dieser Repräsentation liegt darin, dass die konkreten Oberflächeninformationen, anders als bei der Diskretisierung von Occupancy Maps, nicht verloren gehen. Das heißt, dass trotz einer größeren Voxel Struktur stets der Nulldurchgang rekonstruiert werden kann. Neben der Entfernung zur nächsten Oberfläche wird zusätzlich noch ein Gewichtungswert in jedem Voxel

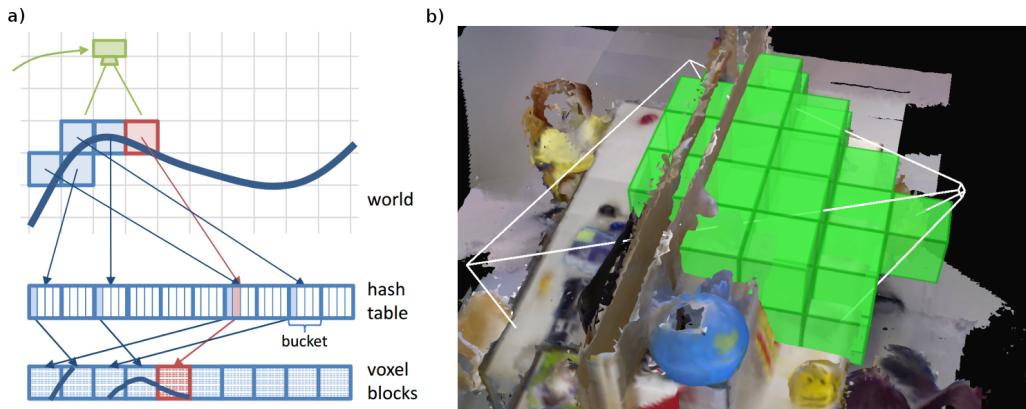
gespeichert. Das ermöglicht es, leichtes Rauschen durch eine einfache Mittelung mehrerer Messergebnisse der Oberfläche zu unterdrücken. Außerdem kann somit eine stetige Optimierung der Oberfläche vorgenommen werden. (Sturm, 2014)

Hoppe u. a. (1992) nutzen in ihrem offline Rekonstruktionsverfahren auch die hier beschriebene TSDF. Jedoch bestimmen sie für jeden festgehaltenen Punkt der Pointcloud die umliegenden Nachbarn, um eine Tangentialebene zu ermitteln, von der aus die auf der Normalen liegenden Voxel mit der entsprechenden Distanz aktualisiert werden. Für die Echtzeitrekonstruktion ist dieses Vorgehen jedoch zu aufwändig. Hier werden die Voxel nicht anhand der exakten euklidischen Distanz aktualisiert, sondern es wird mithilfe des Raycastings, ausgehend von der Tiefenkamera, eine projizierte Distanz als Approximation verwendet (Sturm, 2014). Später bei der detaillierten Beschreibung von Chisel (Kapitel 4.3.4) wird das Vorgehen auch nochmal grafisch in Abbildung 4.7 erläutert.

### 4.3.2 Spatial Hashing

Das Problem der Echtzeitrekonstruktion ist wie bereits angesprochen der Kompromiss zwischen dem Detailgrad, der Skalierung der zu rekonstruierenden Szene und der Performance der Rekonstruktion. Auch die TSDF-Repräsentation ist speicherintensiv und benötigt für die zu scannende Szene reservierten Speicher, der auf mobilen Endgeräten nur begrenzt verfügbar ist. Daher muss für größere Rekonstruktionen oder Rekonstruktionen unbekannter Größe ein dynamischer Ansatz gefunden werden, Speicher zu verwalten. Klingensmith u. a. (2015) erwähnt dazu, dass einige Verfahren Octrees einsetzen, die zwar äußerst dynamisch sind, jedoch einen deutlichen Nachteil hinsichtlich der Zugriffszeiten auf die Voxel bergen.

Nießner u. a. (2013) führen daher eine 2-Ebenen-Struktur ein, die auf der zweiten Ebene eine Menge von Voxel räumlich zusammenfassen. Diese werden hier Chunks genannt und sind Abbildung 4.5 links zu sehen. Auf der ersten Ebene können diese Chunks in einer Hashtabelle räumlich mit einer Hashfunktion identifiziert werden. Das ermöglicht somit einen nahezu direkten Zugriff auf räumliche Voxel und erlaubt zudem die Möglichkeit



Chunks dynamisch zu allozieren. Als Hash der Chunkposition  $x, y$  und  $z$  wird die folgende Hashfunktion aus Gleichung 4.10 verwendet. Bei den Variablen  $p_1$ ,  $p_2$  und  $p_3$  handelt es sich um willkürlich hohe Primzahlen und  $n$  entspricht der Größe der Hashtabelle. Da Kollisionen in der Hashtabelle nicht vollkommen ausgeschlossen werden können, werden die Chunks gegebenenfalls in der Hashtabelle verkettet.

$$H(x, y, z) = (x * p_1 \oplus y * p_2 \oplus z * p_3) \mod n \quad (4.10)$$

### 4.3.3 Marching Cubes

Die meisten Echtzeitrekonstruktionen durch TSDF wie KinectFusion sind GPU-Umsetzungen. Diese haben daher die Möglichkeit ein hardwarebeschleunigtes Rendering durch Raycasting durchzuführen. Das Verfahren Chisel von Klingensmith u. a. (2015), welches eine reine CPU Umsetzung ist, nutzt hingegen einen indirekten Weg zum Rendering durch die Marching Cubes Triangulation.

Marching Cubes nach Lorensen u. Cline (1987) ist ein Algorithmus um aus einer, als Voxel repräsentierten, Isofläche Polygone zu bestimmen, die dieser Fläche möglichst nahe kommen. Hierzu werden zu jedem Voxel die Ecken  $v_1$  bis  $v_8$  anhand der Nachbarvoxel und des Distanzwertes untersucht,

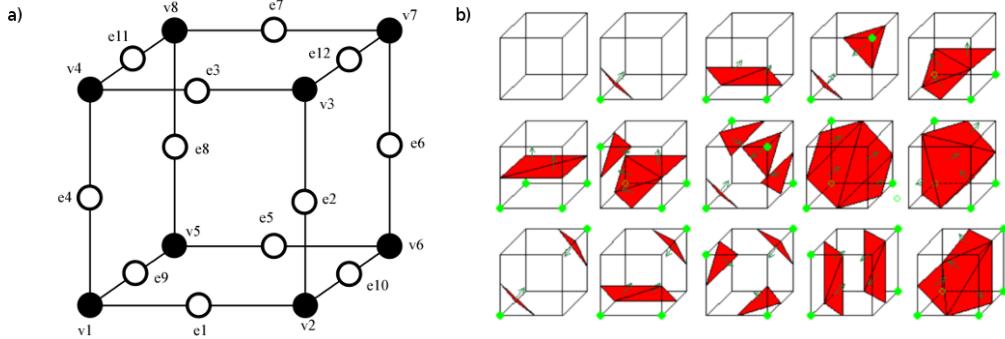


Abbildung 4.6: a) Marching Cubes Voxel Repräsentation mit den Ecken und den Kantenschnittpunkten b) Die 15 möglichen 3D Polygon Varianten. Übernommen von Raphaël u. Karen (2002)

ob sie innerhalb oder außerhalb eines Objektes liegen. Zusätzlich werden zu jeder Kante auf dem Voxel, wenn ein Schnitt der Isofläche existiert, die Schnittpunkte auf den Kanten  $e_1$  bis  $e_{12}$  bestimmt. Abbildung 4.6 a) zeigt die Ecken und Kantenschnittpunkte eines Voxels.

Je nach binärer Gewichtung der Ecken können hiernach aus einem Katalog von 255 Varianten die Polygone nachgeschlagen werden. Inhalt des Katalogs sind die Indizes der Kantenschnittpunkte  $p_{e1}$  bis  $p_{e12}$ , aus denen Polygone generiert werden können. Alle diese 255 Varianten können auf 15 verschiedene Fälle zurückgeführt werden, die sich nur in Rotation oder Symmetrie unterscheiden. Die 15 Varianten sind in Abbildung 4.6 b) zu finden. (Raphaël u. Karen, 2002)

#### 4.3.4 Chisel mit Space Carving

Das bereits erwähnte Verfahren Chisel von Klingensmith u. a. (2015) verwendet alle zuvor erwähnten Techniken der TSDF, Spatial Hashing und der Marching Cubes Überführung. Sie sprechen dabei von einem „dynamic spatial-hashed truncated distance field“. Das für den mobilen Einsatz optimierte Verfahren ist in der Lage eine Echtzeitrekonstruktion von Räumen von bis zu  $300m^2$  mit einem Detailgrad von zwei bis drei Zentimetern zu erstellen. Zudem können neben Tiefeninformationen auch gefärbte Pointclouds verarbeitet werden, wodurch ein gefärbtes Mesh durch Marching Cubes generiert werden kann.

Die Farbinformationen sind jedoch auf die Voxel Auflösung des Verfahrens beschränkt. (Klingensmith u. a., 2015)

Zusätzlich erweitern sie den TSDF Algorithmus um die „space carving“ Funktionalität. Sie betrachten dabei den Strahl von der Kamera zur Oberfläche als eine Art Bedingung, in der alle durchstoßenden Voxel bis zur Oberfläche einen negativen Wert beinhalten müssen. Ist das nicht der Fall, so wird ein Voxel außerhalb der inneren Begrenzung auf den leeren Ursprungswert gesetzt. Der Pseudocode in Listing 4.3 sowie die Abbildung 4.7 erläutern dieses Verhalten. Diese Verbesserung führt dazu, dass die Rekonstruktion bei stark rauschenden Tiefeninformationen, besonders an Objektkanten, deutlich verbessert wird. Außerdem ist das Verfahren hiermit in der Lage dynamische Änderungen in der Umgebung zu detektieren und neue Oberflächen entsprechend zu aktualisieren. So beeinflusst, als Beispiel, sich eine im Bild bewegende Personen nur kurz die Voxel der TSDF. (Klingensmith u. a., 2015)

```

Eingabe: Pointcloud  $C$ , Kameratransformation  $P_{cam}$ ,
          Strahlenbegrenzung  $t$ 

für jeden Tiefenwert  $\vec{p}$  aus  $C$ 
    bestimme die Oberflächenposition  $\vec{z}$  aus  $\vec{p}$  und  $P_{cam}$ 
    bestimme einen Strahl  $\vec{r}$  aus  $\vec{z}$  und  $P_{cam}$ 
    bestimme den Begrenzungsbereich  $t_{vor}, t_{nach}$  mit  $t$  um  $\vec{z}$  auf  $\vec{r}$ 
    # space carving
    für jeden Voxel  $v$  zwischen Kamera und  $t_{vor}$ 
        wenn die Distanz im Voxel negativ ist
            setze Voxel zurück
    # normale TSDF Bestimmung
    für jeden Voxel  $v$  zwischen  $t_{vor}$  und  $t_{nach}$ 
        bestimme die Voxeldistanz zu  $z$ 
        setze das Gewicht  $w$  des Voxels  $v$ 

```

---

Listing 4.3: Chisel TSDF Algorithmus

Neben space carving wurden zudem eine variable Strahlenbegrenzung und Gewichtung der Voxel, abhängig von der jeweils aufgenommenen Tiefe, implementiert. Diese Funktion berücksichtigt die Messungenauigkeiten des Tiefensorsors, die bei größerer Entfernung der Oberfläche zum Tiefensor zunehmen können. (Klingensmith u. a., 2015)

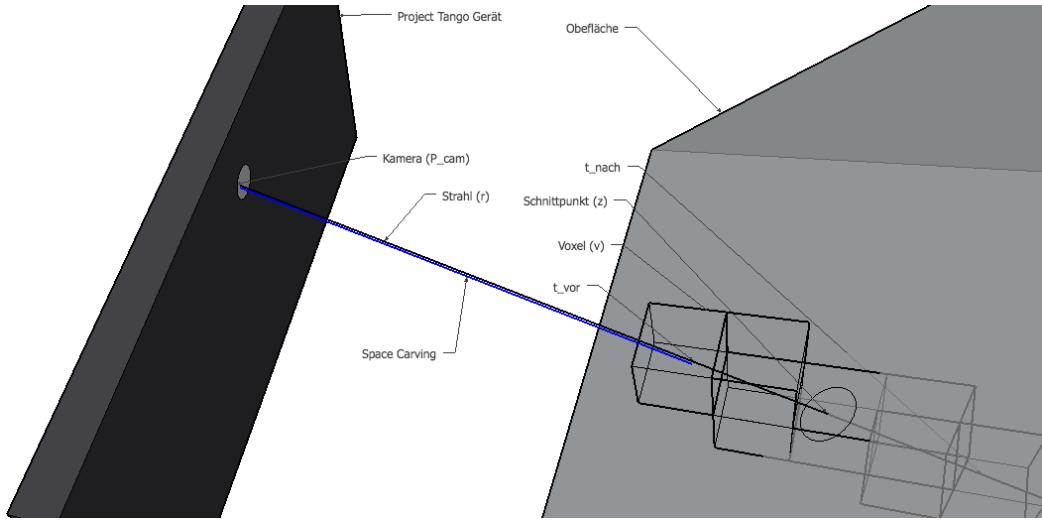


Abbildung 4.7: Exemplarische Visualisierung des Strahls  $\vec{r}$  und der zu aktualisierenden Voxel  $v$  im TSDF Algorithmus. Links ist hier das Project Tango Gerät zu erkennen, welches Tiefeninformationen der Oberfläche eines Objekts rechts aufgenommen hat.

## 4.4 Tiefenanpassungen durch Farbbilder

Aus allen zuvor beschriebenen Verfahren werden letztendlich Tiefeninformationen, in Form von geometrischen Primitiven oder Punkten im Raum gewonnen. Diese werden passend zur aktuellen Kameraposition als Tiefenbild gerendert und füllen den Z-Buffer für entsprechende Aussparungen bei der Überdeckung virtueller Objekte. Auf Grund von Sensorungenauigkeiten und größeren Auflösungen der Rekonstruktionsverfahren können dabei fehlerhafte Tiefeninformationen in den Z-Buffer gelangen, die zu Fehlern bei der Bestimmung der Überdeckung führen können. Dieses Problem ist am Beispiel der Pointcloud Projektion aus Kapitel 4.1 in Abbildung 4.8 zu erkennen.

Die Reduktion von Ungenauigkeiten im Tiefenbild könnte durch einen einfachen Weichzeichner erreicht werden. Dieser würde jedoch die Kanten im Farbbild nicht berücksichtigen und somit fehlerhafte Tiefengradienten an den Kanten erzeugen und einen durchaus größeren Fehler generieren. Newcombe u. a. (2011) wenden einen sogenannten „Bilateralen Filter“ in ihrem KinectFusion Rekonstruktionsverfahren an, bevor sie die Tiefeninformationen in die TSDF Repräsentation einfließen lassen. Dieser Filter von Tomasi u. Manduchi (1998) ermöglicht das Weichzeichnen ohne dabei die Kanten im

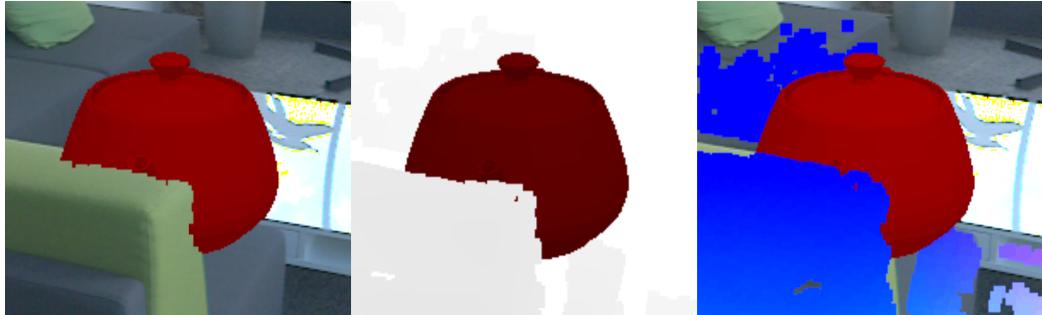


Abbildung 4.8: Überdeckung mit einfacher Pointcloud Projektion. Links: Resultat der Überdeckung. Mitte: Darstellung des Z-Buffers mit dem Ausschluss des virtuellen Objekts. Rechts: Darstellung der Pointcloud.

Bild zu übergehen, bezieht sich jedoch nur auf das selbe Bild, auf dem der Filter angewendet wird.

Liu u. a. (2012) hingegen wenden einen sogenannten „Guided Filter“ in ihrem Verfahren zur Optimierung der Tiefeninformationen für Kinect ähnliche Sensoren auf das Tiefenbild an. Dieser Filter von He u. a. (2010) ist in der Lage, auf Grundlage eines anderen Leitbildes eine Weichzeichnung durchzuführen, ohne dabei die Kanten des Leitbildes zu überschreiten. Auch wenn Petschnigg u. a. (2004) eine Erweiterung, den Joint Bilateral Filter, vorstellen, der auf Basis eines anderen Leitbildes eine Weichzeichnung ohne Kantenüberschreitung ermöglicht, bietet der Guided Filter eine deutlich bessere Performance. Außerdem verhindert der Guided Filter Fehlerartefakte im Resultat, die bei dem Bilateralen Filter an den Kanten auftreten können. (He u. a., 2010)

Ausgehend von der Eingangsgrafik  $p$ , einem Leitbild  $I$  und dem Ergebnisbild  $q$  wird das grundlegende Modell von dieser Art Filter mit der Gleichung 4.11 beschrieben. Diese Gleichung findet für jeden Pixel  $i$  in  $q$  eine gewichtete Summe über jeden Pixel  $j$  einer vordefinierten Ausschnittgröße.  $W_{ij}$  entspricht dabei dem Gewicht, welches für die jeweiligen Pixel  $p_j$  gilt. Bei dieser Faltung gilt üblicherweise  $\sum_j W_{ij}(I) = 1$  für  $\forall i \in [1 \dots |p|]$ . (He u. a., 2010)

$$q_i = \sum_j W_{ij}(I)p_j \quad (4.11)$$

Der Filterkern  $W_{ij}(I)$  des Guided Filter, zu finden in Gleichung 4.12, ist, wie auch beim bilateralen Filter, abhängig von einem Leitbild  $I$ , um die Gewichte entsprechend den Kanten des Leitbildes an der Position ermitteln zu können. Die Variablen  $\mu_k$  und  $\sigma_k^2$  beschreiben jeweils den Mittelwert und die Abweichung des Leitbildes im Bildausschnitt  $w_k$ .  $|w|$  entspricht der Pixelgröße des Ausschnitts. (He u. a., 2010)

$$W_{ij}(I) = \frac{1}{|w|^2} \sum_{k:(i,j) \in w_k} \left( 1 + \frac{(I_i - \mu_k)(I_j - \mu_k)}{\sigma_k^2 + \epsilon} \right) \quad (4.12)$$

Dieser Filterprozess wird auch als eine translationsabhängige Faltung bezeichnet, die üblicherweise aufwändig ist und dessen Berechnungsaufwand abhängig zur Filterkern Größe ( $|w|$ ) ist. He u. a. (2010) stellen jedoch noch eine andere Definition des Filters zur Verfügung, in denen alle Summen der Form  $\sum_{i \in w_k} f_i$  entsprechen und dadurch mit der Bildintegrationstechnik von Crow (1984) in  $O(N)$  gelöst werden können. Der Guided Filter wird in der letztendlichen Implementierung nach Gleichung 4.13 implementiert, in der die Koeffizienten  $\bar{a}_i$  und  $\bar{b}_i$  dem Mittelwert über  $a_k$  aus Gleichung 4.14 und  $b_k$  aus Gleichung 4.15 für jedes Fenster  $w_k$  entspricht. So wird auch  $\bar{p}_k$  durch  $\frac{1}{|w|} \sum_{i \in w_k} p_i$  berechnet.

$$q_i = \bar{a}_i I_i + \bar{b}_i \quad (4.13)$$

$$a_k = \frac{\frac{1}{|w|} \sum_{i \in w_k} I_i p_i - \mu_k \bar{p}_k}{\sigma_k^2 + \epsilon} \quad (4.14)$$

$$b_k = \bar{p}_k - a_k \mu_k \quad (4.15)$$

Der Parameter  $\epsilon$  reguliert im beschriebenen Filter von He u. a. (2010) welcher Bildanteil als beizubehaltende Kante im resultierenden Bild gewertet werden soll und somit stärker oder schwächer in der Gewichtung  $W_{ij}$  Einfluss nimmt. Neben diesem Regulierungsfaktor ist auch die Wahl des Radius  $r$  für den Ausschnitt  $w_k$  als Eingabe für diesen Filter wichtig. Der Radius wirkt sich laut He u. a. (2010) jedoch nicht wie beim bilateralen Filter auf die Laufzeit des Filters aus.

„One more advantage of the guided filter over the bilateral filter is that it automatically has an  $O(N)$  time exact algorithm.  $O(N)$  time implies that the time complexity is independent of the window radius  $r$ , so we are free to use arbitrary kernel sizes in the applications.“ (He u. a., 2010)

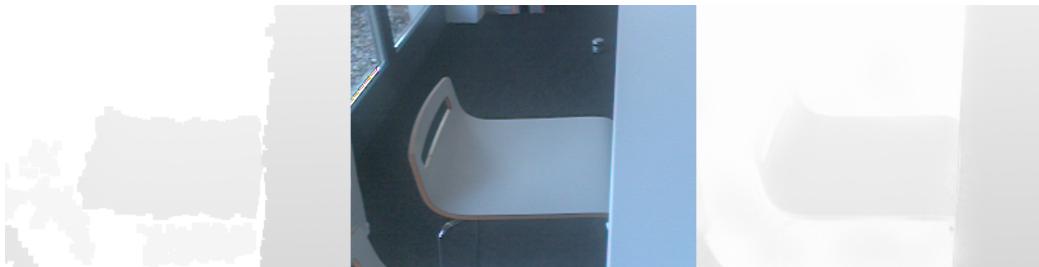


Abbildung 4.9: Guided Filter Anwendungsbeispiel. Das Tiefenbild links ergibt durch den Guided Filter mit dem Leitbild in der Mitte das Ergebnis im rechten Bild.

Mit einer Komplexität von  $O(N)$  findet dieser Filter erfolgreich Anwendung in verschiedensten Bereichen. Er wird zum Beispiel zur Rauschunterdrückung, dem Weichzeichnen oder Verstärken von Details, zur HDR Kompression, dem Entfernen von matteneigenschaften oder, wie in diesem Fall, zum zusammengeführten Anreichern von Bildinformationen verwendet (He u. a., 2010). Angewendet auf das ermittelte Tiefenbild kann dieser Guided Filter, mit dem jeweiligen RGB Bild als Leitbild, ein Rauschen eliminieren und die Kanten der Tiefeninformationen durch einen entsprechend groß gewählten Fensterradius  $r$  und Regulierungsfaktor  $\epsilon$ , an die Kanten der Kameraaufnahme angleichen (Liu u. a., 2012). Ein Beispiel für eine erfolgreiche Anwendung dieses Filters ist in Abbildung 4.9 zu sehen.

# Kapitel 5

## Implementierung

Dieses Kapitel widmet sich der Umsetzung der in Kapitel 4 beschriebenen Verfahren. Hierfür wurden im Laufe dieser Arbeit verschiedene Prototypen entwickelt, die jeweils als Proof of Concept für die einzelnen Verfahren galten. Ziel dieser Umsetzung ist es, einen einzigen Prototypen in Form einer Android Anwendung auf dem Project Tango Gerät zu entwickeln, die den gesamten Funktionsumfang beinhaltet, welcher in Kapitel 5.1 zusammengefasst wird.

Zudem muss ein Programmfluss geschaffen werden, der es ermöglicht, vor dem Rendern in die Daten des Tiefenbuffers mit eigenen Implementierungen eingreifen zu können, um eine Filterung zu ermöglichen. Wie das technisch ermöglicht wird und auf welcher Basis die Anwendung implementiert wird, beschreibt das Kapitel 5.2. Hiernach wird die Umsetzung der jeweiligen Verfahren näher erläutert.

### 5.1 Finaler Prototyp

Der finale Prototyp soll, wie bereits erwähnt, alle zuvor beschriebenen Verfahren zur Realisierung von Überlagerungen in einer Augmented Reality Szene beinhalten. Also muss zunächst eine einfache AR Szene geschaffen werden, in der eine virtuelle Kamera existiert, die den intrinsischen und extrinsischen Eigenschaften der realen Project Tango Kamera zu jeder Zeit entsprechen. Außerdem muss das aktuelle Farbbild der RGB Kamera in der Szene im

Hintergrund dargestellt werden. Für diese Aufgaben existieren, wie bereits in Kapitel 2.2 erwähnt, Schnittstellen, die diese Informationen liefern.

Um eine reale Überdeckung sinnvoll testen zu können, benötigen wir zudem ein virtuelles Objekt in der Szene. Dieses sollte im Idealfall nicht zu einfach gestaltet sein, damit die Verfahren anhand praxisnaher Gegebenheiten verglichen werden können. Zu diesem Zweck sollen Objekte in die App geladen werden können, die im Forschungsbereich der Computergraphik typischerweise eingesetzt werden. Typische Modelle sind zum Beispiel der „Utah Teapot“, „Stanford Bunny“ oder „Blenders Suzanne“<sup>1</sup>. Eines dieser Modelle soll in die Szene geladen werden und es soll die Möglichkeit gegeben sein, dass das Objekt flexibel positioniert werden kann. Um das zu realisieren, wird der beschriebene Raypicking Mechanismus für die Auswahlgeste umgesetzt.

Um die Ergebnisse der realen Überlagerung einfach gegenüberstellen zu können, sollen die beschriebenen Tiefenbildgenerierenden Verfahren flexibel im Betrieb ausgetauscht werden. Dazu gehört das Rendering der Pointcloud Projektion, die TSDF Rekonstruktion durch Chisel und die Ebenenrekonstruktion. Außerdem soll das Filtern des Tiefenbildes mit Hilfe des Guided Filters optional zu jeder Zeit möglich sein. Hilfreich wäre es zudem, die Einstellungen des Guided Filters flexibel anpassen zu können. Wie diese Verfahren und der flexible Austausch umgesetzt werden, wird in den folgenden Kapiteln näher beschrieben.

## 5.2 Technische Umsetzung und Struktur

Wie bereits erwähnt, basiert das Project Tango System auf Googles Android Betriebssystem. Dies ermöglicht es Anwendungen mit bestehenden und bewährten Technologien wie OpenGL, Rajawali<sup>2</sup> oder der Unity Engine entwickeln zu können. Project Tango bietet hierfür drei verschiedene Schnittstellen, in C/C++, Java und Unity (Mono Framework in C#), um auf die Sensordaten in verschiedenen Umgebungen zugreifen zu können. Im Laufe dieser Arbeit wurden alle Schnittstellen mit verschiedensten prototypischen Entwicklungen getestet.

---

<sup>1</sup>List of common 3D test models - <https://goo.gl/MsOtSW> (26.02.16)

<sup>2</sup>Android OpenGL ES 2.0/3.0 Java Engine - <https://goo.gl/r9Ohdj> (27.02.16)

Der finale Prototyp wurde letztendlich in C/C++ entwickelt und basiert auf dem Android NDK<sup>3</sup>. Dabei greifen die anderen, höher angesiedelten Schnittstellen auf genau die selbe native Implementierung zurück, um sie in Java und Unity zur Verfügung zu stellen. Außerdem ermöglicht die native Entwicklung, neben Performancevorteilen, den vollen Zugriff auf OpenGL Mechanismen, die von Rajawali (der OpenGL Java Abstraktion) gegebenenfalls ausgeschlossen sind.

Das Project Tango Team stellt für die native Entwicklung die eigentliche Tango Schnittstellen Bibliothek<sup>4</sup>, eine Support-Bibliothek<sup>5</sup> und eine einfache Kapselung für OpenGL Anwendungen mit dem Namen TangoGL<sup>6</sup> bereit. Die Support-Bibliothek bietet verschiedene Hilfsfunktionen zur Datenverarbeitung und Allokation. TangoGL wiederum erleichtert den Einstieg in die OpenGL Entwicklung und übernimmt die grundlegende Struktur und Interoperabilität zu Project Tango. Zum Beispiel gibt es Methoden, um Tango Positionsdaten in eine Translationsmatrix umrechnen zu können oder Klassen, die das Rendern der aktuellen RGB Kamera Textur übernehmen.

Abbildung 5.1 zeigt grob den strukturellen Aufbau der Android Applikation. Der obere Teil der Grafik bezieht sich dabei auf den in Java implementierten Teil, der die Nutzeroberfläche, ihre Interaktion und den Renderingcanvas beinhaltet. Die Activity stellt jedoch nur einen kleinen Teil der Anwendung dar, denn alle Interaktionen und Ereignisse werden über ein Java Native Interface (JNI) zum nativen Teil der Anwendung geleitet, welcher die Ansprache der Schnittstellen, die Prozesslogik und das Rendering selbst beinhaltet.

Die Hauptklasse „ARApp“ in der Grafik widmet sich in der Anwendung nur der Anreicherung und Weiterleitung von JNI Informationen und der Ansprache der Project Tango Schnittstelle. Kern der Anwendung ist die „Scene“ Klasse, welche die Sensorinformationen an das entsprechend aktive Verfahren zur Tiefenbildgenerierung weiterreicht. So wird zum Beispiel die Pointcloud an das Chisel, Pointcloud oder Plane Drawable weitergereicht, damit sie ein aktualisiertes Tiefenbild rendern oder die Rekonstruktion anreichern können. Auch das Farbbild der Kamera gelangt über die Scene zum RGB Drawable,

---

<sup>3</sup>Android Native Development Kit - <http://goo.gl/ananZT> (27.02.16)

<sup>4</sup>Project Tango C API - <https:// goo.gl/lbBfAp> (27.02.16)

<sup>5</sup>Project Tango C Suppport API - <https:// goo.gl/VGyeKm> (27.02.16)

<sup>6</sup>TangoGL Repository - <https:// goo.gl/ymDCsJ> (27.02.16)

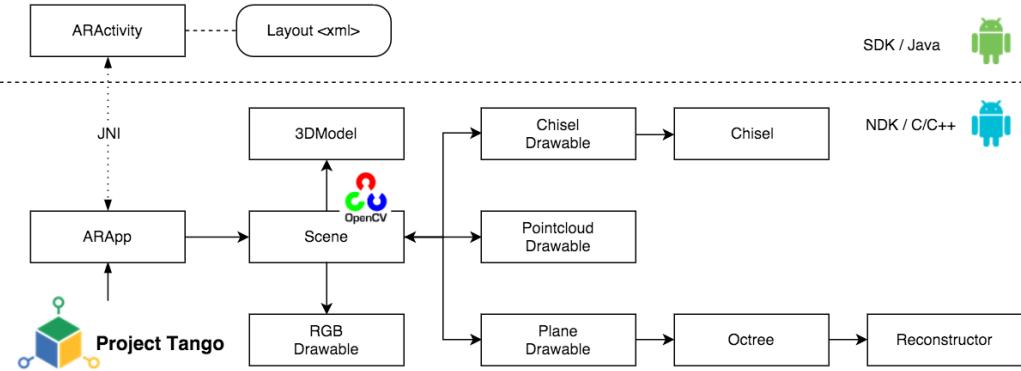


Abbildung 5.1: Struktureller Aufbau des Prototypen

welches letztendlich als eine texturierte Fläche am Ende des Kamera Frustum dargestellt wird. Die Szene selbst ermöglicht durch den Einsatz von OpenCV<sup>7</sup> den optionalen Filter Prozess durch den Guided Filter. Um das zu ermöglichen wird ein OpenGL Framebuffer eingesetzt. Die Abbildung 5.2 zeigt hierzu das Vorgehen beim Rendering der Szene mit Hilfe des zusätzlich eingesetzten Framebuffer (TB). Um den Z-Buffer Ausschluss, also die letztendliche Überlagerung zu nutzen, muss der OpenGL Depth-Test aktiviert werden.

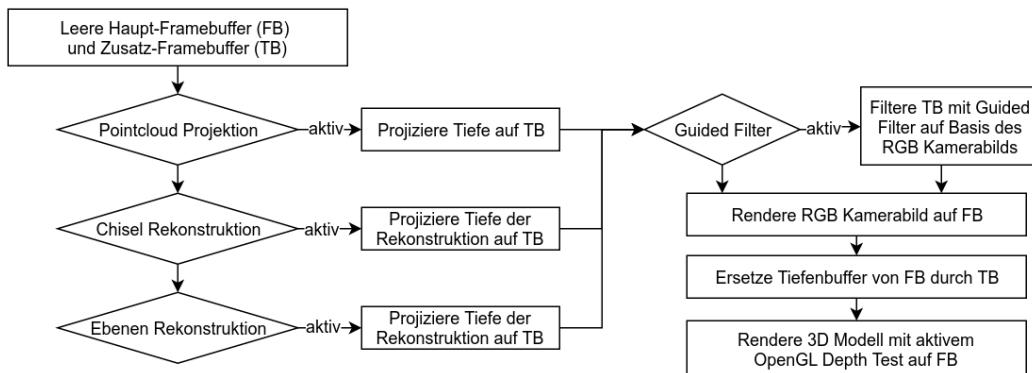


Abbildung 5.2: Prozessdiagramm für das Rendering der Szene

<sup>7</sup>Open Source Computer Vision - <http://opencv.org/> (26.03.16)

## 5.3 Umsetzung der Verfahren

### Tiefe aus der Pointcloud Projektion

Wie im Kapitel 4.1 erwähnt, müssen die Punkte der Project Tango Pointcloud auf die Bildebene projiziert werden und mit einer entsprechenden Tiefenfarbe und einem Radius auf den Tiefenpuffer gezeichnet werden. Dieser Schritt wurde auch bereits in Prototypen mit den angegebenen Gleichungen umgesetzt.

Nachdem der Proof of Concept jedoch fertig gestellt wurde, ist aufgefallen, dass OpenGL neben dem Rendering von Polygonen auch Primitiven, wie Punkte und Linien, unterstützt. Somit konnten die Punkte im finalen Prototyp einfach vor die Kamera positioniert werden und mit Hilfe der symbolischen Konstante „GL\_POINTS“ anstelle von „GL\_TRIANGLES“ gerendert werden. Zudem lässt sich durch einen entsprechenden Vertexshader die Größe der Punkte anpassen. Abbildung 5.3 zeigt links die optionale gefärbte Projektion auf der Bildebene und rechts das resultierende Tiefenbild.

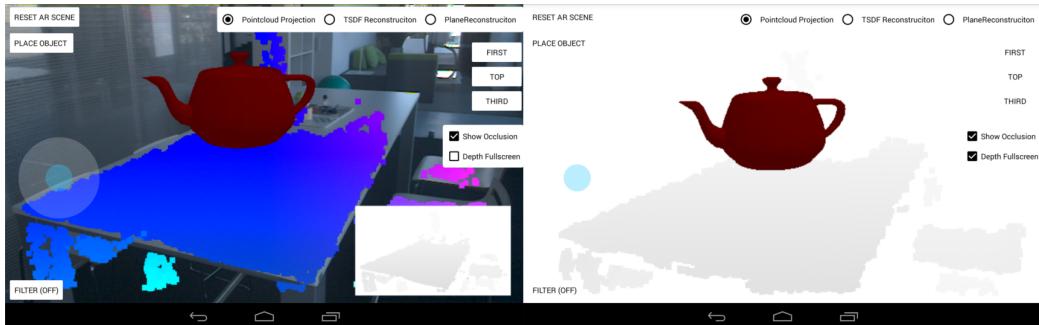


Abbildung 5.3: Pointcloud Projektion Prototyp. Links optionale Projektion auf der Bildebene. Rechts das resultierende Tiefenbild.

### Planare Rekonstruktion

Der erste Proof of Concept der planaren Rekonstruktion wurde zu Beginn dieser Arbeit auf Java Ebene implementiert und entwickelte sich nach und nach zu dem in Kapitel 4.2 beschriebenen Verfahren. Für die finale Umsetzung in dem nativen Prototypen mussten somit alle Algorithmen und Datenstrukturen neu in C/C++ umgesetzt werden. Begonnen wurde mit dem Octree, der in seinen tiefsten Zweigen die Menge aller aufgenommenen Punkte für den

jeweiligen Sektor und eine Instanz der „Reconstructor“ Klasse beinhaltet. Diese beinhaltet alle beschriebenen Algorithmen zur Ebenenrekonstruktion wie RANSAC, die lineare Regression und den Graham Scan.

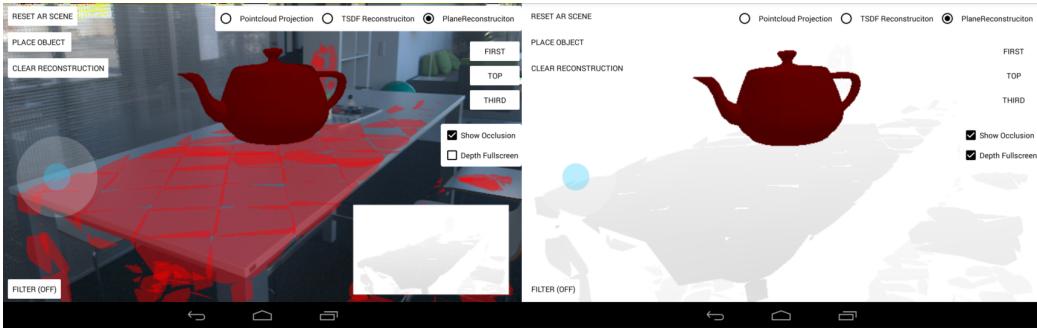


Abbildung 5.4: Planare Rekonstruktion Prototyp. Links optionale Projektion auf der Bildebene. Rechts das resultierende Tiefenbild.

Für die Berechnung mit Vektoren und Matrizen wurde wie auch im gesamten Projekt die OpenGL Mathematics Bibliothek (GLM)<sup>8</sup> verwendet. Sie bietet typische Primitiven mit entsprechenden Operationen für Berechnungen der linearen Algebra. Wie bereits beschrieben wird für die lineare Regression die Berechnung von Eigenvektoren mit ihren Eigenwerten benötigt. Diese Berechnung wird von GLM nicht unterstützt. Hier wurde die Eigen-Bibliothek<sup>9</sup> verwendet, die diese Operation für den Entwickler anbietet. Abbildung 5.4 zeigt die Ergebnisse der Ebenerkonstruktion links und der daraus resultierenden Tiefeninformation rechts.

## TSDF Rekonstruktion

Klingensmith u. a. (2015) erwähnen, dass ihr Verfahren Chisel zunächst als proprietäre Umsetzung im Project Tango Constructor<sup>10</sup>, Googles Demo Anwendung zur räumlichen Rekonstruktion, umgesetzt wurde. Zu ihrer Publikation haben sie jedoch zusätzlich ein Open-Source ROS basiertes Modul zur Verfügung gestellt. Diese Bibliothek mit dem Namen OpenChisel<sup>11</sup> wurde für den Prototypen in dieser Arbeit auf das Android NDK portiert. Dafür wurden einige Module des C++11 Standards, wie zum Beispiel „st::shared\_ptr“, die

<sup>8</sup>OpenGL Mathematics - <http://goo.gl/2oY83s> (27.02.16)

<sup>9</sup>Eigen: template library for linear algebra - <http://goo.gl/TsNOuW> (27.02.16)

<sup>10</sup>Project Tango Constructor - <https://goo.gl/8HdTnY> (27.02.16)

<sup>11</sup>OpenChisel - Chisel chunked TSDF library - <https://goo.gl/nla8hy> (27.02.16)

nach derzeitigen Kenntnisstand vom Android NDK nicht unterstützt werden oder dessen Umsetzung ein anderes Verhalten aufweist, auf die Boost<sup>12</sup> Implementationen migriert. Neben der Boost Bibliothek nutzt OpenChisel auch die Eigen Bibliothek für Primitiven und Berechnungen der linearen Algebra.

Als Eingabe benötigt OpenChisel neben der Kameraposition und Kameraeigenschaften entweder eine Pointcloud oder ein Tiefenbild zum Anreichern der Rekonstruktion. In der Proof of Concept Umsetzung war erkennbar, dass OpenChisel mit der Pointcloud von Project Tango deutlich schlechtere Ergebnisse lieferte, als die Implementation des Constructors von Google. Dadurch, dass die Support Bibliothek von Google seit Februar 2016 eine performante Methode<sup>13</sup> anbietet, um aus einer Punktwolke eine DepthMap mit einer Auflösung von 320x180 Pixel zu interpolieren, wird nun ein Tiefenbild für OpenChisel verwendet. Die resultierenden Ergebnisse kommen dadurch der Constructor Implementation deutlich näher. Abbildung 5.5 zeigt eine exemplarische Rekonstruktion einer Pointcloud links mit dem resultierenden Tiefenbild rechts.

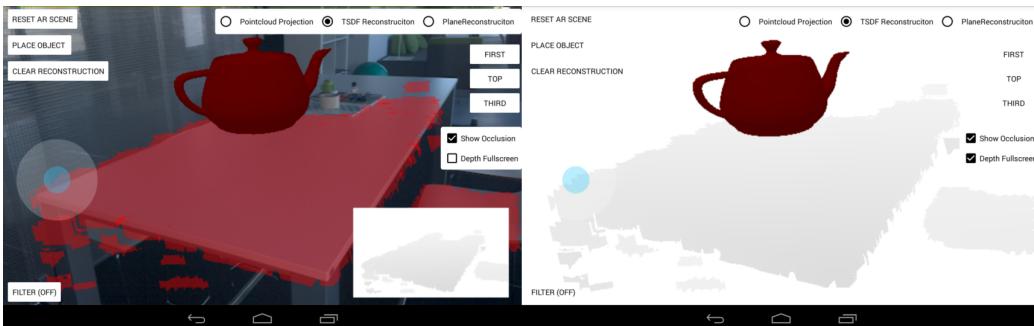


Abbildung 5.5: OpenChisel Rekonstruktion Prototyp. Links optionale Projektion auf der Bildebene. Rechts das resultierende Tiefenbild.

## Guided Filter

Für die Anwendung des Guided Filters wurde, wie bereits erwähnt, die Computer Vision Bibliothek OpenCV verwendet. Um diesen Filter mit OpenCV anwenden zu können, mussten zuvor das RGB Bild und das Tiefenbild in das OpenCV Format gebracht werden. Dies war jedoch mit den Methoden

<sup>12</sup>Boost C++ Libraray - <http://www.boost.org/> (27.02.16)

<sup>13</sup>TangoSupport\_upsampleImageNearestNeighbor - <https://goo.gl/mchIie> (27.02.16)

„glReadPixels“ und „glTexImage2D“ für den aktuell selektierten Framebuffer und der OpenGL Textur problemlos möglich.

Problematisch ist jedoch, dass das OpenGL Tiefenbild eine Farbtiefe von 16Bit nutzt und der OpenCV Guided Filter nur auf 8Bit Graustufen angewendet werden kann. Diese Transformation und die daraus resultierende Ungenauigkeit der Tiefe wurde jedoch zunächst in Kauf genommen, da erst einmal der Mechanismus selbst getestet werden soll. In der späteren Auswertung von Testszenarien muss diese Transformation berücksichtigt werden.

Diese Implementierung ermöglicht es, den Guided Filter dynamisch auf das aktuell generierte Tiefenbild mit dem aktuell aufgenommenen RGB Bild als Leitbild anzuwenden. Außerdem lassen sich die Filter Parameter, der Radius  $r$  und der Einflussfaktor  $\epsilon$  dynamisch variieren. Abbildung 5.6 zeigt das Tiefenbild der TSDF Rekonstruktion links, auf die rechts der Guided Filter mit dem aktuellen RGB Bild angewendet wurde.

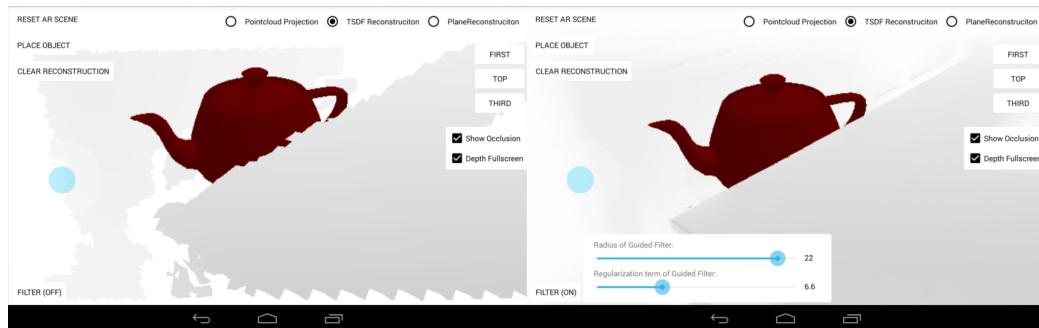


Abbildung 5.6: Anwendung des Guided Filters auf eine TSDF Rekonstruktion. Links vor und rechts nach der Anwendung.

## 5.4 Technische Problemstellungen

Auch wenn schon einige Probleme in der Umsetzung der jeweiligen Verfahren in Kapitel 5.3 näher beschrieben wurden, werden hier noch Einzelheiten aufgegriffen, die bei der Entwicklung für Projekt Tango zu beachten waren.

Alle von Project Tango zurückgegebenen Vektoren besitzen ihre eigene Konvention bezüglich der Achsenanordnungen. Gegenüber der Konvention in OpenGL sind die Achsen  $Z$  und  $Y$  vertauscht. Außerdem zeigt die

resultierende  $Z$ -Achse in die entgegengesetzte Richtung. Aufgrund dieser unterschiedlichen Konventionen müssen alle Vektoren  $\vec{v}$  von Project Tango mit der Transformationsmatrix  $T_{OGL}^{PT}$  aus Gleichung 5.1 konvertiert werden (Google, 2015b). Nachdem Google im Laufe dieser Arbeit neue Schnittstellen<sup>14</sup> zur Verfügung gestellt hat, um diese Transformationen zu abstrahieren, können die „TransformationSupport“ Methoden hierfür genutzt werden.

$$T_{OGL}^{PT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (5.1)$$

Wie bei vielen Echtzeitsystemen ist das Problem der Nebenläufigkeit (engl. Concurrency) auch in der Schnittstelle von Project Tango zu beachten. Die API von Project Tango publiziert die Sensordaten typischerweise asynchron. Außerdem werden die Rekonstruktionsverfahren von den entsprechenden API Publikationen angestoßen. Deshalb muss besonders beim Rendering sichergestellt werden, dass die Daten während dieses Prozesses nicht modifiziert werden. Diese Synchronisation wird durch den Einsatz von „std::mutex“ aus der C/C++ Standard-Bibliothek gewährleistet.

Bei der Anwendung des Guided Filters von OpenCV ist zu beachten, dass die Speicherkonventionen von OpenGL und OpenCV, was die X und Y Achse angeht, genau vertauscht sind. Dadurch können zum Umwandeln der Bilder in das jeweilige Framework nicht die direkten Adressen verwendet werden, da sonst das Bild um  $90^\circ$  gedreht wird. Da dieser Filterprozess jedoch für den Nutzer völlig transparent durchgeführt wird und das Bild wieder in OpenCV zurück konvertiert wird, kann der Filterprozess einfach um  $90^\circ$  gedreht durchgeführt werden.

---

<sup>14</sup>Project Tango API: Transformation Support - <https://goo.gl/N8dapq> (29.02.16)

# Kapitel 6

## Tests

In diesem Kapitel sollen die beschriebenen und prototypisch implementierten Verfahren zur Überlagerung gegenübergestellt werden, um anhand eines direkten Vergleichs eine objektive Aussage über die Qualität der Ergebnisse treffen zu können. Hierzu wird im ersten Teil das Vorgehen zum Testen vorgestellt, welches darauf folgend mit allen Verfahren umgesetzt wird. Danach werden die daraus resultierenden Ergebnisse gegenübergestellt.

### 6.1 Statische Testszenen

Zum Vergleich der Verfahren wurden zwei statische Szenen gewählt, in denen das Project Tango Gerät nicht bewegt wird, um dadurch allen Kandidaten den selben sensorischen Inhalt bieten zu können. Diese Wahl auf statische Szenen wurde getroffen, um eine zuverlässige und reproduzierbare Informationsquelle für das Gerät zu schaffen, denn die Reproduktion eines bewegten und dynamischen Szenarios ist für alle zu vergleichenden Verfahren nur sehr schwer möglich.

Eine Idee für ein dynamisches Testszenario jedoch wäre es, alle sensorischen Informationen der Hardware während einer AR Anwendung einmal aufzunehmen und eine reproduzierbare und simulierte Umgebung dieser Daten zu schaffen. Technologien wie das Robot Operating System (ROS) würden dies ermöglichen, jedoch übersteigt der Aufwand den zeitlichen Rahmen dieser

Arbeit. Auch wenn die Firma Bosch eine exemplarische Implementation<sup>1</sup> für die Aufnahme aller Daten in ROS demonstriert, sind die hier implementierten Verfahren zu sehr in den API Zyklen der Project Tango Schnittstelle involviert, um diese in kurzer Zeit auf eine Desktop Umgebung zu portieren. Um dennoch eine Aussage über die Verfahren in einer dynamischeren Anwendung treffen zu können, sollen neben den statischen Tests auch manuelle dynamische Szenarien mit den Verfahren durchgegangen werden.

Die erste gewählte Szene, welche in Abbildung 6.1 links zu sehen ist, beinhaltet einen Hocker, in Form eines einfachen Würfels, und einen Sitzball. Der Sitzball wurde gewählt, um auch runde Formen zur Tiefenaufnahme zu testen, welche gegebenenfalls für die Verfahren schwerer zu rekonstruieren sind. Das Project Tango Gerät ist etwas höher in einem Stativ plaziert. Das virtuelle Objekt wird, wie in Abbildung 6.1 rechts, zwischen die beiden realen Objekte platziert, sodass es von beiden Seiten durch die realen Objekte überdeckt wird.

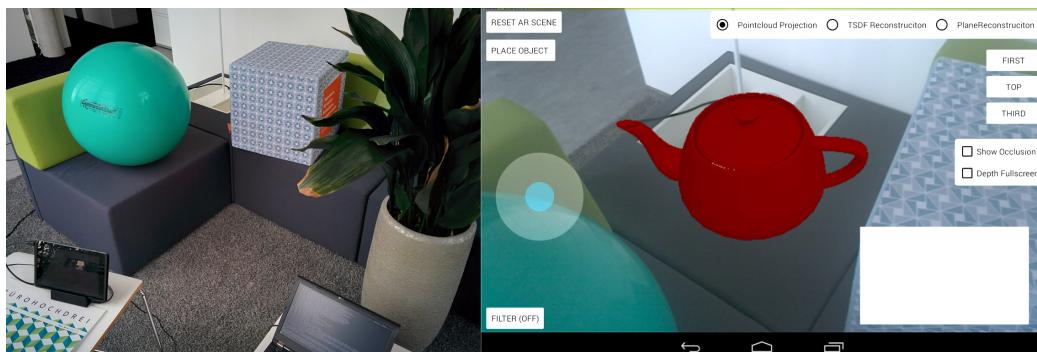


Abbildung 6.1: Links: Erste statische Szene mit einem Hocker und einem Sitzball. Rechts: Platzierung des virtuellen Objekts.

Die zweite gewählte Szene, welche in Abbildung 6.2 links zu sehen ist, soll als Herausforderung die Überdeckung von komplexeren Strukturen testen. Sie besteht daher aus einer Pflanze, die sich, wie rechts im Bild zu sehen, vor dem virtuellen Objekt befindet. Auch hier befindet sich das Project Tango Gerät in einem Stativ, damit sich die Position während der Tests nicht ändert.

Für beide Szenen sollen alle Kombinationen der Verfahren getestet werden. Somit ergeben sich sechs verschiedene Kombinationen pro Szene, in denen die

---

<sup>1</sup>Tango Output to Rosbag Files - <https://goo.gl/hhnciZ> (26.03.16)

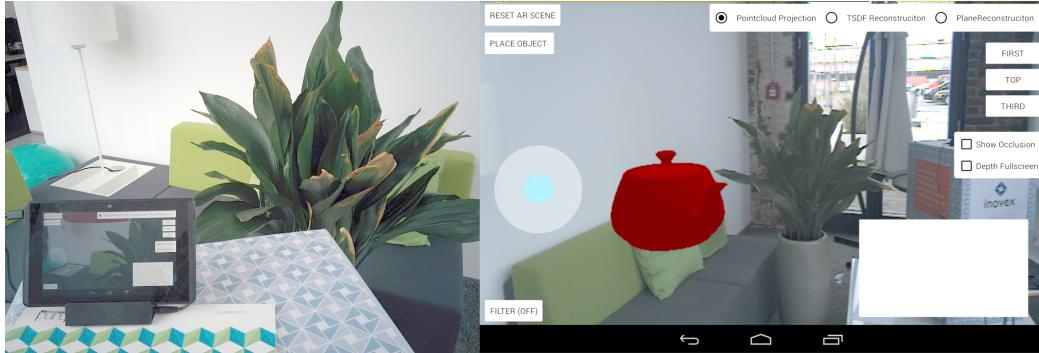


Abbildung 6.2: Links: Zweite statische Szene mit einer Pflanze im Vordergrund. Rechts: Platzierung des virtuellen Objekts hinter der Pflanze.

Pointcloud Projektion, die TSDF Rekonstruktion und die Ebenenrekonstruktion jeweils mit und ohne die Anwendung des Guided Filter auf das Tiefenbild getestet werden. Für alle Kombinationen soll ein gerendertes Bild und ein Tiefenbild mit dem virtuellen Objekt festgehalten werden. Zur Auswertung werden die jeweils gerenderten Ergebnisbilder  $p$  mit einem manuell zugeschnittenen Ergebnisbild  $q$  für jeden Pixel  $i$  verglichen. Für die Gegenüberstellung wird die Anzahl der abweichenden Pixel wie in Gleichung 6.1 bestimmt. Ein Pixel wird nur als Veränderung gewertet, wenn die absolute Differenz der Pixel einen Faktor  $threshold$  übersteigt. Dies verhindert, dass leichte Variationen in der Farbintensität der Umgebung mit in die Wertung eingehen.

$$d = \sum_i \begin{cases} 1, & \text{if } |p_i - q_i| > threshold \\ 0, & \text{otherwise} \end{cases} \quad (6.1)$$

## 6.2 Durchführung der Tests

Die beiden Testszenen konnten wie beschrieben aufgebaut und mit allen Verfahren durchgetestet werden. Hierzu wurde mit der „Android Debug Bridge“ (adb)<sup>2</sup> eine Videoaufnahme gestartet, in der im Prototypen für jede Szene alle Verfahren durchgeschaltet wurden. Die Verfahren mussten dabei entsprechend zeitnah durchlaufen werden, um einen potentiellen Drift von Project Tangos Motion Tracking so minimal wie möglich zu halten, denn dieser negative Effekt würden die Ergebnisse stark beeinträchtigen.

<sup>2</sup>Android Debug Bridge - <http://goo.gl/ffH51x> (01.03.16)

Bei der Anwendung des Guided Filters wurde immer der Radius  $r = 24px$  und der Einflussfaktor  $\epsilon = 0.8$  gewählt. Diese Werte versprachen nach einigen manuellen Tests erfolgreiche Ergebnisse. Der Radius wurde so groß gewählt, damit die bei den Rekonstruktionen üblichen Fehler auch weitreichend in den Gewichtungskern einfließen können. Der Einflussfaktor  $\epsilon$  wurde auch größer als in den üblichen Beispielen von (He u. a., 2010) gewählt, damit nur die offensichtlich sichtbaren Kanten im Farbbild nicht vom Weichzeichnen betroffen sind. Denn ein  $\epsilon >> \sigma^2$  für einen Bildausschnitt  $w_k$  ohne ersichtliche Kante wirkt hier wie ein zweidimensionaler Tiefpass und reduziert das Rauschen.

Abbildung A.1 und A.2 im Anhang zeigen jeweils die aus dem Video extrahierten Bildausschnitte. Die obere Reihe zeigt die drei tiefengenerierenden Verfahren ohne den Guided Filter und die untere Reihe jeweils mit dem Filter. In der untersten Reihe sind jeweils die Projektionen der generierten Primitiven in der Szene zu sehen, um sich eine Vorstellung der Rekonstruktion machen zu können.

Nach der Ausführung dieser statischen Tests wurde jedes Verfahren zusätzlich mit einem bewegten Einsatz des Gerätes getestet. Dazu wurde das virtuelle Objekt zunächst neben einen Würfel gestellt, um danach das Gerät räumlich um den Würfel zu führen, sodass das virtuelle Objekt in der Bewegung überdeckt wurde. Da diese Tests aus zuvor beschriebenen Gründen nicht eindeutig reproduzierbar sind, werden auch keine Testdaten hiervon erhoben. Dennoch soll diese nicht rein objektive Beobachtung während dieser Tests mit in die Evaluation der Verfahren fließen.

## 6.3 Auswertung der Ergebnisse

Der Vergleich der Ergebnisse, welcher mit dem bereits beschriebenen Bilddifferenz Ansatz aus der Gleichung 6.1 durchgeführt werden soll, wurde mit Hilfe der OpenCV Bibliothek durchgeführt und ist in Listing B.1 im Anhang zu finden. Für jede Szene wurde ein Referenzbild manuell konstruiert, welches dem idealen Ergebnis entsprechen soll. Zur Konstruktion der Vergleichsbilder wurden Ausschnitte mit leichten Anpassungen aus einigen Ergebnisbildern gewählt und zusammengeschnitten. Diese Referenzbilder sind in Abbildung

6.3 oben zu finden. Mit dem Referenzbild wurden alle zuvor passend zugeschnittenen Grafiken einer Szene verglichen. Außerdem wurde eine Maske für jede Szene für den Bereich des Ergebnisbildes bestimmt, die überhaupt für die Bestimmung der Differenz in Betracht gezogen wird. Die Masken sind jeweils in Abbildung 6.3 unten zu finden.

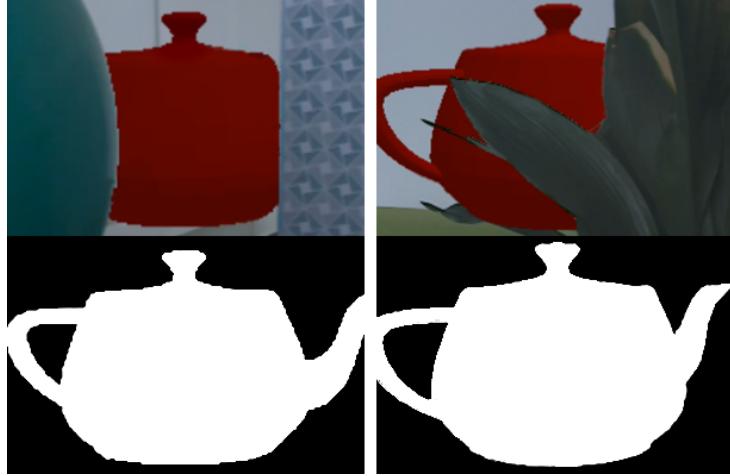


Abbildung 6.3: Manuell konstruierte Referenzbilder der idealen Überlagerung mit Filtermaske in Szene 1 (links) und 2 (rechts).

Das gezeigte Skript zum Vergleich der Ergebnisbilder mit den Referenzbildern ergibt neben den Differenzwerten, welche in Tabelle 6.1 zu finden sind, auch die absoluten Differenzbilder für jedes Verfahren. Diese Ergebnisbilder sind auch im Anhang in Abbildung A.3 zu finden. In der Abbildung werden beide Szenen mit je zwölf Bildern dargestellt, wobei jedes der drei Überlagerungsverfahren (horizontal) mit und ohne Anwendung des Guided Filters (vertikal) festgehalten sind. Sie sind also genau wie die Ergebnisse in Tabelle 6.1 angeordnet. Diese enthält wiederum die Differenzwerte für jede Kombination der Überlagerungsumsetzungen.

	Pointcloud Projektion	TSDF Rekonstruktion	Ebenen Rekonstruktion
<b>Szene 1</b>	1400	3165	2193
<b>Szene 1 + GF</b>	1506	85	1545
<b>Szene 2</b>	2478	4007	4478
<b>Szene 2 + GF</b>	67	1816	1933

Tabelle 6.1: Distanzwert zwischen dem Referenzbild und den Ergebnisbildern in der jeweiligen Szene.

# Kapitel 7

## Fazit

In diesem abschließenden Kapitel werden zunächst die implementierten Verfahren anhand ihrer Testergebnisse und Eigenschaften für den möglichen Einsatz in mobilen Augmented Reality Anwendungen eingeschätzt. Dazu werden sowohl die Ergebnisse aus den statischen Testszenen als auch die Einschätzungen in der dynamisch getesteten Anwendung in die Ergebnisse einfließen. Hiernach soll zudem noch ein Ausblick gegeben werden, wie eine potentielle Weiterentwicklung der Verfahren aussehen könnte und welche technischen Änderungen an diesen einen verbesserten Einsatz der Verfahren begünstigen würden.

### 7.1 Evaluation

Die prototypische Umsetzung der Verfahren zeigt, dass mit dem grundlegenden Ansatz der AR Tiefenbild Überdeckung von Wloka u. Anderson (1995) eine Echtzeitüberdeckung virtueller Objekte auch auf der mobilen Project Tango Hardware erfolgreich umgesetzt werden kann. Dabei wird im Folgenden auf jedes Verfahren sowie ihre Vor- und Nachteile im Kontext der anderen Verfahren und auf Basis der durchgeführten Tests eingegangen.

## **Pointcloud Projektion**

Die Überlagerung durch die Pointcloud Projektion bietet gegenüber den anderen Verfahren den Vorteil, dass sie zu jeder Zeit eine dynamische und aktualisierte Repräsentation der Tiefe von der Szene liefert und somit auch Änderungen in der Szene sofort berücksichtigt. Außerdem ist das Verfahren nicht auf Clustergrößen beschränkt und kann dadurch auch komplexe Strukturen abbilden. Diese Eigenschaft ist besonders bei der Überlagerung der komplexeren zweiten Szene hilfreich. Dort generiert die Pointcloud Projektion das beste Ergebnis mit 2478 geänderten Pixeln, ohne ein Filtern durch den Guided Filter. Die beiden anderen Rekonstruktionsverfahren generieren hingegen einen Distanzwert über 4000.

Dadurch, dass die Sensordaten von Project Tango Fehler in Form von Ausreißern und einem gewissen Rauschen enthalten können, spiegeln sich diese Fehler jedoch auch in der berechneten Projektion wider. Das führt dazu, dass zum Beispiel die Kante einer realen Überlagerung durchgehend in Bewegung ist und ihre Struktur mit jedem neuen Tiefenbild variiert. Neben den kontinuierlich variierenden Kanten betrachtet eine Pointcloud Projektion nur den aktuellen Aufnahmezeitpunkt. Dadurch können Überlagerungen nur für den aktuellen Messbereich des Tiefensensors bestimmt werden. Die Überlagerungen von realen Objekten innerhalb der ersten 50 Zentimeter und ab vier Metern können somit nicht mehr bestimmt werden. Durch das Zeichnen der Tiefe mit den Punkt Primitiven eines bestimmten Radius entstehen zusätzliche Fehler, denn die Abbildung von, zum Beispiel, schräg angesiedelten Flächen enthalten sichtbare Quantisierungsstufen.

## **Ebenenrekonstruktion**

Die Ebenenrekonstruktion löst die Schwächen der Pointcloud Projektion in dem Sinne, dass sie die Ungenauigkeit der Tiefeninformation als eine Oberflächen Approximation mit Hilfe von RANSAC in Form von Ebenen abbildet. Hierdurch werden Ausreißer weitestgehend ignoriert und auch das Rauschen bezüglich der Oberfläche wird durch eine lineare Regression gemittelt. Dadurch verschwindet das kontinuierliche Rauschen an den Kanten der

Überlagerungen. Zusätzlich ermöglicht das Vorgehen der Ebenenrekonstruktion eine kontinuierliche Verbesserung, indem alle bereits aufgenommenen Pointclouds in die aktuelle Rekonstruktion einfließen.

Diese Rekonstruktion wird immer pro Octree Cluster durchgeführt und birgt somit die Gefahr, dass Strukturen mit den Ebenen zu grob abgebildet werden. Dadurch dass sich die Ebenen aber nicht wie bei der Ebenenrekonstruktion von Yang u. Förstner (2010) auf die gesamte Würfelgröße ausbreiten, sondern die Ausbreitung anhand der konvexen Hülle bestimmt wird, können auch etwas detailliertere Formen abgebildet werden. So reicht der Detailgrad aus, um in Szene eins eine angemessene Rekonstruktion zu erstellen. In Szene zwei führt das Ergebnisbild der Ebenenrekonstruktion jedoch zu dem größten Distanzwert zum optimalen Ergebnisbild.

In einem manuellen und damit dynamischeren Test mit Bewegungen weist dieses Verfahren zusätzlich einige Schwächen auf. So sind durch die begrenzte Dichte der Pointcloud Lücken zwischen den Ebenen zu sehen, die zwar von Aufnahme zu Aufnahme kleiner werden aber üblicherweise nicht komplett schließen. Das führt dazu, dass zum Beispiel große Oberflächen, die ein virtuelles Objekt überlagern, das Objekt vereinzelt nicht aussparen, da keine Tiefe an den Stellen durch Lücken zwischen den Ebenen vorhanden ist. Außerdem ist das Verfahren nur bedingt in der Lage runde Strukturen wie den Sitzball aus Szene eins zu rekonstruieren. Diese Fehler werden besonders dann sichtbar, wenn man sich um diesen Ball dreht und er eine Überlagerung mit den Ecken und Kanten der Ebenen aus der Rekonstruktion auf ein virtuelles Objekt ausübt. Neben der fehlenden Unterstützung für runde Konturen, besitzt dieses Verfahren keine Möglichkeit, Messungen zu revidieren, wenn reale Objekte in der Szene verändert wurden oder ein Drift Fehler von Project Tango auftritt.

## TSDF Rekonstruktion

Wie zu erwarten liefert Chisel als eine TSDF Implementierung, aufgrund der großen Voxelgröße, nicht die Qualität, die zum Beispiel ein KinectFusion liefern kann. Dafür ist es performant genug, um als eine CPU Implementierung eine Echtzeitrekonstruktion auf der mobilen Project Tango Hardware zu

ermöglichen. Genau wie die Ebenenrekonstruktion bietet Chisel den Vorteil eine Rekonstruktion pro Tiefenbild anzureichern und stetig zu verbessern. Dadurch können Überlagerungen auch außerhalb des Messbereichs des Tiefensensors ermöglicht werden. Anders als bei der Ebenenrekonstruktion generiert die TSDF Rekonstruktion stets eine geschlossene Oberfläche. Außerdem können runde Strukturen festgehalten werden, wodurch der in Szene eins stehende Sitzball abgebildet werden kann.

Die große Voxelgröße führt jedoch dazu, dass, wie in beiden getesteten Szenen zu erkennen, die Strukturen der Rekonstruktionen sehr grob ausfallen und dadurch die Differenzergebnisse ohne eine Filterung auf einen hohen Fehler hinweisen. Auch wenn Chisel nicht in der Lage ist, die detaillierten Kantenabbildungen wie die Ebenenrekonstruktion zu generieren, besitzt Chisel einen Vorteil: Durch den Space Carving Mechanismus können Rekonstruktionen wieder revidiert werden. Das hilft dabei den Problematiken des Drift Effekts von Project Tango entgegenzuwirken. Außerdem könnte dieses Rekonstruktionsverfahren durch eine GPU Implementierung mit deutlich kleinerer Voxelgröße und dadurch höherem Detailgrad realisiert werden.

## Guided Filter

Der Guided Filter war in den Tests häufig in der Lage selbst grobe Fehler im Tiefenbild an die Kanten der Farbbilder anzugeleichen und somit auch, wie in den Messergebnissen zu erkennen, den Differenzwert zum Optimum zu reduzieren. Jedoch führte der Einsatz des Filters zu deutlichen Performanceeinbußen, denn der Filterprozess selbst benötigt im Durchschnitt 220 ms. Der Einsatz von OpenCV erschwert zudem den Einsatz des Filters für die Echtzeitumsetzung, da die Bildebene pro Bild aus dem OpenGL Framebuffer heraus und wieder hinein geladen werden muss. Dieser Prozess benötigt zusätzliche 80 ms, was die Wiederholrate der prototypischen Implementierung auf 3 Hertz reduziert.

Zusätzlich sind unter gewissen Umständen, bei denen ein virtuelles Objekt nah an der Oberfläche eines realen Objekts, aber immer noch räumlich hinter dem realen Objekt liegt, Artefakte aufgefallen, an denen das eigentlich überlagerte virtuelle Objekt durchschimmert. Dieser Effekt ist in Abbildung

7.1 zu erkennen. Denn neben den eigentlichen Kanten für die Überlagerung im Farbbild werden auch Kanten von flachen Strukturen berücksichtigt. Wie im Bild zu sehen, beeinflusst das Muster vom Würfel das resultierende Tiefenbild soweit, dass eine fehlerhafte Überlagerung nach der Filterung stattfindet. Zudem ist zu beachten, dass das Tiefenbild durch die Konvertierung von 16 zu 8 Bit Zwischenschritte verliert, wodurch dieser Effekt begünstigt wird. So liegt bei 16 Bit die Auflösung der Tiefe, mit einer hinteren Clippingebene von  $10m$ , noch bei  $0.0152cm$ , wo bei einer 8 Bit Kodierung diese nur noch bei  $3.90cm$  liegt.



Abbildung 7.1: Fehlerhafte Überdeckung bei der Anwendung des Guided Filters. Links: Reales Objekt mit Rekonstruktion. Mitte: Tiefenbild. Rechts: Sichtbare Fehler nach Guided Filter.

Angewendet auf die Tiefeninformationen der Pointcloud Projektion konnte der Filter in der komplexeren zweiten Szene nahezu das Optimum der Überlagerung erreichen. Schwieriger war jedoch der Einsatz bei der runden Kontur in der ersten Szene, wo der Filter nicht in der Lage war, den initialen groben Fehler des Tiefensensors zu revidieren. Das selbe Verhalten ist auch bei der Ebenenrekonstruktion in Szene eins zu beobachten.

Besonders beachtenswert ist die Tatsache, dass bei den zu weit reichenden Tiefeninformationen der TSDF Rekonstruktion, in der Mitte der Messergebnisse von Szene eins, ein etwa gleichgroßer Fehler durch die Filterung behoben werden konnte. Eine weitere Besonderheit, die während der Tests zu erkennen war, ist, dass der Filter in der Lage war, bei der Anwendung auf die Tiefeninformationen der Ebenenrekonstruktion, die Lücken zwischen den Ebenen unkenntlich zu machen. Somit würde dieser Filter einen Nachteil der Ebenenrekonstruktion lösen.

## 7.2 Einsatz der Verfahren

Grundsätzlich ist festzuhalten, dass sich die Pointcloud Projektion ohne den Guided Filter, trotz des erreichbaren Detailgrads, höchstens in einzelnen Bildaufnahmen für eine Überlagerung in Augmented Reality eignet, da das Rauschen der Eingangsdaten durchgehend sichtbar ist. Außerdem ist die Sichtweite auf die Erreichbarkeit des Tiefensors des aktuellen Ausschnitts begrenzt. Auch die Ebenenrekonstruktion ist ohne Filterung bedingt geeignet, da Lücken zwischen den Ebenen zu erkennen sind, die die Illusion von AR zerstören würden. Auch wenn die TSDF Rekonstruktionen durch Chisel nach den statischen Testszenen oft mit Fehlern behaftet waren, existieren entschiedene Vorteile gegenüber den anderen Vorgehensweisen. Denn durch Chisel werden geschlossene Flächen gebildet, welche sich dynamisch der Szene anpassen können. Betrachtet man zudem den Einsatz von Chisel in größeren Flächen, wie in Räumen oder sogar im ganzen Gebäude, fällt der gemessene Fehler deutlich weniger auf.

Angenommen, man würde den Guided Filter im Prototypen für jedes Verfahren in Echtzeit anwenden können, so würde die Pointcloud Projektion durchaus Anwendung finden. Es könnte zum Beispiel in einer AR Applikation genutzt werden, die sich nur in einem bestimmten Sichtbereich bewegt und die eine gewisse komplexe und dynamische Szene bedienen muss. Ausgehend von den Testergebnissen als Entscheidungsgrundlage zwischen der Ebenenrekonstruktion und der TSDF Rekonstruktion mit dem Guided Filter würde, wie auch ohne Filter, Chisel die bessere Alternative darstellen. Denn wenn das aktuelle Project Tango Gerät in der Lage wäre, die Leistung für den Echtzeiteinsatz der aktuellen Umsetzung bereitzustellen, könnten auch die Cluster in Chisel deutlich kleiner gewählt werden. Die Verkleinerung der Cluster bei der Ebenenrekonstruktion würde dagegen deutlich mehr False-Positives bei der RANSAC Ebenenerkennung generieren. Denn es lägen deutlich weniger Punkte in einem Cluster, was wiederum die Wahrscheinlichkeit erhöht, dass pro RANSAC Iteration die falschen Stichproben gewählt werden und somit falsche Ebenen für die Oberfläche bestimmt werden.



Abbildung 7.2: Unity Prototyp mit Schattenschlag auf den Polygonen der Chisel Rekonstruktion.

Ein weiterer Grund sich für ein Rekonstruktionsverfahren zu entscheiden, welches geometrische Primitiven der Szene generiert, ist die Möglichkeit, Interaktionen oder Schattenschlag mit den verfügbaren Techniken von OpenGL umzusetzen. Denn wenn ausschließlich die Tiefeninformationen wie bei der Pointcloud Projektion bekannt sind, müssen beispielsweise für den Schattenschlag die entsprechenden Normalen der Oberfläche pro Tiefenwert bestimmt werden. Bei den Rekonstruktionen werden die Normalen bereits im Verfahren bestimmt oder sind nachträglich über die Polygone einfach zu bestimmen. Hierzu wurde in dieser Arbeit ein weiterer Proof of Concept umgesetzt, der, mithilfe einer Chisel Rekonstruktion und dem Shadowmapping der Unity Engine, die Projektion einer virtuellen Kugel mit Schattenschlag in einer realen Umgebung ermöglicht. Zwei Beispiele der Anwendung sind in Abbildung 7.2 zu sehen.

### 7.3 Ausblick

Im Laufe der Umsetzung und der Auswertung der Verfahren sind noch weitere Ideen entstanden, die nicht näher verfolgt werden konnten, aber durchaus zu einer Verbesserung der Qualität und Performance in den Verfahren beitragen könnten. Um die Qualität der Rekonstruktionen zu verbessern, könnte näher untersucht werden, wie sich das Filtern der Pointcloud vor der Integration in die Rekonstruktion auswirken könnte. Dazu kann die Pointcloud zunächst zu einem Tiefenbild projiziert werden und daraufhin mit dem Guided Filter

oder dem „Guided Depth Upsampling“ Verfahren von Ferstl u. a. (2013) auf Basis des Farbbildes verbessert werden, denn bis jetzt wurde immer erst nach der Rekonstruktion eine Optimierung der Tiefeninformationen vorgenommen. Hierdurch könnten Ungenauigkeiten und Ausreißer schon vor der Rekonstruktion reduziert werden.

Für den produktiven Einsatz des Guided Filters wäre eine eigene Implementierung sinnvoll, um den OpenCV Anteil aus der prototypischen Umsetzung herausnehmen zu können. Das würde zunächst den Transformationsaufwand zu OpenCV von  $80ms$  vermeiden. Außerdem wäre es denkbar, den Guided Filter komplett im OpenGL Fragmentshader umzusetzen, wodurch eine parallele Verarbeitung durch die GPU diesen Prozess deutlich beschleunigen würde. Zusätzlich könnten hier weitere Verfahren zum Upsampling der Tiefeninformationen näher evaluiert werden, denn neben dem Guided Filter können noch deutlich komplexere Verfahren eingesetzt werden, wie das zuvor erwähnte „Guided Depth Upsampling“ Verfahren von Ferstl u. a. (2013), das ein Tiefenbild, wie auch beim Guided Filter, auf Basis der Farbbilder optimiert.

Während der Implementierung und Recherche zu den verschiedenen Ansätzen zur Realisierung der AR Überdeckung mit Project Tango wurden viele neue Schnittstellen in der Project Tango Bibliothek von Google veröffentlicht<sup>1</sup>. Unter anderem wurde zuletzt eine Methode für das Filtern des Tiefenbildes mit Hilfe des Farbbildes durch den Bilateral Filter veröffentlicht<sup>2</sup>. Nach den Recherchen zu den Filtern ist hier eigentlich der Guided Filter der leistungsfähigste Ansatz. Außerdem wurde die interne Chisel Implementierung für eine Rekonstruktion unter der Unity Engine veröffentlicht. Diese scheint speziell auf die Charakteristika der Tango Hardware abgestimmt zu sein, und liefert im zuvor beschriebenen Prototypen aus Abbildung 7.2 präzise Rekonstruktionen mit einer Auflösung von  $5cm$ . Bei den speziellen Charakteristika ist hier die maximal zu verarbeitende Tiefe und die tiefenabhängige Gewichtung der Voxel gemeint, wodurch dem Rauschen der Sensordaten entgegengewirkt werden soll.

---

<sup>1</sup>Project Tango Release Notes - <https://goo.gl/ueGBtn> (21.03.16)

<sup>2</sup>Depth Interpolation Support Functions - <https://goo.gl/kDdq21> (21.03.16)

Seitdem die Firma Lenovo eine Kooperation<sup>3</sup> mit Google angekündigt hat, verspricht Project Tango eine einheitliche Technologie für das Motion Tracking, die Tiefenwahrnehmung und die Umgebungswiedererkennung auf mobilen Endgeräten zu werden. Lenovo wird im Sommer 2016 das erste Endverbrauchergerät mit Project Tango Hardware und Software auf den Markt bringen. Auch Intel arbeitet mit Google zusammen<sup>4</sup> und ermöglicht in ihrer Konkurrenztechnologie RealSense™ den Betrieb von Project Tango Anwendungen.

---

<sup>3</sup>Lenovo News - <http://goo.gl/jFLNyn> (21.03.16)

<sup>4</sup>Intel®RealSense™ Developer Kits - <http://goo.gl/j4Y18A> (21.03.16)

# Abbildungsverzeichnis

1.1	AR Projektion mit Project Tango - Links: Erfolgreiche Projektion. Rechts: Fehlerhafte Darstellung ohne Überdeckung.	3
2.1	Vereinfachte Darstellung des Realitäts-Virtualitäts-Kontinuum von Milgram, Takemura, Utsumi, u. Kishino (1995)	6
2.2	Links: schematischer Aufbau der Google Project Tango Hardware. Rechts: Das aktuelle Entwickler Gerät im Tablet Format (oben) und das alte Entwickler Gerät im Smartphone Format (unten). Übernommen von Google (2015c)	15
2.3	Links: Lokalisierungsprozess durch Area Learning. Rechts: Korrektur von Motion Tracking anhand gelernter Merkmale. Übernommen von Google (2015c)	18
3.1	Sortierung der Punkte nach Winkel zum Startpunkt (links). Das Unterscheidungskriterium für die Sortierung (rechts). Übernommen von Sunday (2001)	23
3.2	Schematische Darstellung eines Octrees mit zwei Untergliederungen. Übernommen von Dumusc u. a. (2011)	23
3.3	Raypicking Visualisierung. Übernommen von Marsh (2011)	25
4.1	Visualisierung der Methode zur Vedeckung durch Depth Maps. Übernommen von Kanbara u. a. (2000)	28
4.2	Veranschaulichung des Gesamtprozesses zur Bestimmung der Ebenenausbreitung durch RANSAC und der konvexen Hülle.	34
4.3	Links: Ebenenrekonstruktion ohne Clustering. Rechts: Rekonstruktion mit K-Mean Clustering.	35
4.4	a) Beispielhafte Voxel Füllung von Occupancy Maps; b) Beispielhafte Voxel Füllung durch TSDF; c) Exemplarische 2D Darstellung der Oberfläche mit entsprechenden Strahlensatz für die TSDF. Übernommen von Sturm (2014)	37

4.5	a) Voxel Hashing Datenstruktur. Übernommen von Nießner u. a. (2013)	b) Darstellung der relevanten Voxel Chunks für die Aktualisierung. Übernommen von Klingensmith u. a. (2015)	39
4.6	a) Marching Cubes Voxel Repräsentation mit den Ecken und den Kantenschnittpunkten	b) Die 15 möglichen 3D Polygon Varianten. Übernommen von Raphaël u. Karen (2002)	40
4.7	Exemplarische Visualisierung des Strahls $\vec{r}$ und der zu aktualisierenden Voxel $v$ im TSDF Algorithmus. Links ist hier das Project Tango Gerät zu erkennen, welches Tiefeninformationen der Oberfläche eines Objekts rechts aufgenommen hat.		42
4.8	Überdeckung mit einfacher Pointcloud Projektion. Links: Resultat der Überdeckung. Mitte: Darstellung des Z-Buffers mit dem Ausschluss des virtuellen Objekts. Rechts: Darstellung der Pointcloud.		42
4.9	Guided Filter Anwendungsbeispiel. Das Tiefenbild links ergibt durch den Guided Filter mit dem Leitbild in der Mitte das Ergebnis im rechten Bild.		45
5.1	Struktureller Aufbau des Prototypen		49
5.2	Prozessdiagramm für das Rendering der Szene		49
5.3	Pointcloud Projektion Prototyp. Links optionale Projektion auf der Bildebene. Rechts das resultierende Tiefenbild.		50
5.4	Planare Rekonstruktion Prototyp. Links optionale Projektion auf der Bildebene. Rechts das resultierende Tiefenbild.		51
5.5	OpenChisel Rekonstruktion Prototyp. Links optionale Projektion auf der Bildebene. Rechts das resultierende Tiefenbild.		52
5.6	Anwendung des Guided Filters auf eine TSDF Rekonstruktion. Links vor und rechts nach der Anwendung.		53
6.1	Links: Erste statische Szene mit einem Hocker und einem Sitzball. Rechts: Platzierung des virtuellen Objekts.		56
6.2	Links: Zweite statische Szene mit einer Pflanze im Vordergrund. Rechts: Platzierung des virtuellen Objekts hinter der Pflanze.		57
6.3	Manuell konstruierte Referenzbilder der idealen Überlagerung mit Filtermaske in Szene 1 (links) und 2 (rechts).		59

7.1	Fehlerhafte Überdeckung bei der Anwendung des Guided Filters. Links: Reales Objekt mit Rekonstruktion. Mitte: Tiefenbild. Rechts: Sichtbare Fehler nach Guided Filter. . . . .	64
7.2	Unity Prototyp mit Schattenschlag auf den Polygonen der Chisel Rekonstruktion. . . . .	66
A.1	Ergebnisaufnahmen aus der ersten statischen Szene . . . . .	73
A.2	Ergebnisaufnahmen aus der zweiten statischen Szene . . . . .	74
A.3	Differenzbilder der Verfahren in ersten (oben) und zweiten Szene (unten) . . . . .	75

# Listings

3.1	Der RANSAC Algorithmus . . . . .	21
3.2	Graham Scan Algorithmus . . . . .	22
4.1	Planare Echtzeitrekonstruktion . . . . .	31
4.2	Bestimmung der Ebenenausbreitung und Triangulation . . . .	33
4.3	Chisel TSDF Algorithmus . . . . .	41
B.1	Python Implementierung der Bilddifferenz . . . . .	76

# **Anhang A**

## **Ergebnisaufnahmen**

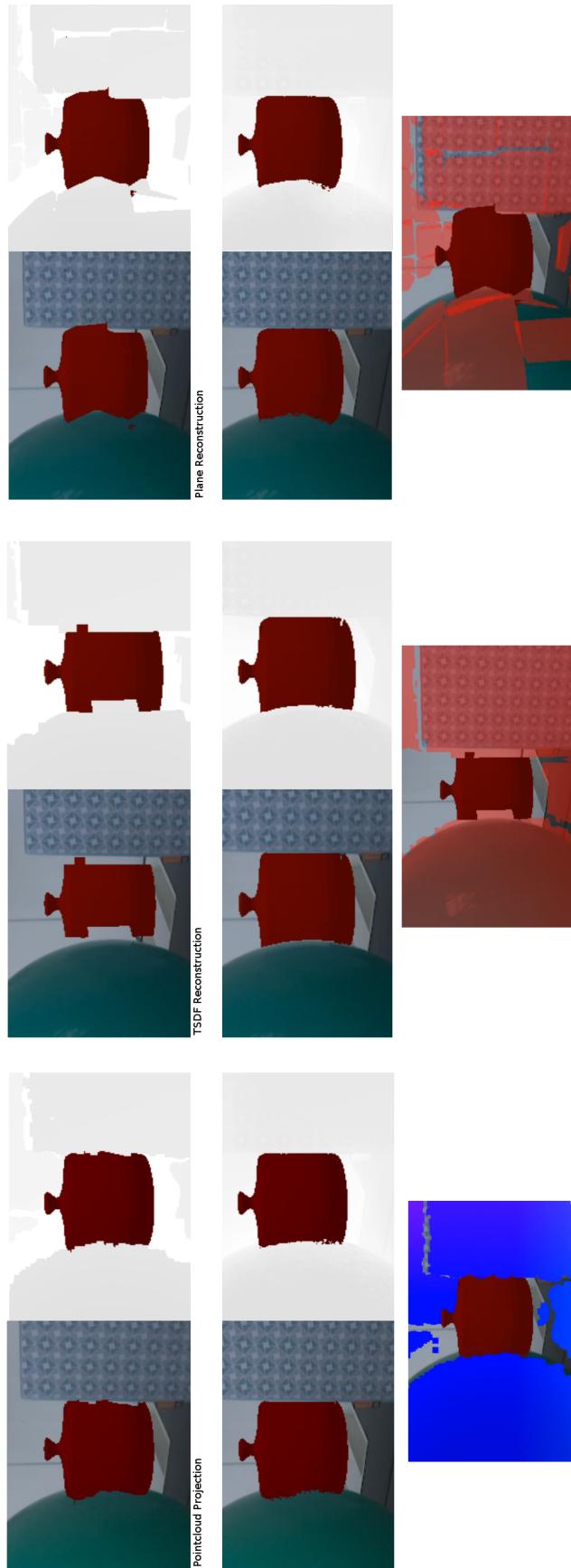


Abbildung A.1: Ergebnisaufnahmen aus der ersten statischen Szene



Abbildung A.2: Ergebnisaufnahmen aus der zweiten statischen Szene

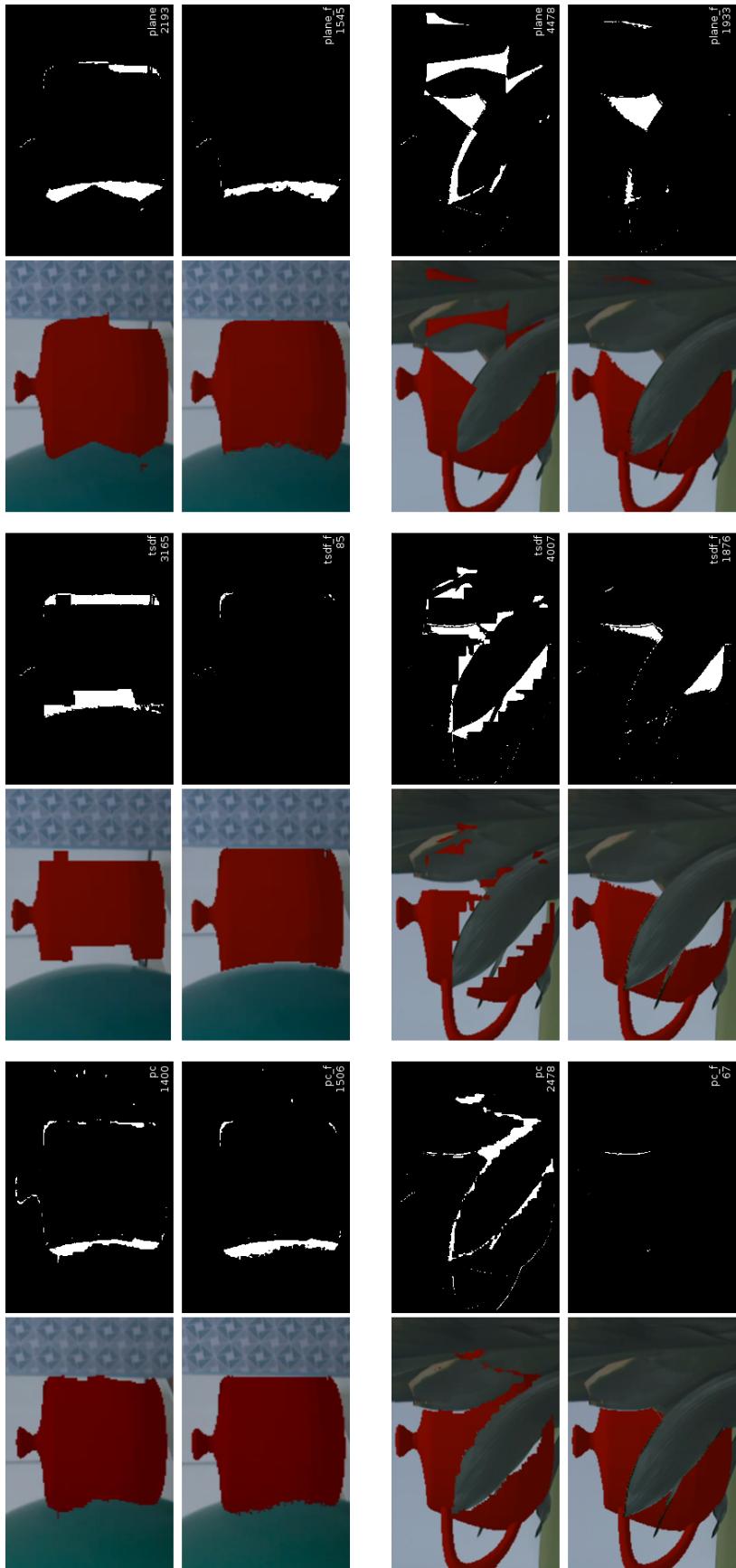


Abbildung A.3: Differenzbilder der Verfahren in ersten (oben) und zweiten Szene (unten)

## Anhang B

# Differenz OpenCV Python Skript

```
import numpy as np
from cv2 import *
from os import listdir
from os.path import isfile, join

reference = imread("reference.png")
mask = imread("mask.png")
mask = cvtColor(mask, COLOR_BGR2GRAY)
all_images = [f for f in listdir("./")
              if isfile(join("./", f))
              and f.endswith(".png")
              and not f.startswith("mask")
              and not f.startswith("reference")]
results = []

for img_path in all_images:
    img = imread(img_path)
    result = absdiff(reference, img)
    result = inRange(result, (30,30,30), (255, 255, 255))
    result = bitwise_and(result, mask)
    imwrite('result_'+img_path, result)
    difference = countNonZero(result)
    results.append([img_path, difference])

for result in results:
    print str(int(result[1])) + "\t" + result[0]
```

---

Listing B.1: Python Implementierung der Bilddifferenz

# Literaturverzeichnis

- [Andrew 1979] ANDREW, Alex M.: Another efficient algorithm for convex hulls in two dimensions. In: *Information Processing Letters* 9 (1979), Nr. 5, S. 216–219
- [Azuma u. a. 2001] AZUMA, Ronald ; BAILLOT, Yohan ; BEHRINGER, Reinhold ; FEINER, Steven ; JULIER, Simon ; MACINTYRE, Blair: Recent advances in augmented reality. In: *Computer Graphics and Applications, IEEE* 21 (2001), Nr. 6, S. 34–47
- [Berger 1997] BERGER, M-O: Resolving occlusion in augmented reality: a contour based approach without 3D reconstruction. In: *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on IEEE*, 1997, S. 91–96
- [Breen u. a. 1996] BREEN, David E. ; WHITAKER, Ross T. ; ROSE, Eric ; TUCERYAN, Mihran: Interactive occlusion and automatic object placement for augmented reality. In: *Computer Graphics Forum* Bd. 15 Wiley Online Library, 1996, S. 11–22
- [Bylow u. a. 2013] BYLOW, Erik ; STURM, Jürgen ; KERL, Christian ; KAHL, Fredrik ; CREMERS, Daniel: Real-time camera tracking and 3d reconstruction using signed distance functions. In: *Robotics: Science and Systems (RSS) Conference 2013* Bd. 9, 2013
- [Catmull 1974] CATMULL, Edwin: A subdivision algorithm for computer display of curved surfaces / DTIC Document. 1974. – Forschungsbericht
- [Crow 1984] CROW, Franklin C.: Summed-area tables for texture mapping. In: *ACM SIGGRAPH computer graphics* 18 (1984), Nr. 3, S. 207–212
- [Curless u. Levoy 1996] CURLESS, Brian ; LEVOY, Marc: A volumetric method for building complex models from range images. In: *Proceedings of the 23rd*

*annual conference on Computer graphics and interactive techniques* ACM, 1996, S. 303–312

[Derpanis 2010] DERPANIS, Konstantinos G.: Overview of the RANSAC Algorithm. In: *York University, Toronto, Canada* (2010)

[Dumusc u. a. 2011] DUMUSC, Raphael ; GONZÁLEZ, Germán ; LUCCHI, Aurélien ; FUÀ, Pascal: Multi-Scale 3D Rendering of Big Biological Images CVLab-EPFL. (2011)

[Dunston u. a. 2008] DUNSTON, Phillip S. u. a.: Identification of application areas for Augmented Reality in industrial construction based on technology suitability. In: *Automation in Construction* 17 (2008), Nr. 7, S. 882–894

[Eddy 1977] EDDY, William F.: A new convex hull algorithm for planar sets. In: *ACM Transactions on Mathematical Software (TOMS)* 3 (1977), Nr. 4, S. 398–403

[Elmqvist u. Fekete 2008] ELMQVIST, Niklas ; FEKETE, Jean-Daniel: Semantic pointing for object picking in complex 3D environments. In: *Proceedings of Graphics Interface 2008* Canadian Information Processing Society, 2008, S. 243–250

[Feng u. a. 2014] FENG, Chen ; TAGUCHI, Yasuhiro ; KAMAT, Vineet R.: Fast plane extraction in organized point clouds using agglomerative hierarchical clustering. In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on* IEEE, 2014, S. 6218–6225

[Ferstl u. a. 2013] FERSTL, David ; REINBACHER, Christian ; RANFTL, Rene ; RUETHER, Matthias ; BISCHOF, Horst: Image Guided Depth Upsampling Using Anisotropic Total Generalized Variation. In: *The IEEE International Conference on Computer Vision (ICCV)*, 2013

[Fischler u. Bolles 1981] FISCHLER, Martin A. ; BOLLES, Robert C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In: *Communications of the ACM* 24 (1981), Nr. 6, S. 381–395

[Google ] GOOGLE: *TangoCameraIntrinsics Struct Reference — Project Tango C API — Google Developers.* <https://goo.gl/iNUJdh>, . – (Besucht am 02/25/2016)

- [Google 2015a] GOOGLE: *Known Issues — Project Tango — Google Developers*. <https://goo.gl/gWW7ih>, 2015. – (Besucht am 12/06/2015)
- [Google 2015b] GOOGLE: *Project Tango - Coordinate System Conventions*. <https://goo.gl/i341Ib>, 03 2015. – (Besucht am 12/03/2015)
- [Google 2015c] GOOGLE: *Project Tango — Google Developers*. <https://goo.gl/Ucsuc9>, 12 2015. – (Besucht am 12/03/2015)
- [Google 2015d] GOOGLE: *Project Tango — Google*. <https://goo.gl/pOJ2Vf>, 03 2015. – (Besucht am 12/03/2015)
- [Google 2015e] GOOGLE: *Tango Concepts — Project Tango — Google Developers*. <https://goo.gl/sjwkLv>, 2015. – (Besucht am 12/03/2015)
- [Graham 1972] GRAHAM, Ronald L.: An efficient algorithm for determining the convex hull of a finite planar set. In: *Information processing letters* 1 (1972), Nr. 4, S. 132–133
- [Greene u. a. 1993] GREENE, Ned ; KASS, Michael ; MILLER, Gavin: Hierarchical Z-buffer visibility. In: *Proceedings of the 20th annual conference on Computer graphics and interactive techniques* ACM, 1993, S. 231–238
- [He u. a. 2010] HE, Kaiming ; SUN, Jian ; TANG, Xiaouo: Guided image filtering. In: *Computer Vision—ECCV 2010*. Springer, 2010, S. 1–14
- [Hoppe u. a. 1992] HOPPE, Hugues ; DEROSSE, Tony ; DUCHAMP, Tom ; McDONALD, John ; STUETZLE, Werner: *Surface reconstruction from unorganized points*. ACM, 1992
- [Kalman 1960] KALMAN, Rudolph E.: A new approach to linear filtering and prediction problems. In: *Journal of basic Engineering* 82 (1960), Nr. 1, S. 35–45
- [Kanbara u. a. 2000] KANBARA, Masayuki ; OKUMA, Takashi ; TAKEMURA, Haruo ; YOKOYA, Naokazu: A stereoscopic video see-through augmented reality system based on real-time vision-based registration. In: *Virtual Reality, 2000. Proceedings. IEEE* IEEE, 2000, S. 255–262
- [Kazhdan u. a. 2006] KAZHDAN, Michael ; BOLITHO, Matthew ; HOPPE, Hugues: Poisson surface reconstruction. In: *Proceedings of the fourth Eurographics symposium on Geometry processing* Bd. 7, 2006

- [Klein u. Drummond 2004] KLEIN, Georg ; DRUMMOND, Tom: Sensor fusion and occlusion refinement for tablet-based AR. In: *Mixed and Augmented Reality, 2004. ISMAR 2004. Third IEEE and ACM International Symposium on* IEEE, 2004, S. 38–47
- [Klingensmith u. a. 2015] KLINGENSMITH, Matthew ; DRYANOVSKI, Ivan ; SRINIVASA, Siddhartha ; XIAO, Jizhong: Chisel: Real Time Large Scale 3D Reconstruction Onboard a Mobile Device. In: *Robotics Science and Systems 2015*, 2015
- [Kottas u. a. 2013] KOTTAS, Dimitrios G. ; HESCH, Joel A. ; BOWMAN, Sean L. ; ROUMELIOTIS, Stergios I.: On the consistency of vision-aided inertial navigation. In: *Experimental Robotics* Springer, 2013, S. 303–317
- [LaMarca u. a. 2005] LAMARCA, Anthony ; CHAWATHE, Yatin ; CONSOLVO, Sunny ; HIGHTOWER, Jeffrey ; SMITH, Ian ; SCOTT, James ; SOHN, Timothy ; HOWARD, James ; HUGHES, Jeff ; POTTER, Fred u. a.: Place lab: Device positioning using radio beacons in the wild. In: *Pervasive computing*. Springer, 2005, S. 116–133
- [Lee u. Hollerer 2008] LEE, Taehee ; HOLLERER, Tobias: Hybrid feature tracking and user interaction for markerless augmented reality. In: *Virtual Reality Conference, 2008. VR'08. IEEE* IEEE, 2008, S. 145–152
- [Liu u. a. 2012] LIU, Junyi ; GONG, Xiaojin ; LIU, Jilin: Guided inpainting and filtering for Kinect depth maps. In: *Pattern Recognition (ICPR), 2012 21st International Conference on* IEEE, 2012, S. 2055–2058
- [Lorensen u. Cline 1987] LORENSEN, William E. ; CLINE, Harvey E.: Marching cubes: A high resolution 3D surface construction algorithm. In: *ACM siggraph computer graphics* Bd. 21 ACM, 1987, S. 163–169
- [Marsh 2011] MARSH, Andrew: *gluUnproject() for P3D and OPENGL Sketches — andrewmarsh.com.* <http://goo.gl/5h3gjI>, 2011. – (Besucht am 01/14/2016)
- [Milgram u. a. 1995] MILGRAM, Paul ; TAKEMURA, Haruo ; UTSUMI, Akira ; KISHINO, Fumio: Augmented reality: A class of displays on the reality-virtuality continuum. In: *Photonics for Industrial Applications* International Society for Optics and Photonics, 1995, S. 282–292

- [Mourikis u. a. 2007] MOURIKIS, Anastasios ; ROUMELIOTIS, Stergios u. a.: A multi-state constraint Kalman filter for vision-aided inertial navigation. In: *Robotics and Automation, 2007 IEEE International Conference on* IEEE, 2007, S. 3565–3572
- [Narzt u. a. 2006] NARZT, Wolfgang ; POMBERGER, Gustav ; FERSCHA, Alois ; KOLB, Dieter ; MÜLLER, Reiner ; WIEGHARDT, Jan ; HÖRTNER, Horst ; LINDINGER, Christopher: Augmented reality navigation systems. In: *Universal Access in the Information Society* 4 (2006), Nr. 3, S. 177–187
- [Newcombe u. a. 2015] NEWCOMBE, Richard A. ; FOX, Dieter ; SEITZ, Steven M.: DynamicFusion: Reconstruction and Tracking of Non-rigid Scenes in Real-Time. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, S. 343–352
- [Newcombe u. a. 2011] NEWCOMBE, Richard A. ; IZADI, Shahram ; HILLGES, Otmar ; MOLYNEAUX, David ; KIM, David ; DAVISON, Andrew J. ; KOHI, Pushmeet ; SHOTTON, Jamie ; HODGES, Steve ; FITZGIBBON, Andrew: KinectFusion: Real-time dense surface mapping and tracking. In: *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on* IEEE, 2011, S. 127–136
- [Nießner u. a. 2013] NIESSNER, Matthias ; ZOLLHÖFER, Michael ; IZADI, Shahram ; STAMMINGER, Marc: Real-time 3d reconstruction at scale using voxel hashing. In: *ACM Transactions on Graphics (TOG)* 32 (2013), Nr. 6, S. 169
- [Nistér 2004] NISTÉR, David: An efficient solution to the five-point relative pose problem. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 26 (2004), Nr. 6, S. 756–770
- [Nistér u. a. 2004] NISTÉR, David ; NARODITSKY, Oleg ; BERGEN, James: Visual odometry. In: *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on* Bd. 1 IEEE, 2004, S. I–652
- [OpenGL 2016] OPENGL: *OpenGL FAQ / 20 Picking and Using Selection.* <https://goo.gl/D4USqd>, 2016. – (Besucht am 01/14/2016)

- [Petschnigg u. a. 2004] PETSCHNIGG, Georg ; SZELISKI, Richard ; AGRAWALA, Maneesh ; COHEN, Michael ; HOPPE, Hugues ; TOYAMA, Kentaro: Digital photography with flash and no-flash image pairs. In: *ACM transactions on graphics (TOG)* Bd. 23 ACM, 2004, S. 664–672
- [Preparata u. Shamos 1985] PREPARATA, Franco P. ; SHAMOS, Michael I.: Convex hulls: Basic algorithms. In: *Computational geometry*. Springer, 1985, S. 95–149
- [Raphaël u. Karen 2002] RAPHAËL, Gervaise ; KAREN, Richard: *Marching Cubes Tutorial: Development report*. <http://goo.gl/egNPuO>, 06 2002. – (Besucht am 01/06/2016)
- [Seo u. Lee 2013] SEO, Dong W. ; LEE, Jae Y.: Direct hand touchable interactions in augmented reality environments for natural and intuitive user experiences. In: *Expert Systems with Applications* 40 (2013), Nr. 9, S. 3784–3793
- [Straßer 1974] STRASSER, W: *Schnelle Kurven-und Flachendarstellung auf graphischen Sichtgeräten*, Ph. D.-Thesis, Diss., 1974
- [Sturm 2014] STURM, Juergen: *Computer Vision Group - Summer Semester 2013 - Visual Navigation for Flying Robots*. <https://vision.in.tum.de/teaching/ss2013/visnav2013>, 2014. – (Besucht am 01/05/2016)
- [Sunday 2001] SUNDAY, Dan: *The Convex Hull of a Planar Point Set*. <http://goo.gl/lIQpj8>, 2001. – (Besucht am 01/07/2016)
- [Tomasi u. Manduchi 1998] TOMASI, Carlo ; MANDUCHI, Roberto: Bilateral filtering for gray and color images. In: *Computer Vision, 1998. Sixth International Conference on IEEE*, 1998, S. 839–846
- [Trajković u. Hedley 1998] TRAJKOVIĆ, Miroslav ; HEDLEY, Mark: Fast corner detection. In: *Image and vision computing* 16 (1998), Nr. 2, S. 75–87
- [Trevor u. a. 2011] TREVOR, Alexander J. B. ; ROGERS, John G. ; CHRISTENSEN, Henrik I.: *Planar Surface Mapping*. <https://goo.gl/5NyDn7>, 2011. – (Besucht am 01/05/2016)

[Trevor u. a. 2012] TREVOR, Alexander J. ; ROGERS III, John G. ; CHRISTENSEN, Henrik u. a.: Planar surface SLAM with 3D and 2D sensors. In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on* IEEE, 2012, S. 3041–3048

[Tsai 1987] TSAI, Roger Y.: A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. In: *Robotics and Automation, IEEE Journal of* 3 (1987), Nr. 4, S. 323–344

[Van Krevelen u. Poelman 2010] VAN KREVELEN, DWF ; POELMAN, R: A survey of augmented reality technologies, applications and limitations. In: *International Journal of Virtual Reality* 9 (2010), Nr. 2, S. 1

[Wloka u. Anderson 1995] WLOKA, Matthias M. ; ANDERSON, Brian G.: Resolving occlusion in augmented reality. In: *Proceedings of the 1995 symposium on Interactive 3D graphics* ACM, 1995, S. 5–12

[Yang u. Förstner 2010] YANG, Michael Y. ; FÖRSTNER, Wolfgang: Plane detection in point cloud data. In: *Proceedings of the 2nd int conf on machine control guidance, Bonn* Bd. 1, 2010, S. 95–104