

Bachelorarbeit

*Entwicklung eines
Lasertriangulationssensor zur
Oberflächen-Rekonstruktion mit ROS2*

zur Erlangung des akademischen Grades

Bachelor of Science (B.Sc.)

vorgelegt dem

Fachbereich Mathematik, Naturwissenschaften und Informatik
der Technischen Hochschule Mittelhessen.

Tristan Elias Wolfram

xx.xx.202x, Gießen

Referentin: Prof. Dr.-Ing. Seyed Eghbal Ghobadi
Korreferent: Moritz Schauer, M.Sc.

Kurzfassung

Titel

Titel auf deutsch

Zusammenfassung

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Abstract

Titel

Title in english

Summary

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Inhaltsverzeichnis

1 Einleitung	1
1.1 Aufgabe / Motivation	1
1.2 Stand der Technik	2
1.2.1 Methodik	2
1.2.2 Geräte	3
1.3 Kriterien	3
1.4 Lösungsansatz	3
2 Hauptteil - Projektbeschreibung und Projektergebnisse	5
2.1 Bibliotheken	5
2.2 Der Grundlegende Aufbau	5
2.3 OpenCv Pinhole Camera Model	6
2.4 Mathematische Grundlage	8
2.4.1 Homogene Koordinaten	8
2.4.2 Transformationen	9
2.5 Bildverarbeitung	11
2.5.1 Das Erkennen der Laserlinie	12
2.5.2 Grauwert-Bild	13
2.5.3 Gauss-Filter	14
2.5.4 Das Erstellen von Subpixeln	15
2.6 Kalibrierung	18
2.6.1 Intrinsische Kalibrierung	18
2.6.2 Extrinsische	21
2.7 Aufbau / Hardware	21
2.8 Aufbau / Software	21
2.8.1 Python (Bibliothek)	21
2.8.2 ROS2	21
2.9 Qualitative Ergebnisse	21
2.10 Evaluation	21
2.10.1 Testen von Genauigkeit	21
2.10.2 Fehler	21
2.10.3 Probleme und Schwierigkeiten	21
3 Fazit	22
4 Ausblick	24

Inhaltsverzeichnis

4.1 Erweiterung	24
4.2 Anpassungen	26
A Anhang	27
A.1 Anhang A	27
A.2 Anhang B	28
A.3 Anhang C	29
Abbildungsverzeichnis	31
Tabellenverzeichnis	32
Quellcodeverzeichnis	33
Abkürzungsverzeichnis	34

Danksagung

An dieser Stelle kann man Danke sagen. :-)

DANKE

1. Einleitung

Die Möglichkeit eine Oberfläche in ihrer Beschaffenheit 3D zu erfassen, birgt viele Anwendungsfälle. Sei es, um ein 3D-Abbild von einem Raum digital zu erhalten oder in der Industrie Bauteile zu vermessen und auf Fehler zu prüfen. Dabei sind viele verschiedene Verfahren über die Zeit entwickelt und optimiert wurden. 3D Kameras oder auch RGB-D Kameras finden heute vielfältig ihren Einsatz. Diese Arbeit beschäftigt sich mit der Dokumentation und Evaluation eines im Rahmen meiner Praktikumsphase an der Technischen Hochschule Mittelhessen entwickelten Lasertriangulationssensor. Ziel dieser Arbeit ist es dabei, die Funktionalität und Architektur dieses Sensors zu erklären. Zusätzlich soll der Sensor evaluiert und auf bestimmte Aspekte mit dem aktuellen Stand der Technik im Bereich der 3D-Kameras bzw. RGB-D Kameras verglichen werden. Sinn und Zweck eines Lasertriangulationssensor ist es eine Oberfläche zu rekonstruieren und dabei nicht nur die Tiefen-Informationen, sondern auch die Farbinformationen gemeinsam aufzunehmen. Als Verfahren wird, wie im Namen genannt, die Lasertriangulation verwendet.

1.1. Aufgabe / Motivation

Die grundsätzliche Aufgabe und Motivation entstand durch ein Projekt in Zusammenarbeit mit der Firma RINNTECH – „Technik zur Prüfung von Bäumen“. Die Firma bezeichnet sich selbst wie folgt: „Als Anwender und Entwickler verfügen wir über jahrelange Erfahrung und zahlreiche Patente auf dem Gebiet der Baum- und Holzanalyse. Für diese Anforderungen können wir Ihnen daher ausgereifte Technik und umfassenden Service anbieten“ [Vgl.: RIN]. Beispielanwendungen, für die Geräte und Software bereitgestellt werden sind zum Beispiel Bäume kontrollieren, Jahrringe analysieren, Holzkonstruktionen kontrollieren und Holzqualität und Zuwachs im Wald kontrollieren [Vgl.: RIN]. Das zugrundeliegende Projekt wurde als Forschungs- und Entwicklungsprojekt zusammen mit dem Institut für Technik und Informatik (ITI) an der THM gestartet. Es trägt den Titel *„Entwicklung einer Messmethodik zur Ermöglichung einer schnellen Bestimmung von Holzart und -herkunft anhand von Jahrring- und Farbanalyse. Entwicklung der Messwerterfassung und -auswertung der neuen Messmethodik“*

und beschäftigt sich konkret mit der Jahrringanalyse. Durch diese soll Holzart und Herkunft bestimmt werden. RINNTECH verkauft das Produkt „LINTAP“ für diesen Zweck [Vgl.: LINTAP]. Das schon existierende Produkt soll in dem Projekt erweitert, optimiert und automatisiert werden. Die Grundidee besteht darin, dass ein Roboter-Arm mit einer hochauflösenden Kamera über das Objekt fährt und entsprechende Bilder aufnimmt, die für die Jahrringanalyse erforderlich sind. Hier wird eine Kamera eingesetzt, die von

dem Roboterarm nah an das Holz herangeführt werden muss. Die Pfade, die der Roboter dementsprechend abfahren muss, müssen also mit hinreichender Genauigkeit errechnet werden. Dazu ist eine digitale Abbildung des Holzes unverzichtbar. Anhand einer rekonstruierten Oberfläche können die entsprechenden Pfade für den Roboter ohne Probleme errechnet werden. Es muss also initial eine Oberflächen-Rekonstruktion des Objektes stattfinden. An diesem Punkt setzt meine Arbeit an. Es wurde eine Software entwickelt, die mithilfe einer Kamera und einen Linienlaser einen 3D-Scan durchführt. Dabei werden die Tiefeninformationen und auch Farbinformationen aufgenommen und verarbeitet. Man erhält eine Punktwolke der gescannten Oberfläche die entsprechend gefärbt ist. Ein typischer Output für eine 3D-Kamera in der Industrie. Die Aufgabenstellung wurde noch etwas konkretisiert. Als Methode soll eine Lasertriangulation verwendet werden, dazu wurde die Kamera und der Linienlaser bereitgestellt. Zusätzlich soll nur Open-Source-Software verwendet werden und die Anwendung soll über ROS2 (Robot Operating System) laufen.

1.2. Stand der Technik

1.2.1. Methodik

Lasertriangulation ist nicht die einzige Methode eine Oberflächen-Rekonstruktion durchzuführen. Die zentralen Technologien zu diesem Zweck sind Triangulation oder Time-of-Flight [Vgl.: SotA]. Bei der Triangulation gibt es zwei Ansätze. Einen direkten über Structured Light. Dabei wird ein bekanntes Muster mit einem Laser oder Ähnlichem auf die Oberfläche projiziert. Eine Kamera nimmt das Muster als Bild auf, welches sich durch die variierende Höhe und Form der Oberfläche verzerrt. Diese Verzerrung wird als Anhaltspunkt verwendet, um die Unterschiedlichkeiten in der Höhe zu ermitteln. Lasertriangulation und der hier verwendete Lösungsansatz gehören zu dieser Möglichkeit. Einen indirekten Ansatz der Triangulation bietet Stereo-Vision. Dabei werden zwei Kameras verwendet, die zwei aufgenommenen Bildern aus unterschiedlichen Positionen liefern. Ebenfalls wird ein fester Punkt benötigt, der auch mit einem Laser oder Ähnlichem im Bild projiziert werden kann. Über die zwei unterschiedlichen Bilder kann die Position ermittelt werden.

Die Time-of-Flight-Technologie benutzt eine andere Lösungsmöglichkeit. Es wird Licht auf einen Punkt auf der Oberfläche gestrahlt. Dort wird es zurück reflektiert und von einem Sensor registriert. Dieser misst die verstrichene Zeit. Durch die bekannte Geschwindigkeit von Licht, kann über die gebrauchte Zeit der zurückgelegte Weg errechnet werden. Dieser entspricht der Höhe.

1.2.2. Geräte

Diese Methoden finden ihre Anwendung auch in der Industrie. Diverse Geräte und Anwendungen zur Oberflächen-Rekonstruktion sind bereits auf dem Markt. Die Rede ist von sogenannten 3D-Kameras bzw. RGB-D Kameras. Dabei steht RGB (Red, Green, Blue) für die Farbinformationen und das D (Depth) für die Tiefeninformation. Angefangen mit der von Microsoft entwickelten „Kinect“ über die „Intel RealSense“ zur „Google Tango“ folgen viele weitere Geräte. Diese sind nicht nur meist für einen geringen Preis verfügbar, sie können auch die entsprechenden Pixelfarben in einer guten Auflösung aufnehmen. Zusätzlich geschieht die Aufnahme in Echtzeit, was bedeutet, dass sich die herausgegebene Punktewolke ändert, sobald sich die aufgenommene Oberfläche ändert bzw. die Kamera bewegt wird.

1.3. Kriterien

Für das Projekt wurde zu Beginn eine „Intel RealSense d415“ verwendet. Die Industrie-3D-Kamera erfüllt die Anforderungen. Es kann eine 3D-Rekonstruktion des Objektes mit Farbinformationen durchgeführt werden. Die RealSense verwendet Stereo-Vision-Technologie. Sie verfügt über zwei Kameras und einen Projektor für ein vom Menschen nicht sichtbares Infrarot-Muster. Die Kamera berechnet so für jeden Pixel seine 3D-Position in Echtzeit. In dem Forschungsprojekt wurde jedoch auch überlegt, ob man die 3D-Rekonstruktion jenseits einer Industrie-Lösung erhalten kann. Grundsätzlich ist dafür eine einfache Webcam mit einem Linienlaser ausreichend. So kam die Idee einen Open-Source Lasertriangulationssensor zu entwickeln. Dieser soll mit der Intel RealSense und dem generellen Industriestandard von RGB-D Kameras verglichen werden. Dabei ist die Genauigkeit (der Tiefeninformationen), Schnelligkeit des Scans und die Auflösung der Farbinformationen ausschlaggebend. Sowohl die Intel RealSense als auch die Open-Source-Variante sollen am Ende eine Punktewolke liefern. Diese kann direkt verglichen werden. Genauso können die Position der Punkte auf eine Genauigkeit untersucht werden.

1.4. Lösungsansatz

Um den genaueren Erklärungen im Hauptteil folgen zu können, soll einmal oberflächlich der Lösungsansatz der entwickelten Anwendung erläutert werden. Die Grundlegende Idee ist die Lasertriangulation. Dafür ist eine Kamera und ein Projektor für eine Laserlinie notwendig.

Die resultierende Punktewolke der 3D-Rekonstruktion muss im Bezug zu einem Koordi-

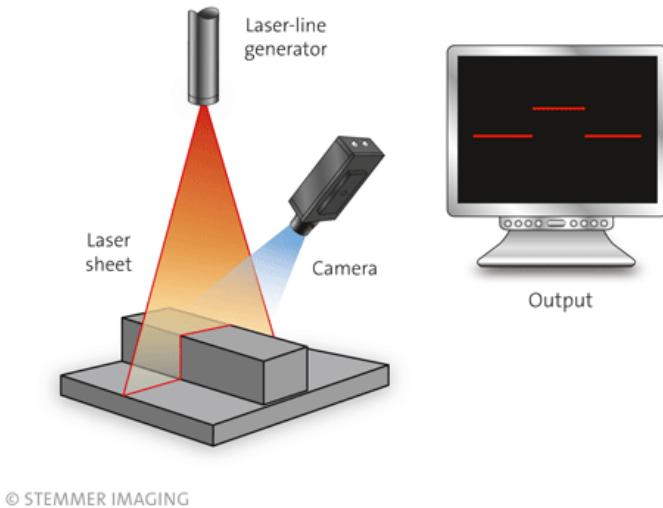


Abbildung 1: Lasertriangulation

natensystem stehen. Als Bezugspunkt wird die Kamera gewählt. Das bedeutet, dass sich die Koordinaten auf die Kamera beziehen und sich der Ursprung des Koordinatensystems an den optischen Sensor der Kamera befindet. Für die Berechnung von einem Pixel zu einem 3D-Punkt wird die Ebene bestimmt, die der Laser projiziert. Ausgehend von dem Startpunkt des Projektors und der projizierten Linie kann das Laserlicht als Ebene begriffen werden. Diese Ebene kann aus Kamerasicht als Ebenengleichung berechnet werden. In der Theorie wird dann, sobald die Ebenengleichung bekannt ist, eine Laserlinie auf die zu scannende Oberfläche projiziert. Die Kamera nimmt ein Bild von dieser Oberfläche auf. Über Bildverarbeitung wird aus dem aufgenommenen Bild die rote Laserlinie herausgearbeitet. In Abb. 1 wird unter Output Beispielhaft das Ergebnis dieses Prozesses gezeigt. Die Pixel der Laserlinie werden im nächsten Schritt auf die Ebene gespannt, indem sie in die Ebenengleichung eingesetzt werden. Dadurch wird eine dritte Dimension für die Pixel errechnet. Die 3D-Punkte sind dann, wie gefordert, aus Sicht der Kamera. Diese Methodik liefert die 3D-Repäsentation der Laserlinie, jedoch nicht von dem kompletten Objekt. Der Sensor (bestehend aus der Kamera, den Laser und der Software zusammen) muss zusätzlich über das Objekt bewegt werden. Die Aufgenommenen Linien werden dann zu einer gesamten Punktwolke zusammengefügt. Dabei muss bekannt sein, welche Strecke an Bewegung von dem Sensor zurückgelegt wurde, um die neue Linie in der richtigen Position einzufügen.

2. Hauptteil - Projektbeschreibung und Projektergebnisse

Der Hauptteil soll damit beginnen, die Grundlagen des Lösungsansatzes zu erläutern. Dazu werden zuerst die Mathematischen Grundlagen aufgezeigt. Danach werden die verwendeten selbst erstellten Algorithmen besprochen. Da dabei gewisse Bibliotheken mit Python zum Einsatz kamen, sollen diese im ersten Schritt kurz erwähnt werden. Der finale Software-Aufbau wird vollständig in einem späteren Kapitel vorgestellt.

2.1. Bibliotheken

Der Lasertriangulationssensor wurde vor Allem mit OpenCv und ROS2 entwickelt. OpenCv bietet die perfekte Unterstützung für die benötigte Bildverarbeitung. Diverse Funktionen, um Bilder zu bearbeiten, zur Kamerakalibrierung und zum Errechnen der Ebenengleichung sind in OpenCv implementiert. ROS2 kümmert sich um die Automatisierung und den generellen Ablauf eines Scavorgangs. In der Forschung und Entwicklungs-Projekt mit RINNTECH wird zusätzlich auch ROS2 als übergeordnetes System genutzt. Deshalb war ROS2 auch eine Anforderung an das Projekt. Die vorher benutzte RGB-D-Kamera Intel RealSense ist ebenfalls in der Lage über ROS2 angesprochen zu werden. Da der OpenSource-Lasertriangulationssensor diese ersetzen soll, ist die Verwendung von ROS2 ein logischer Schritt.

2.2. Der Grundlegende Aufbau

Der grundlegende Aufbau orientiert sich an den Lösungsansatz. Notwendig sind dafür nur ein Linienlaser und eine Kamera. Zuerst wurde für einen Prototyp zum Testen eine Webcam verwenden, später eine Industriekamera. Ausschlaggebend zum Funktionieren des theoretischen Lösungsansatz ist, dass die Kamera einen Linienversatz aufnehmen kann. Um das zu erreichen werden Kamera und Laser in einem gewissen Winkel zueinander gesetzt. Durch die Perspektive der Kamera entsteht der Linienversatz. Dabei ist egal, ob die Kamera von oben auf das Objekt zeigt und der Laser schräg sitzt oder andersrum. Die Lasertriangulationssensoren aus der Industrie weisen zumeist den Aufbau aus Abb. 1 auf.

Der erste Schritt in der Erarbeitung des Sensors war es, die Möglichkeit zu entwickeln, eine aufgenommene Laserlinie in eine korrekte Punktewolke umzusetzen. Danach muss es ermöglicht werden den Sensor über bzw. das Objekt unter dem Sensor bewegen zu

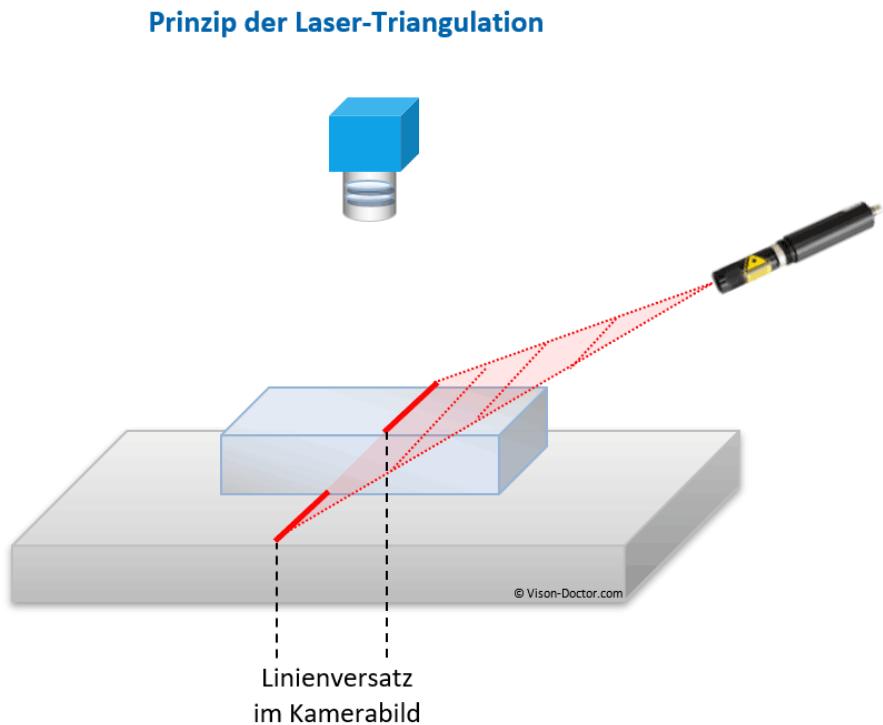


Abbildung 2: Positionen bei der Lasertriangulation

können. Mehr zu dem genauen Aufbau des fertigen 3D-Scanner wird im Kapitel 2.7 Aufbau / Hardware erläutert. Für die folgenden Erklärungen zur mathematischen Grundlage, Bildverarbeitung und Kalibrierung ist nur wichtig zu wissen, dass die Kamera und Laser in einem Winkel zueinander über dem Objekt angebracht sind. Ebenfalls ist wichtig, dass Kamera und Laser fest angebracht sind und sich zueinander nicht bewegen. Nur der ganze Sensor ist bewegbar, dabei bleiben dann aber Kamera und Laser zueinander in der gleichen Position.

2.3. OpenCv Pinhole Camera Model

Der Anfang aller Berechnungen des Lasertriangulationssensors ist immer ein Bild. Die Kamera als optischer Sensor ist die einzige Informations-Quelle. Ein Bild besteht aus Pixeln. Diese sind 2-Dimensional. Es muss also eine Möglichkeit gebunden werden, die dritte Dimension zu finden. Für die gesuchten 3D-Punkte des Objektes ist dafür eine Ebenen-Gleichung für die Laser-Ebene notwendig. Diese ist nicht von Anfang an bekannt und es gilt, diese herauszufinden. Trotzdem wird nach einem Vorgehen gesucht, welches nur aus dem Bild und mithilfe der Kamera 3D-Informationen liefern kann. Genau diese Informationen sind zugänglich mit den Pinhole Camera Model von OpenCv. In OpenCv ist eine Kamera genauso, wie eine Lochkamera begriffen.

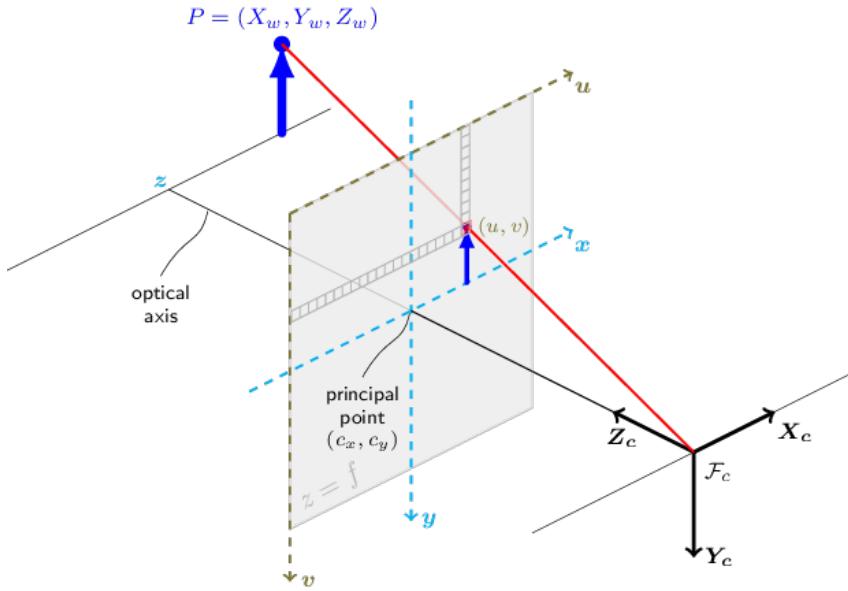


Abbildung 3: Pinhole Camera Model
(tesaugdsakif sd)

In Abb. 2 ist dieses Modell dargestellt. Dabei gilt als optisches Zentrum, welches bei einer Lochkamera das Lochblende ist. Daran orientiert sich das Kamera-Koordinatensystem. Bei einer Lochkamera ist es üblich, dass eine Szene aus der Echten Welt aufgenommen wird. Diese wird über die Öffnung auf dem Kopf gespiegelt auf einem Schirm abgebildet.

Zum Vergleich ist in Abb. 3 das Modell einer Lochkamera zu sehen. Wenn man die beiden Darstellungen vergleicht, fällt auf, dass der „Schirm“, auf dem das Bild als Reflektion dargestellt wird bei der Lochkamera hinter der Lochblende ist und bei dem Modell von OpenCv davor. Die physikalisch korrekte Darstellung ist die der Lochkamera. Jedoch ist OpenCv nicht auf die richtige physikalische Darstellung angewiesen. Mathematisch macht es keinen Unterschied, ob die Bild-Ebene vor oder hinter dem optischen Zentrum ist. So kann ein Punkt in der Szene als Vektor begriffen werden, der durch die Bild-Ebene einen Pixel definiert und zum optischen Zentrum führt. Zu sehen in Abb.2 als die rote Linie. In unserem Fall kennen wir das Bild und die genaue Position eines Pixels mit u und v . Das Ziel ist es 3D-Koordinate zu erhalten.

2.4. Mathematische Grundlage

Grundlegend für die Umrechnung des Pixels in der Bild-Ebene ist die folgende Formel:

$$s p_{pix} = A [R|t] p_w \quad (1)$$

Sie beschreibt die Transformation Projektion eines 3D-Punktes in eine Szene zu einem Punkt in der Bild-Ebene. Hierbei ist p_{pix} der Pixel im Bild. p_w ist die 3D-Koordinate, welche gesucht wird. A ist die Kamera-Matrix und $[R|t]$ die Rotation und Translation vom Kamerakoordinatensystem zum Weltkoordinatensystem. Die genau Entstehung der Formel beschreibt OpenCv in [OpenCv-CamCalib]. Kamera-Matrix, Rotation und Translation sind hierbei neu. Die genaue Bedeutung wird in dem Kapitel 2.6 Kalibrierung genannt. Zum Verstehen der Formel ist hier nur wichtig, dass diese durch eine Kamerakalibrierung herausgefunden werden können. Die Variablen sind also bekannt. Kamera-Matrix und Rotation sind beide jeweils 3x3 Matrizen. Die Translation wird durch einen Vektor (3x1) beschrieben.

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (2)$$

In der Formel (2) wird dies noch einmal genauer gezeigt. Bekannt sind also p , A und $[R|t]$. Unbekannte Variablen sind s der Scale-Factor und p_w die Weltkoordinate.

2.4.1. Homogene Koordinaten

In der Formel (2) sind die Rotation (R), die Translation (t) als homogenen Matrix und der Punkt im Weltkoordinatensystem (p_w) als homogener Vektor dargestellt. Der Vorteil vom Verwenden einer homogenen Matrix ist, dass beliebig viele Transformationen im dreidimensionalen Raum in dieser zusammengefasst werden können. Die Matrix kann dann auf einen Punkt im dreidimensionalen Raum, dargestellt als homogener Vektor, angewandt werden. In diesem Fall sind die Transformationen eine Rotation und eine Translation von dem Punkt im Weltkoordinatensystem zu dem Kamerakoordinatensystem. Das Zusammenführen von Rotation und Translation ergibt Sinn. Die Transformation von dem einen Koordinatensystem in das andere ist nur abgeschlossen, wenn sowohl die Rotation als auch die Translation auf den Zielvektor angewandt wurden. Die Formel (1) soll aber im Folgendem nach dem Punkt im Weltkoordinatensystem umgestellt werden (siehe (5)). Für

diese Umstellung wird die homogene Matrix wieder aufgeteilt. Damit dieser Vorgang keine nachvollziehbar ist soll kurz die Entstehung der homogenen Matrix für dieses Beispiel erläutert werden.

Die grundsätzliche Transformation (Rotation + Translation), die angewandt werden soll ist:

$$\begin{aligned} v' &= R v + t \\ &= \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} v + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \end{aligned} \quad (3)$$

Dabei ist v ein Beispielvektor.

Eine homogene Matrix ist eine 4x4-Matrix. Die unterste Zeile ist immer $(0, 0, 0, 1)$. Beide Transformationen können in ihr eingebunden werden. Die Matrix wird dann mit den homogenen Vektor verrechnet:

$$\begin{aligned} v'_{homogen} &= \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ 1 \end{bmatrix} \\ &= [R|t] v_{homogen} \end{aligned} \quad (4)$$

2.4.2. Transformationen

Die Grundlegende Formel soll jetzt umgestellt werden, um den 3D-Punkt im Weltkoordinatensystem anhand eines Pixels zu errechnen. Die folgende Abbildung zeigt nochmal das Pinhole Camera Model und verdeutlicht dabei die angewandten Transformationen.

Das Ziel ist es den 3D-Punkt im Weltkoordinatensystem zu errechnen. Die Rotation (R) und Translation (t) werde für die Umrechnung in das Kamerakoordinatensystem benötigt. Das sind die sogenannten extrinsischen Parameter. In Abb.: (4) wird diese Transformation unter P der **Camera Pose Matrix** dargestellt.

Die Kamera-Matrix beschreibt die Transformation zur Bild-Ebene bzw. den Pixelkoordinatensystem. Die Matrix enthält die sogenannten intrinsischen Parameter. Das ist die in Abb.: (4) gezeigte **Focal Length f** , welche den Abstand vom Kamera-Koordinatensystem zur Bild-Ebene beschreibt. Hinzu kommt der **Principal Point**. Dieser befindet sich normalerweise in dem Zentrum der Bild-Ebene. Beides zusammen ergibt die gezeigte Matrix

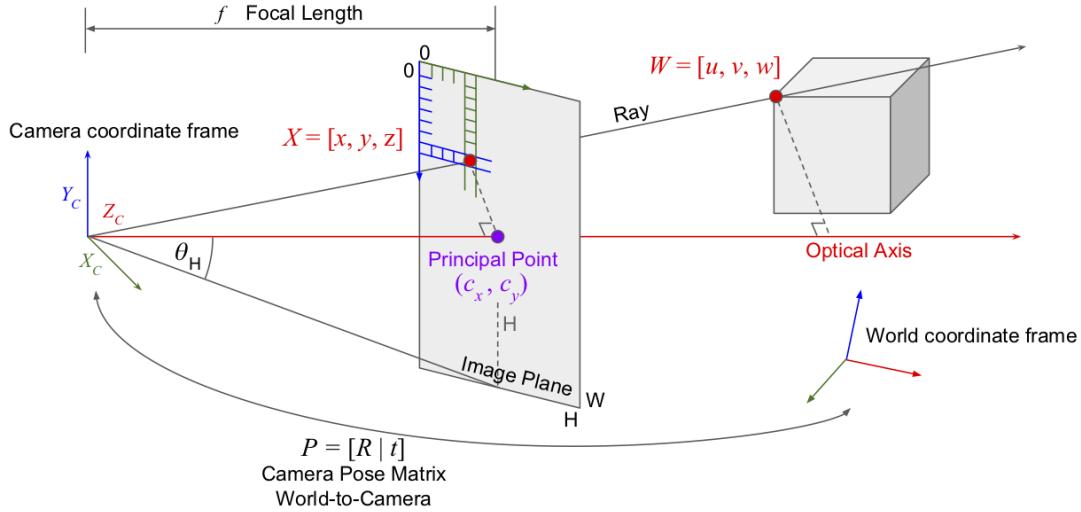


Abbildung 4: Transformationen im Pinhole Camera Model

(<https://www.oreilly.com/library/view/mastering-opencv-4/9781789533576/848e4e77-32ec-499d-9945-cb0352e28236.xhtml>)

aus (2). Letztendlich wird damit die Umrechnung vom Kamerakoordinatensystem (in Abb.: (4) **Camera coordinate frame**) zur Bild-Ebene (in Abb.: (4) **Image Plane**) dargestellt [OpenCv-CamCalib].

Wir können einen Pixel im Bild auswählen und mithilfe dieser Parameter den entsprechenden Punkt im Weltkoordinatensystem errechnen. Dazu muss die Formel (1) nach dem Punkt im Weltkoordinatensystem umgestellt werden.

$$\begin{aligned}
 \begin{bmatrix} A \\ R|t \end{bmatrix} p_w &= s p_{pix} \\
 \begin{bmatrix} R|t \end{bmatrix} p_w &= s \begin{bmatrix} A \end{bmatrix}^{-1} p_{pix} \\
 \begin{bmatrix} R \end{bmatrix} p_w &= s \begin{bmatrix} A \end{bmatrix}^{-1} p_{pix} - t \\
 p_w &= s \begin{bmatrix} R \end{bmatrix}^{-1} \begin{bmatrix} A \end{bmatrix}^{-1} p_{pix} - \begin{bmatrix} R \end{bmatrix}^{-1} t \\
 p_w &= s \vec{a} - \vec{b}
 \end{aligned} \tag{5}$$

wobei : $\vec{a} = \begin{bmatrix} R \end{bmatrix}^{-1} \begin{bmatrix} A \end{bmatrix}^{-1} p_{pix}$
 $\vec{b} = \begin{bmatrix} R \end{bmatrix}^{-1} t$

Diese Gleichung ist die Grundlage der Errechnung von 3D-Informationen. \vec{a} und \vec{b} dienen zur Vereinfachung. Wenn $\begin{bmatrix} R \end{bmatrix}$, $\begin{bmatrix} A \end{bmatrix}^{-1}$ und p_{pix} miteinander verrechnet werden entsteht ein Vektor (\vec{a}). Genauso entsteht aus $\begin{bmatrix} R \end{bmatrix}^{-1}$ und t der Vektor (\vec{b}).

Der neue Ausgangspunkt ist die errechnete Weltkoordinate. Nach Aufgabenstellung sollen errechnete Koordinaten immer aus Sicht der Kamera dargestellt werden. Dazu wird die folgende Transformation benötigt.

Weltkoordinate zu Kamerakoordinate:

$$p_{cam} = [R] p_w + t \quad (6)$$

Kamerakoordinate zur Weltkoordinate:

$$p_w = [R]^{-1} (p_{cam} - t) \quad (7)$$

Anzumerken ist noch, das s der Scale-Factor immer noch eine Unbekannte ist. Es scheint also, als ob die Gleichung (5) noch nicht lösbar sei. Sobald konkrete Werte ausgerechnet werden sollen, muss s bekannt sein. Das passiert zum ersten Mal beim Erstellen der Ebenengleichung für die Laser-Ebene bei der extrinsischen Kalibrierung. Dabei wird auch darauf eingegangen, wie s errechnet werden kann.

2.5. Bildverarbeitung

Bekannt sind nun gewisse Grundlagen, wie wir mit einem ausgewählten Pixel im Bild umgehen können. Die Rechnungen und Transformationen sollen aber nicht auf zufällige oder sogar alle Pixel im Bild angewandt werden. Sie sollen auf ganz bestimmte ausgewählte Pixel erfolgen. Und zwar ganz genau diese, die zur abgebildeten Laserlinie gehören. Die Laserlinie ist immer unser Ausgangspunkt für die 3D-Informationen. Alle anderen Pixel interessieren uns im Grunde nicht. Oder anders; die Pixel, die wir brauchen, sind die, die die Laserlinie abbilden.

Pixel können einfach und eindeutig als Tupel benannt werden. Sie können als Punkt in einem zweidimensionalen Koordinatensystem begriffen und dann mit einem horizontalen und vertikalen Wert genau gekennzeichnet werden. Benötigte wird eine Menge an diesen Tupeln, für die gilt, dass sie teil der Laserlinie im Bild sind. Mit diversen Methoden der Bildverarbeitung also jene Pixel herausgefunden und abgespeichert, um mit ihnen weiterarbeiten zu können. Das folgende Kapitel bildet somit einen Algorithmus ab, dessen Ziel es ist ein Bild entgegen zu nehmen und die Pixel der Laserlinie zurückzugeben.

2.5.1. Das Erkennen der Laserlinie

Der erste Schritt des Algorithmus muss sein, die Laserlinie im Bild zu erkennen. Am Anfang des Praktikums bin ich hier experimentell vorgegangen. Das bedeutet, dass ich mir ein Lösungsansatz des Problems ausgedacht und diesen mit einem einfachen Python-Script geprobt habe. Danach habe ich diesen kurz evaluiert und die aufgekommenen Probleme zusammengefasst. Der nächste Schritt war dann, nach einer anderen Methode zu suchen, die weniger Probleme oder auch höhere Genauigkeit aufweist. Dabei war ein wichtiges Kriterium die Genauigkeit der ausgewählten Pixel. Jeder Pixel, der vom Algorithmus markiert wird, aber nicht zur eigentlichen Laserlinie gehört wird in einem Fehler in der am Ende erstellten Punktewolke enden. Es wird also ein Punkt im Raum gezeigt, der nicht zur eingescannten Oberfläche passt. Dieser hängt dann beispielsweise in der Luft bzw. ist an einer Stelle im Koordinatensystem wo sich eigentlich nichts befindet. Dieses sogenannte Rauschen soll möglichst gering sein. Getestet habe ich das, indem ich mit den ausgewählten Pixeln eine Bitmaske erstellt und ein Binärbild erstellt. Vorliegen hatte ich dann also ein Bild in dem alle Pixel schwarz sind außer die der durch die Maske getroffene Pixel. Diese sind dann weiß. Die Laserlinie soll also in weiß gut sichtbar sein. Alle Pixel die auch weiß sind, aber offensichtlich im Vergleich zum Originalbild nicht zur Laserlinie gehören fallen aber auch sofort auf.

Die erste Idee war es, nach der offensichtlichen Erscheinung des Lasers im Bild zu suchen. Die Laserlinie ist Rot. Somit war die Methode das Bild nach roten Pixeln zu filtern und diese auszuwählen. Das schien zuerst auch gut zu funktionieren. Über OpenCv kann man eine Maske auf die RGB-Werte der Pixel anwenden. Man kennzeichnet also einen zulässigen Bereich für alle drei Werte. Dieser Bereich soll logischerweise die roten Farben abbilden. Diese Methode wird allerdings sofort problematisch, sobald sich andere rote Gegenstände im Bild befinden. Diese werden dann als Teil der Laserlinie angesehen. Ein anderes Teil so hell war, dass die Mittleren Pixel der Linie schon eher Weiß waren und nicht mehr zu dem gewählten roten Bereich gehörten. Da diese Probleme zu hohem Rauschen führen und auch durch geeignetes Anpassen der RGB-Maske nicht zu beheben gehen, wurde diese Methode verworfen. Es gab noch kurze Überlegungen die Maske statt auf den RGB-Farbraum auf den HSV-Farbraum anzuwenden, diese führten aber nicht zu auffallend besseren Ergebnissen.

Im Zuge einer Umfangreichen Recherche zum Thema Lasertriangulation und Opensource-Produzierten Laserlinienscannern bin ich auf ein Paper von Bajpai und Perelman, die auch einen Laserlinienscanner entwickelt haben. Nicht nur hier, auch in diversen anderen Schritten der Errechnung von 3D-Punkten aus den Laserlinien-Pixeln ist dieses Paper die Grundlage und Hilfestellung. Durch das Paper kam die Idee, zwei Bilder zu machen. In einem ist der Laser angeschaltet, in dem anderen nicht. Wenn diese Bilder voneinander

abgezogen werden, kommen im Grunde genau die Veränderungen hervor. Da sich nichts anders im Bild verändern sollte, außer das Erscheinen der Laserlinie, wird genau diese perfekt aufgezeigt. Voraussetzung dafür ist, dass die Umgebung der zu scannenden Fläche gleich bleibt. Einwirkungen wären zum Beispiel Änderungen bei den Lichtverhältnissen, bzw. bei der Beleuchtung oder auch eine Bewegung von Objekten zwischen der Aufnahme der zwei Bilder. Solche Einwirkungen würden auch in Rauschen enden oder sogar die Laserlinie falsch positionieren. Weitere Informationen hierzu befinden sich ebenfalls im Kapitel 2.10.3 Probleme und Schwierigkeiten. Diese Anforderungen werden für den zu entwickelnden Laserlinien-Scanner akzeptiert. Ebenfalls sind es Anforderungen, die nicht direkt von der Software beeinflusst werden können. Sie sollten somit beim Aufbau des Scanners genauer beachtet werden und auf die Bildverarbeitung keinen Einfluss mehr haben dürfen. Das proben dieser Methode führte auch mit dem Testen über das Binärbild zu sehr guten Ergebnissen. Zusätzlich ist diese Methode sehr einfach über OpenCv umzusetzen. Festzuhalten ist damit, dass die Kamera zwei Bilder aufnehmen muss, eins Bild mit Laserlinie und ein Bild ohne. Das Bild ohne Laserlinie wird dann von dem mit Laserlinie angezogen. Der Algorithmus zum finden der Laserlinien-Pixel arbeitet dann mit der Differenz.

2.5.2. Grauwert-Bild

Das Ergebnis der Differenz von zwei Bildern ist, wie bei einer Maske für den roten Farbbereich, jedoch kein Binärbild. Es ist also nun erkennbar, wo sich die Laserlinie befindet, jedoch müssen eindeutig die Pixel die weiterverarbeitet werden sollen ausgewählt werden. Aber nicht nur die Laserlinie, sondern auch die Pixel, die nicht dazu gehören sind nicht eindeutig schwarz mit einem RGB-Wert von ($R=0, G=0, B=0$). Einen kleinen für den Menschen zumeist nicht sichtbaren Unterschied in zwei nacheinander aufgenommenen Bildern wird es immer geben. Es muss also immer noch genau festgelegt werden, welche Pixel für die Laserlinie ausgewählt werden. Die Differenz von den zwei Bildern macht dies allerdings einfacher. Gleichbleibende Stellen zwischen den beiden Bildern wie zum Beispiel auch sehr helle Stellen oder auch andere rote Stellen sind in dem Differenz-Bild nicht mehr erkennbar, wobei sie in der vorherigen Methode zu Problemen führten, da man sie nicht genau von der Laserlinie unterscheiden konnte. Gleichzeitig ist die Laserlinie in einem gut aufgelösten Bild mehrere Pixel dick. Ideal wäre allerdings eine dicke von genau einem Pixel. Das Differenz-Bild muss demnach noch nachbearbeitet werden

Als erste Schritt wird dabei das Bild zu einem Grauwert-Bild konvertiert. Vorteil davon ist, dass jetzt nicht mehr der RGB-Wert mit drei eigenen Werten ausschlaggebend ist, sondern nur noch die Intensität, bzw. der Grauwert eines Pixels. Die jeweilige Intensität

für einen Pixel berechnet OpenCv nach der folgenden Formel [Vgl: OpencvDoc]:

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \quad (8)$$

Damit hält OpenCv sich an die ITU-R BT 601 [vgl.: CCIR 601] und dem darin festgelegten Farbmodell YC_bC_r . Dabei handelt es sich um eine Konvention, wie digitale Video-Signale zu kodieren sind.

Die Formel (8) zeigt ebenfalls, dass aus den drei Farbwerten nun ein einzelner Grauwert errechnet wird. Allgemein kann man dabei festhalten, dass je höher die einzelnen Farbinformationen waren, um so höher wird auch der Grauwert bzw. die Intensität sein. Das betrifft die Pixel der Laserlinie. Die anderen Pixel besitzen Farbwerte von ungefähr 0-5 und fallen nicht weiter auf.

Die Entwicklung der folgenden Methodik, die Laserlinie auf eine Breite von einem Pixel zu reduzieren zeigte eine neue Anforderung auf. Bekannt sind nun die Intensitäten der Pixel. Die Laserlinie kann lokalisiert werden, indem man die höchsten Intensitäten im ganzen Bild sucht. Dabei braucht man ein Schema, wie das Bild abgesucht wird. Zum Beispiel sucht man in jeder Zeile im Bild die höchste Intensität. Das ist jedoch nur sinnvoll, wenn die Laserlinie im Bild von oben nach unten geht. Das ist wieder eine Anforderung an den Aufbau. Im Kapitel 2.2 wurde beschrieben, dass Kamera und Laser zueinander in der gleichen Position bleiben. Demnach kann der Laser entsprechend angebracht werden.

2.5.3. Gauss-Filter

Die einfachste Methode wäre also jede Zeile nach der höchsten Intensität zu suchen und den betreffenden Pixel auszuwählen. Dieser Pixel befindet sich je nach Aufnahme jedoch nicht genau in der Mitte der Laserlinie. Die erarbeiteten Linien können so sehr "wellig" werden, da Pixel untereinander zum Teil stark in ihrer Position in der Zeile abweichen. Den Pixel mit der höchsten Intensität als Ausgangspunkt zu nehmen ist gut, jedoch sollen die umliegenden Pixel einen Einfluss machen können, um die Genauigkeit der Laserlinie zu erhöhen. Dazu wurden zwei Vorgänge entwickelt.

Die erste Methode ist es, einen Gauss-Filter über das Grauwert-Bild zu legen. Die genaue Beschreibung dieser Operation würde in diesem Zusammenhang zu weit gehen. Nähere Informationen befinden sich in [?, ?, ?]. Der Gauß-Filter wird auf jeden Pixel im Bild angewandt. Dabei ändert er den Wert des ausgewählten Pixels in Abhängigkeit der umliegenden Pixel. Die Auswirkung auf das ganze Bild ist dabei, dass die Strukturen etwas ineinander verschwimmen. Das ist eine Standard-Bildverarbeitungsoperation, die auch mögliches Rauschen im Bild verringert.

2.5.4. Das Erstellen von Subpixeln

Nachdem der Filter verwendet wurde, kommt die Operation, in der jede Zeile des Bildes nach dem Laserlinien-Pixel abgesucht wird. Hier wird zuerst der Pixel mit der höchsten Intensität gesucht. Dabei kann immer noch mehr Genauigkeit erzielt werden, wenn man die Pixel links und recht neben dem Ausgewählten untersucht. So fließt nicht nur die Intensität als Auswahlkriterium ein, sondern auch die restlichen Pixel der Laserlinie in dieser Zeile. Da die erschlossenen Pixel in 3D-Koordinaten umgewandelt werden und somit vom Bild und der Pixel-Darstellung getrennt werden, ist es nicht mehr notwendig nur ganze Zahlen zur Zuverwendung. Die Werte können also auch Nachkommastellen haben und sogenannte Subpixel ergeben. Für dieses errechnen des passenden Subpixel wurde ein Algorithmus erstellt, der wie folgt funktioniert:

1. Threshold

Zuerst wird ein passender Threshold gesucht um eine Grenze festzulegen, ab welchen Wert überhaupt ein Pixel gefunden werden soll. Wenn die maximale Intensität in einer Zeile unter dem gewählten Threshold liegt, wird diese übersprungen. Damit wird auch ungewolltes Rauschen entfernt, da eine Intensität, die so niedrig ist, nicht zur Laserlinie gehören kann. Zuerst wurde dabei ein fester Threshold gewählt. Damit wäre dieser nicht an das Bild angepasst. Bilder können allerdings unterschiedlich ausfallen und nicht wie gewollt auf den Threshold reagieren. Der Threshold soll genau auf das aktuelle Bild angepasst sein. Die Otsu-Methode ist ein Schwellenwertverfahren, welches genau den gewollten Threshold liefern kann. Dabei wird ...

So wird ein angepasster Threshold gefunden und die höchste Intensität der aktuellen Zeile wird auf diesen geprüft.

2. Einbeziehen der benachbarten Pixel

Der Ausgangspunkt ist jetzt der Pixel mit der höchsten Intensität und dieser liegt über dem Threshold. Die benachbarten Pixel sollen jetzt in einen endgültigen Wert mit einbezogen werden. Um das zu erreichen wird eine Parabel genutzt. Es gilt eine quadratische Funktion in der Form $ax^2 + bx + c = I_x$, wobei I die Intensität und x die Position in der jeweiligen Zeile ist. Bei einem Bild, das beispielsweise 1920 Pixel breit ist, ist das ein Wert zwischen 0 und 1919. Die quadratische Funktion wird auf den ausgewählten Pixel und seine Nachbarn angewandt. Somit gilt:

$$\begin{aligned}
 ax^2 + bx + c &= I_x \\
 a(x+1)^2 + b(x+1) + c &= I_{x+1} \\
 a(x-1)^2 + b(x-1) + c &= I_{x-1}
 \end{aligned}$$

daraus folgt : (9)

$$\underbrace{\begin{pmatrix} x^2 & x & 1 \\ (x+1)^2 & (x+1) & 1 \\ (x-1)^2 & (x-1) & 1 \end{pmatrix}}_X \underbrace{\begin{pmatrix} a \\ b \\ c \end{pmatrix}}_{\vec{abc}} = \begin{pmatrix} I_x \\ I_{x+1} \\ I_{x-1} \end{pmatrix}$$

Das Ziel ist es den Vektor \vec{abc} herauszufinden. Da die Intensitäten und die x-Werte bekannt sind, ist \vec{abc} die einzige Unbekannte und kann errechnet werden. Der nächste Schritt ist es die Nullstelle der Parabel herauszufinden. Sie befindet sich dort, wo die tatsächliche Intensität am höchsten ist. Dieser x-Wert befindet sich dann in den meisten Fällen zwischen zwei Pixeln. Dafür gilt:

$$\begin{aligned}
 2a \cdot x + b &= 0 \\
 x &= \frac{-b}{2a}
 \end{aligned}$$
(10)

b und a sind durch das Errechnen von \vec{abc} bekannt und ein Wert für x kann gefunden werden.

In der Rechnung fällt auf, dass x für jede Zeile individuell ausgerechnet wird. Dabei ändert sich in (9) die Werte für x und die Intensitäten. Hier kann man die Berechnung vereinfachen. wählt man für $x 0$, entsteht die folgende Matrix:

$$X = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & -1 & 1 \end{pmatrix}$$
(11)

Wenn \vec{abc} mit dieser Matrix und den individuellen Intensitäten berechnet wird, errechnet man immer die Abweichung zu Mitte. Dabei ist die Mitte 0 und das entspricht den ausgewählten Pixel mit der höchsten Intensität. Man erhält eine Zahl zwischen -1 und 1. Diese wird mit der x-Position des Ausgangspixel in der Zeile verrechnet. Der Algorithmus wurde somit vereinfacht, da sich in jeder Rechnung nur der Intensitäten-Vektor ändert und immer die gleiche Matrix X gewählt werden kann.

Mit dem Finden des x-Wert ist die Bearbeitung einer Zeile fertig. Es gibt nun eine eindeutige 2D-Koordinate für einen Punkt der Laserlinie. Der Algorithmus geht über jede Zeile (y) und findet einen x-Wert. Dabei fügt er den gefundene Subpixel (x, y) in ein Array ein. Die Menge dieser Subpixel ist die Abbildung der Laserlinie in einen 2-dimensionalen Raum. Diese wurde als Output gefordert.

2.6. Kalibrierung

Die Kamerakalibrierung ist ausschlaggebend, um die Formel (2) benutzen zu können. Durch sie wird die Kamera-, Rotationsmatrix und der Translationsvektor gefunden. Dabei unterscheidet man zwischen intrinsische und extrinsischer Kalibrierung, bei denen auch ein unterschiedliches Kalibrierverfahren angewandt wird.

2.6.1. Intrinsische Kalibrierung

Die intrinsische Kalibrierung beschäftigt sich mit dem *Inneren* der Kamera. Ergebnis der Kalibrierung ist die Kameramatrix A . Die Kamerakalibrierung ist in OpenCv implementiert und wurde in diesem Projekt genutzt [OpenCv-CamCalib]. Die Implementierung richtet sich nach der Kamerakalibrierung nach Zhang [Zhang] und nach Bouguet [Bouguet].

Der grundlegende Vorgang ist es, ein bekanntes Muster in verschiedenen Positionen mit der Kamera aufzunehmen. So ein Muster ist beispielsweise ein Schachbrett, welches auch in dieser Arbeit für das Kalibrieren benutzt wurde. Die Maßen von den Kacheln sind dementsprechend dann bekannt und der Druck muss genau sein. Dabei muss es sich nicht um ein herkömmliches Schachbrett handeln. Die Größe und die Menge an Kachel kann zum kalibrieren angegeben werden. Man muss OpenCv also sagen, nach welchem Schachbrett der Algorithmus suchen muss.

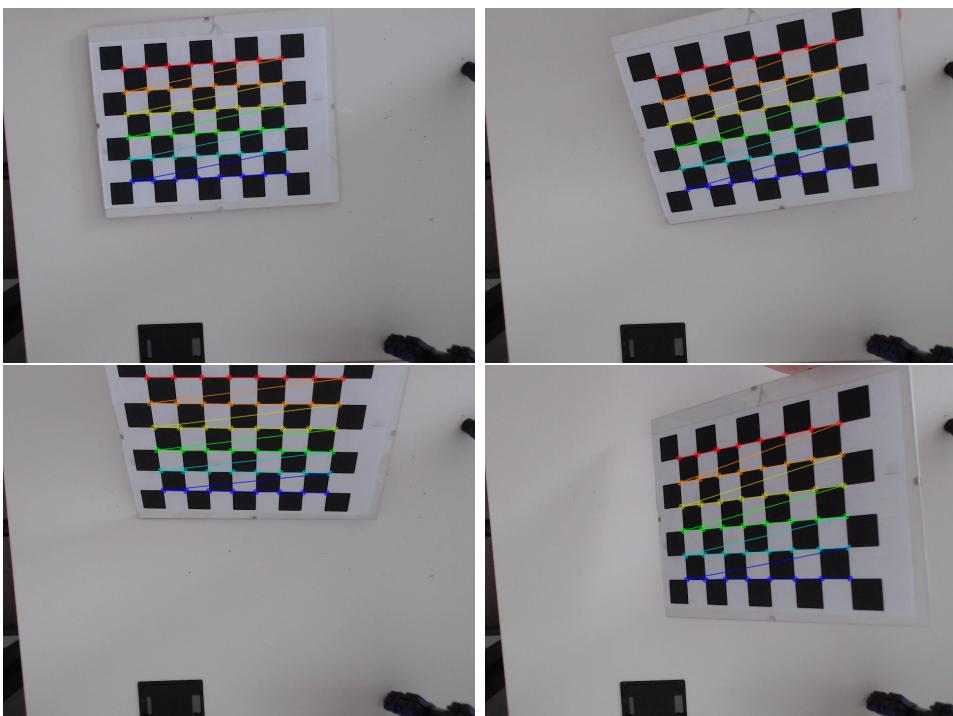


Abbildung 5: Schachbrett-Kalibrierung

Die benutzte Methode heißt **calibrateCamera()**. Sie nimmt beliebig viele Bilder entgegen. In der eigenen Implementierung wurde gemäß [OpenCv-Cam] vorgegangen. Dabei handelt es sich um die offizielle Einführung zur Kamerakalibrierung von OpenCv. In dem Beispiel in Abb.: (5) wurden der Einführung folgend auch die Ecken des definierten Schachbrettes eingezeichnet. Damit kann man im Grunde sichbar machen, dass der Algorithmus das Schachbrett richtig erkannt hat. Gemäß [OpenCv-Cam] werden mindestens 10 Bilder benötigt, um gute Ergebnisse zu liefern. Daran wurde sich ebenfalls gehalten. Die Methode gibt die folgenden Werte zurück:

Kameramatrix

Die **Kameramatrix** wird errechnet und ausgegeben. Dabei handelt es sich um die vollwertige Matrix A aus den Formeln (1) und (2).

Distortion – Koeffizienten

Die Distortion-Koeffizienten werden gemäß [OpenCv-Cam] geliefert. Distortion bedeutet Verzerrung und bezieht sich auf das aufgenommene Bild. Dies Auswirkungen sind, dass aufgenommene gerade Linien in der Szene im Bild nicht mehr gerade dargestellt werden. Sie biegen sich bzw. sind verzerrt. Es gibt zwei verschiedene Verzerrungen, beschrieben in [Dist] und von OpenCv selbst in [OpenCv-CamCalib].

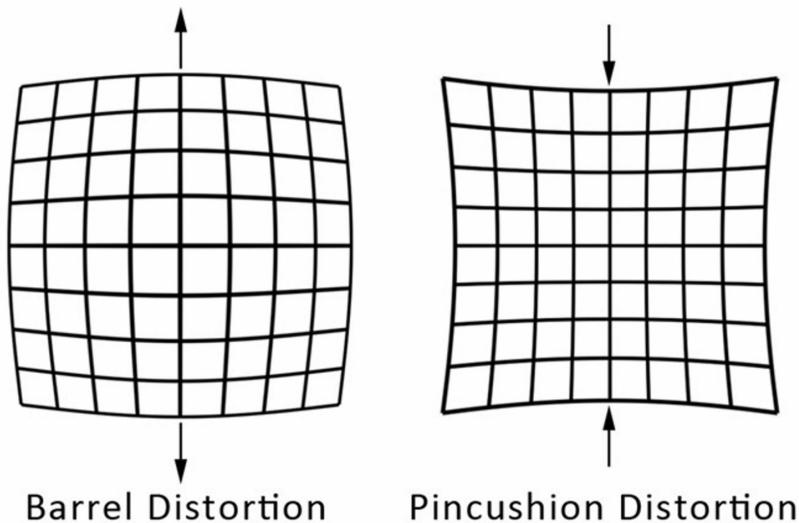


Abbildung 6: Verzerrung bei Bildern
(<https://www.learningwithexperts.com/photography/blog/look-out-for-lens-distortion>)

Barrel Distortion bedeutet zu deutsch Faß-Verzerrung und Pincushion Distortion zu deutsch Nadelkissen-Verzerrung. Um dieser Verzerrung entgegenzuwirken gibt es in OpenCv die Methode **undistort()**. Diese bringt ein Bild wieder in den normalen Zustand und benutzt dabei die gefundenen Koeffizienten [OpenCv-Cam]. Das benutzen dieser Methode ist

unumgänglich. Die Laserlinie ist bei einem ebenen Untergrund gerade. Wenn diese allerdings gebogen ist, führt das beim Aufspannen auf die Ebene zu Höhenunterschieden zwischen den Punkten. Die Verzerrung ist nicht Teil des Pinhole Camera Models und somit auch kein Teil der Formel (1). Bevor mit aufgenommenen Bildern gearbeitet wird, werden die jedoch mit der Methode `undistort()` entzerrt. Ein Beispiel direkt mit den aktuellen Aufbau soll das verdeutlichen:

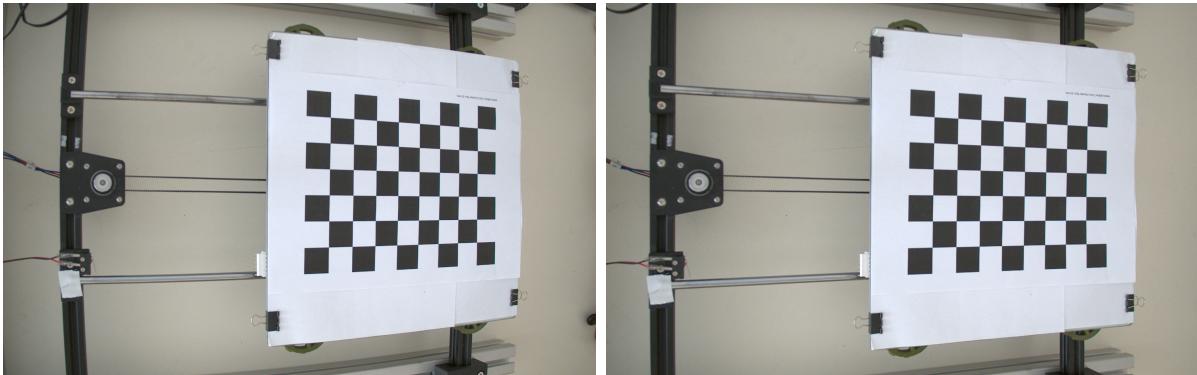


Abbildung 7: Bildbeispiel

Ebenso ist erkennbar, dass es sich bei den verwendeten Kameramodell um eine *Barrel Distortion* handelt.

Rotationen und Translationen

Die Methode gibt ebenfalls eine Rotationsmatrix und einen Translationsvektor für jedes übergebene Schachbrett zurück. Das sind extrinsische Parameter. In jedes Schachbrett wird ein Koordinatensystem gelegt. Die Rotation und Translation beschreiben dann die Transformation von der Kamera zu diesem Koordinatensystem. Es werden also auch schon extrinsische Parameter geliefert. Diese sind allerdings für den Lasertriangulationssensor selbst nicht ausschlaggebend. Damit dieser mit dem Laser kalibriert werden kann, also eine Ebenen-Gleichung für den Laser gefunden wird, muss der Laser mit auf den Bildern abgebildet sein. Dann kann man die Linie in Bezug auf das gefundene Weltkoordinatensystem weiterverwenden. Für die extrinsische Kalibrierung wurde eine extra Methode entwickelt, welche im nächsten Kapitel beschrieben wird.

Die intrinsische Kalibrierung ist nur dazu da, die Kameramatrix zu finden und die Distortion-Koeffizienten zu bestimmen, damit die verwendeten Bilder nicht verzerrt sind. Die Rotation und Translation zu den verschiedenen Positionen des Schachbrettes sind nicht von Interesse.

2.6.2. Extrinsische

2.7. Aufbau / Hardware

2.8. Aufbau / Software

2.8.1. Python (Bibliothek)

2.8.2. ROS2

2.9. Qualitative Ergebnisse

2.10. Evaluation

2.10.1. Testen von Genauigkeit

2.10.2. Fehler

2.10.3. Probleme und Schwierigkeiten

3. Fazit

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier

ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

4. Ausblick

4.1. Erweiterung

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten!

4.2. Anpassungen

wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

4.2. Anpassungen

A. Anhang

A.1. Anhang A

A.2. Anhang B

A.3. Anhang C

Abbildungsverzeichnis

Abb. 1	Lasertriangulation	4
Abb. 2	Positionen bei der Lasertriangulation	6
Abb. 3	test	7
Abb. 4	Transformationen	10
Abb. 5	Schachbrett-Kalibrierung	18
Abb. 6	Verzerrung bei Bildern	19
Abb. 7	Beispiel für die Verzerrung in Bildern	20

Tabellenverzeichnis

Quellcodeverzeichnis

Abkürzungsverzeichnis

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Gießen, 20.08.2020

Ort, Datum

Unterschrift