
Institut für Informatik XVII
Robotik

*Masterarbeit*

# Multi-Sensor Kalibrierung

Luca Anteunis

Dezember 2022

Erstgutachter: Prof. Dr. Andreas Nüchter
Zweitgutachter: Prof. Dr.-Ing. Sergio Montenegro
Betreuer: MSc. Sven Jörissen

# Zusammenfassung

Die vorliegende Arbeit thematisiert die Kalibrierung einer Kamera und eines Laserprojektors zu einem Laserscanner. Der hier verwendete Laserscanner weist ein relativ starkes Rauschen auf, was sich negativ auf die Kalibrierung auswirkt. Um dieses Problem anzugehen, verwendet die Kalibrierung hierfür einen Quader als Kalibrierobjekt, wodurch eine genauere Lokalisierung möglich ist. Die Kamera wird hier sowohl intrinsisch als auch extrinsisch kalibriert. Für Projektor und Laserscanner ist eine intrinsische Kalibrierung nicht notwendig, da diese werkskalibriert sind. Die Experimente zeigen, dass die hier vorgestellte Methode eine erfolgreiche Kalibrierung der Kamera zu lässt und sowohl die intrinsische als auch die extrinsische Kalibrierung einen Reprojektionsfehler von weniger als einen Pixel aufweist. Wie in dieser Arbeit zu sehen ist, konnte die Kalibrierung des Projektors zwar erfolgreich durchgeführt werden, da der der Reprojektionsfehler am Kalibrierort unter 10 mm. Damit ist eine Kalibrierung der Sensoren grundsätzlich möglich. Außerhalb des Kalibrierorts hat die Projektion des Projektors zum Teil eine Abweichung größer als 100 mm.

# Inhaltsverzeichnis

# Tabellenverzeichnis

# Abbildungsverzeichnis

**DigSmart** Digitalisierung und Smart Services im Tiefbau

**GUI** grafischen Benutzeroberfläche (engl. Graphical User Interface)

**ICP** Iterative Closest Points

**CCD** Charge-Coupled Device

**CMOS** Complementary Metal–Oxide–Semiconductor

**ROS** Robotic Operating System

**RANSAC** RANdom SAmple Consensus

**PCA** Principal Component Analysis

**DLT-Algorithmus** Direct Linear Transformation Algorithmus

**RMSE** "Mittleren Reprojektionfehler"

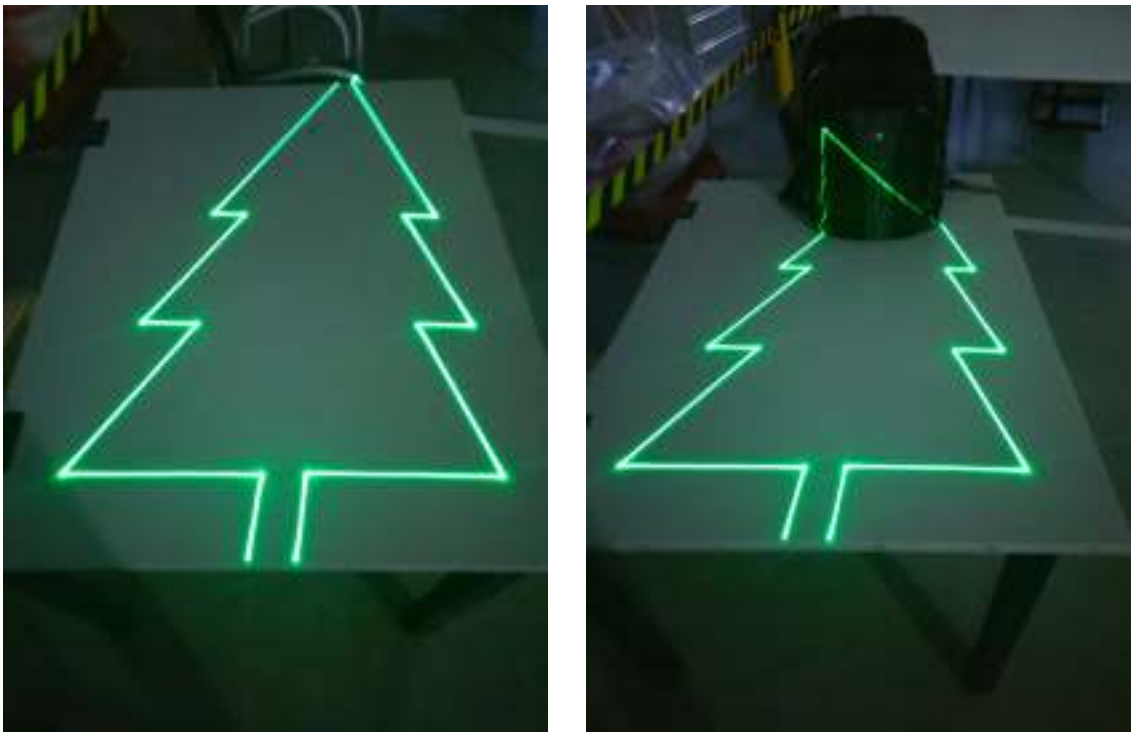
**PoE** Power over Ethernet

# Kapitel 1

# Einleitung

Science-Fiction Filme wie Star Trek oder Star Wars sowie Computerspiele wie Factorio zeigen immer wieder das Potential auf welches in der Interaktion zwischen Mensch und Maschine steckt. Alleine in dem Computerspiel Factrio ist der Spieler in der Lage eine große Schar an Robotern für sich einzusetzen, um die eigene Fabrik durch Roboter beliebig zu skalieren. In der Realität nimmt der Einsatz von Robotern, künstlicher Intelligenz und Augmented Reality immer mehr zu. Hierdurch übernehmen Maschinen immer mehr Aufgaben und unterstützen dabei den Menschen bei komplexen Aufgaben. Durch Augmented Reality ist es z.B. möglich durch Visualisierung den Nutzer auf wichtige Dinge hinzuweisen, ihn zu warnen oder ihm den nächsten Arbeitsschritt anzuzeigen. Auch im Tiefbau kann ein solches System z.B. einem Baggerfahrer anzeigen, in welchem Bereich er graben muss oder wo Rohre und Leitungen verlaufen, die nicht beschädigt werden dürfen. Eine Beschädigung dieser kann abgesehen vom finanziellen Schaden auch eine Gefahr für die Arbeiter darstellen, wenn z.B. ein Gasleck dadurch entsteht. Genau hierum dreht sich das Projekt Digitalisierung und Smart Services im Tiefbau (DigSmart), welches einen ersten Schritt der Digitalisierung des Tiefbaus ermöglichen und untersuchen soll. Ein Laserprojektor unterstützt den Baggerfahrer bei seinen Arbeiten, indem er den o.g. Verlauf von Rohren, Leitung etc. visualisiert. Da auch eine falsche Visualisierung zu den bereits beschriebenen Problemen und Gefahren führen kann, ist eine genaue Visualisierung notwendig. Hierzu ist es wichtig, dass das Augmented Reality System die Projektionsfläche, die Sicht der Arbeiter und die Ausrichtung des Baggers und der Sensoren miteinbezieht, da nur so eine genaue Visualisierung möglich ist. Vernachlässigt das System z.B. die Form des Objektes, auf welches der Projektor projiziert, und die Perspektive des Betrachters, so kann die Projektion für den Betrachter verformt wirken, was Abbildung 1.1 veranschaulicht. Abbildung 1.1(a) zeigt eine auf ein ebenes Objekt projizierte Tanne, während Abbildung 1.1(b) die gleiche Projektion zeigt, jedoch befindet sich eine Tasche auf dem Tisch, wodurch das Objekt nicht mehr eben ist. Trotz der Tasche sollte sich die Projektion für den Betrachter nicht verändern, sondern die Form aus Abbildung 1.1(a) beibehalten. Stellt die Tanne zum Beispiel den Verlauf einer Leitung dar, so sieht der Verlauf durch die Tasche anders aus, was zu den o.g. Gefahren und Problemen führen kann.

Um die Form des Objektes in die Projektion miteinzubeziehen, muss dessen Form erfasst und die



(a) Erwartete Projektion

(b) durch Hindernis erzeugt Verformung der Projektion

**Abbildung 1.1:** Projektion einer Tanne auf einen Tisch

Projektion daran angepasst werden. Die Form kann durch einen Laserscanner erfasst werden. Da sich die Perspektive des Laserscanners vom Projektor unterscheidet, ist es notwendig die erfasste Form aus der Perspektive des Laserscanners in die Perspektive des Projektors zu bringen. Nicht nur die Form spielt eine Rolle für die Projektion, sondern auch welche Daten am Arbeitsplatz visualisiert werden sollen. Gibt es z.B. Bauarbeiten in einer Straße, wäre es schlecht, wenn der Projektor die Leitungen und Rohre aus der Nachbarstraße projiziert. Daher ist eine Lokalisierung des Baggers wichtig. An sich kann bereits mit einem Laserscanner die Lokalisierung durchgeführt werden, jedoch kann der Einsatz von Kameras die Genauigkeit der Lokalisierung erhöhen. Daher sollen in diesem Projekt auch mehrere Kameras zum Einsatz kommen. Genauso wie Laserprojektor und Laserscanner haben auch die Kameras eine eigene Perspektive auf die Szene. Jede Perspektive impliziert ein eigenes lokales Koordinatensystem. Damit die Sensordaten in die Perspektive eines anderen Sensors übertragen werden kann, muss die Transformation zwischen den Perspektiven und damit zwischen Koordinatensystemen bekannt sein. Die Berechnung dieser Transformationen ist Teil der Kalibrierung der Sensoren. Die Kalibrierung berechnet die Transformation zwischen zwei Koordinatensystemen.

Genau das thematisiert diese Thesis, sie untersucht die Qualität der Kalibrierung mit den in Kapitel 2 vorgestellten Methoden. Die für die Kalibrierung verwendete Software beschreibt

Kapitel 3, während Kapitel 4 Experimente vorstellt, die die Qualität der Kalibrierung untersuchen und deren Ergebnisse darlegt. Abschließend fasst Kapitel 5 die Ergebnisse dieser Arbeit zusammen und gibt einen Ausblick auf die Fortführung des DigSmart Projektes.

# Kapitel 2

# Grundlagen

Im Zuge dieser Thesis wird ein Mini-Bagger mit einem Lidar Ouster “OS1 - 64” Laserscanner, einem Z-Laser ZLP1 Projektor und einer Flir “Blackfly S GigE” Kameras ausgestattet. Die Funktionsweise der jeweiligen Sensoren sind in den folgenden Abschnitten genauer erläutert. Wie auf dem Bild 4.1(a) zu sehen ist, befinden sich die Sensoren auf der Fahrerkabine. Bei diesem Bagger ist es möglich den Arm zur Seite zu klappen, wodurch dieser zeitweise die Sicht der Sensoren in einem Bereich weniger einschränkt.

## 2.1 Vorstellung der Hardware

### 2.1.1 Flir Kamera

Bei der montierten Kamera handelt es sich um die Blackfly S GigE (Modell: BFS-PGE-23S3C-C) von der Firma Teledyne Flir. Die technischen Spezifikationen sind die Folgenden und stammen aus [16]: Die Kamera hat eine Auflösung von $1920 \times 1200$, eine Bildrate von 53 fps und kann nur sichtbares Licht (Wellenlänge $380 - 720\,\text{nm}$) verarbeiten. Des Weiteren verwendet sie einen Sony IMX392 CMOS Sensor. Die Stromversorgung erfolgt durch Power over Ethernet (PoE).

Im Folgenden ist die Funktionsweise von Kameras dargelegt. Die Grundlagen basieren auf Kapitel 2.2 der Dissertation von Borrmann [3] und auf Kapitel 3.1.1 der Dissertation von Leutert [14].

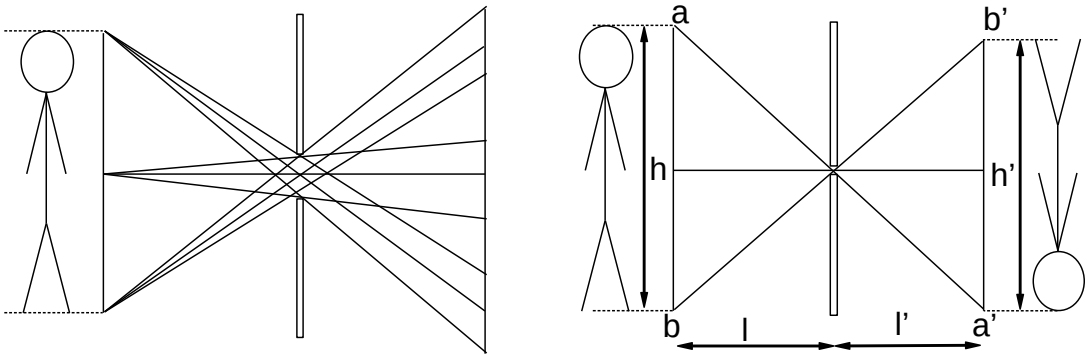
Eine Digitalkamera erzeugt Bilder, indem ein Linsensystem Lichtstrahlen auf eine Sensorfläche bündelt. Hierbei beschränkt eine Blende, wie viel Licht in das System einfällt. Bei den verwendeten Sensoren handelt es sich heutzutage meistens um Complementary Metal–Oxide–Semiconductor (CMOS)- oder Charge-Coupled Device (CCD)-Sensoren.

Um die Funktionsweise einer Kamera zu beschreiben, greift die moderne Physik auf zwei Modelle zurück: das Lochkamera-Modell (engl. pinhole camera Modell) und die Modellierung von optischen Linsen. Bei einer Lochkamera fällt Licht durch ein kleines Loch und trifft anschließend auf eine dahinter liegende Fläche (die Bildebene), wodurch dort ein Bild zu sehen ist. Je kleiner die Öffnung, desto schärfer und dunkler ist die Abbildung auf der Bildebene. Dieses Prinzip ist in Abbildung 2.1 zu sehen. Bei (b) ist die Öffnung kleiner als bei (a), wodurch weniger Strahlen

vom selben Punkt durch die Öffnung fallen und somit die Abbildung schärfer und dunkler ist. Nach dem Satz der "Ähnlichen Dreiecke" ergibt sich die Größe der Abbildung $h'$ aus:

$$h' = \frac{l' \cdot h}{l} \tag{2.1}$$

wobei $h$ die Größe des projizierenden Objekts, $l$ der Abstand zwischen Loch und Objekt und $l'$ der Abstand zwischen dem Loch und der Bildebene ist. Abbildung 2.1 zeigt auch, dass das projizierte Bild auf der Bildebene auf dem Kopf steht. Der Punkt $a$ bildet die Projektion auf den Punkt $a'$ ab, sowie den Punkt $b$ auf den Punkt $b'$.




(a) Erweiterte Lochkamera (b) Kalibrierkörer

**Abbildung 2.1:** Lochkamera Modell basierend auf [3, 7]

Die folgenden Abschnitte befassen sich mit sogenannten Sammellinsen. Eine Sammellinse bündelt das Licht auf einen Punkt. Dies ist in Abbildung 2.2 zu sehen. Die Sammellinse lenkt die Lichtstrahlen, die von dem Punkt (a) ausgehen, so ab, dass sie wieder im Punkt (a') zusammentreffen. An sich haben Linsen die Eigenschaft Licht abzulenken, weil sich die Geschwindigkeit von Licht je nach Medium verändert. Da sich Licht innerhalb der Linse langsamer bewegt, lenkt die Linse das Licht ab, was sich Brechung nennt. Die Form der Sammellinse sorgt dafür, dass die Linse die Lichtstrahlen auf einen Punkt fokussiert. An sich gibt es zwei Modellierungen von Linsen: dünne und dicke Linsen. Im Modell der dünnen Linse, lenkt die Linse das Licht nur einmal ab, sodass die Strahlen, die sich parallel zur optischen Achse bewegen, durch den Punkt $F'$ verlaufen. Die Strahlen, die durch den Punkt $F$ verlaufen, lenkt die Linse so ab, dass sie danach parallel zur optischen Achse verlaufen und der Strahl, der durch das optische Zentrum $O$ der Linse fällt, erfährt keine Ablenkung. Durch die Bündelung der Strahlen bildet sich bei einer idealen Linse ein scharfes Bild in Abstand $l'$ zur Linse. Aus dem "Satz der ähnlichen Dreiecke" ergibt sich

$$m = \frac{h'}{h} = \frac{l'}{l} = \frac{x'}{f'} = \frac{f}{x} \tag{2.2}$$

wobei $m$ der Vergrößerung entspricht. Im dicken Linsenmodell erfährt der Lichtstrahl zwei Ablenkungen: einmal beim Eintritt in die Linse und einmal beim Austritt. Hierbei geht man aber davon aus, dass die Ablenkung die Gleiche wie bei der dünnen Linse ist und somit die gleichen

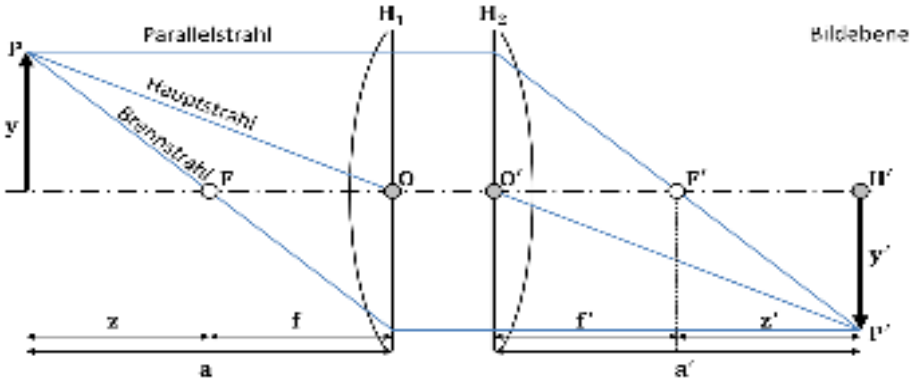



**Abbildung 2.2:** Modellierung mit dünner Linse [14]

mathematischen Grundlagen der dünnen Linse auch auf die dicke Linse zutreffen. Gleiches gilt auch für ein System mit mehreren Linsen, wie es häufig bei Kameras der Fall ist.

### 2.1.2 Laserscanner

Dieser Abschnitt beschreibt die Funktionsweise des verwendeten Laserscanners. Die Grundlagen basieren auf Kapitel 2.4 von [3]. Im Gegensatz zu Kameras generieren Laserscanner ein 3D-Abbild ihrer Umgebung, d.h. er bildet 3D-Koordinaten auf 3D-Koordinaten ab. Hierzu sendet der Laserscanner einen oder mehrere Lichtimpulse aus und misst die Dauer $t$ bis ein Reflexionssignal empfangen wird. Generell bewegt sich Licht mit einer Geschwindigkeit von $c = 2,99792458 \cdot 10^8 \frac{m}{s}$. Das Licht hat somit $\frac{t}{2}$ benötigt, um ein Objekt zu erreichen. Die zurückgelegte Strecke $s$ ergibt sich aus $s = \frac{c \cdot t}{2}$. Diese Methode trägt den Namen “direct time measurement”.

Im Allgemeinen ist es möglich, dass der Empfänger des Laserscanners mehrere Lichtimpulse erhält. Dies kann durch Teilreflexionen entstehen. Das bedeutet, trifft der Lichtstrahl $A$ auf eine Kante oder eine Licht durchlässige Fläche, so teilt sich der Strahl in die Strahlen $B$ und $C$ auf. Der Strahl $B$ bewegt sich weiter in die gleiche Richtung wie ursprünglich $A$, während die Kante $C$ reflektiert und zurück zum Laserscanner sendet. Trifft $B$ anschließend wieder auf eine Kante oder solch eine Fläche, so teilt sich $B$ wieder auf. Falls $B$ aber auf eine lichtundurchlässige Fläche trifft und nicht nur deren Kante, so reflektiert diese $B$ vollständig (abgesehen von absorbierten Teilen des Lichtstrahls). Hieraus ergeben sich mehrere Impulse, die der Empfänger des Laserscanners wahrnimmt, wodurch ein ausgesendeter Lichtimpuls mehr Informationen liefert. Dies passiert zum Beispiel auch bei Regen. Um Regentropfen herauszufiltern verwendet der hier verwendete Laserscanner das stärkste, empfangene Signal [8].

Der Laserscanner liefert nicht nur Informationen über die Distanz zum erfassten Objekt, sondern auch dessen Reflektivität $r \in [0, 1]$. Diese lässt sich aus dem Intensitätsunterschied zwischen dem ausgesendeten und empfangenen Lichtimpuls berechnen. Liegt die Reflektivität bei 1, so gibt es keinen Intensitätsunterschied zwischen gesendeten und empfangenen Impuls, d.h. das Objekt hat den Strahl vollständig reflektiert. Falls der Empfänger keinen Impuls wahrnimmt, so ist die Reflektivität gleich 0. Die Reflektivität ist abhängig vom Material und der Farbe des Objekts als

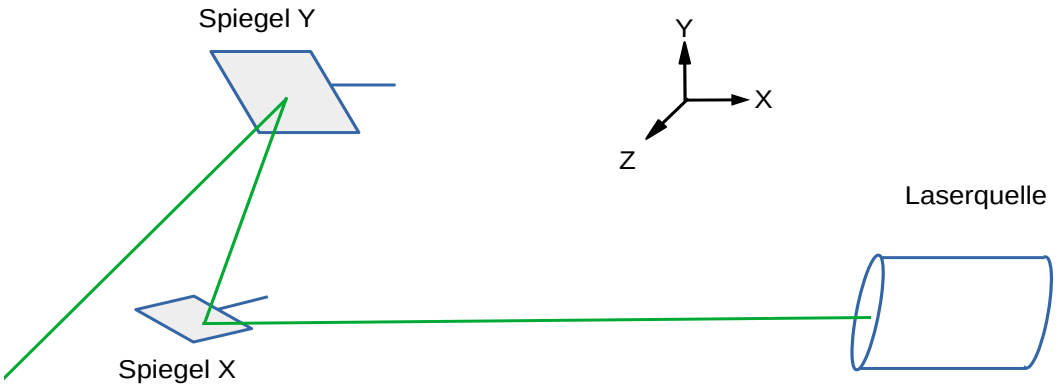



**Abbildung 2.3:** Projektor Modell - Der Laserstrahl wird durch Spiegel x abgelenkt, sodass der Strahl auf Spiegel y verschoben entlang der x-Achse aufkommt. Spiegel y reflektiert den Strahl anschließend, sodass der Strahl entlang der y-Achse verschoben wird.

auch vom Auftrittswinkel und dessen Entfernung. Wie bereits zu Beginn dieses Kapitels erwähnt, handelt es sich bei dem hier verwendeten Laserscanner, um einen Ouster OS1-64 Lidar mit einer vertikalen Auflösung von 64 Strahlen. Hierbei handelt es sich um einen so genannten "Puls LiDAR", der mit der o.g. Methode Abstände misst. Die technischen Spezifikationen stammen aus [12]. Die Horizontale Auflösung $hres \in \{512, 1024, 2048\}$ ist konfigurierbar, diese Arbeit verwendet jedoch $hres = 512$. Des Weiteren hat der Laserscanner einen vertikalen Öffnungswinkel von insgesamt $33,2°$, d.h. $16,6°$ nach oben und nach unten. Die Wellenlänge des Lasers beträgt 850 nm und hat eine Reichweite von $0,8\,\text{m} - 40\,\text{m}$ bei einem 10 % reflective lambertian target und 100 klx Sonnenlicht.

### 2.1.3 Laserprojektor

Bei dem Laserprojektor handelt es sich, um den ZLP 1 Projektor von dem Unternehmen Z-Laser. Die technischen Spezifikationen stammen aus [28]. Der Projektor hat eine Genauigkeit von $\pm 3mm$ in einem Arbeitsbereich zwischen ein bis drei Metern. Die prinzipielle Funktionsweise ist in Abbildung 2.3 dargestellt. Für die Projektion verwendet der Projektor zwei Spiegel: Spiegel x steuert den Laser entlang der x-Achse, Spiegel y entlang der y-Achse. Sei das Koordinatensystem rechts-händisch und die z-Achse deute in Richtung Projektionsebene und die y-Achse nach oben. Spiegel x ist der erste Spiegel, auf den der Laserstrahl trifft. Dieser ist nur in der Lage sich um die z-Achse zu drehen. Der Spiegel lenkt den Laserstrahl nach oben ab und der Strahl trifft anschließend auf Spiegel y. Spiegel x ist so angebracht, dass die Ausrichtung festlegt, welchen x-Wert der Auftrittspunkt des Lasers auf Spiegel y und somit auch auf der Projektionsebene hat. Der zweite Spiegel kann sich nur um die x-Achse drehen, d.h. der Spiegel lenkt den Laserstrahl je nach Ausrichtung in die positive bzw. negative y-Richtung ab. Somit ist Spiegel y für den y-Wert des Auftrittspunktes verantwortlich.

Im Allgemeinen kann nach [14] der Laserprojektor als eine inverse Kamera modelliert werden. Anstelle 3D-Koordinaten auf 2D-Koordinaten abzubilden, bildet der Laserprojektor 2D-Koordinaten auf 3D-Koordinaten ab. Der Projektor kann aber nur auf Flächen projizieren, d.h. die Projektion

ist davon abhängig wie die Fläche aussieht, in welchem Winkel der Projektor zur Fläche steht und wo sich der Beobachter befindet. Je nach Position des Beobachters und Flächenform kann die Projektion verzerrt oder sogar lückenhaft sein.

## 2.2 Kalibrierungsalgorithmen

Nach [3, S. 156] ist es notwendig, dass die geometrischen Eigenschaften der Sensoren bekannt sind, um diese in einem System zu kombinieren. Das Bestimmen von diesen Eigenschaften nennt sich Kalibrierung. Dabei besteht die Gesamtkalibrierung aus zwei Schritten, der intrinsischen und der extrinsischen Kalibrierung. Bei der intrinsischen Kalibrierung geht es um die interne Kalibrierung, also wie bildet sich ein Punkt aus dem einen Koordinatensystem z.B. dem Weltkoordinatensystem in das andere Koordinatensystem z.B. das Kamerakoordinatensystem ab. Sind nur die Daten von einem Sensor nötig, genügt diese Kalibrierung. Wenn ein Programm jedoch von mehreren Sensoren die Daten benötigt, dann muss es erst die Daten in ein Koordinatensystem zusammenführen. Hierfür ist die extrinsische Kalibrierung gedacht. Sie bestimmt die relative Position von zwei gegebenen Sensoren und ermöglicht das Zusammenführen der Sensordaten. Die extrinsische Kalibrierung verwendet dabei Referenzpunkte aus den lokalen Koordinatensystemen der Sensoren, die sich auf das selbe Objekt beziehen. Die Kalibrierung der Kamera verwendet hierbei ein Schachbrettmuster, wobei dessen geometrische Eigenschaften bekannt sind: die Dimension des Schachbrettes, also die Anzahl der Schachbrettecken in der Breite und Höhe, und die Dimension der Schachbrettfelder (Breite und Höhe), sowie die Koordinaten der jeweiligen Schachbrettecken. Um den Laserprojektor extrinsisch zu kalibrieren, kommen Reflektoren zum Einsatz von denen die 3D-Koordinaten als auch die 2D-Koordinaten im Projektorkoordinatensystem bekannt sein müssen. Diese Arbeit kalibriert die Kamera zum Laserscanner und den Laserprojektor zum Laserscanner. Das Koordinatensystem des Laserscanners ist in dieser Arbeit äquivalent zum Weltkoordinatensystem. Aus der relativen Position von zwei Sensoren zum Laserscanner ergibt sich die relative Position zwischen den Sensoren durch die Differenz der relativen Positionen zum Laserscanner. Sei also $\vec{p_1}$ die relative Position von Sensor $s_1$ zum Laserscanner und $\vec{p_2}$ die relative Position von Sensor $s_2$ zum Laserscanner, dann ist die relative Position $p_{12}$ von Sensor $s_1$ nach Sensor $s_2$ definiert als

$$p_{12} = \vec{p_1} - \vec{p_2} \tag{2.3}$$

Da diese Arbeit die Kamera bzw. den Laserprojektor zum Laserscanner extrinsisch kalibriert, ergibt sich das Problem, dass durch das Rauschen des Ouster Laserscanners die gemessenen Punkte nicht akkurat sind. Daher stabilisiert diese Arbeit die Messungen des Laserscanners zunächst, damit die Messergebnisse konsistent bleiben. Weiterhin verwendet diese Arbeit ein Modell, um die Pose des Kalibrierobjekts und damit die Koordinaten der Referenzpunkte zu bestimmen. Dies bestimmt diese Arbeit, indem das Preprocessing das Modell an das Kalibrierobjekt in der Punktwolke anpasst, was den Einfluss des Rauschens auf die bestimmten Referenzpunkte reduziert.

Der Laserprojektor und der Ouster Laserscanner sind intrinsisch werkskalibriert, d.h. für diese beiden Sensoren ist nur eine extrinsische Kalibrierung notwendig, weshalb diese Arbeit die intrinsische Kalibrierung dieser Sensoren nicht näher behandelt.

Die Gesamtkalibrierung des Systems läuft also wie folgt ab:

1. intrinsische Kamerakalibrierung
2. Punktwolken Aggregation
3. Bestimmung der Pose des Kalibrierobjekts
4. extrinsische Kamerakalibrierung
5. extrinsische Projektorkalibrierung

Die folgenden Abschnitte dieses Kapitels orientieren sich an der Reihenfolge der Gesamtkalibrierung. Jedoch sind die intrinsische und extrinsische Kamerakalibrierung in einem Abschnitt zusammengefasst, nämlich Abschnitt 2.2.3. Der erste Abschnitt 2.2.1 beschäftigt sich aber mit den hier verwendeten Ansätzen, um das Rauschen des Ousters zu reduzieren bzw. konsistenter zu machen. Da die extrinsische Kalibrierung die Pose des Kalibrierobjekts benötigt, beschreibt Abschnitt 2.2.2 die Grundlagen wie diese bestimmt wird. Abschließend beschreiben die Abschnitte 2.2.3 und 2.2.4 die bereits erwähnte Kalibrierung der Kamera und die extrinsische Kalibrierung des Laserprojektors.

### 2.2.1 Punktwolken Stabilisierung

Wie bereits erwähnt, besitzt der o.g. Laserscanner ein relativ starkes Rauschen. Um eine gute Kalibrierung zu gewährleisten, ist es wichtig, dass die Pose des Kalibrierobjekts bekannt ist. Das Rauschen behindert jedoch die Bestimmung der Pose. Um dieses Problem zu beheben, untersucht diese Arbeit mehrere Stabilisierungsmaßnahmen, die die folgenden Abschnitte beschreiben.

#### Arithmetisches Mittel

Sei $M \subset \mathbb{R}$, dann ist der Mittelwert auf $\mathbb{R}$ definiert durch:

$$\bar{m} = \frac{\sum_{x \in M} x}{|M|} \tag{2.4}$$

Auf den $\mathbb{R}^3$ Vektorraum berechnet sich der Mittelwert über mehrere Vektoren, indem für die jeweiligen Vektorkomponenten der Mittelwert über die Vektoren gebildet werden. Sei $\vec{a}, \vec{b} \in M \subset \mathbb{R}^3$, dann ist der Mittelwert $\vec{m}$ in diesem Fall:

$$\bar{m} = \frac{\vec{a} + \vec{b}}{2} = \frac{\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}}{2} = \begin{pmatrix} \frac{a_1+b_1}{2} \\ \frac{a_2+b_2}{2} \\ \frac{a_3+b_3}{2} \end{pmatrix} \tag{2.5}$$

Der Mittelwert zeichnet sich dadurch aus, dass dieser die Gesamtheit der gesammelten Punktwolken besser repräsentiert, da alle Werte das gleiche Gewicht bekommen. Jedoch haben große Abweichungen in den Messwerten einen großen Einfluss auf den Mittelwert. Daher kann der Median in solchen Situationen eine bessere Wahl sein.

### Median

Der Median ist der mittlere Wert einer geordneten Menge. Zum Beispiel wäre aus der Menge $M = \{1, 2, 3, 4, 10000\}$ die 3 der mittlere Wert, der Mittelwert ist hier jedoch $\bar{m} = 2002$. Im $\mathbb{R}^3$ gibt es nicht direkt einen mittleren Wert, da ein Element aus drei Komponenten besteht. Daher definiert diese Arbeit den Median im $\mathbb{R}^n$ folgendermaßen: Sei $med(M)$ die Funktion, die den Median der Menge $M \subset \mathbb{R}$ bestimmt, dann ist der Median der Menge $P \subset \mathbb{R}^n$ mit $n \in \mathbb{N}$ definiert als

$$med_n : 2^{\mathbb{R}^n} \to \mathbb{R}^n \tag{2.6}$$

$$med_n(P) = \vec{a} \tag{2.7}$$

$$\vec{a}_i = med(\{\vec{x}_i | \vec{x} \in P\}) \tag{2.8}$$

mit $i < n \wedge i \in \mathbb{N}$. Hier entspricht $\vec{a}_i$ der $i$-ten Komponente von a.

### Punktwolken Vereinigung

Als Alternative gibt es noch die Möglichkeit mehrere aufgezeichnete Punktwolken in eine Punktwolke zu vereinigen. Angenommen die Messfehler sind normalverteilt und unabhängig voneinander, dann häufen sich die Messpunkte im direkten Umfeld des eigentlichen Messpunktes. Der Iterative Closest Points (ICP) Algorithmus findet dann eine Transformation, sodass das transformierte Modell sich in der Mitte der Messpunkte befindet.

## 2.2.2 Bestimmung der Kalibrierobjekt Pose

Die extrinsische Kalibrierung von Kamera und Laserprojektor benötigt die 3D Koordinaten der verwendeten Referenzpunkte (Reflektor, Schachbrettecken). Diese haben eine fixe Position auf dem Kalibrierobjekt, d.h die 3D-Position lässt sich aus der Pose des Kalibrierobjektes herleiten. Um die Pose des Kalibrierobjekts zu bestimmen, verwendet diese Arbeit die Methode von [26].

Der Algorithmus erhält als Eingabe eine 3D-Punktwolke vom Laserscanner, in welchem sich das Kalibrierobjekt befindet. Die Methode von [26] verwendet ein auf einem Gestell befestigtes Brett mit einem aufgedruckten Schachbrettmuster. Das Kalibrierobjekt befindet sich an einer festen Position und hat eine fixe Höhe und Breite. Das machen sich die Autoren zu nutzen und schneiden die gegebene Punktwolke auf das Kalibrierobjekt zu. Übrig bleibt nur das Kalibrierobjekt. Dies ermöglicht dem RANdom SAmple Consensus (RANSAC) Algorithmus eine Ebene zu bestimmen, welche im Kalibrierobjekt liegt. Alle Punkte, die in einem gewissen Bereich um die Ebene liegen, verarbeitet die Methode von [26] weiter. Hierzu reduziert sie die Anzahl der Punkte weiter, indem sie durch das DB-Scan Clustering sowie dem Voxel-Downsampling Verfahren weitere Punkte entfernt. Aus den übrig gebliebenen Punkten bestimmt der Algorithmus eine initiale Schätzung der Pose. Abschließend verfeinern sie die Schätzung durch den ICP Algorithmus.

Die in dieser Arbeit verwendete Methode unterscheidet sich etwas zu [26], da diese Arbeit ein Quader anstelle eines Brettes verwendet. Wie auch in [26] schneidet der hier verwendete Algorithmus zunächst die Punktwolke auf das Kalibrierobjekt zu. Um die initiale Schätzung der

Pose zu bestimmen, benötigt [26] nur eine Ebene. Die hier verwendete Methode bestimmt nun diese Ebene, indem zunächst der RANSAC Algorithmus auf den Quader angewendet wird. Da nur zwei der langen Seiten im Laserscan vorkommen, findet RANSAC eine der beiden. Anschließend entfernt der Algorithmus die Punkte, die sich in einem gewissen Abstand zu der bestimmten Ebene befinden. Durch das erneute Ausführen dieser beiden Schritte findet RANSAC die nächste große Ebene und mit einem weiteren letzten Mal die kleine Ebene auf der Oberseite des Quaders. Mit Hilfe des Centroids – also dem Mittelwert aller Punkte der Ebene – und dem Durchschnitt der Normalenvektoren der Punkte der jeweiligen Ebene kann der Algorithmus unterscheiden, welche der langen Ebenen die Methode von [26] nutzen soll. Die Normalenvektoren der Punkte berechnen sich aus dem lokalen Umfeld des jeweiligen Punktes. Hierzu nutzt der Algorithmus eine begrenzte Anzahl an Nachbarn und berechnet auf deren Basis den Normalenvektor des jeweiligen Punktes.

Damit das von den Normalenvektoren aufgespannte Koordinatensystem rechtshändisch ist, korrigiert der Algorithmus die Ausrichtung des Normalenvektors der Ebene. Die Methode zur Orientierungskorrektur basiert auf der Annahme, dass die berechneten Normalenvektoren annäherungsweise den tatsächlichen Normalenvektoren der Ebene entsprechen und somit in einem 90° Winkel zu deren Ebene stehen. Da sich der Normalenvektor je Ebene aus dem Mittelwert mehrerer Normalenvektoren berechnet, macht dies die Ausrichtung stabiler. Zur Korrektur addiert der Algorithmus den Normalenvektor auf den Centroid der Ebene. Sollte das Ergebnis eine größere Distanz zum Centroid der kleinen, oberen Ebene haben als der Centroid der betrachteten Ebene, so ist der Normalenvektor nach Außen und somit richtig orientiert. Andernfalls hat der invertierte Vektor die richtige Ausrichtung. Dies gilt nur wegen der Annahme, dass die Normalenvektoren rechtwinklig zu deren Ebenen stehen. Die relevante Seite für [26] ist hier die rechte Seite des Quaders. Diese Seite findet der Algorithmus, indem er den Winkel zwischen dem Normalenvektor der aktuell betrachteten Ebene und dem Kreuzprodukt zwischen dem Normalenvektoren der kleinen Seite mit der anderen langen Seite berechnet. Ist dieser kleiner als 90° dann ist die aktuelle Ebene die relevante Ebene. Also sei $n_0$ der Normalenvektor der kleinen, oberen Seite des Quaders und $n_1$ und $n_2$ die Normalenvektoren von den großen Seiten. Damit $n_2$ die relevante Seite ist, muss folgendes gelten:

$$\arccos\left(\frac{(n_0 \times n_1) \cdot n_2}{||(n_0 \times n_1)|| \cdot ||n_2||}\right) < 90° \tag{2.9}$$

Mit der relevanten Seite findet die Methode von [26] die initiale Pose des Quaders.

#### Random Sample Consensus

Wie bereits erwähnt verwendet die Arbeit [26] den RANSAC Algorithmus, um eine Ebenen-Segmentierung durchzuführen. Nach [3] beschreibt RANSAC hierbei mehrere zufallsbasierte Algorithmen, welche bzgl. auf die Ebenen-Segmentierung eine gute Beschreibung für die vorhandenen Ebenen findet, sodass möglichst viele Punkte in dieser Ebene liegen. Hierzu betrachtet der Algorithmus drei zufällig bestimmte Punkte aus der Punktwolke und bestimmt die Ebenengleichung der hierdurch aufgespannten Ebene. Befinden sich genug Punkte in dem festgelegten Umfeld der Ebene, so terminiert der Algorithmus. Andernfalls wiederholt der Algorithmus das Prozedur bis er eine gute Lösung findet oder eine maximale Anzahl an Wiederholungen durchgeführt hat.

Sei $P = \{p_1, p_2, p_3\}$ die o.g. minimale Stichprobe an Punkten aus der Punktwolke $M$. Dann ist die Ebenengleichung laut [3, S.95] und [22] definiert als:

$$E = \vec{n} \cdot (\vec{x} - \vec{p_1}) = \frac{\vec{a} \times \vec{b}}{|\vec{a} \times \vec{b}|} \cdot (\vec{x} - \vec{p_1}) \tag{2.10}$$

mit $\vec{a} = \vec{p_3} - \vec{p_2}$ und $\vec{b} = \vec{p_1} - \vec{p_2}$ und $\vec{x} \in \mathbb{R}^3$. Mit dieser Gleichung prüft der Algorithmus nun die Qualität dieser Ebene, indem er die Menge $M_h = \{q | q \in M \wedge d_q <= t\}$ bestimmt, wobei $d_q$ den Abstand zur Ebene $E$ und $t \in \mathbb{R}$ den o.g. maximalen Abstand beschreibt, den der Punkt $q$ zur Ebene haben darf. Hat $M_h$ eine ausreichende Größe von mindestens $s_{min}$, gilt also $s_{min} <= |M_h|$, so terminiert der Algorithmus, ansonsten zieht er eine neue Stichprobe $P$ und berechnet für diese wieder $M_h$. Dies wiederholt der Algorithmus maximal $i$-mal oder bis $M_h$ ausreichend groß ist. Zum Schluss berechnet der Algorithmus noch die Ebene, die $M_h$ am besten beschreibt. Somit benötigt der Algorithmus die folgenden Parameter: die Punktwolke $M$, den maximalen Abstand $t$, die Mindestanzahl an Punkten um $M_h$ zu akzeptieren und die maximale Anzahl $i$ an Iterationen.

### DB-Scan

Der RANSAC Algorithmus liefert eine Menge an Punkten, die zur gefundenen Ebene passen. Der DB-Scan Algorithmus verfeinert diese Menge, indem er Ausreißer entfernt. Hierbei entspricht das größte gefundene Cluster der Ebene. Nach [23] stellt Algorithmus 1 eine zusammengefasste Version des Algorithmus dar. Im Endeffekt bestimmt der Algorithmus sogenannte Kernpunkte. Damit ein Punkt $p$ ein Kernpunkt ist, muss in der Umgebung dieses Punktes mindestens $minPts$ Punkte liegen, die jeweils höchstens einen Abstand von $\epsilon$ zu $p$ haben. Der Kernpunkt und dessen Nachbarn sind ein Cluster. Der Algorithmus erweitert dieses Cluster, indem er die Nachbarpunkte des Clusters zum Cluster hinzufügt, falls diese ebenfalls im besagten Umfeld liegen. Alle Punkte, die am Ende zu keinem Cluster gehören, sind als Rauschen klassifiziert.

**Algorithm 1** Abstract DBSCAN Algorithm[23]

**Require:** Datenpunkte $DB$
**Require:** Radius $\epsilon$
**Require:** mindest Anzahl an Punkten $minPts$
**Require:** Distanzfunktion $dist$
  Bestimme Nachbarn von jedem Punkt und identifiziere Kernpunkte
  Vereinige benachbarte Kernpunkte zu einem Cluster
  **for all** nicht-Kernpunkte p **do**
    **if** Abstand zum nächsten Kernpunkt ¡= $\epsilon$ **then**
      Füge p zum benachbarten Kernpunkt hinzu
    **else**
      füge p zur Rauschen-Menge hinzu
    **end if**
  **end for**

### Voxel-Downsampling

Wie bereits erwähnt reduziert die Methode von [26] die Punktmenge noch einmal zusätzlich, indem das Voxel-Downsampling Verfahren von [31] die verbliebenen Punkte zusätzlich reduziert. Hierzu gruppiert der Algorithmus die 3D-Punkte, die räumlich beieinander liegen, in sogenannte Voxel. Der Mittelwert aller Punkte im Voxel ergibt dann den neuen Punkt.

### Principal Component Analysis

Principal Component Analysis (PCA) liefert die geschätzte Pose für die übriggebliebenen Punkte. Die Grundlagen dieses Abschnitts basieren auf [1, s. Kapitel 4]. Bei PCA handelt es sich um eine Methode zur Reduktion von Dimensionen. PCA berechnet hierfür die Kovarianz Matrix der Daten. Auf 3D-Punktwolken bezogen, handelt es sich um eine $3 \times 3$ Matrix, da es hier drei Dimensionen gibt, nämlich die Dimensionen des $\mathbb{R}^3$ Vektorraums. Also beziehen sich die Variablen auf die x, y und z-Komponente der Punkte. Die Kovarianzmatrix $CM$ ist eine quadratische Matrix und ist definiert als:

$$CM = \begin{pmatrix} CoVar(x,x) & CoVar(x,y) & CoVar(x,z) \\ CoVar(y,x) & CoVar(y,y) & CoVar(y,z) \\ CoVar(z,x) & CoVar(z,y) & CoVar(z,z) \end{pmatrix} \tag{2.11}$$

wobei $CoVar(q,p)$ die Kovarianz von $q,p$ ist. Im Allgemeinen ist die Kovarianz symmetrisch, d.h. es gilt:

$$CoVar(q,p) = CoVar(p,q) \ . \tag{2.12}$$

Weiterhin gilt

$$CoVar(q,q) = Var(q) \tag{2.13}$$

wobei $Var(q)$ die Varianz von $q$ ist. Die Kovarianz zwischen einer Variable und sich selbst entspricht also der Varianz von der Variable. Durch Einsetzen der Gleichungen 2.12 und 2.13 in 2.11 ergibt sich:

$$CM = \begin{pmatrix} Var(x) & CoVar(x,y) & CoVar(x,z) \\ CoVar(x,y) & Var(y) & CoVar(y,z) \\ CoVar(x,z) & CoVar(y,z) & Var(z) \end{pmatrix} \tag{2.14}$$

d.h. die Matrix selbst ist symmetrisch da $CM = CM^T$ gilt, wobei $CM^T$ die transponierte Matrix von $CM$ ist.

Da die Kovarianz Matrix quadratisch ist, besitzt sie Eigenvektoren und Eigenwerte. Die Eigenvektoren der Kovarianz Matrix werden auch als Principal Components bezeichnet. Nach [1] geben die Eigenvektoren die Richtung der größten Varianz in den Datenpunkten an, während der Eigenwert die Signifikanz eines Eigenvektors angibt. Umso größer die Varianz der Datenpunkte ist, desto höher ist auch der entsprechende Eigenwert. Die verwendeten Kalibrierobjekte haben rechteckige, nicht quadratische Flächen. Ein generiertes Modell samt eingezeichneten Eigenvektoren ist in Abbildung 2.4 zu sehen. Die Eigenvektoren zeigen in Richtung Länge, Breite und Tiefe des Modells. Der Eigenvektor mit dem kleinsten Eigenwert zeigt von der Ebene weg und ist orthogonal

**Abbildung 2.4:** Eigenvektoren einer generierten Ebene. Die rote Linie entspricht dem Normalenvektor der Ebene.

zu dieser, da alle Datenpunkte in einer Ebene liegen und somit die Varianz in dieser Richtung am kleinsten ist. Er entspricht hier auch dem Normalenvektor der Ebene. Hat die Ebene zu viele Punkte in der Tiefe, dann ist nicht mehr gewährleistet, dass der Normalenvektor orthogonal zur Ebene ist. Den zweitgrößten Eigenwert hat der Eigenvektor in Richtung Breite, während der Eigenvektor in Richtung Länge den größten Eigenwert besitzt.

In der Praxis treten bei einem Laserscan Messfehler auf, wodurch Länge, Breite und Tiefe des gescannten Objekts variieren können. Dies beeinflusst auch die Ausrichtung der Eigenvektoren. Das Clustering-Verfahren und das Downsampling in den vorherigen Schritten reduziert jedoch dieses Rauschen, weshalb der Laserscan mehr dem hier dargestellten idealen Modell entspricht.

Die berechneten Eigenvektoren repräsentieren nicht nur die Pose der gefundenen Ebene, sondern entsprechen auch einer Transformation eines Koordinatensystems in ein anderes. Befinde sich ein Modell dieser Ebene mit dem Centroid im Ursprung nur in der X-Y-Ebene. Die Eigenvektoren bilden dann eine Transformationsmatrix, die das Modell an die Position und Ausrichtung der gescannten Ebene transformiert. Da die gefundenen Eigenvektoren abhängig von der Qualität des Scans sind, ist die Transformation des Modells im Scan nur grob. Um die Qualität der Transformation zu verbessern, passt der im folgenden Abschnitt erklärte ICP Algorithmus, das transformierte Modell weiter an den Scan an.

### Iterative Closest Point Algorithm

Dieser Abschnitt beschreibt die theoretischen Grundlagen des ICP Algorithmus (vgl. 2) und basiert auf [3]. Der ICP Algorithmus ist einer der am häufigsten genutzten Algorithmen zur Punktwolken Registrierung. Als Eingabe benötigt der Algorithmus eine initiale Pose und zwei Punktwolken $M, D \subset \mathbb{R}^3$. Anschließend berechnet er durch Minimieren der Distanz korrespondierender Punkte

die neue Pose $(R, t)$ der Punktwolke. $R$ entspricht hier der Rotationsmatrix und $t$ der Translation der Punktwolke. Ein Punkt $a \in M$ korrespondiert mit einem Punkt $b \in D$ mit Abstand $x$, wenn es keinen Punkt $c \in D$ mit Abstand $y$ zu $a$ gibt, für den $y < x$ gilt – es also keinen anderen Punkt mit geringerem Abstand gibt. Da sich die Pose der Punktwolke während der Durchführung des Algorithmus ändert, können sich auch die Korrespondenzen ändern. Daher handelt es sich hier, um ein iteratives Verfahren bei welchem ICP $k$-mal zunächst die Punktkorrespondenzen bestimmt und anschließend den Fehler $E_{ICP}$ (vgl. Algorithmus 2) minimiert. Da der Algorithmus $E_{ICP}$ in jedem Schritt verringert, konvergiert ICP zu einem lokalen Minimum.

**Algorithm 2** Der ICP Algorithmus[3, vgl. S. 114]

**Require:** $k \in \mathbb{N}$, Punktwolken $M, D$
bestimme Punktkorrespondenzen $Z$, mit $N = |Z|$
minimiere Rotation $R$ und Translation $t$ mit $E_{ICP}(R,t) = \frac{1}{N} \sum_{i=0}^{N} ||m_i - (Rd_i + t)||^2$, wobei $m_i \in M, d_i in D$ gilt
iteriere Schritt (1), (2) k-mal
return pose(R,t)

Damit der Algorithmus bessere Ergebnisse liefert, verwendet eine Abwandlung des Algorithmus einen Parameter $t_{dist}$, der die erlaubte maximale Distanz einer Punktkorrespondenz festlegt. Ist die Distanz größer, so bezieht der Algorithmus die betroffenen Punkte nicht mit ein. Somit kann die Anzahl der Korrespondenzen in jedem Schritt variieren und der Fehler $E_{ICP}$ kann sogar ansteigen. Diese Methode verbessert die Ergebnisse bei Punktwolken mit Verdeckungen. Abbildung 2.5 zeigt ein solches Szenario. Der Laserscanner nimmt zwei Scans an den Stellen $A$ und $B$ auf. Die gestrichelten Linien markieren die Sicht des Laserscanners auf das Objekt. In den zwei Scans fehlt jeweils die lange Seite des anderen Scans, aber beide Scans haben die kurze Seite gemein. Ohne die Verwendung des Thresholds würde der ICP Algorithmus beide Scans so aneinander anpassen, dass die langen Seiten zum Großteil übereinander liegen. Durch den Threshold haben die langen Seiten einen kleinen Einfluss auf das Matching, da nur die kurzen Seiten in beiden Scans miteinander korrespondieren.

Diese Arbeit verwendet die ICP Implementierung von [31], welche auf [2] basiert.

### 2.2.3 Intrinsische und Extrinsische Kamerakalibrierung

Dieser Abschnitt beschreibt die Funktionsweise der intrinsischen und extrinsischen Kamerakalibrierung. Wie bereits erwähnt benötigt die Kalibrierung hierzu Referenzpunkte, deren geometrische Eigenschaften bekannt sind. Diese sind für die Kalibrierung nötig, da ohne diese kein Unterschied zwischen dem wahrgenommenen Bild und der Realität erkennbar ist. Für die Kamerakalibrierung kommt hierfür meistens ein Schachbrettmuster zum Einsatz, welches sich auf einer flachen Ebene befindet. Nach [30] lässt sich der Algorithmus für die Kalibrierung wie folgt gliedern:

1. Aufnehmen von mehreren Fotos eines Objektes, welches ein vordefiniertes aufgedrucktes Muster besitzt (hier Schachbrettmuster)

2. Detektion der Referenzpunkt, also Hauptmerkmale wie z.B. die Ecken der Schachbrettfelder

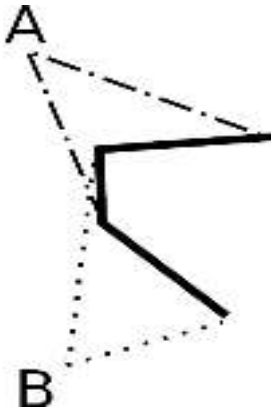



**Abbildung 2.5:** Scans werden an den Stellen (A) und (B) gemacht, aber können nur den markierten Bereich scannen.

3. Schätzen der intrinsische und extrinsischen Parameter durch den Direct Linear Transformation Algorithmus (DLT-Algorithmus) (vgl. Abschnitt 2.2.3)
4. Parameter durch Fehler-Minimierung verbessern.

Die folgenden Abschnitte beschreiben wie die Kamerakalibrierung die intrinsischen und extrinsischen Parameter bestimmt und basieren auf [4, 20, 30].

Aus den beiden Modellen aus Abschnitt 2.1.1 lässt sich die Abbildung eines Punktes im Weltkoordinatensystem ins Kamerakoordinatensystem wie folgt berechnen:

$$\begin{pmatrix} \bar{x} \\ \bar{y} \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x/z \\ y/z \\ 1 \end{pmatrix} \tag{2.15}$$

Die $3 \times 3$ Matrix wird als Kameramatrix bezeichnet und entspricht den intrinsischen Parametern der Kamera. Hierbei handelt es sich bei $f_x$ und $f_y$ jeweils, um die Brennweite in $x$ und $y$ Richtung, sowie bei $c_x$ und $c_y$ um die Verschiebung des Kamera Zentrums.

Die Kalibrierung nutzt, um die geometrischen Eigenschaft der Kamera festzustellen, die wahrgenommenen und die tatsächlichen Eigenschaften der Referenzpunkte. Hierbei ist die Position der tatsächlichen Referenzpunkte im Allgemeinen in einem anderen Koordinatensystem als dem Kamerakoordinatensystem angegeben. Die Transformation zwischen diesen beiden Koordinatensystemen ist durch eine Rotation und Translation definiert und beschreibt die Abbildung der Referenzpunkte in das Kamerakoordinatensystem wie folgt:

$$sx_i = s \begin{pmatrix} \bar{x} \\ \bar{y} \\ 1 \end{pmatrix} = A \cdot E \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = H \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = HX_i \tag{2.16}$$

Die Matrix $A$ entspricht dabei der Kameramatrix und $E = [R|t]$ entspricht den extrinsischen Parametern und ist eine $3 \times 4$ Matrix, wobei $R = [r_1\ r_2\ r_3]$ gilt und $r_1, r_2, r_3$ jeweils Spaltenvektoren von $R$ sind. Hierbei ist zu beachten, dass die o.g. extrinsischen Parameter zwar mit der extrinsischen Kalibrierung im Zusammenhang stehen, hier aber erst einmal für die intrinsische Kalibrierung genutzt werden. Bei der intrinsischen Kalibrierung können die berechneten extrinsischen Parameter als Abstand und Orientierung des Referenzkoordinatensystems angesehen werden.

Beide Matrizen können in einer $3 \times 4$ Matrix $H$ zusammengefasst werden. O.b.d.A. kann Gleichung 2.16 durch die Annahme vereinfacht werden, dass die Modell Ebene in $Z = 0$ liegt. Hierdurch entspricht der dritte Spaltenvektor der Matrix $R$ dem Nullvektor. Damit lassen sich $X_i$ und $R$ darstellen als

$$X_i = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix},\ R = [r_1\ r_2]. \tag{2.17}$$

Dadurch handelt es sich bei $E$, $A$ und $H$ jeweils um $3 \times 3$ Matrizen. Der folgende Abschnitt beschreibt einen Algorithmus zur Bestimmung der o.g. Parameter.

### Direct Linear Transform Algorithmus

Aus den Gleichungen 2.16, 2.17 und der Tatsache das $r_1$ und $r_2$ orthogonal zueinander sind, ergeben sich die folgenden beiden Bedingungen:

$$\begin{aligned} h_1^T (A^{-1})^T A^{-1} h_2 &= 0 \\ h_1^T (A^{-1})^T A^{-1} h_1 &= h_2^T (A^{-1})^T A^{-1} h_2. \end{aligned} \tag{2.18}$$

Diese beiden Bedingungen lassen sich durch das Zusammenfassen von $(A^{-1})^T A^{-1}$ zur Matrix $B$ vereinfachen, die wie folgt definiert ist:

$$B = (A^{-1})^T A^{-1} = \begin{pmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{pmatrix} \tag{2.19}$$

wobei $B$ symmetrisch ist und sich hierdurch als 6D-Vektor $b = [B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33}]$ beschreiben lässt. Dadurch gilt

$$h_i^T B h_j = v_{ij}^T b = \begin{pmatrix} h_{i1}h_{j1} \\ h_{i1}h_{j2} + h_{i2}h_{j1} \\ h_{i2}h_{j2} \\ h_{i3}h_{j1} + h_{i1}h_{j3} \\ h_{i3}h_{j2} + h_{i2}h_{j3} \\ h_{i3}h_{j3} \end{pmatrix}^T b \ . \tag{2.20}$$

Mit den Bedingungen aus Gleichung 2.18 ergibt sich

$$\begin{bmatrix} v_{12}^T \\ (v_{11} - v_{22})^T \end{bmatrix} b = 0 \ . \tag{2.21}$$

Bei der Verwendung von $n$ Bildern des Modells ergeben sich $n$ solcher Gleichungen wie in Gleichung 2.21. Diese bilden die $2n \times 6$-Matrix $V$ für die demnach $Vb = 0$ gilt. Gilt $n > 3$, so gibt es eine eindeutige, beliebig skalierte Lösung, nämlich den Eigenvektor von $V^T V$ mit dem kleinsten Eigenwert.

Die intrinsischen und extrinsischen Parameter ergeben sich dann aus

$$\begin{gathered} c_y = (B_{12}B_{13} - B_{11}B_{23})/(B_{11}B_{22} - B_{12}^2) \\ \lambda = B_{33} - [B_{13}^2 + c_y(B_{12}B_{13} - B_{11}B_{23})]/B_{11} \\ f_x = \sqrt{\lambda / B_{11}} \\ f_y = \sqrt{\lambda B_{11}/(B_{11}B_{22} - B_{12}^2)} \\ \gamma = -B_{12} f_x^2 f_y / \lambda \\ c_x = \gamma c_y / f_x - B_{13} f_x^2 / \lambda \\ r_1 = \lambda A^{-1} h_1 \\ r_2 = \lambda A^{-1} h_2 \\ r_3 = r_1 \times r_2 \\ t = \lambda A^{-1} h_3 \end{gathered} \tag{2.22}$$

mit $\lambda = 1/||A^{-1}h_1|| = 1/||A^{-1}h_2||$.

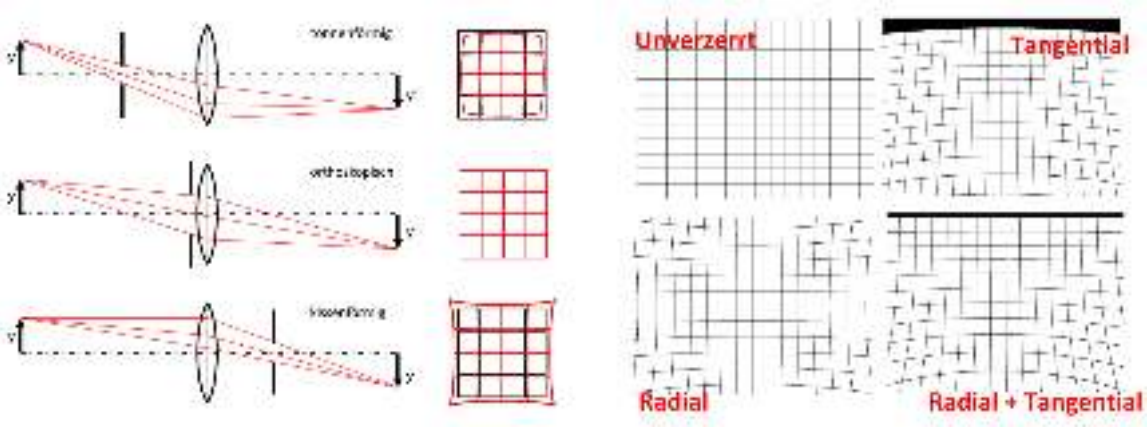



**Abbildung 2.6:** Verzeichnungen [14]

### Fehlerminimierung

Die obige Lösung ist nur eine Schätzung der tatsächlichen Parameter, welche durch das Minimieren der folgenden Fehlerfunktion verbessert wird:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} ||m_{ij} - \bar{m}(A, R_i, t_i, M_j)||^2 \tag{2.23}$$

wobei $\bar{m}(A, R_i, t_i, M_j)$ die Projektion des Punktes $M_j$ in Bild $i$ ist, $n$ entspricht der Anzahl an Bildern und $m$ der Anzahl an Referenzpunkten im Bild. Hierbei handelt es sich um ein nicht-lineares Minimierungsproblem, welches der Levenberg-Marquardt Algorithmus löst. Der Algorithmus benötigt eine initiale Schätzung von $A$, $R_i$ und $t_i$, welche eben die obige Lösung des Gleichungssystems ist.

### Verzeichnungen

Bilder von Kameras weisen in der Praxis Verzeichnungen auf, welche die bisherigen Gleichungen noch nicht berücksichtigen. Hierbei gibt es mehrere Einflussfaktoren, die die Verzeichnung beeinflussen. Wie in Abbildung 2.6 zu sehen ist, beeinflusst die Position der Blende das Bild so, dass die Verzeichnung tonnen- oder kissenförmig ist. Durch eine Verschiebung der Linse kommt es wiederum zu tangentialer Verzeichnungen. Beide Effekte können gleichzeitig vorkommen. Das Aufnehmen der Verzeichnungen ins Modell reduziert den Einfluss von diesen auf das Abbildungsergebnis.

Nach [4, 14, 20] stellt Gleichung 2.24 die mathematische Modellierung der Verzerrung dar:

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = (1 + d_1 r^2 + d_2 r^4 + d_3 r^6) \begin{pmatrix} x_n \\ y_n \end{pmatrix} + \begin{pmatrix} p_1(2x_n y_n) + p_2(r^2 + 2x_n^2) \\ p_1(r^2 + 2y_n^2) + p_2(2x_n y_n) \end{pmatrix} \tag{2.24}$$

wobei $d_1$ - $d_3$ die radialen, $p_1$ und $p_2$ die tangentialen Verzerrungsparameter, $(x_n, y_n)$ die idealen, verzerrungsfreien und $(x_d, y_d)$ die tatsächlichen Bildpunkte sind. Die Verzerrungsparameter

fasst der Vektor $dist = [d_1, d_2, p_1, p_2, d_3]$ zusammen. Der Bildradius $r$ ist wie folgt definiert: $r = \sqrt{x_n^2 + y_n^2}$.

Seien $(u, v)$ die idealen Bildpunkte ohne Verzerrung und $(\bar{u}, \bar{v})$ die tatsächlich, wahrgenommenen Pixel-Bildpunkte. Das Zentrum der radialen Verzerrung liegt im optischen Zentrum der Linse. Hinsichtlich der radialen Verzerrung folgt aus $\bar{u} = u_0 + f_x x_d + \gamma y_d$ und $\bar{v} = v_0 + f_y y_d$:

$$\bar{u} = u + (u - u_0)[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2] \tag{2.25}$$

$$\bar{v} = v + (v - v_0)[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2] \tag{2.26}$$

Aus den Gleichungen 2.25 und 2.26 ergeben sich die folgenden zwei Gleichungen:

$$\begin{bmatrix} (u - u_0)(x^2 + y^2) & (u - u_0)(x^2 + y^2)^2 \\ (v - v_0)(x^2 + y^2) & (v - v_0)(x^2 + y^2)^2 \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = \begin{bmatrix} \bar{u} - u \\ \bar{v} - v \end{bmatrix}. \tag{2.27}$$

Mit $m$ Punkten in $n$ Bildern ergeben sich $2mn$ Gleichungen und das Gleichungssystem $Dk = d$, wobei $k = [k_1, k_2]^T$ gilt. Die Lösung von $k$ ist gegeben durch

$$k = (D^T D)^{-1} D^T d. \tag{2.28}$$

Durch das Integrieren der Verzeichnungen in Gleichung 2.23 kann der Levenberg-Marquardt Algorithmus ebenfalls die radialen und tangential Parameter bestimmen. Die modifizierte Gleichung ist dann

$$\sum_{i=1}^{n} \sum_{j=1}^{m} ||m_{ij} - \bar{m}(A, dist, R_i, t_i, M_j)||^2 \tag{2.29}$$

wobei $\bar{m}(A, dist, R_i, t_i, M_j)$ die Abbildung des Punktes $M_j$ des Bildes $i$ unter Einbezug der Verzeichnung $dist$ nach Gleichung 2.24 ist.

Mit den bisher beschriebenen Methoden lassen sich also die intrinsischen und extrinsischen Parameter berechnen. Diese Arbeit verwendet dabei die OpenCV [21] Implementierung der Kamerakalibrierung, welche auf [4] basiert. Die theoretische Grundlage von [4] basiert auf [10, 30].

Hierbei unterscheidet sich der Algorithmus von [4] zu [30] in drei Punkten: Erstens verwendet die Implementierung das intrinsische Modell von [10], welches zusätzlich zu den radialen auch die tangentialen Verzeichnungsparameter verwendet, welche im Abschnitt 2.2.3 Verzeichnungen dargelegt sind. Des Weiteren nutzt [4] ausdrücklich die Orthogonalität der Fluchtpunkte zur Schätzung der initialen Parameter, wobei die Verzerrungsparameter dabei nicht initial geschätzt werden.

Wie bereits erwähnt sind die extrinsischen Parameter gleichbedeutend mit dem Abstand und der Orientierung des Referenzkoordinatensystems. Die intrinsische Kalibrierung setzt voraus, dass die Referenzpunkte in der XY-Ebene liegen. Hierdurch entsprechen die extrinsischen Parameter dem Abstand und dem Orientierungunterschied in jedem Bild. Die extrinsische Kalibrierung verwendet stattdessen die Koordinaten der Referenzpunkte aus dem Koordinatensystem des

anderen Sensors. Des Weiteren setzt die extrinsische Kalibrierung voraus, dass die Verzerrung der Bilder korrigiert ist. Folglich ist die intrinsische Kalibrierung Voraussetzung für die extrinsische Kalibrierung.

### 2.2.4 Kalibrierung Laserprojektor

Der Algorithmus nutzt das mitgelieferte SDK von Z-Laser, um den Projektor zu kalibrieren. Hierzu nutzt das SDK mehrere Reflektoren im Arbeitsbereich, deren 3D-Position bekannt ist. Der Projektor beleuchtet anschließend einen zuvor festgelegten Bereich, in welchem sich die Reflektoren befinden. Die Kalibrierung erfolgt in dieser Thesis semi-automatisch. Der Projektor projiziert ein Rechteck, welches den Reflektor einschließen soll und den zu scannenden Bereich festlegt. Der Anwender bewegt das Rechteck über eine grafischen Benutzeroberfläche (engl. Graphical User Interface) (GUI) zum Reflektor. Um die Transformation von der Werkskalibrierebene zur neuen Kalibrierebene zu berechnen, benötigt die Software die 3D-Punkte der Reflektoren. Da die Position der Reflektoren auf dem Kalibrierobjekt fest ist, leiten sich die Koordinaten aus der Pose des Kalibrierobjekts ab.

Der Projektor scannt, nach dem Festlegen der Daten, den festgelegten Bereich mehrfach, mindestens einmal entlang der x-Achse und y-Achse. Für das Scannen entlang der x-Achse projiziert der Projektor eine Linie, deren Anfang- und Endpunkte den gleichen x-Wert im Projektor-Koordinatensystem besitzen, und eine Länge hat, die der Höhe des Rechtecks entspricht. Analog dazu geschieht dies auch für die y-Achse. Hier entspricht die Länge der Linie der Breite des Rechtecks und deren Anfang- und Endpunkte besitzen den gleichen y-Wert im Projektor-Koordinatensystem. Der Projektor besitzt mehrere Sensoren, die das reflektierte Licht des Reflektors wahrnehmen. Diese befinden sich rundum das Spiegelsystem (vgl. Abbildung 2.7). Mit der Berechnung der Transformation ist die Kalibrierung damit abgeschlossen. Soll der Projektor Figuren im neuen Koordinatensystem projizieren, so werden diese in das Koordinatensystem der Werkskalibrierebene transformiert und anschließend projiziert. Die Werkskalibrierebene befindet sich drei Meter von dem Laserprojektor entfernt, sodass der Laserstrahl zum Punkt $P(0, 0)$ orthogonal zur Kalibrierebene ist.

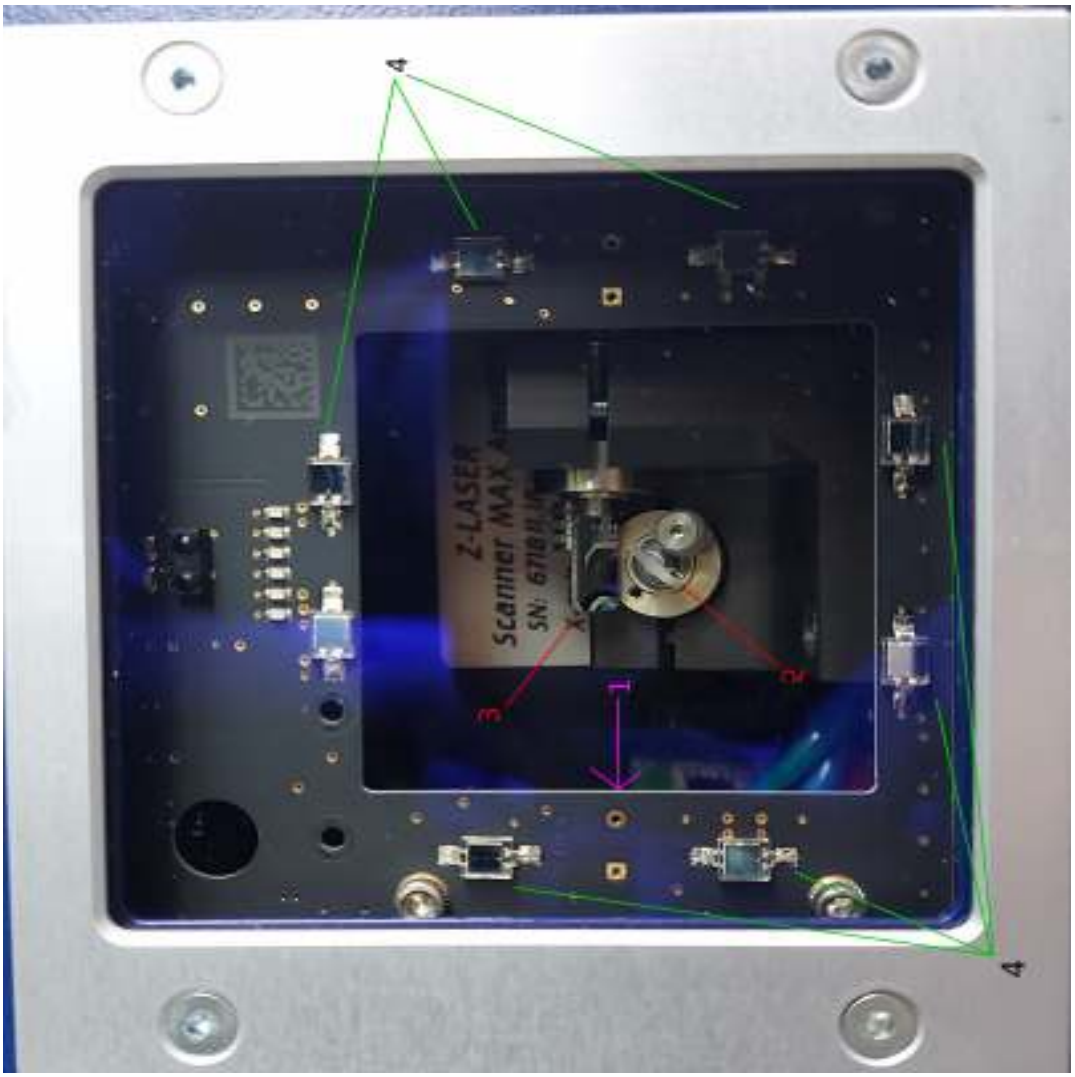



**Abbildung 2.7:** (1) Laserquelle (2) Spiegel x (3) Spiegel y (4) Lichtsensoren

# Kapitel 3

# Software

Dieses Kapitel beschreibt die verwendete Software, die die Kalibrierung der verschiedenen Sensoren durchführt. Abschnitt 3.1 beschreibt dabei die Grundlage des gesamten Systems, das sogenannte Robotic Operating System (ROS). Anschließend beschreiben die darauffolgenden Abschnitte 3.2 - 3.5 die verschiedenen Pakete, die jeweils eine Aufgabe in der Gesamtkalibrierung übernehmen.

## 3.1 Robotic Operating System

Die gesamte Kalibrierung ist in ROS implementiert. Nach [24] ist ROS eine Open Source Sammlung von Software-Bibliotheken und Werkzeugen, die bei der Entwicklung von Software für Roboter hilft. Darin sind z.B. Treiber als auch Algorithmen enthalten, die auf dem neuesten Stand der Technik sind. Jedes Stück Software ist als ein Paket verpackt, welches eine Menge von Abhängigkeiten angibt. Die dort enthaltenen Programme werden als sogenannte Nodes gestartet und kommunizieren über den sogenannten “ROS Master” miteinander. Dieser managt die Kommunikation, indem jeder Node seine Daten als sogenannte “topic” registriert. Andere Nodes können anschließend diese “topic” abonnieren und erhalten jede dort veröffentlichte Nachricht. Wie bei jedem Producer-Consumer System kann ein Producer schneller Daten zur Verfügung stellen als der Consumer bearbeiten kann. ROS verwendet hier ein Warteschlangensystem, welches managt wie viele veröffentlichte Daten maximal auf die Bearbeitung warten dürfen. Sollte die Warteschlange gefüllt sein, so verwirft ROS jede weitere erhaltene Nachricht bis die Warteschlange wieder einen Platz frei hat. In ROS heißen Producer auch Publisher und die Consumer heißen Subscriber. Sowohl Producer als auch Consumer verwenden Warteschlangen, die die Nachrichten speichern. Jeder dieser Nodes repräsentiert ein Prozess, der Daten erzeugt und veröffentlicht oder von anderen Nodes die Daten weiterverarbeitet.

Diese Thesis implementiert die folgenden ROS Pakete:

- point_cloud_aggregation
- detect_calibration_pattern

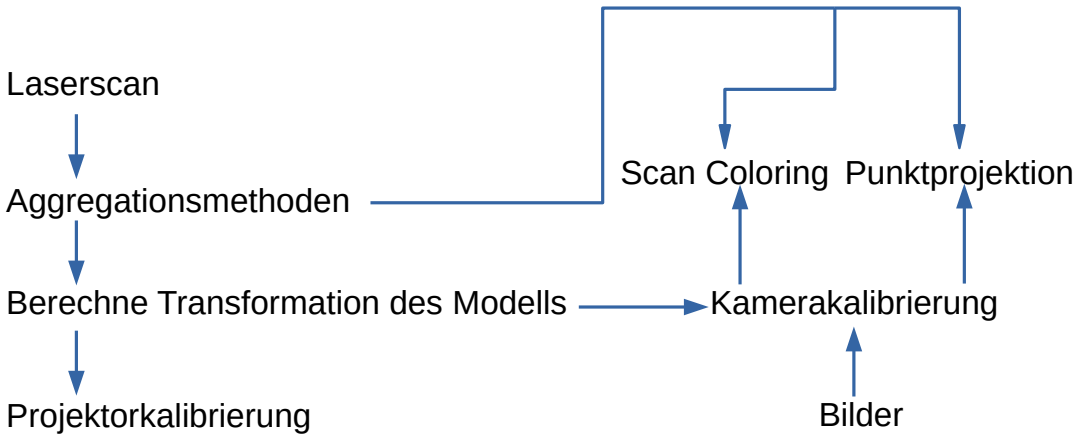



**Abbildung 3.1:** Zusammenfassung des ROS Dataflows während der Kalibrierung.

- jmu_camera_calibration
- zlp-projector-calibration

Außerdem verwendet die Implementierung weitere Packages z.B. zum Verwenden der Kamera, die diese Arbeit nicht näher beschreibt, aber hier genannt sind [5, 13, 19, 27]. Die folgenden Abschnitte beschreiben den Inhalt der Pakete und die Funktionsweise der verschiedenen Nodes. Abbildung 3.1 zeigt den Datenfluss zwischen den verschiedenen Nodes. Der Laserscanner veröffentlicht Punktwolken, welche die Aggregationsmethoden aggregieren und das Ergebnis wiederum veröffentlichen. Daraus wird anschließend die Transformation des Modells in den Scan berechnet, welche anschließend von der Projektor- und Kamerakalibrierung genutzt wird. Die Kamerakalibrierung erhält als Sensordaten Bilder von der Kamera und berechnet aus diesen die intrinsischen und extrinsischen Parameter, wobei die Transformation des Modells nur für die extrinsischen Parameter nötig ist. Die extrinsischen und intrinsischen Parameter können anschließend genutzt werden, um die Projektion von 3D-Koordinaten in das Kamerakoordinatensystem zu projizieren und diese im Bild zu markieren. Des Weiteren kann der Laserscan auch mit den Informationen aus den Bildern eingefärbt werden, wodurch ersichtlich ist, welche 3D-Punkte zu welchem Pixel im Bild gehören.

## 3.2 point_cloud_aggregation

Das Paket “point_cloud_aggregation” stellt mehrere Nodes zur Verfügung, die das Rauschen vom Ouster reduzieren sollen. Diese Nodes abonnieren die Topic des Ousters und wenden anschließend verschiedene Methoden hierfür an, bevor sie die überarbeitete Punktwolke auf einer weiteren Topic veröffentlichen. Der Node “point_cloud_averager” sammelt mehrere Punktwolken und bildet für jeden Punkt den Durchschnitt über diese Punktwolken. Da der Mittelwert stärker auf Ausreißer und Messfehler reagiert, verwendet der Node “pcl_median” den Median anstelle des

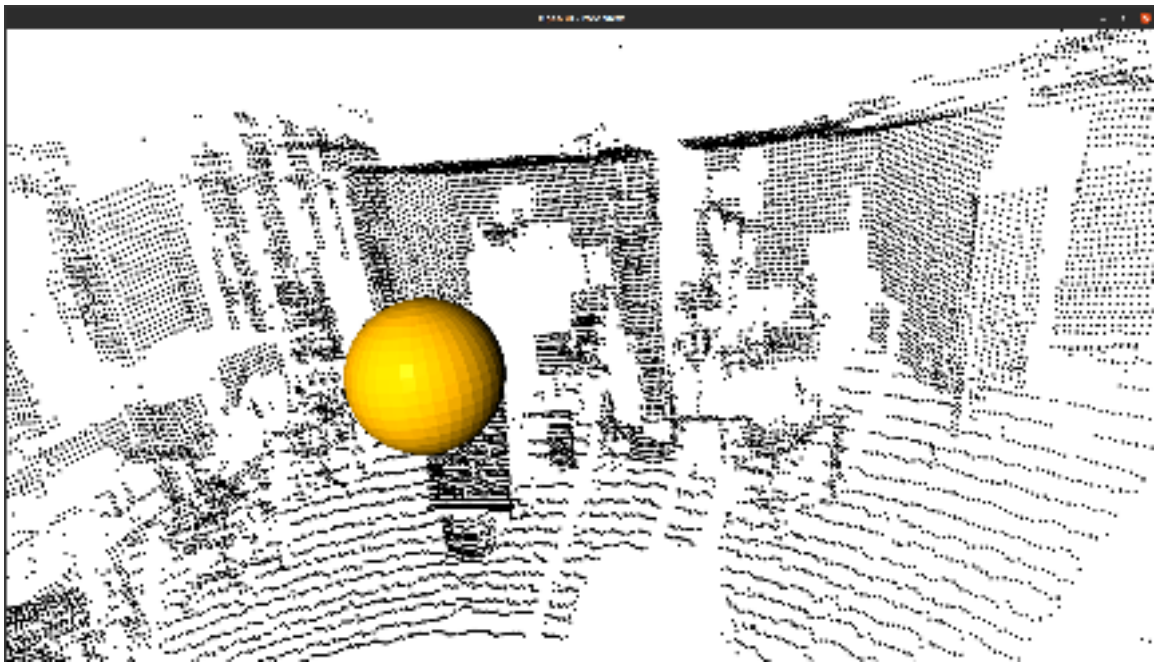


**Abbildung 3.2:** GUI zum Festlegen des Bereichs, in welchem sich das Kalibrierpattern befindet. Mit shift linksklick kann der Nutzer einzelne Punkte der angezeigten 3D-Punktwolke selektieren und mit shift rechtsklick wieder deselektieren. Hiermit legt der Nutzer einen Quader fest, der das Kalibrierobjekt umfassen soll. Die Implementierung verwendet für diese Interaktion und der weiteren Verarbeitung Open3d [31].

Mittelwerts. Als Alternative steht noch der Node “point_cloud_union” zur Verfügung, welcher mehrere Punktwolken zu einer Punktwolke zusammenfügt. Ein Parameter steuert die Anzahl der gesammelten und aggregierten Punktwolken.

## 3.3 detect_calibration_pattern

Wie bereits in Kapitel 2 erwähnt, benötigt die Kamera- und Projektorkalibrierung die 3D-Koordinaten der Referenzpunkte des Schachbretts sowie der Reflektoren. Der Node “RosDetect-CalibrationObject.py” bestimmt die Transformation des verwendeten Modells zu der aufgezeichneten Punktwolke. Hierzu muss der Nutzer zunächst den Bereich festlegen, in welchem sich das Kalibrierobjekt befindet. Dies geschieht über die GUI in Abbildung 3.2. Open3D [31] übernimmt die Anzeige der GUI als auch die weitere Verarbeitung der Punktwolke. Der Nutzer wählt mehrere Punkte über die GUI aus, die den Bereich in der 3D-Punktwolke festlegen, in dem sich das Kalibrierobjekt befindet. Anschließend berechnet der Algorithmus [26] die Transformation, die das verwendete Modell an die Punktwolke anpasst. Der Algorithmus betrachtet dabei nur den ausgewählten Teil der Punktwolke. Abschließend veröffentlicht der Node die Transformation, sowie die transformierten Positionen der Reflektoren und Schachbrettecken. Das hier verwendete Modell hat [17] mit [15] erstellt.

## 3.4 jmu_camera_calibration

Das Paket “jmu_camera_calibration” ist für die Kamera Kalibrierung verantwortlich und verwendet [21], um die Kamera zu kalibrieren. In diesem ist ein Node enthalten, welcher eine Bild-Topic abonniert und die erhaltenen Bilder nach Schachbrettmustern absucht. Findet dieser ein Schachbrett, so speichert der Node dieses ab. Anschließend berechnet das Programm “IntrinsicCameraCalibration” die intrinsischen Parameter der Kamera und speichert diese auf dem Dateisystem ab. Auch hier steuern Parameter die verwendete Topic als auch den Zielordner für die Bilder und den Speicherort der intrinsischen Parameter. Des Weiteren abonniert der Node “ExtrinsicCameraCalibration” die veröffentlichte Topic des “detect_calibration_pattern” Nodes als auch eine Bild Topic. Anhand von der Transformation berechnet der Node die Koordinaten der Schachbrettecken. Anschließend bestimmt er die extrinsischen Parameter mithilfe dieser Koordinaten und der erhaltenen Bilder. Da jedes Bild ein Ergebnis für die extrinsischen Parameter liefert, nutzt der Node verschiedene Methoden, um eine gute Lösung für alle Bilder zu finden. Die ursprüngliche Berechnung der guten Lösung basiert Code von [26]. Die o.g. Methoden sind in Abschnitt 4.2.1 genauer beschrieben. Zusätzlich bestimmt der Node noch den Reprojektionsfehler. Beides speichert der Node im Dateisystem ab.

Des Weiteren enthält das Package den Node “MarkPointsInImage.py”, um 3D-Punkte in der Punktwolke in das Kamerakoordinatensystem zu projizieren und diese in Bildern von der Kamera einzuzeichnen. Hierzu abonniert der Node die Topic für die extrinsischen Parameter, die der Node “ExtrinsicCameraCalibration” berechnet, die Punktwolken Topic und die Bilder Topic. Der Node fordert den Nutzer auf Punkte in der erhaltenen Punktwolke auszuwählen, welche der Node anschließend ins Kamerakoordinatensystem projiziert.

Der Node “ColorPointCloud.py” abonniert wie der vorherige Node die gleichen Topics. Statt aber 3D-Punkte in die erhaltenen Bilder einzuzeichnen, färbt der Node die Punktwolke mit den erhaltenen Bildern ein. Hierzu projiziert der Node die 3D-Punkte der Punktwolke in das Kamerakoordinatensystem und überprüft, ob die erhaltenen Punktkoordinaten innerhalb des Bildes liegen. Falls es zutrifft, erhält der 3D-Punkt die Farbe des entsprechenden Pixels, andernfalls färbt der Node die Punkte Pink ein. Das Ergebnis einer solche Einfärbung ist in Abbildung 3.3 zu sehen. Der Node berücksichtigt aktuell noch nicht, die Ausrichtung der Kamera für die Einfärbung. Jedoch zeichnet er das Kamerakoordinatensystem in die Abbildung mit ein. Abbildung 3.3 ist im Nachhinein mit CloudCompare [6] entstanden, weshalb das Kamerakoordinatensystem in dieser Abbildung nicht zu sehen ist.

Das Skript “CalculateResiduen.py” berechnet mit Hilfe von [9, 21] die Abweichungen der projizierten 3D-Punkte im Kamerakoordinatensystem von den detektierten Positionen der Schachbrettecken. Aus den resultierenden Residuen erstellt das Skript mit [11] mehrere Graphen.

## 3.5 zlp-projector-calibration

Das Paket “zlp-projector-calibration” ermöglicht die Kalibrierung des ZLP Projektors. In diesem ist der Node “InteractiveZLPCalibration” enthalten, welche den ZLP Projektor kalibriert. Dieser abonniert nach dem Start eine Punktwolken Topic, deren Nachrichten das Kalibrierobjekt

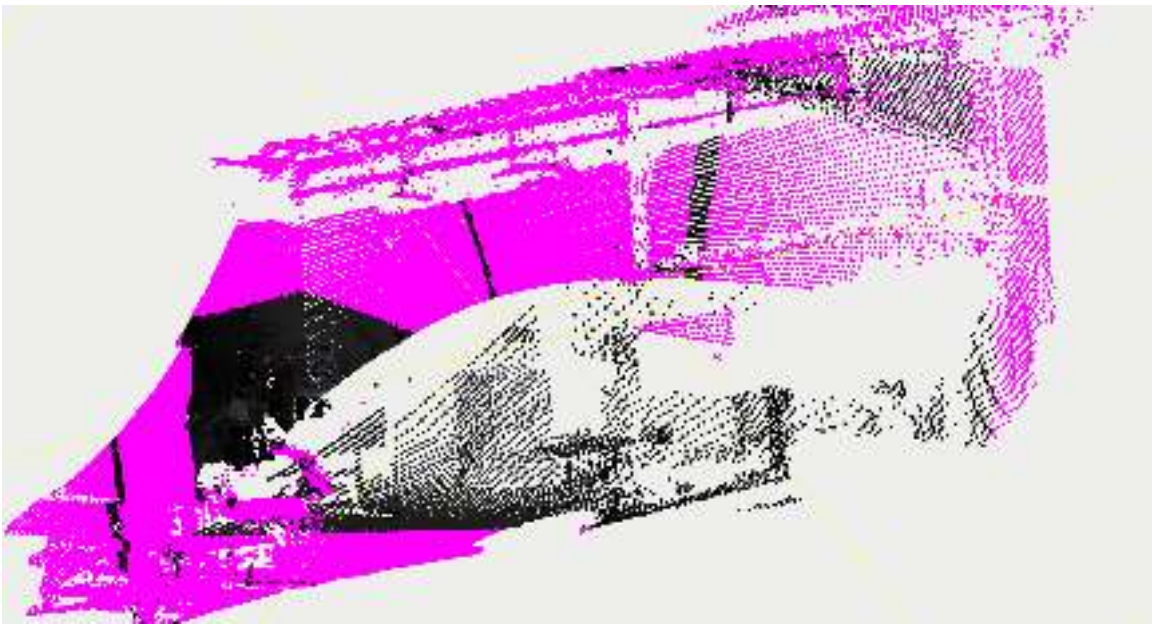


**Abbildung 3.3:** Mit Kamerabild eingefärbter Scan

beinhalten. Außerdem fordert der Node den Nutzer auf, die Reflektoren des Kalibrierobjekts mit dem Projektor zu markieren. Dies geschieht über die GUI aus Abbildung 3.4. Die GUI zeigt ein Fadenkreuz an, welches der Nutzer über Drag & Drop steuern kann. Parallel dazu projiziert der Projektor dieses Fadenkreuz in die Gegend. Der Projektor aktualisiert die Projektion, während der Nutzerinteraktion. Über den Button "Set Trace Point" speichert der Nutzer einen Trace Point in der Liste rechts. Diese nutzt der Node anschließend für die Kalibrierung wie in Kapitel 2 beschrieben. Da die Position der Reflektoren auf dem Kalibrierobjekt fix sind, ergibt sich deren 3D-Position aus der Transformation des Modells in die Punktwolke. Zum Schluss führt der Node die Registrierung des neuen Koordinatensystems aus, indem der Node über das ZLP Sdk [29] eine Punktsuche startet.

Der Node "ProjectOuterChessboard" ist für die Projektion zweier Linien zuständig, die Abschnitt 4.2.2 für die Auswertung der Projektorkalibrierung verwendet. Dieser Node abonniert die Topic für die Transformation des Modells in die Punktwolke und berechnet die Position der äußeren Schachbrettecken. Damit die Linien auf dem Kalibrierobjekt besser zu erkennen sind, verschiebt der Node die angepeilten Punkte nochmal, um ein halbes Schachbrettfeld nach oben und nach innen, wodurch sich der Anfang und das Ende der projizierten Linie in der Mitte eines weißen Kästchens befinden sollte.

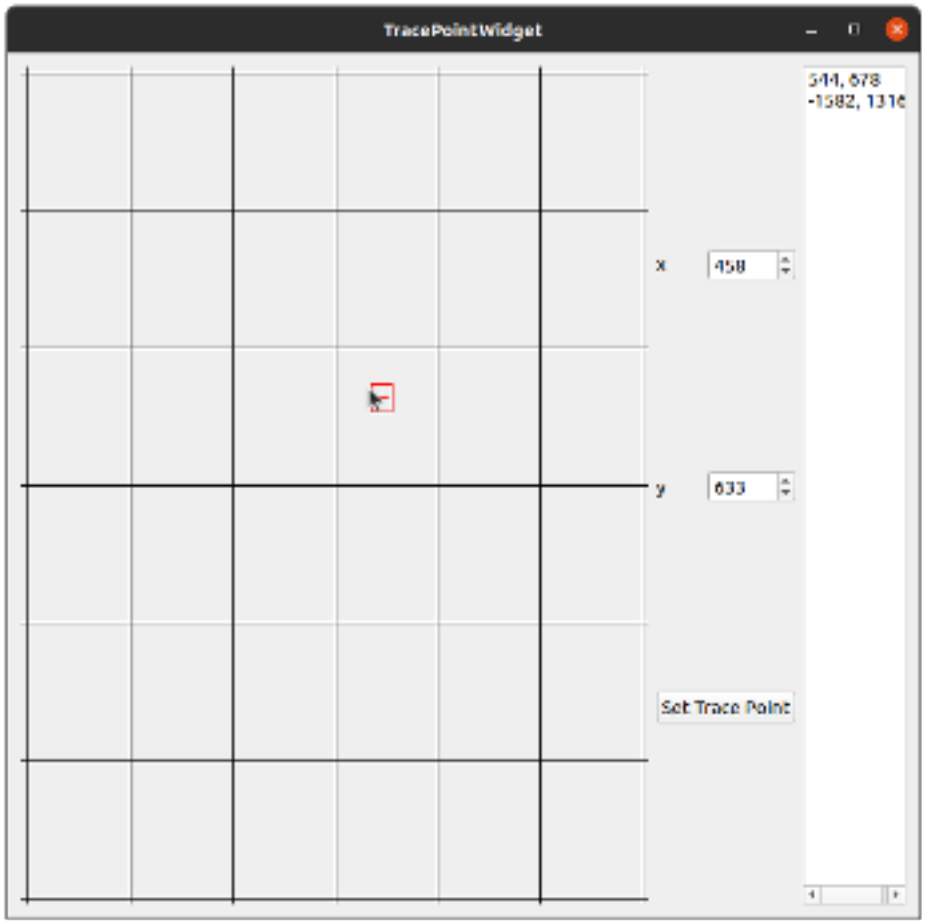



**Abbildung 3.4:** GUI für die Projektor Kalibrierung: Links ist ein Koordinatensystem zu sehen, welches ein rotes Fadenkreuz beinhaltet. Mit der Maus kann man über Drag & Drop das Fadenkreuz an eine beliebige Stelle ziehen. Fürs Finetuning kann über die Spinboxen, die x und y Koordinaten ebenfalls angepasst werden. Der Zahlenbereich des Koordinatensystems ist der Folgende : (-3000, 3000). Der "Set Trace Point" Button speichert den aktuellen Trace Point in der Liste rechts. Die "Entf" Taste entfernt das ausgewählte Element wieder aus der Liste. Falls nötig können in der Liste, die Punkte noch nachträglich editiert werden. Die Kalibrierung benötigt mindestens 3 Punkte. Der (x) Button oben rechts beendet die Auswahl der Punkte [25].

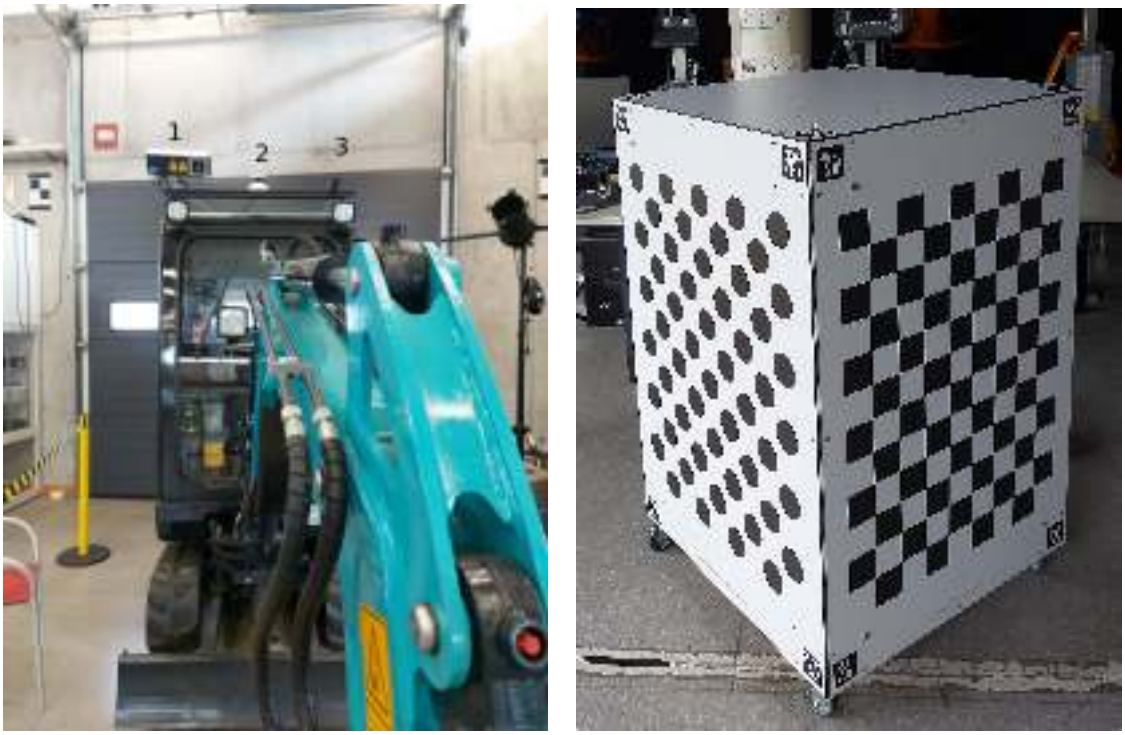
# Kapitel 4

# Experimente und Auswertung

Das Ziel dieser Arbeit ist es, die Qualität der Kalibrierung der verschiedenen Sensoren zu evaluieren. Dieses Kapitel beschreibt den Versuchsaufbau für mehrere Experimente, um die Kalibrierung qualitativ auszuwerten. Hierzu untersucht diese Arbeit die Qualität der Kamera- und Projektorkalibrierung und den Einfluss der Punktwolken- Stabilisierungsmethode auf die Kalibrierung. Da die Stabilisierungsmethode die Kalibrierungsergebnisse direkt beeinflusst, lässt sich ein Zusammenhang zwischen der Qualität der Kalibrierung und der Stabilisierungsmethode feststellen. Da es bereits viele Arbeiten wie z.B.[14] gibt, die sich mit dem Thema intrinsische Kamerakalibrierung befassen und die intrinsische Kalibrierung den Ouster Laserscanner nicht verwendet, führt diese Arbeit keine weiteren Untersuchungen zu diesem Thema durch. Wie bereits in Kapitel 2.2.3 beschrieben ist die intrinsische Kalibrierung Voraussetzung für die extrinsische, weshalb sie auch Teil der Ergebnisse ist.

## 4.1 Versuchsaufbau

Die Kalibrierung der Sensoren muss für das Projekt DigSmart auch dann funktionieren, wenn die Sensoren auf dem Bagger befestigt sind, weshalb der experimentelle Teil dieser Arbeit dies untersucht. In Abbildung 4.1(a) ist der verwendete Bagger mit den montierten Sensoren abgebildet. Hierbei ist der Laserprojektor links, der Laserscanner in der Mitte und die Kamera rechts angebracht. Wie bereits in Abschnitt 2.1.2 erwähnt, besitzt der Laserscanner einen Scanbereich von $33,2°$. Damit der Scanner auch in dieser hohen Lage den Bodenbereich unterhalb von 4 m scannen kann, ist dieser in einem 45° Winkel angebracht. Dies schränkt aber auch den Bereich ein, in welchem sich der Kalibrierkörper befinden darf, damit dieser noch für die extrinsische Kalibrierung wahrnehmbar ist. Da der Abstand zwischen Kalibrierkörper und Sensoren Einfluss auf die Kalibrierung hat, untersuchen die Experimente drei Abstände: Die Position "vorne" hat einen Abstand von $3 - 3,6$ m zum Laserscanner, "mittig" hat einen Abstand von $4,4 - 5$ m und "hinten" hat einen Abstand von $6,5 - 8$ m. Je weiter weg sich das Kalibrierobjekt befindet, desto weniger Punkte hat es in der 3D-Punktwolke. Dies hat zur Folge, dass die Anzahl der Punkte für die obere Ebene bei einer horizontalen Auflösung von $hres = 512$ zwischen 5 und 15 Punkten liegt. RANSAC ist hier für die Ebenenerkennung so eingestellt, dass dieser die gefundene




(a) Bagger mit Projektor (1), Laserscanner (2) und Kamera (3)

(b) Kalibrierkörer

**Abbildung 4.1:** Versuchsaufbau

Ebene erst ab einer Größe von 10 Punkten akzeptiert. Falls die obere Ebene zu wenige Punkte beinhaltet, findet RANSAC diese Ebene nicht. Daher verwenden die Experimente in diesem Fall eine höhere horizontale Auflösung von $hres = 1024$ für den Ouster Laserscanner, um die kleinere Anzahl an Punkten auszugleichen. Die Experimente aggregieren jeweils 10 Punktwolken, bevor die Kalibrierung diese verwendet.

### 4.1.1 Kalibrierobjekt

Als Kalibrierobjekt verwendet diese Arbeit einen Quader mit den Maßen $0,8\,\text{m} \times 0,8\,\text{m} \times 1,2\,\text{m}$ (Breite × Tiefe × Höhe). Der Quader besitzt vier Rollen, die jeweils $0,1\,\text{m}$ hoch sind und die Konstruktion beweglich machen. Wie in Abbildung 4.1(b) zu sehen ist, sind auf drei Seiten Platten angebracht, wobei die seitlichen Platten verschiedene Kalibriermuster besitzen. Die Experimente verwenden davon nur das Schachbrettmuster. Etwas oberhalb und unterhalb des Schachbrettmusters sind jeweils zwei Metallreflektoren angebracht, die die Projektorkalibrierung benötigt. Die Position der Metallreflektoren ist fix und durch den Aufdruck des Kalibriermusters festgelegt. Im Laserscan ist das Kalibrierobjekt so positioniert, dass alle drei Seiten erkennbar sind.

## 4.2 Metrik

Die folgenden Abschnitte stellen kurz vor, wie diese Arbeit die Qualität der Kamera- und Projektorkalibrierung bestimmt. Diese Arbeit verwendet bei sowohl der Kamera- als auch der Projektorkalibrierung den sogenannte "Mittleren Reprojektionfehler" (RMSE), welcher für Kameras laut [3, 14] wie folgt definiert ist:

$$RMSE = \sqrt{\frac{\sum_{i=0}^{n} \sum_{j=0}^{m} ||x_{ij} - \prod_P(X_{ij})||^2}{nm}} \tag{4.1}$$

wobei $x_{ij}$ die $j$-ten 2D-Koordinaten und $X_{ij}$ die $j$-ten 3D-Koordinaten des Modells in Bild $i$ sind und $n$ der Anzahl an Bildern und $m$ der Anzahl an Modellpunkten im Bild entspricht. Die Abbildungsfunktion $\prod_P$ bildet den Punkt $X_i$ in das Kamerakoordinatensystem ab und verwendet hierfür die intrinsischen und extrinsischen Parameter.

Analog ist der RMSE für die Projektorkalibrierung definiert, da der Projektor auf eine Fläche etwas projizieren kann. Hierzu lässt sich ein lokales 2D-Koordinatensystem auf der Projektionsfläche definieren, mit dem sich die Abweichungen von den angepeilten Punkten berechnen lässt.

Die Qualität der intrinsischen Kamerakalibrierung lässt sich ebenfalls durch den RMSE beschreiben. Ebenfalls betrachtet die Auswertung die Verteilung, Länge und Ausrichtung von sogenannten Residuen, welche die Abweichung der projizierten Punkte zu den erwarteten Messpunkten beschreiben. Nach [14, S. 164] und [18] sollten die Residuen möglichst klein sein und keine systematische Ausrichtung aufweisen. Andernfalls könne dies ein Anzeichen für systematische Fehler sein.

Um die Ergebnisse der extrinsische Kamerakalibrierung zu überprüfen, lassen sich 3D-Punkte wie bei der intrinsischen Kamerakalibrierung ins Kamerakoordinatensystem abbilden. Beispielsweise sollten die projizierten Punkte der Schachbrettecken, die gleichen Koordinaten wie die Schachbrettecken im Bild haben. Die Auswertung betrachtet wendet dieses Prinzip auch auf weitere markante Stellen in der Szene an, damit die Abbildung nicht nur für die Schachbrettecken gilt. Die Abweichung sollte hier möglichst gering sein, sonst können Fehler in der Kalibrierung vorhanden sein.

### 4.2.1 Extrinsische Kamerakalibrierung

Diese Thesis bestimmt für jedes Bild die extrinsischen Parameter der Kamera. Hierdurch ergibt sich eine Menge an extrinsischen Parametern, die eine mögliche Lösung für die extrinsische Kalibrierung sind. Die Autorin von [3, S. 178] schlägt mehrere Methoden vor, um die besten Parameter zu bestimmen. Diese Arbeit verwendet die folgenden Methoden:

**mean** – der Mittelwert über die gefundenen extrinsischen Parameter

**emed** – der Median über die Komponenten der extrinsischen Parameter

**min** – die Parameter, die insgesamt den kleinsten RMSE haben.

Zu beachten ist, dass nach [3, S. 178] mehrere von den oben genannten Methoden ausprobiert werden müssen, da die Ergebnisse situationsabhängig sind.

### 4.2.2 Laserprojektor

Um die Kalibrierung des Laserprojektors auszuwerten, projiziert dieser zwei Linien auf das Schachbrettmuster, wobei nur die Start- und Endpunkte der Linie für die Auswertung relevant sind. Diese vier Punkte markieren die Mitte der äußeren Ecken des Schachbrettmusters. Wie die Projektion nach der Kalibrierung am Kalibrierort aussehen kann, ist in Abbildung 4.2 zu sehen. Die Abweichung des projizierten Punktes von der tatsächlichen Mitte wird mithilfe eines Geodreiecks abgemessen. Da diese je nach Abstand zum Projektor variieren kann, werden mehrere Messungen an drei Stellen durchgeführt: im Bereich "vorne", "mittig" und "hinten". In jedem Experimenten wird der Projektor auch in diesen Bereichen mit jeder Stabilisierungmethode kalibriert. An diesen drei Stellen wird für jede Stabilisierungsmethode jeweils das o.g. Muster projiziert und die Abstände gemessen. Anhand der eigentlichen Projektionspunkten lässt sich die Genauigkeit des Projektors beurteilen.

## 4.3 Auswertung

### 4.3.1 Kamerakalibrierung

#### Intrinsische Kalibrierung

Die Ergebnisse, die der Algorithmus anhand von 497 Bildern für die intrinsische Kamerakalibrierung berechnet hat, sind in Tabelle 4.1 zusammengefasst. Zusätzlich zu den berechneten Werten zeigt Tabelle 4.1 auch die Standardabweichungen dieser Parameter. "Diese können als erster Anhaltspunkt genutzt werden, ob die Berechnung der Parameter signifikant ist: laut Luhmann sollten hierfür mindestens eine Größenordnung kleiner als der Parameter selbst [18], und generell möglichst klein sein" [14, S.154].

Wie in der o.g. Tabelle zu sehen ist, gilt dies für alle Parameter der Kameramatrix. Für die Verzerrungsparameter gilt dies zum größten Teil auch, jedoch hat die Standardabweichung des Parameters $d_1$ die gleiche Größenordnung wie er selbst. Dieser Parameter ist betragsweise bereits kleiner als ein tausendstel Pixel, weshalb der Einfluss dieses Parameters auf das Endergebnis gering ist und deshalb die mangelnde Signifikanz der Berechnung von diesem Parameter vernachlässigbar ist. Der auf diesen Bildern beruhende $RMSE$ liegt bei $0,180437$ Pixel.

Des Weiteren liegen die berechneten Parameter der Kameramatrix auch ungefähr bei den Werten des Herstellers.

Abbildung 4.3 zeigt die Residuen der intrinsischen Kamerakalibrierung. Dabei ist schon auf Abbildung 4.3(a) zu sehen, dass die Ausrichtung der Residuen nicht systematisch ist. Abbildung 4.3(b) und 4.3(c) zeigen zwei Ausschnitte von dichteren Stellen in Abbildung 4.3(a) und wie zu sehen ist, sind auch an diesen Stellen die Residuen unsystematisch orientiert.

Hinsichtlich der Länge der Residuen, so ist in Abbildung 4.4 die Verteilung der Länge der Residuen abgebildet. Wie zu erkennen ist, haben mehr als 90 % der Residuen eine Länge kleiner als $0,25$ Pixel. Von diesen sind ca. 42 % kleiner als $0,1$ Pixel. Ca. $4,5$ % der Residuen haben eine Länge größer als $0,25$ Pixel und kleiner als $0,5$ Pixel. Der kleinste Anteil der Residuen liegt über einer Länge von $0,5$ Pixel.

**Abbildung 4.2:** Projektor projiziert Linie auf verschobene Schachecken

Da der Großteil der Residuen eine Länge kleiner als $0,5$ Pixel hat, die Residuen unsystematisch orientiert sind, der $RMSE$ auch kleiner eins ist, die Berechnung der intrinsischen Parameter signifikant ist und die aufgenommenen Schachpunktecken gut verteilt sind, sprechen diese Beobachtungen für eine erfolgreiche intrinsische Kalibrierung. Damit ist die Voraussetzung für die extrinsische Kalibrierung erfüllt, welche der folgende Abschnitt untersucht.

### Extrinsische Kamerakalibrierung

Wie bereits in den Kapitel 2 erwähnt, benötigt die extrinsische Kalibrierung zwischen Laserscanner und Kamera die 3D-Koordinaten der Referenzpunkte im Laserscannerkoordinatensystem. Daher verwendet diese Arbeit zwei Experimente, in denen das Kalibrierobjekt sich in den Bereichen "mittig" und "hinten" befindet. Durch die Ausrichtung der Kamera ist das Kalibrierpattern im

**Tabelle 4.1:** endgültig verwendete intrinsische Kameraparameter

| Parameter | Wert | Standardabweichung |
|---|---|---|
| $f_x$ | $1220,754071$ | $\pm 3,555470$ |
| $f_y$ | $1220,986427$ | $\pm 3,533398$ |
| $c_x$ | $938,342769$ | $\pm 3,365458$ |
| $c_y$ | $579,819761$ | $\pm 3,300025$ |
| $p_1$ | $-0,055190$ | $\pm 0,003051$ |
| $p_2$ | $0,060820$ | $\pm 0,006485$ |
| $p_3$ | $-0,017299$ | $\pm 0,005088$ |
| $d_1$ | $-0,000232$ | $\pm 0,000587$ |
| $d_2$ | $-0,002655$ | $\pm 0,000639$ |

**Tabelle 4.2:** intrinsische Parameter nach Hersteller

| $f_x$ | $f_y$ | $c_x$ | $c_y$ |
|---|---|---|---|
| $1159,42$ | $1159,42$ | $960,0$ | $600,0$ |

Bereich “vorne” nicht vollständig erkennbar, weshalb dieser Bereich für die Kamerakalibrierung und dem dazugehörigen Experiment wegfällt. Beide Experimente verwenden jeweils die drei Aggregationsmethoden für die Punktwolken, sowie die drei Aggregationsmethoden für die extrinsischen Parameter. Der Algorithmus bestimmt anhand von 10 Bildern die extrinsischen Parameter, die die o.g. Methoden aggregieren. Die Ergebnissen in Tabelle 4.3 zeigen, dass diese Anzahl an Bildern eine gute Schätzung für die Parameter liefert, da auch hier der RMSE kleiner als ein Pixel ist. Ebenfalls lässt sich erkennen, dass der RMSE bei fast allen Methoden im Bereich “hinten” am besten ist. Hier fällt nur die Kombination der Aggregationsmethoden “emed” und “median” aus dem Muster und ist in dem Fall mit $0,521352$ Pixel schlechter im Bereich “hinten” als mit $0,328582$ Pixel im Bereich “mittig”. Insgesamt besitzt “emed” die schlechtesten Ergebnisse. An sich haben die Methoden “mean” und “min” in allen Experimenten die besten Ergebnisse und weichen von einander um weniger als $0,01$ Pixel ab. Das beste Ergebnis hat die Kombination “Average”, “hinten” und “mean” mit $0,245598$ Pixel gefolgt von der Kombination “Average”, “hinten” und “min” $0,24588$ Pixel.

Da nur die besten extrinsischen Parameter benötigt werden, berechnet die Software automatisch für die 10 Bilder den RMSE für die Aggregation der extrinsischen Parameter. Hierdurch kann sie die besten Parameter mit dem kleinsten RMSE Wert auswählen und für die Nutzung speichern. Wie die Ergebnisse in Tabelle 4.3 zeigen, hat “mean” in allen Versuchen den besten RMSE, weshalb nur für diesen die Projektion von weiteren Punkten in der Szene im Folgenden betrachtet wird.

Die Ergebnisse der Projektion sind in Abbildung 4.6 zu sehen. Abbildung 4.5 zeigt die in der Punktwolke markierten Punkte, die hierfür verwendet werden. Teil dieser Punkte sind die Ecken

**Tabelle 4.3:** RMSE der extrinsischen Kamerakalibrierung

| Punktwolken Aggregation | Position | Aggregation extrinsischer Parameter | RMSE |
|---|---|---|---|
| Median | hinten | mean | 0,277782 |
| | | emed | 0,521352 |
| | | min | 0,278309 |
| Median | mitte | mean | 0,323399 |
| | | emed | 0,328582 |
| | | min | 0,324273 |
| Average | hinten | mean | 0,245598 |
| | | emed | 0,302985 |
| | | min | 0,24588 |
| Average | mitte | mean | 0,320563 |
| | | emed | 0,441939 |
| | | min | 0,322238 |
| Union | hinten | mean | **0,21298** |
| | | emed | 0,304882 |
| | | min | **0,213783** |
| Union | mitte | mean | 0,317921 |
| | | emed | 0,36323 |
| | | min | 0,318229 |

des Kalibrierkörpers sowie die Ecken eines Tisches, als auch zwei Punkte von den Stellwänden und ein Punkt von der Stuhllehne. Wie Abbildung 4.6 zeigt, sind die abgebildeten Punkte der Tischecken und der Stellwände ungefähr an den Stellen an denen sie in der Punktwolke auch ausgewählt sind (Abweichung $\leq 100$ Pixel vom ausgewählten Objekt/Punkt). Jeder der projizierten Punkte befindet sich an dem Objekt, welches in der Punktwolke markiert ist. Mit dieser Abweichung wird die hier gezeigte extrinsische Kalibrierung als Erfolg gewertet. In den Abbildungen 4.6(a), 4.6(b) und 4.6(c) sind die Markierungen der unteren Ecken des Kalibrierkörpers oberhalb der eigentlichen Ecken zu finden. Das kann durch eine ungenaue Auswahl der Ecken in der Punktwolke auftreten. Aus dem gleichen Grund ist die Markierung der hintersten oberen Ecke des Kalibrierobjekts in den Abbildungen 4.6(d), 4.6(e) und 4.6(f) verschoben. Die Abbildungen sind auch im Anhang A in groß zu finden.

### 4.3.2 Projektorkalibrierung

Wie auch bei der extrinsischen Kamerakalibrierung betrachten die Experimente mit dem Laserprojektor mehrer Position des Kalibrierobjektes. In Experiment 1 befindet sich das Kalibrierobjekt im Bereich “mittig”, während es sich im Experiment 2 im Bereich “hinten” und im Experiment 3 “vorne” befindet. Die Kalibrierung des Laserprojektors ist in Experiment “2” fehlgeschlagen, da

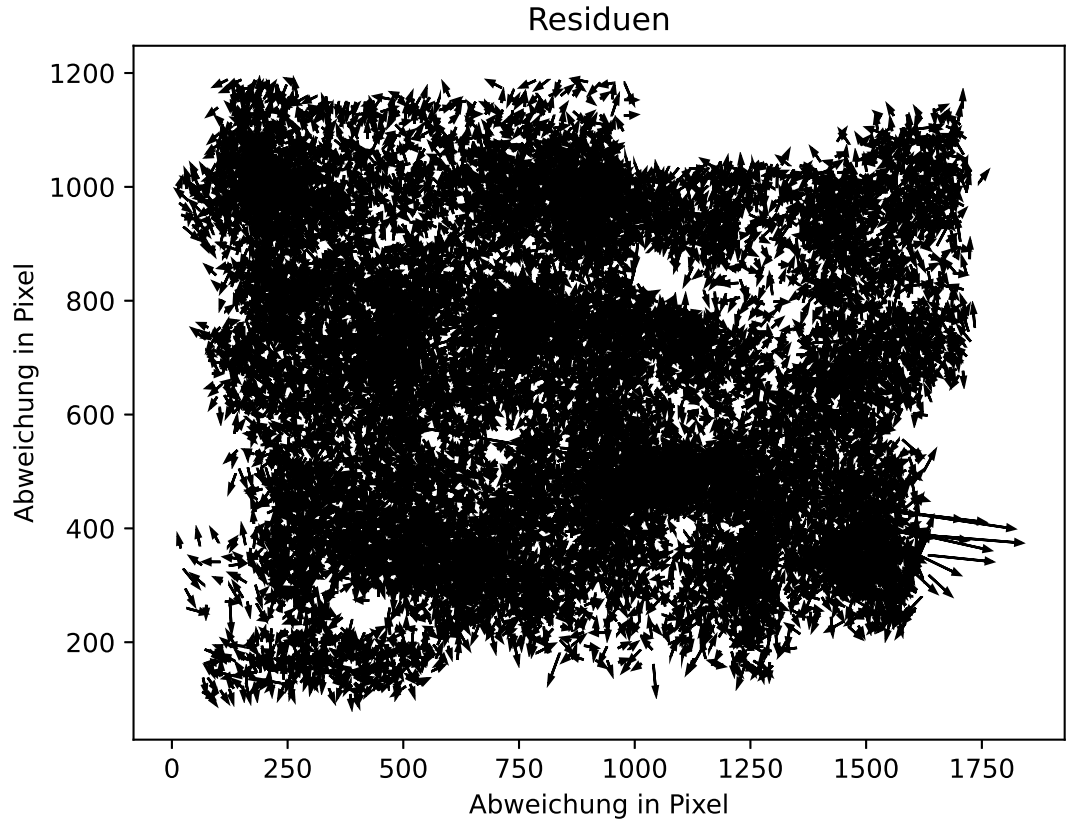
**Tabelle 4.4:** Experiment 1 - RMSE in *mm* - Kalibrierort Mitte

| Aggregation | RMSE (vorne) | RMSE (mitte) | RMSE (hinten) | RMSE (total) |
|---|---|---|---|---|
| Average | 38, 24 | 6, 18 | 30, 83 | 28, 58 |
| Median | 51, 20 | 4, 85 | 40, 86 | 37, 92 |
| Union | 43, 51 | 2, 41 | 40, 28 | 34, 26 |

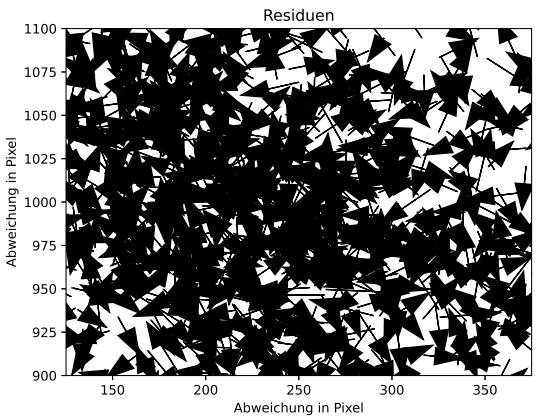
**Tabelle 4.5:** Experiment 3 - RMSE in *mm* - Kalibrierort Vorne

| Aggregation | RMSE (vorne) | RMSE (mitte) | RMSE (hinten) | RMSE (total) |
|---|---|---|---|---|
| Average | 6, 03 | 23, 16 | 45, 70 | 29, 78 |
| Median | 8, 99 | 20, 73 | 40, 70 | 26, 88 |
| Union | 1, 60 | 16, 61 | 28, 15 | 18, 89 |

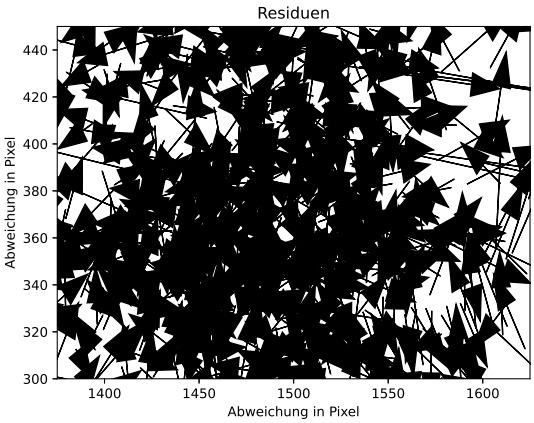
nie alle Reflektoren detektiert werden konnten. Dies lässt sich mit der Entfernung des Bereichs "hinten" zum Laserprojektor begründen, da sich das Kalibrierobjekt um mehr als 2 m außerhalb des Arbeitsbereichs des Projektor befindet. Tabelle 4.4 und 4.5 enthalten die Ergebnisse der Tests. Wie darin zu sehen ist, ist die Abweichung der Projektion am Kalibrierort am geringsten. Der RMSE liegt dort unter 9 mm. Die Aggregationsmethode "Union" weist hierbei die besten Ergebnisse auf, wenn sich der Kalibrierkörper bei der Kalibrierung im Bereich "vorne" befindet. Hier hat "Union" sowohl in den einzelnen Tests als auch insgesamt die besten Ergebnisse. Abgesehen von dem Bereich "vorne" hat die Methode "Median" die zweitbesten Ergebnisse. Falls sich der Kalibrierort aber im Bereich "mittig" befindet, so weist "Union" mit 2, 41 mm nur am Kalibrierort die besten Ergebnisse auf. In diesem Fall ist die Methode "average" insgesamt am besten.




(a) alle Residuen




(b) Ausschnitt 1




(c) Ausschnitt 2

**Abbildung 4.3:** Residuen mit skalierter Länge um den Faktor 100

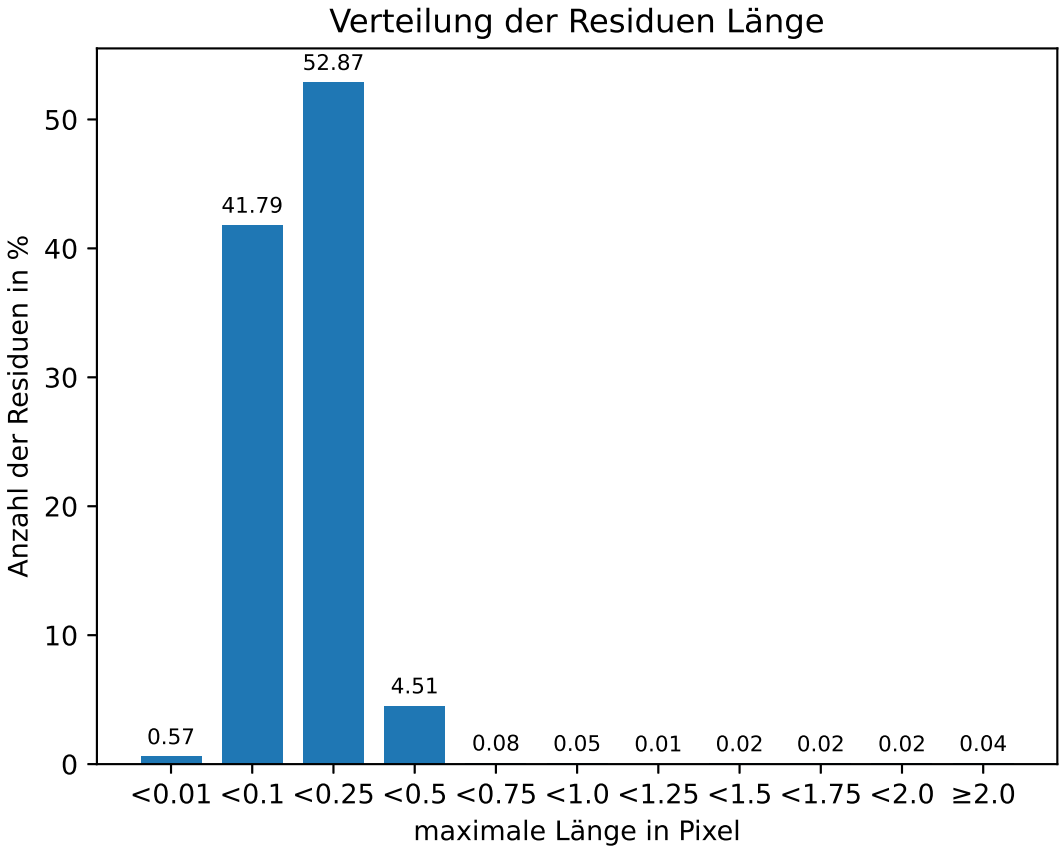
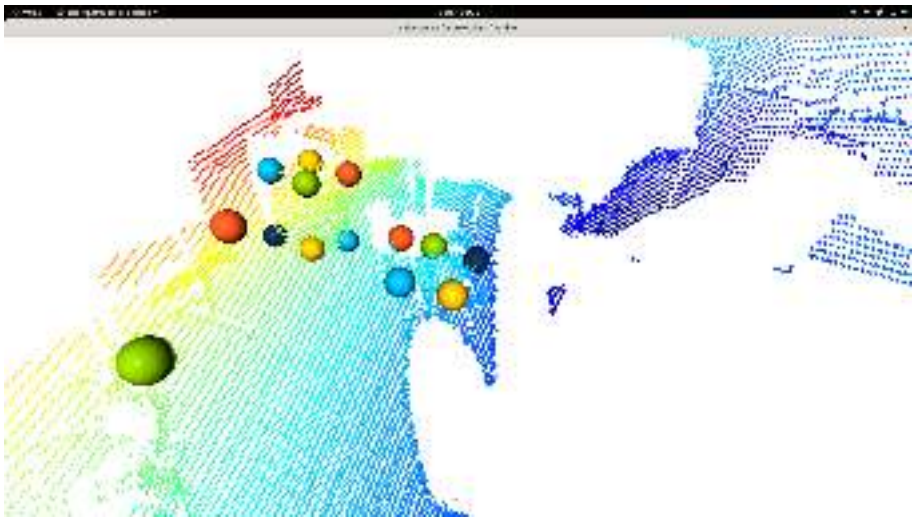
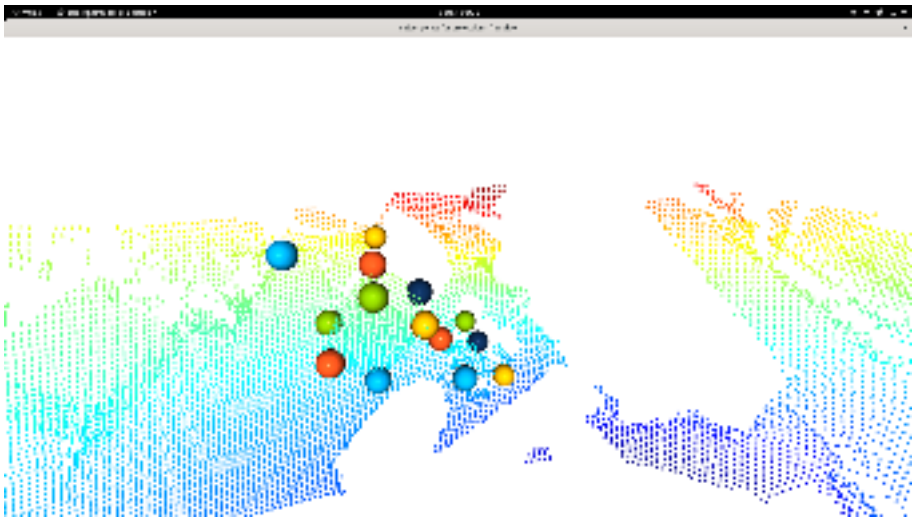



**Abbildung 4.4:** Verteilung der Residuen Länge



(a) Markierungen Position hinten



(b) Markierungen Position mittig

**Abbildung 4.5:** Markierte Punkte für die Projektion vom Laserscan Koordinatensystem ins Kamerakoordinatensystem

(a) Position “hinten”, Punktwolken Aggregation “average”

(b) Position “hinten”, Punktwolken Aggregation “median”

(c) Position “hinten”, Punktwolken Aggregation “union”

(d) Position “mittig”, Punktwolken Aggregation “average”

(e) Position “mittig”, Punktwolken Aggregation “median”

(f) Position mittig, Punktwolken Aggregation “union”

**Abbildung 4.6:** Projektion weiterer Punkte in das Kamerakoordinatensystemder anhand der besten extrinsischen Parametern (mean) je Aggregationsmethode der Punktwolken

# Kapitel 5

# Fazit

## 5.1 Zusammenfassung

Wie bereits in der Einleitung beschrieben ist es das Ziel dieser Thesis, die Qualität der Kalibrierung zu untersuchen. Die Ergebnisse der Experimente aus Kapitel 2.2.3 zeigen, dass die in Kapitel 2 vorgestellten Methoden die Kamera intrinsisch und extrinsisch kalibrieren können. Die Residuen der intrinsischen Kalibrierung sind weder systematische orientiert, noch hat der Großteil der Residuen eine Länge größer als 0.5 Pixel. Auch ist der RMSE kleiner als ein Pixel. Diese Beobachtungen spricht wie bereits erläutert für eine erfolgreiche intrinsische Kalibrierung der Kamera. Ebenfalls legen die Experimente dar, dass auch die extrinsische Kamerakalibrierung erfolgreich ist. Die Kalibrierung im Bereich "hinten" liefert für alle Aggregationsmethoden die besten Ergebnisse. Der RMSE ist auch für die extrinsische Kalibrierung kleiner als ein Pixel, was auch für eine erfolgreiche Kalibrierung spricht. Die Projektion von mehreren ausgewählten Punkten vom Laserscan- ins Kamerakoordinatensystem bestätigt, dass die Punkte richtig abgebildet werden. Hinsichtlich der Kalibrierung des Laserprojektors, so deuten die Ergebnisse aus 4.2.2 an, dass die Kalibrierung nur gute Ergebnisse für den Kalibrierort liefert und nicht für andere Bereiche. Dies ist für das Projekt DigSmart problematisch, da dies den Bereich sehr einschränkt, in welchem der Projektor projizieren kann. Eine Projektion auf ein Objekt außerhalb des Kalibrierbereichs weist zum Teil Abweichungen auf, die größer als 10 cm sind. Das Projekt DigSmart verlangt für Projektionen aber eine Abweichung von maximal 10 cm. Vermutlich liegt das daran, dass der Kalibrierwürfel im Vergleich zum Field of View des Projektors relativ klein ist. Außerdem arbeitet der Projektor außerhalb seines empfohlenen Arbeitsbereichs von 3 m, was sich ebenfalls negativ auf die Projektion auswirkt.

## 5.2 Ausblick

Mit der hier vorgestellten Kalibrierungsmethode, konnten sowohl Kamera als auch Projektor erfolgreich kalibriert werden. Im Zuge des DigSmart Projektes sollen weitere Kameras für die Lokalisierung auf dem Bagger montiert werden. Hierzu ist es ebenfalls notwendig, diese zu kalibrieren und die extrinsischen Parameter festzustellen. Da der Laserscanner nur einen gewissen

Bereich scannen kann, kann die in dieser Arbeit vorgestellten Methode nicht für alle Kameras genutzt werden. In diesem Fall muss eine Kamera-zu-Kamera-Kalibrierung stattfinden, wie es bei Stereo-Kameras der Fall ist. Des Weiteren soll der Laserprojektor dafür eingesetzt werden, z.B. den Verlauf von Rohren sichtbar zu machen. Damit die Projektion nicht verzehrt ist, müssen zwei Dinge beachtet werden. Zum einen ist die Position der Betrachter wichtig, zum anderen ist die Form der Ebene wichtig auf der die Projektion zu sehen ist. Da die Projektion aus verschiedenen Blickwinkeln anders aussieht, kann hierdurch der Eindruck entstehen, dass z.B. Rohre anders verlaufen als sie eigentlich tun. Daher muss der Projektor die Projektion je nach Perspektive anpassen. Das Gleiche gilt auch für die Form der Projektionsebene, eine projizierte Linie auf einer gewölbten Fläche kann von außen gebogen wirken, wodurch es so aussehen kann, dass z.B. ein Rohr einen Bogen macht. Daher muss die Form der Ebene mit dem Laserscanner extrahiert werden und in der Projektion mit einbezogen werden. Wie diese Arbeit gezeigt hat, hat die Textur der Umgebung aber auch Auswirkungen auf die Abstandsmessung des Laserscanners, weshalb eine flache Ebene trotzdem ein starkes Rauschen im Laserscan des Ousters aufweisen kann, was die Extraktion der Ebenenform erschwert. Aus den Ergebnissen der Experimente lässt sich des Weiteren schließen, dass ein alternativer Kalibrierungsalgorithmus für den Laserprojektor notwendig ist, da die Abweichung außerhalb des Kalibrierortes größer als

# Anhang A

# Bilder

**Abbildung A.1:** Kalibrierkörper

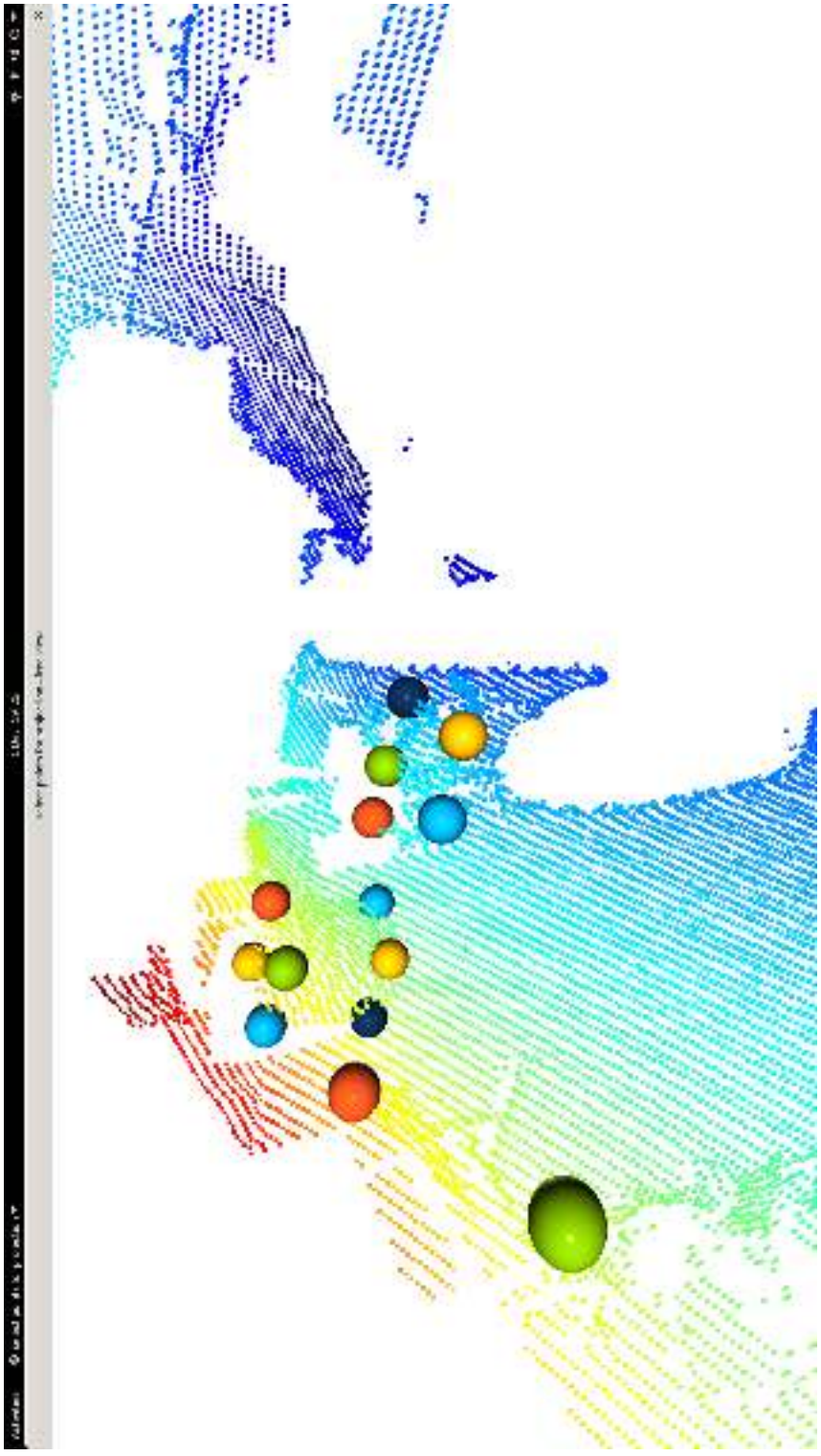


**Abbildung A.2:** Punktmarkierungen im Laserscan für die Projektion ins Kamerakoordinatensystem – Position hinten

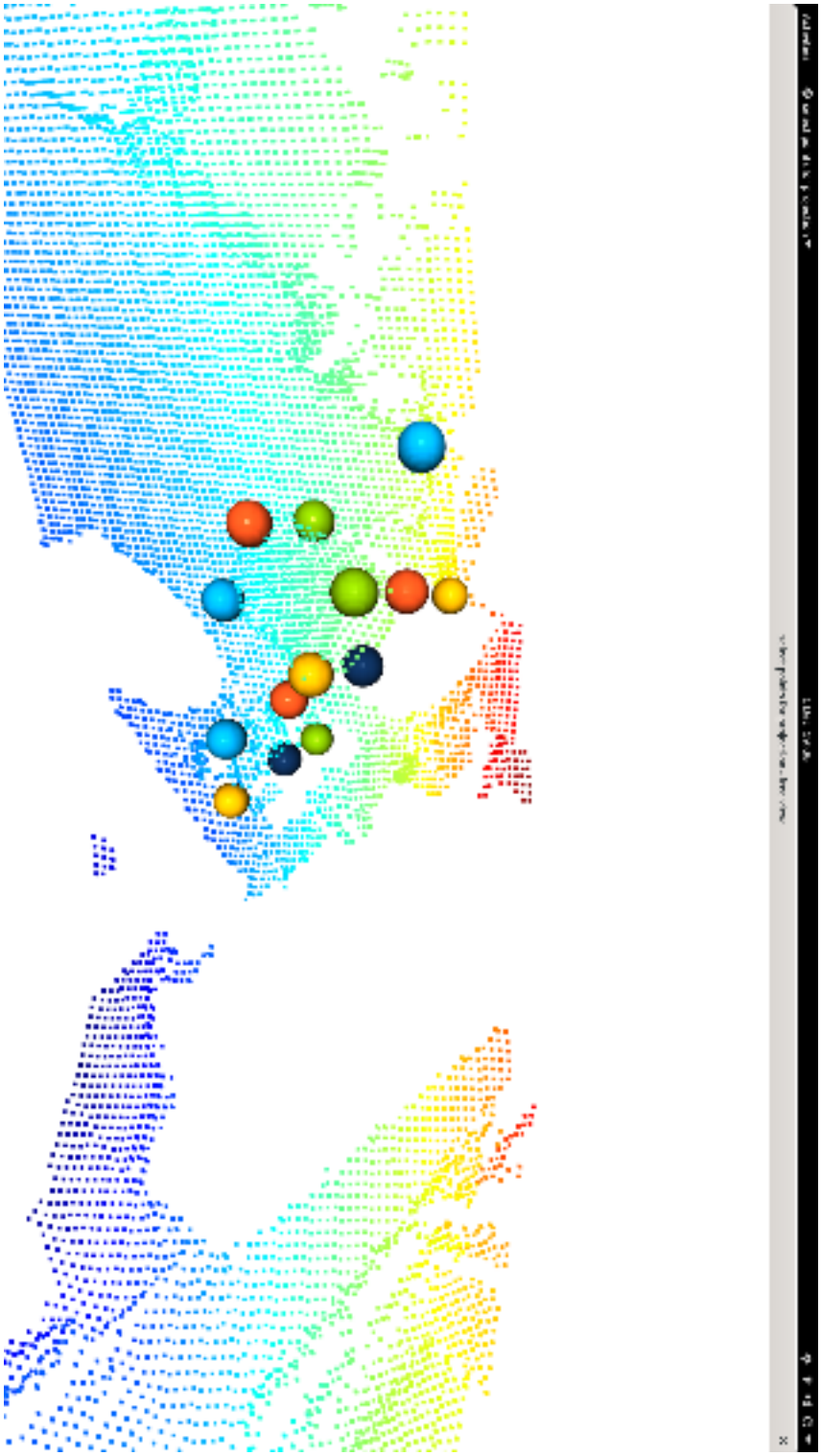


**Abbildung A.3:** Punktmarkierungen im Laserscan für die Projektion ins Kamerakoordinatensystem – Position mittig

**Abbildung A.4:** projizierte 3D-Koordinaten ins Kamerakoordinatensystem – Position “hinten”, Punktwolken Aggregation “average”

**Abbildung A.5:** projizierte 3D-Koordinaten ins Kamerakoordinatensystem – Position "hinten", Punktwolken Aggregation "median"

**Abbildung A.6:** projizierte 3D-Koordinaten ins Kamerakoordinatensystem – Position “hinten”, Punktwolken Aggregation “union”

**Abbildung A.7:** projizierte 3D-Koordinaten ins Kamerakoordinatensystem – Position “mittig”, Punktwolken Aggregation “average”

**Abbildung A.8:** projizierte 3D-Koordinaten ins Kamerakoordinatensystem – Position “mittig”, Punktwolken Aggregation “median”

**Abbildung A.9:** projizierte 3D-Koordinaten ins Kamerakoordinatensystem – Position mittig, Punktwolken Aggregation "union"

# Literaturverzeichnis


[1] Alok Malik, Bradford T.: *Applied unsupervised learning with R.* Birmingham : Packt

[2] Besl, P.J.; McKay, Neil D.: A method for registration of 3-D shapes. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14 (1992), Nr. 2, S. 239–256. http://dx.doi.org/10.1109/34.121791. – DOI 10.1109/34.121791

[3] Borrmann, Dorit: *Multi-modal 3D mapping - Combining 3D point clouds with thermal and color information*, Universität Würzburg, doctoralthesis, 2018. http://dx.doi.org/10.25972/OPUS-15708. – DOI 10.25972/OPUS–15708

[4] Bouguet, Jean-Yves: *Camera Calibration Toolbox for Matlab.* http://robots.stanford.edu/cs223b04/JeanYvesCalib/index.html

[5] Carnegie Mellon University, Inc. Clearpath R. Clearpath Robotics: *flir_camera_driver.* https://github.com/ros-drivers/flir_camera_driver

[6] CloudCompare: *CloudCompare.* https://www.cloudcompare.org/

[7] Freeman, M.H.; Hull, C.C.; Charman, W.N.; Freeman, M.H. (Hrsg.); Hull, C.C. (Hrsg.); Charman, W.N. (Hrsg.): *Optics (Eleventh Edition).* Eleventh Edition. London : Butterworth-Heinemann, 2003. – ISBN 978–0–7506–4248–4

[8] Grey, Tom: *Lidar vs. camera: driving in the rain.* https://ouster.com/blog/lidar-vs-camera-comparison-in-the-rain/

[9] Harris, Charles R.; Millman, K. J.; Walt, Stéfan J.; Gommers, Ralf; Virtanen, Pauli; Cournapeau, David; Wieser, Eric; Taylor, Julian; Berg, Sebastian; Smith, Nathaniel J.; Kern, Robert; Picus, Matti; Hoyer, Stephan; Kerkwijk, Marten H.; Brett, Matthew; Haldane, Allan; Río, Jaime F.; Wiebe, Mark; Peterson, Pearu; Gérard-Marchant, Pierre; Sheppard, Kevin; Reddy, Tyler; Weckesser, Warren; Abbasi, Hameer; Gohlke, Christoph; Oliphant, Travis E.: Array programming with NumPy. In: *Nature* 585 (2020), September, Nr. 7825, 357–362. http://dx.doi.org/10.1038/s41586-020-2649-2. – DOI 10.1038/s41586–020–2649–2

[10] Heikkilä, Janne; Silvén, Olli: A Four-step Camera Calibration Procedure with Implicit Image Correction. In: *CVPR*, IEEE Computer Society, 1997. – ISBN 0–8186–7822–4

[11] Hunter, J. D.: Matplotlib: A 2D graphics environment. In: *Computing in Science & Engineering* 9 (2007), Nr. 3, S. 90–95. http://dx.doi.org/10.1109/MCSE.2007.55. – DOI 10.1109/MCSE.2007.55

[12] Inc., Ouster: *OS1 - High Resolution Imaging LIDAR (Datasheet)*. https://storage.cloud.google.com/data.ouster.io/downloads/datasheets/datasheet-rev06-v2p4-os1.pdf

[13] Inc., Ouster: *https://github.com/ouster-lidar/ouster-ros*. 2022

[14] Leutert, Florian: *Flexible Augmented Reality Systeme für robotergestützte Produktionsumgebungen*, Universität Würzburg, doctoralthesis, 2021. http://dx.doi.org/10.25972/OPUS-24972. – DOI 10.25972/OPUS–24972

[15] LLC, Agisoft: *Agisoft Metashape*. https://www.agisoft.com/

[16] LLC, Teledyne F.: *FLIR BLACKFLY S - P/N BFS-PGE-23S3*. https://flir.app.box.com/s/a92yhdbw5lg3qfiktfjuoiicssd5v4as

[17] Lucht, Joschka van d.: *Modell des Kalibrierobjekts*. 2022

[18] Luhmann, Thomas: *Erweiterte Verfahren zur geometrischen Kamerakalibrierung in der Nahbereichsphotogrammetrie*, Diss., 01 2010

[19] Nikhil Khedekar, Steve M. Pranay Mathur M. Pranay Mathur: *perception_open3d*. https://github.com/ros-perception/perception_open3d

[20] OpenCV: *Camera calibration With OpenCV*. https://docs.opencv.org/4.x/d4/d94/tutorial_camera_calibration.html

[21] OpenCV: *OpenCV*. https://opencv.org/

[22] Schneider, Andreas: *Hessesche Normalform*. https://www.mathebibel.de/hessesche-normalform#beispiel-5

[23] Schubert, Erich; Sander, Jörg; Ester, Martin; Kriegel, Hans P.; Xu, Xiaowei: DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN. In: *ACM Trans. Database Syst.* 42 (2017), jul, Nr. 3. http://dx.doi.org/10.1145/3068335. – DOI 10.1145/3068335. – ISSN 0362–5915

[24] Stanford Artificial Intelligence Laboratory et al.: *Robotic Operating System*. https://www.ros.org

[25] The Qt Company: *Qt*. https://www.qt.io/

[26] Tsoulias, Nikos; Jörissen, Sven; Nüchter, Andreas: An approach for monitoring temperature on fruit surface by means of thermal point cloud. In: *MethodsX* 9 (2022), 101712. http://dx.doi.org/https://doi.org/10.1016/j.mex.2022.101712. – DOI https://doi.org/10.1016/j.mex.2022.101712. – ISSN 2215–0161

[27] University, Carnegie M.: *pointgrey_camera_driver*. https://github.com/ros-drivers/pointgrey_camera_driver

[28] Z-Laser: *Datasheet ZLP1*. https://z-laser.com/wp-content/uploads/_download/Datasheets/English/Z-LASER_Datasheet_ZLP1.pdf

[29] Z-Laser: *ZLP Sdk*. 2022

[30] Zhang, Zhengyou: Flexible Camera Calibration by Viewing a Plane from Unknown Orientations. In: *ICCV*, IEEE Computer Society, 1999. – ISBN 0–7695–0164–8

[31] Zhou, Qian-Yi; Park, Jaesik; Koltun, Vladlen: Open3D: A Modern Library for 3D Data Processing. In: *arXiv:1801.09847* (2018)

# Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Würzburg, Dezember 2022