HafenCity Universität Hamburg

Universität für
Baukunst und Metropolenentwicklung
Henning-Voscherau-Platz 1
20457 Hamburg

Aufbau eines photogrammetrischen Messsystems mit Raspberry-Pi-Kameras als Low-Cost-Sensoren

für die Aufnahme von kleinen Objekten

Geodäsie und Geoinformatik

Masterthesis
Sommersemester 2024

Florian Timm

Abgabedatum: 28. Oktober 2024
mit Korrekturen vom 09. Februar 2026

Verfasser

Florian Timm

E-Mail: florian.timm@hcu-hamburg.de / info@florian.timm.de

Erstprüfer

Prof. Dr.-Ing. Thomas Kersten
HafenCity Universität Hamburg
Henning-Voscherau-Platz 1, 20457 Hamburg

E-Mail: thomas.kersten@hcu-hamburg.de

Zweitprüfer

Dipl.-Ing. Kay Zobel
HafenCity Universität Hamburg
Henning-Voscherau-Platz 1, 20457 Hamburg

E-Mail: kay.zobel@hcu-hamburg.de

Kurzzusammenfassung

Die Photogrammetrie ermöglicht die Erstellung von 3D-Modellen unter Zuhilfenahme relativ einfacher Technik. Allerdings ist der Zeitaufwand für die Aufnahme der Bilder oft hoch, sodass sich diese Methode nicht für die Erfassung einer großen Anzahl von Objekten eignet, beispielsweise für die Digitalisierung von Museumsbeständen. Systeme, die auf mehreren fest installierten Kameras basieren, verwenden in der Regel hochwertige Kameras. Dies führt jedoch zu einem signifikanten Anstieg der Hardwarekosten.

Im Rahmen dieser Arbeit erfolgt eine Untersuchung des Lösungsansatzes, mehrere kostengünstige Kameras zu verwenden, die fest auf einem Rahmen montiert sind. Den Kern dieser Untersuchung bildet der Aufbau eines photogrammetrischen Messsystems für kleine Objekte, welches aus Raspberry-Pi-Kameras besteht. Im Rahmen der Entwicklung ist die Programmierung einer Schnittstelle zur Synchronisation der Kameras sowie die Entwicklung einer Möglichkeit zur Kalibrierung der Kameras vorgesehen. In einem nächsten Schritt wird die Genauigkeit der Erfassung überprüft. Das Endergebnis soll im Idealfall einem photogrammetrischen Laien ermöglichen, schnell und ohne lange Einarbeitungszeit 3D-Modelle in ausreichender Auflösung und Qualität zu erzeugen.

Abstract

Photogrammetry enables the creation of 3D models using relatively simple technology. However, the time required to capture the images is often high, meaning that this method is not suitable for capturing numerous objects, for example when digitizing museum collections. Systems based on several permanently installed cameras generally use high-quality cameras. However, this results in a significant increase in hardware costs.

This thesis investigates the solution of using several low-cost cameras mounted on a fixed frame. The core of this investigation is the construction of a photogrammetric measurement system for small objects, which consists of Raspberry Pi cameras. The programming of an interface to synchronize the cameras and the development of a way to calibrate the cameras are planned as part of this. The next step will be to check the accuracy of the recording. Ideally, the end result should also enable a photogrammetric layman to generate 3D models in acceptable resolution and quality quickly and without a long familiarization period.

Inhaltsverzeichnis

1. Einleitung

Im Rahmen dieser Arbeit wurde ein Photogrammetrie-System auf Basis von Raspberry-Pi-Kameras entwickelt. Das System soll es ermöglichen, kleine Objekte bis etwa 40 cm mit geringem finanziellem und personellem Aufwand zu erfassen und daraus 3D-Modelle zu erstellen.

Die folgenden Abschnitte geben einen Überblick über die Motivation und das Konzept der Arbeit und stellen den bisherigen Stand der Technik und Forschung dar.

1.1. Motivation

In Museen besteht vielfach der Wunsch, den Bestand an Exponaten zu digitalisieren, beispielsweise um diese online in virtuellen Ausstellungen zu präsentieren. Entsprechende Handreichungen des Deutschen Museumsbundes legen auch die Digitalisierung als 3D-Modelle nahe, verweisen aber auf den großen Aufwand und Formatprobleme (vgl. Deutscher Museumsbund e. V., 2022, S. 43). Neben der reinen Präsentation der Bestände ist auch die Dokumentation und Erforschung der Exponate von Interesse. Beispielsweise durch Naturkatastrophen, Brände und bewaffnete Konflikte können Kulturgüter jederzeit verloren gehen oder beschädigt werden. Die Digitalisierung ermöglicht es, diese Objekte zu bewahren und der Öffentlichkeit zugänglich zu machen. Neben der vorsorglichen Digitalisierung ist aber auch die schnelle Erfassung von Objekten nach einem Schadensfall wichtig (vgl. Deutsches Archäologisches Institut, 2024).

Weiterhin besteht in vielen weiteren Bereichen der Bedarf, dreidimensionale Modelle einfach und kostengünstig zu erfassen, beispielsweise in der Archäologie, der Spiele- und Filmindustrie für die 3D-Modellierung oder auch der Industrie zur Entwicklung und Qualitätskontrolle (vgl. Luhmann, 2023, S. 37f).

1.2. Konzept

Das zu entwickelnde System soll dabei von Laien mit kurzer Einarbeitungszeit bedienbar sein. Dazu ist eine weitgehende Automatisierung der Schritte erforderlich. Auch ein eigener Nachbau des Systems soll einfach möglich sein. Um Lizenzkosten zu sparen, ist die Möglichkeit der Nutzung von Open-Source-Software zu prüfen. Das System soll kleine Objekte in einer Größenordnung bis etwa 40 cm Durchmesser erfassen können.

Nach einer Analyse der Anforderungen an ein solches System ist die Entwicklung eines Prototyps vorgesehen, welcher aus mehreren Kameras besteht, die auf einem Rahmen montiert sind. Die Anordnung der Kameras erfolgt dabei so, dass eine Erfassung des Objektes aus verschiedenen Blickwinkeln gewährleistet ist. Die Kameras sind dabei synchron auszulösen und die Daten anschließend automatisch zu übertragen, sodass eine direkte Verarbeitung der Aufnahmen zu einem 3D-Modell erfolgen kann.

Zur Untersuchung möglicher Optimierungen des Systems ist - neben der eigentlichen Entwicklung und Genauigkeitsuntersuchung - eine Analyse der Anzahl der Kameras auch unter Nutzung eines Drehtellers vorgesehen. Ziel ist die Identifikation von Potenzialen zur Senkung der Hardwarekosten oder alternativ zur Steigerung der Auflösung und Genauigkeit.

Zusammengefasst sollen folgende Anforderungen erfüllt werden:

1. Erfassung von kleinen Objekten bis etwa 40 cm Durchmesser
2. automatisierte Erfassung der Bilder
3. automatisierte Übertragung der Bilder
4. automatisierte Verarbeitung der Bilder zu einem 3D-Modell
5. einfache Bedienbarkeit
6. geringe Kosten
7. einfacher Nachbau
8. Nutzung von Open-Source-Software
9. Transportmöglichkeit/Nutzung in anderen Ländern
10. Möglichkeit zur Erweiterung

1.3. Stand der Technik und Forschung

Der Ansatz, Kameras auf einem festen Rahmen zu montieren, Bilder aufzunehmen und anschließend automatisiert 3D-Modelle auf Basis von Photogrammetrie zu erzeugen, ist bereits weit verbreitet und erprobt. Hauptsächlich unterscheiden sich diese im Vergleich zu dem hier untersuchten Ansatz in der Wahl der Kameras und der Zielgruppe. Häufig werden hochwertige Kameras verwendet, die jedoch entsprechend hohe Kosten in der Anschaffung verursachen.

Bereits Anfang der 1990er Jahre wurde ein modulares System von Leica entwickelt, das die Objekterfassung mittels mehrerer Digitalkameras ermöglichte, das sogenannte Leica POM. Dieses System wurde für industrielle Anwendungen entwickelt, beispielsweise für die Qualitätskontrolle von Bauteilen. Es ermöglicht die flexible Nutzung verschiedener Kamerasysteme, Drehteller und Lichtquellen. Wie bei dem hier geplanten System werden die Daten ebenfalls automatisch übertragen und verarbeitet. Auch können bereits einige Punkte automatisch gemessen werden. Ähnlich wie das aktuelle System TubeInspect (vgl. Bösemann, 1996) von Hexagon - dem Mutterkonzern von Leica und AICON - nutzt es jedoch kein Structure-from-Motion (SfM) zur Erstellung der 3D-Modelle, sondern Kantendetektion und anschließenden Bildvergleich. Hierfür wird das Objekt von hinten beleuchtet und die Kanten erfasst. Die Genauigkeit des Systems wird mit 0,1 mm angegeben. (vgl. Luhmann, 1990)

Auch Raspberry-Pi-Kameras wurden bereits in der Photogrammetrie eingesetzt, beispielsweise bei dem Projekt Pi3DScanner. Dieses verfolgt das Ziel, ein kostengünstiges Photogrammetrie-System auf Basis mehrerer Raspberry-Pi-Kameras zu entwickeln. Der Ansatz ist ähnlich, jedoch kommerziell und aufgrund der Nutzung einer größeren Anzahl von Kameras höherpreisiger als das hier geplante Vorhaben. Zudem liegt der Fokus, wie bei den meisten Projekten dieser Art, auf der Erfassung von größeren Objekten oder Personen. (vgl. Garsthagen, 2021)

An der HafenCity Universität wurde 2015 ein Innenraum-Erfassungssystem entwickelt, das auf Basis einer Raspberry-Pi-Kamera und einem Laserentfernungsmessgerät arbeitet. Hierbei wurde auch die Genauigkeit der Kameras untersucht und die Möglichkeit der Kalibrierung geprüft. Bei dem verwendeten, älteren Raspberry Pi Camera Module v2 handelt es sich jedoch um eine Kamera mit Fixfokus, wodurch die Kalibrierung vereinfacht ist (siehe Abschnitt 3.1). Hier wurde dennoch ein instabiler Bildhauptpunkt festgestellt, dass Genauigkeitspotenzial nach Simultankalibrierung aber mit einer Spiegelreflexkamera vergleichbar bezeichnet. (vgl. Kersten et al., 2016b)

Ansätze für die Erfassung von kleineren Objekten in Kombination von Raspberry-Pi-Kamera-Modulen sind bisher nicht bekannt. Problematisch ist hierbei die geringe Schärfentiefe im Makrobereich (vgl. Unterabschnitt 2.2.4). Dieses Problem tritt auch bei höherwertigen Kameras und noch kleineren Objekten auf, bei denen ein Abblenden des Objektivs nicht mehr für ausreichend Schärfentiefe sorgt. Auch im Bereich der Mikroskopie ist dieses Problem weit verbreitet. Clini et al. (2016) zeigt hier eine Lösung durch die Nutzung von Fokusstacking, bei der mehrere Bilder mit unterschiedlichen Fokussierungen aufgenommen und anschließend zusammengefügt werden. Dieser Ansatz wurde auch in dieser Arbeit geprüft.

Der Einsatz von Structure-from-Motion (SfM) – die automatische Erstellung von 3D-Modellen aus Bildern (siehe Kapitel 2) – wurde für die Erfassung von Kulturgütern bereits 2012 an der HafenCity Universität untersucht (vgl. Kersten et al., 2012). Hierbei wurden bei der Erfassung von Objekten verschiedener Größenordnungen wie Gebäude oder kleine Artefakte zum Teil Ergebnisse erzielt, welche mit Laserscannern vergleichbar sind. Manche Testobjekte, vor allem mit texturarmen Oberflächen, führten jedoch zu unbrauchbaren Ergebnissen, aber eine stetige Verbesserung der Software war schon damals absehbar. Auch die Nutzung von Open-Source-Software wurde bereits untersucht. Genauigkeitsuntersuchungen von Nikolov & Madsen (2016) an der Universität Aalborg zeigten schon bessere Ergebnisse, jedoch war auch hier je nach Software bei einigen Objekten keine Erzeugung des 3D-Modells möglich, beispielsweise bei glänzenden oder texturlosen Oberflächen.

In der praktischen Arbeit zur Dokumentation von Kulturgütern ist Structure-from-Motion (SfM) bereits angekommen. Bei archäologischen Grabungen ist die Nutzung zur Dokumentation der Grabungsflächen inzwischen üblich (vgl. Reinhard, 2013). Hier hat sich SfM als Ergänzung zur Tachymetrie und als Alternative zum Laserscanning durchgesetzt.

Auch der Einsatz zur Rettung von mobilem Kulturgut wird geprüft. Das Projekt KulturGutRetter des Deutschen Archäologischen Instituts und des Technischen Hilfswerks hat sich zum Ziel gesetzt, durch verschiedene Gefahren wie Naturkatastrophen und Kriege bedrohte Kulturgüter zu dokumentieren und nach Möglichkeit zu retten. Dabei wird auch SfM als mögliche Technik zur Dokumentation von mobilen und immobilen Kulturgütern genannt (vgl. Busen & Wedekind, 2023, S. 48). Dazu wird ein mobiles Rettungslabor eingesetzt, das als Luftfracht zum Einsatzort transportiert werden kann. Für die Basisdokumentation der mobilen Kulturgüter wird bisher eine Fotobox mit einer Kamera eingesetzt. Die Systeme befinden sich noch in der Entwicklung, sodass eine Integration von SfM denkbar ist. Die Bedienung ist hier durch Fachpersonal vorgesehen. (vgl. Deutsches Archäologisches Institut, 2024)

2. Photogrammetrische Grundlagen

Dieses Kapitel beschreibt die notwendigen Bedingungen und die Grundlagen der Rekonstruktion des Objektes als 3D-Modell. Hierfür wird Photogrammetrie in Form einer Structure-from-Motion-Pipeline genutzt. Der allgemeine Ablauf ist in Abbildung 2.1 dargestellt. Grau dargestellte Schritte werden nicht von der entwickelten Software aus Kapitel 5, sondern von externer Software durchgeführt.

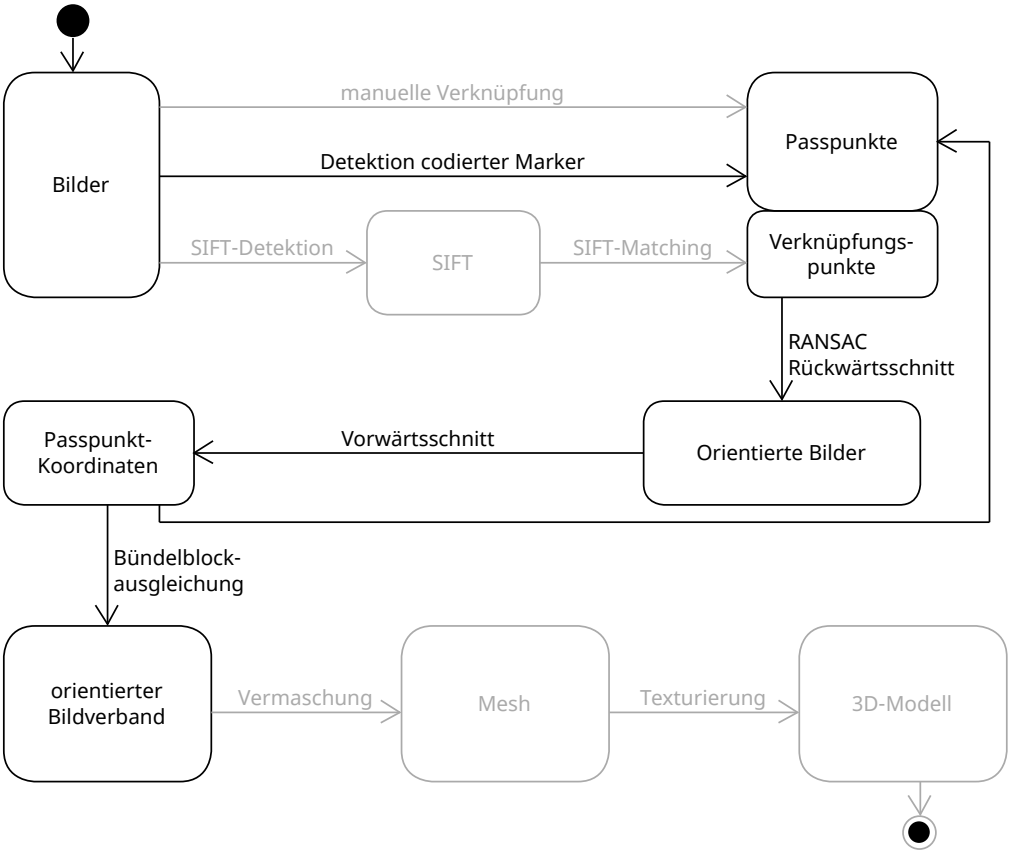


Abbildung 2.1.: Ablauf der Bildauswertung mittels Structure-from-Motion (SfM), nach Luhmann 2023, S. 492

Zunächst werden die Bilder aufgenommen (siehe Abschnitt 2.2). Hierbei ist wichtig, dass sich die Bildinhalte überlappen. Die Bilder werden dann verknüpft, indem identische Punkte in den Bildern identifiziert werden (siehe Abschnitt 2.4). Aus den identifizierten Punkten werden dann die Positionen und Ausrichtung der Kameras und

Verknüpfungspunkte in einem lokalen Koordinatensystem ohne bekannten Maßstab berechnet. Die so erzeugten Daten werden anschließend in einer Bündelblockausgleichung gemeinsam optimiert (siehe Abschnitt 2.6). Durch die Nutzung von bekannten Größen, beispielsweise kalibrierten Maßstäben, kann dieses System transformiert werden.

2.1. Abbildung

Grundlage der Photogrammetrie ist die Abbildungsgleichung. Diese beschreibt die Abbildung eines Punktes im Raum auf dem Sensor beziehungsweise dem Film. Umgekehrt kann mit ihrer Hilfe aus der Position eines Punktes in einem Bild, ähnlich einer Messung mit einem Theodolit, die Richtung des Punktes in Relation zu der Kamera bestimmt werden. Die Abbildungsgleichung ist abhängig von der inneren und äußeren Orientierung der Kamera. Die innere Orientierung beschreibt die Abbildungseigenschaften der Kamera, die äußere Orientierung die Position und Ausrichtung der Kamera. Im Folgenden wird zuerst auf die innere und äußere Orientierung eingegangen, bevor die Abbildungsgleichung selbst vorgestellt wird.

2.1.1. Innere Orientierung

Die innere Orientierung beschreibt die Abbildungseigenschaften der Kamera auf mathematische Weise. Parameter sind hierbei die Lage des Bildhauptpunktes, die Kamerakonstante sowie die Parameter, die die Verzeichnung beschreiben. (vgl. Luhmann, 2023, S. 179f)

Normalerweise wird die Kamera hierbei vereinfacht als Lochkamera unter Verwendung der Zentralperspektive betrachtet (siehe Abbildung 2.2). Als Projektionszentrum O' wird hierbei der Punkt beschrieben, durch den alle Bildstrahlen geradlinig laufen. Sein Lot auf dem Sensor wird als Bildhauptpunkt H' bezeichnet und beispielsweise durch seine Lage in Pixeln oder Mikrometern beschrieben. Durch eine schiefe optische Achse kann dieser vom Mittelpunkt des Sensors abweichen. Die Kamerakonstante c beschreibt idealisiert den Abstand des Projektionszentrums zum Sensor. (vgl. Luhmann, 2023, S. 177)

Als Verzeichnung wird die Abweichung der Abbildung von der idealen Lochkamera bezeichnet. Diese kann durch die Nutzung von Linsen oder durch die Bauweise der Kamera entstehen. Die Verzeichnung wird in radialer und tangentialer Verzeichnung unterschieden. Die radiale Verzeichnung beschreibt die Verschiebung der Bildpunkte zur Bildmitte hin. Dieser Punkt wird Symmetriepunkt genannt und muss nicht zwangsläufig mit dem Bildhauptpunkt zusammenfallen, für die Korrekturen wird dies jedoch üblicher-

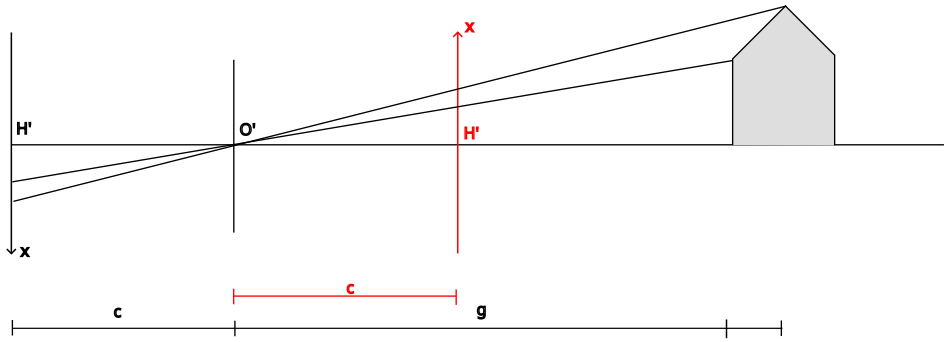


Abbildung 2.2.: Abbildung des Lochkamera-Modells mit Spiegelung am Projektionszentrum (rot), nach Hartley & Zisserman (2004, S. 154), Beschriftung nach Luhmann (2023, S. 177)

weise so angenommen. Die tangentiale Verzeichnung beschreibt die Verschiebung quer zur radialen Richtung. Am wichtigsten ist hierbei die radial-symmetrische Verzeichnung, da diese die größte Auswirkung hat. Sie entsteht durch den nicht symmetrischen Aufbau des Objektivs vor und hinter der Blende. Die radial-asymmetrische und die tangentiale Verzeichnungen entstehen hauptsächlich aus einer Dezentrierung und Schiefstellung der Linsen zur optischen Achse. (vgl. Luhmann, 2023, S. 178)

Jede Einstellung der Kameraoptik verändert die innere Orientierung und jede Kamera, selbst in derselben Modellreihe, unterscheidet sich in ihren Parametern. Änderungen können sich sowohl durch Umfokussierung oder die Nutzung eines optischen Zoom ergeben, als auch durch einen mechanisch instabilen Aufbau der Kameras oder Erwärmung und der damit verbundenen Ausdehnung. Daher sollten die Bilder normalerweise möglichst mit nur einer Kamera mit festen Einstellungen (Brennweite, Fokus, Blende, Objektiv) aufgenommen werden. Änderungen der Empfindlichkeit (ISO-Zahl) oder Belichtungszeit sind unproblematisch für die innere Orientierung, da hierbei die Abbildungseigenschaften der Kamera nicht verändert werden. (vgl. Luhmann, 2023, S. 176)

Die genaue Bestimmung der inneren Orientierung, auch Kalibrierung genannt, kann während der Messung beispielsweise als Parameter in der Bündelblockausgleichung erfolgen. Je nach Anzahl der Kameras und Stabilität der inneren Orientierung sind hier verschiedene Varianten möglich. Bei stabilen Kameras wird ein Parameter-Satz pro Kamera für eine ganze Messreihe ausgeglichen, bei instabilen Kameras kann die Kalibrierung jedes einzelnen Bildes notwendig werden. (vgl. Luhmann, 2023, S. 181f)

2.1.2. Äußere Orientierung

Die äußere Orientierung beschreibt die Lage der Kamera im Raum und ihre Ausrichtung. Sie wird durch die Position des Projektionszentrums und die Richtung der optischen Achse beschrieben. Die Position wird hierbei durch 3D-Koordinaten beschrieben, in der Nahbereichsphotogrammetrie oft in einem lokalen System. Die Richtung der optischen Achse wird mit der Rotation der Kamera zu diesem Koordinatensystem beschrieben, beispielsweise durch eine Rotationsmatrix, eulersche Winkel oder Quaternionen. Durch die Nutzung von Passpunkten oder bekannten Koordinaten der Kamera kann die äußere Orientierung bestimmt werden. (vgl. Luhmann, 2023, S. 273ff)

Die Darstellung der Rotation der Kamera als Quaternionen hat den Vorteil, dass diese dann mit nur vier Parametern in eine Bündelblockausgleichung eingeht. Bei der Nutzung der Rotationsmatrix hätte man hier neun bzw. acht unabhängige Parameter. Die Nutzung von eulerschen Winkeln hat den Nachteil, dass hier ein sogenannter Gimbal-Lock auftreten kann, bei dem zwei Achsen zusammenfallen und die Rotation nicht mehr eindeutig bestimmt werden kann. (vgl. Luhmann, 2023, S. 63)

2.1.3. Abbildungsgleichung

Die Abbildung eines Punktes auf einem Bild wird durch die Abbildungsgleichung beschrieben. Hierfür gibt es verschiedene Schreibweisen. In der Matrizenrechnung besteht diese aus der Multiplikation der homogenen Punktkoordinaten X mit der Projektionsmatrix P (Gleichung 2.1). P ergibt sich aus der Kalibrier- oder Kameramatrix K, der Rotation R und dem Vektor zum Projektionszentrum X_0 (Gleichung 2.2). Die Kalibriermatrix wiederum besteht aus der Kamerakonstante und der Lage des Bildhauptpunktes (Gleichung 2.3, nach Hartley & Zisserman, 2004, S. 244 und Luhmann, 2023, S. 290).

Die äußere Orientierung ist durch die Matrix $[R|X_0]$ beschrieben (Gleichung 2.4). Diese besteht aus der Rotationsmatrix R und dem Vektor zum Projektionszentrum X_0. Die Rotationsmatrix beschreibt die Ausrichtung der Kamera im Raum, das Projektionszentrum beschreibt die Position der Kamera bzw. des Projektionszentrums im Raum.

$$x' = P \cdot X \tag{2.1}$$

$$P = K \cdot [R|X_0] \tag{2.2}$$

$$K = \begin{bmatrix} c_x & 0 & x'_0 \\ 0 & c_y & y'_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

$$[R|X_0] = \begin{bmatrix} r_{11} & r_{21} & r_{31} & X_0 \\ r_{12} & r_{22} & r_{32} & Y_0 \\ r_{13} & r_{23} & r_{33} & Z_0 \end{bmatrix} \tag{2.4}$$

mit x' : Bildkoordinate

P : Projektionsmatrix

X : Homogene Punktkoordinaten

K : Kalibrier- oder Kameramatrix

R : Rotationsmatrix

X_0 : Vektor zum Projektionszentrum

c_x, c_y : Kamerakonstante in x- und y-Richtung

x'_0, y'_0 : Lage des Bildhauptpunktes in x- und y-Richtung

r_{mn} : Parameter der Rotationsmatrix

Grundlage dieser Formel ist das Lochkamera-Modell (siehe Abbildung 2.2). Dabei wird rechnerisch das Bild an dem Projektionszentrum in den Objektraum gespiegelt (rot im Bild). Dadurch kann mit einem aufrechten Bild gerechnet werden.

Darüber hinaus spielen die Verzeichnungsparameter. Diese können in die Abbildungsgleichung integriert werden, in dem x' durch $\Delta x'$ korrigiert wird. (vgl. Luhmann, 2023, S. 277)

$$x' = P \cdot X + \Delta x' \tag{2.5}$$
$$\Delta x' = \Delta x_{rad} + \Delta x_{tan} + ...$$

mit $\Delta x'$: Verzeichnung

Δx_{rad} : radial-symmetrische Verzeichnung

Δx_{tan} : tangentiale Verzeichnung

Die Verzeichnung wird meist mit den Parametern k_1 bis k_4 für die radial-symmetrische Verzeichnung und p_1 und p_2 für die tangentiale Verzeichnung bezeichnet. Die Faktoren stellen dabei die Parameter einer Taylor-Reihe dar. Sie basieren auf dem Kameramodell von Brown (1971, S. 859) und denen von ihm vorgestellten Formeln (Gleichung 2.6 und 2.7).

$$\Delta x_{rad} = k_1\, r^2 + k_2\, r^4 + k_3\, r^6 + k_4\, r^8 \tag{2.6}$$
$$\Delta y_{rad} = k_1\, r^2 + k_2\, r^4 + k_3\, r^6 + k_4\, r^8$$
$$\Delta x_{tan} = p_1\,(r^2 + 2\,\bar{x}^2) + 2\,p_2\,\bar{x}\,\bar{y} \tag{2.7}$$
$$\Delta y_{tan} = 2\,p_1\,\bar{x}\,\bar{y} + p_2\,(r^2 + 2\,\bar{y}^2)$$
$$r = \sqrt{\bar{x}^2 + \bar{y}^2}$$
$$\bar{x} = x' - x'_0$$
$$\bar{y} = y' - y'_0$$

mit r : Abstand des Punktes zum Bildhauptpunkt

x', y' : Bildkoordinaten

$\bar{x}, \bar{y}$: Abstand von Bildhauptpunkt in x- bzw. y-Richtung

k_1, k_2, k_3, k_4 : radial-symmetrische Verzeichnungsparameter

p_1, p_2 : tangentiale Verzeichnungsparameter

Diese Parameter werden so beispielsweise auch in Photogrammetrie- und Computer-Vision-Software wie Agisoft Metashape, OpenDroneMap und OpenCV genutzt.

2.2. Bilder

Die Berechnung der Position eines Objektes im Bild ist nur möglich, sofern der Punkt in mindestens einem weiteren Bild abgebildet ist. Die Genauigkeit der Berechnung ist vom Schnittwinkel dieser beiden Strahlen abhängig. Um möglichst gute Grundlagen zur Verfügung zu haben und die innere und äußere Orientierung gut berechnen zu können, müssen diese Bilder einige Bedingungen erfüllen - auf diese wird hier eingegangen.

2.2.1. Überlappung und Bildinhalte

Da die Bilder durch identische Punkte verbunden werden, müssen sich die Bildinhalte überlappen und gemeinsame Punkte in den Bildern identifiziert werden. Die automatische Detektion von identischen Punkten ist auf verschiedene Weisen möglich: Entweder durch die Nutzung von codierten und uncodierten Passpunkten oder über eine Merkmalsextraktion. Für letzteres muss die Oberfläche genügend Textur bzw. Struktur aufweisen. Detailliert wird in Abschnitt 2.4 hierauf eingegangen. (vgl. Luhmann, 2023, S. 478)

2.2.2. Position und Ausrichtung der Kamera

Damit die Schnitte der Bildstrahlen optimal und die Tiefeninformationen möglichst genau sind, müssen die Kameras gleichmäßig um das Objekt verteilt sein. Bilder, die vom gleichen Standpunkt aufgenommen wurden, sind oft nur ungenau verknüpfbar. Daher empfiehlt es sich, Bilder aus verschiedensten Richtungen zu machen, also bei der manuellen Photographie um das Objekt herumzugehen und eine Mehrbildaufnahme im Rundum-Verband zu erzeugen. (vgl. Luhmann, 2023, S. 170)

Entsprechend müssen die Kameras gleichmäßig um das Objekt positioniert werden und dabei auch an die Form des Objektes wie Einschnitte anpassbar sein.

2.2.3. Belichtung

Um identische Punkte in den Bildern identifizieren zu können, müssen die Bilder eine gleichmäßige Beleuchtung aufweisen. Hierfür sollte es vermieden werden, dass Schatten auf das Objekt fallen. Eine gleichmäßige Beleuchtung kann durch die Nutzung von mehreren Lichtquellen erreicht werden. Auch sollte keine Blendwirkung entstehen, die durch direkte Sonneneinstrahlung oder Reflexionen entstehen kann. Die Belichtung der einzelnen Bilder sollte möglichst identisch sein, damit später auch eine zusammenhän-

gende Texturierung möglich ist. Da die Objekte und die Kameras sich während der Aufnahme nicht bewegen, kann die Belichtungszeit verlängert werden, um Rauschen durch eine zu hohe Sensorempfindlichkeit zu vermeiden und so die Bildqualität zu erhöhen.

2.2.4. Fokussierung und Schärfentiefe

Um genaue Punktwolken erzeugen zu können, muss das Objekt scharf abgebildet werden. Die Kameras müssen dafür entsprechend fokussiert sein. Jedoch wird durch die Fokussierung auch die innere Orientierung verändert. Normalerweise wird daher auf eine Umfokussierung verzichtet und mittels der Blende (kleine Blendenöffnung) die Schärfentiefe erhöht, sodass ein großer Bereich scharf abgebildet wird. Dieses ist aber bei vielen einfachen Kameras wie dem verwendeten Raspberry Pi Camera Module 3 nicht möglich.

Der Schärfebereich ergibt sich aus der Größe des tolerierbaren Unschärfekreises auf dem Sensor. Bei digitalen Kameras wird dieser durch die Pixelgröße bestimmt. Nach Luhmann (2023, S. 148f) berechnet sich die Schärfentiefe nach folgender Formel:

$$
\begin{aligned}
t &= a_h - a_v \\
a_v &= \frac{a}{1+K} \\
a_h &= \frac{a}{1-K} \\
K &= \frac{k(a-f)u'}{f^2}
\end{aligned}
\tag{2.8}
$$

mit t : Länge des Schärfebereichs

a_v, a_h : vordere bzw. hintere Grenze der Schärfentiefe

a : fokussierte Dingweite

k : Blendenzahl

f : Brennweite

Die Schärfentiefe des Raspberry Pi Camera Module 3 ist in Abbildung 2.3 für einen Unschärfekreis von drei Pixeln dargestellt (Plot der Gleichung 2.8). Es zeigt sich, dass die Schärfentiefe im verwendeten Makrobereich relativ klein ist. Problematisch ist dies vor allem bei größeren Objekten, die hierdurch näher an die Kameras rücken, aber durch ihre Größe eine größere Schärfentiefe benötigen. Ein Lösungsansatz durch Kombination mehrerer Aufnahmen wird in Abschnitt 4.4 untersucht.

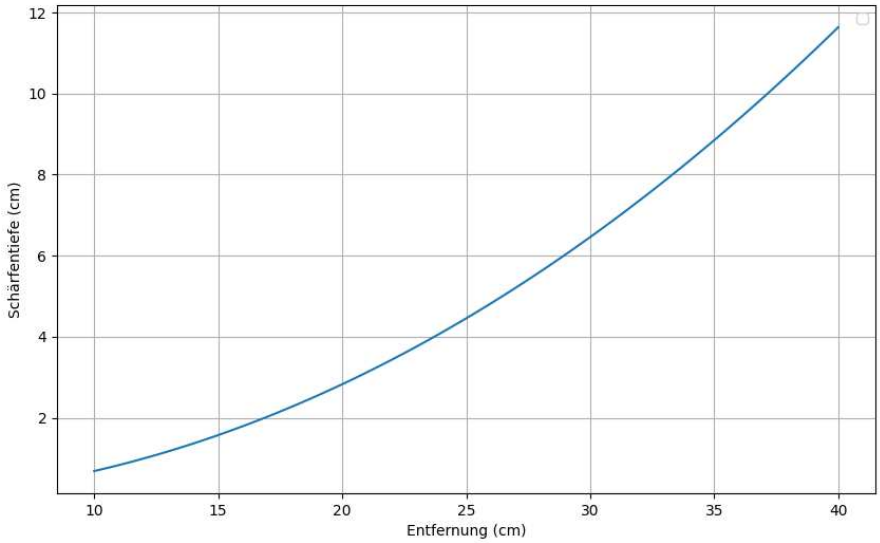


Abbildung 2.3.: Schärfentiefe in Abhängigkeit von der fokussierten Entfernung

2.3. Skalierung/Maßstab

Die Skalierung eines rein photogrammetrisch bestimmten 3D-Modells ist nicht bekannt, da die Berechnungen nur auf Richtungen ohne Längenangaben basieren - es ist im mathematischen Sinne ähnlich dem realen Körper. Daher werden Referenzen in Form einer bekannten Länge benötigt, um die Skalierung zu bestimmen. Alternativ können auch Passpunkte (siehe Abschnitt 2.4) mit bekannten Koordinaten verwendet werden oder im Falle von Mehrkamerasystemen bekannte äußere Orientierungen der Kameras. (vgl. Luhmann, 2023, S. 546)

2.4. Verknüpfungs- und Passpunkte

Um die einzelnen Bilder verknüpfen zu können, werden identische Punkte zwischen zwei oder mehr Bildern benötigt. Diese können klassisch manuell erfasst werden, jedoch ist dies schon bei kleineren Projekten sehr zeitaufwändig. Daher wird meist die Möglichkeit genutzt, automatisch Verknüpfungspunkte zu erzeugen. Hierfür gibt es unterschiedliche Methoden, die im Folgenden vorgestellt werden.

2.4.1. Zielmarker

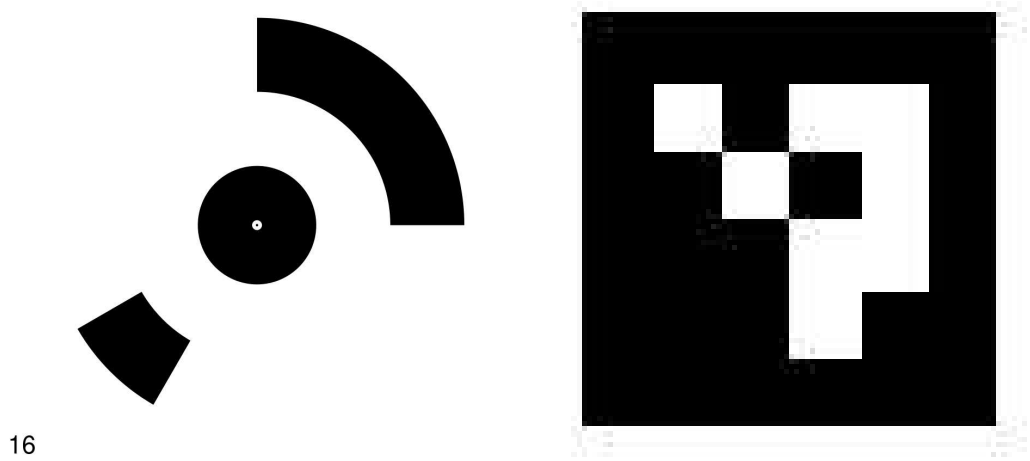
Es gibt verschiedenste Formen von Markern, die automatisch erfasst werden können. Grob unterschieden werden kann in codierte und nicht codierte Zielmarker. Beispiele für nicht codierte sind einfache kreisförmige Klebepunkte oder Marker, die aus Linien bestehen und ihren Mittelpunkt durch dessen Schnitt definieren. Vorteilhaft ist jedoch die Verwendung von codierten Zielmarken. Hier können die Punkte direkt zugeordnet werden, sodass keine weitere Filterung und Berechnung zur Zuordnung notwendig ist. (vgl. Luhmann, 2023, S. 535ff)

Zielmarken nach Schneider

Eine Form der codierten Zielmarken sind die Marker nach Schneider. Diese bestehen aus mehreren konzentrischen Kreisen (siehe Abbildung 2.4a) und werden dementsprechend als Concentric circular coded target (CCCT) bezeichnet (vgl. Liu et al., 2021). Die Mitte des Markers wird durch den gemeinsamen Mittelpunkt der Kreise definiert (vgl. Schneider & Sinnreich, 1992). Vorteilhaft ist, dass die Marker auch bei unscharfen Bildern erkannt werden können und ihr Zentrum manuell identifiziert werden kann, beispielsweise zum Aufmaß mit einem Tachymeter. Sie haben sich daher als Standardmarker in der Photogrammetrie etabliert und werden zum Beispiel von Agisoft Metashape verwendet.

ArUco-Marker

Eine andere Variante der codierten Verknüpfungspunkte sind die sogenannten ArUco-Marker (siehe Abbildung 2.4b). Diese werden häufig für die Orientierung bei Augmented-Reality-Anwendungen genutzt. Jede Ecke kann hierbei automatisch im Subpixelbereich identifiziert werden, sodass ein erkannter Marker vier Verknüpfungspunkte liefern kann. Wenn deren (lokale) Koordinaten und die innere Orientierung bekannt sind, kann mit nur einem Marker die äußere Orientierung bestimmt werden. (vgl. Luhmann, 2023, S. 545)



(a) CCCT nach Schneider & Sinnreich (1992), aus Agisoft Metashape

(b) ArUco-Marker, generiert mit OpenCV (2023)

Abbildung 2.4.: Codierte Passpunkte

Nachteilig zeigte sich in den Untersuchungen des Fokus (siehe Abschnitt 4.2), dass die Ecken nicht eindeutig identifiziert werden können, wenn die Aufnahme unscharf ist. Durch ihre Verwendung in Augmented-Reality-Anwendungen sind viele kostenfreie Bibliotheken für ihre Erkennung verfügbar, sodass beispielsweise OpenCV (2023) ArUco-Marker identifizieren kann.

2.4.2. Merkmalsextraktion

Ohne das Anbringen von Markern können Verknüpfungspunkte erzeugt werden. Hierfür wird Merkmalsextraktion verwendet, beispielsweise Scale-Invariant Feature Transform (SIFT), welches hier stellvertretend kurz vorgestellt wird. Es liefert Verknüpfungspunkte aus Mustern auf den photographierten Oberflächen. Es ist meist nicht notwendig, explizit Marker an dem aufzunehmenden Objekt anzubringen, sofern seine Oberfläche nicht strukturlos ist (glatte weiße Wände etc.).

Zur Erkennung von Merkmalen setzt SIFT auf die Detektion von Kanten. Diese werden in verschiedenen Stufen einer Bildpyramide erkannt und ihre Extrema berechnet. Sie werden anschließend weiter ausgedünnt, beispielsweise über den Kontrast. Sofern ein möglicher Marker identifiziert wurde, wird eine Beschreibung erzeugt. Diese erfolgt durch Analyse der Helligkeitsabweichungen zu den Nachbar-Pixeln und wird an der stärksten

Abweichung ausgerichtet. Hierdurch wird die Beschreibung dann richtungsunabhängig. Mit den Beschreibungen kann dann die Übereinstimmung von zwei Markern in zwei Bildern identifiziert werden, auch wenn diese zueinander gekippt oder gedreht sind. (vgl. Luhmann, 2023, S. 484f)

2.5. Verknüpfung von Bildern

Durch die beschriebenen Verfahren und die hieraus entstandenen Verknüpfungspunkte können die Bilder miteinander verknüpft werden. Da die Kamerapositionen am Rahmen veränderlich sind und die Montage auch keine ausreichend genaue Fixierung garantiert, können die bekannten Positionen aus vorherigen Messungen maximal als Näherungswerte genutzt werden. Die genaue Bestimmung der Position und Ausrichtung - die äußere Orientierung - muss daher mindestens für die verschobenen oder gedrehten Kameras neu berechnet werden. Gleiches gilt für neue Verknüpfungspunkte, für die noch keine Koordinate bekannt ist. Die verwendeten Methoden, die zur Verknüpfung der Bilder beitragen, werden im Folgenden vorgestellt: der Rückwärts- und Vorwärtsschnitt sowie die relative Orientierung.

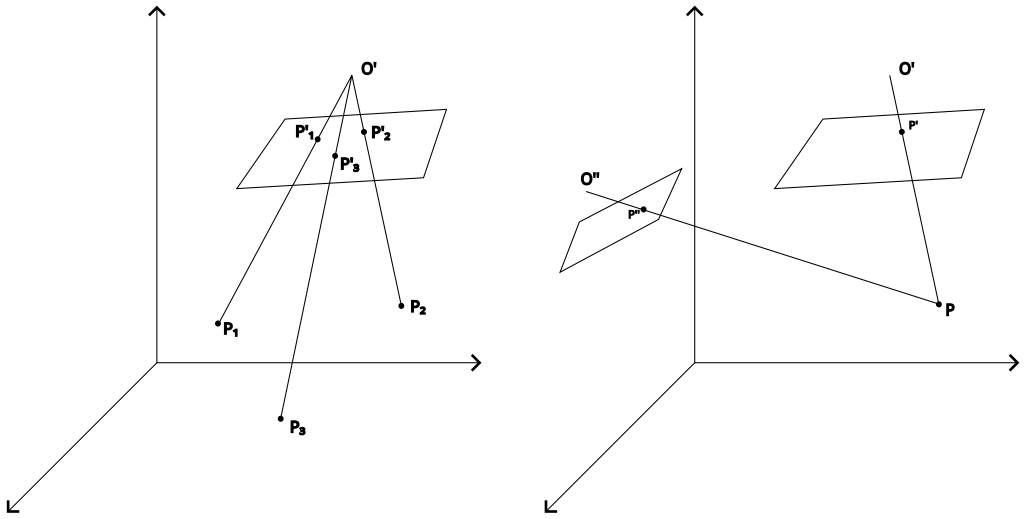
2.5.1. Rückwärtsschnitt

Sofern Koordinaten von Passpunkten bekannt sind, können die Positionen und Ausrichtungen der Kameras berechnet werden. Hierfür wird der sogenannte Rückwärtsschnitt genutzt. Die Berechnung erfolgt auf Basis der Abbildungsgleichung, die die Position eines Punktes in einem Bild in Beziehung zur Kamera setzt. Für die Berechnung selbst gibt es verschiedene Methoden. Verwendet wurde hier der von OpenCV genutzte Ansatz von Lepetit et al. (2008) in Kombination mit einem RANSAC-Ansatz.

Durch die Nutzung des RANSAC-Ansatzes können veränderte Passpunkte identifiziert und als Ausreißer markiert werden. Hierzu werden mehrere Stichproben der Punkte gewählt, die Berechnung durchgeführt und die Lösung gewählt, zu denen die meisten Punkte passen. (vgl. Luhmann, 2023, S. 134)

Abbildung 2.5a zeigt die geometrischen Grundlagen des Rückwärtsschnittes. Es müssen mindestens 3 Passpunkte ($P_1 - -P_3$) mit Koordinaten bekannt und im Bild sichtbar sein. Außerdem muss die innere Orientierung bei Verwendung von nur drei Passpunkten bekannt sein. Die Position der Punkte $P'_1 - -P'_3$ wird dann bestimmt und

ergibt im Beispiel 6 Koordinatenkomponenten (jeweils x' und y'). Durch Berechnungen – auf die hier nicht weiter eingegangen und auf entsprechende Literatur verwiesen wird – kann dann die Position (drei Parameter) und Ausrichtung (drei Parameter) der Kamera bestimmt werden. (vgl. Luhmann, 2023, S. 284)



(a) Rückwärtsschnitt, nach Luhmann 2023, S. 284

(b) Vorwärtsschnitt, nach Luhmann 2023, S. 339

Abbildung 2.5.: Geometrische Grundlagen des Rückwärts- und Vorwärtsschnittes

2.5.2. Vorwärtsschnitt

Um wiederum aus einem Stereopaar mit bekannter innerer und äußerer Orientierung Punktkoordinaten zu berechnen, wird der Vorwärtsschnitt genutzt. Abbildung 2.5b zeigt die geometrischen Grundlagen des Vorwärtsschnittes. Es werden die Bildkoordinaten der Punkte P_1 und P_2 benötigt, die im Bild sichtbar sind. Die Position des Punktes P wird dann berechnet. Wie bereits in Abschnitt 2.1 erwähnt, kann jeder Punkt im Bild durch die innere Orientierung als Horizontal- und Vertikalrichtungsmessung interpretiert werden. Durch die äußere Orientierung kann diese ins übergeordnete System überführt werden. Hieraus ergibt sich dann zur Berechnung der Position P ein räumlicher Schnitt zweier (windschiefer) Geraden (vgl. Luhmann, 2023, S. 95).

Mit dem Vorwärtsschnitt können neben markierten Verknüpfungspunkten auch die Neupunkte, die mittels SIFT oder ähnlicher Bilderkennungsalgorithmen erkannt wurden, berechnet werden. Dadurch kann eine dünne Punktwolke erzeugt werden, welche dann bei der Bündelblockausgleichung (siehe Abschnitt 2.6) genutzt werden kann. Da die Berechnung bereits bei Bildkoordinaten aus zwei Bildern überbestimmt ist und oft Punkte in mehr Bildern abgebildet sind, bietet sich die Nutzung einer Ausgleichung auch schon bei der Bestimmung an. (vgl. Luhmann, 2023, S. 385)

2.5.3. Relative Orientierung

Wenn keine Passpunkte vorhanden sind und durch zum Beispiel ArUco-Marker keine lokalen Passpunkt-Koordinaten erzeugt werden können, kann die Bildorientierung auch in einem lokalen System erfolgen. Hierbei wird die relative Orientierung der Bilder zueinander bestimmt. Eines der Bilder wird als Ursprung des Systems festgelegt (Projektionszentrum in (0,0,0) und die optische Achse in z-Richtung). Die Position und Ausrichtung des zweiten Bildes wird dann relativ zu diesem bestimmt. (vgl. Luhmann, 2023, S. 316)

Die klassische Variante der relativen Orientierung ist die Bestimmung der Essenziellen Matrix, welche identische Punkte in einem Bild mit denen in einem anderen Bild in Beziehung setzt. Diese kann aus der inneren Orientierung und der Fundamentalmatrix berechnet werden, welche wiederum aus den Bildkoordinaten von mindestens 8 Verknüpfungspunkten berechnet werden kann. (vgl. Luhmann, 2023, S. 328ff)

Mittels Einzelwertzerlegung kann dann aus der Essenziellen Matrix die Rotation und Translation des zweiten Bildes bestimmt werden. (vgl. Hartley & Zisserman, 2004, S. 275)

Alternativ kann mit der Homographie die relative Orientierung über eine Ebene, die in beiden Bildern abgebildet wird, bestimmt werden. Dieser Fall tritt beispielsweise bei der Verwendung von flächenhaften Kalibriermustern auf. Die Homographie beschreibt die Abbildung einer Ebene auf eine andere Ebene. Sie wird durch eine 3x3-Matrix beschrieben, die die Transformation der Punkte beschreibt. Für die Bestimmung werden nur 4 Verknüpfungspunkte benötigt, die in beiden Bildern sichtbar sind und auf einer Ebene liegen. Die innere Orientierung der Kamera wird hierbei nicht benötigt. (vgl. Hartley & Zisserman, 2004, S. 33ff).

Aus der Homographie kann zusammen mit der inneren Orientierung die Rotation und Translation des zweiten Bildes bestimmt werden. (vgl. Malis & Vargas, 2007, S. 74)

2.6. Bündelblockausgleichung

Mittels Bündelblockausgleichung können die grob mit den vorher genannten Verfahren bestimmten Positionen und Drehungen in einer Ausgleichung optimiert werden. Hierzu gehen alle Parameter der Bilder und die Positionen der Passpunkte in die gemeinsame Ausgleichung ein. Grundlage der Ausgleichung ist die in Unterabschnitt 2.1.3 beschriebene Abbildungsgleichung. Als Ergebnis erhält man die ausgeglichenen Parameter und Genauigkeitsangaben für diese. (vgl. Luhmann, 2023, S. 343ff)

Neben der Optimierung der Parameter hat die Bündelblockausgleichung auch den Vorteil, dass instabile Kameras, wie sie in diesem Projekt verwendet werden, simultan kalibriert werden können. Hierbei werden die inneren und äußeren Orientierungen der Kameras gemeinsam optimiert. Die in vorherigen Untersuchungen festgestellten, jedoch nicht genauen Parameter der Kameras werden hierbei als Näherungswerte genutzt und die endgültigen Parameter erst durch die Ausgleichung bestimmt. Es ist daher unkritisch, wenn sich die Parameter der Kameras im geringen Rahmen zwischen den Aufnahmen verändern. (vgl. Luhmann, 2023, S. 357f)

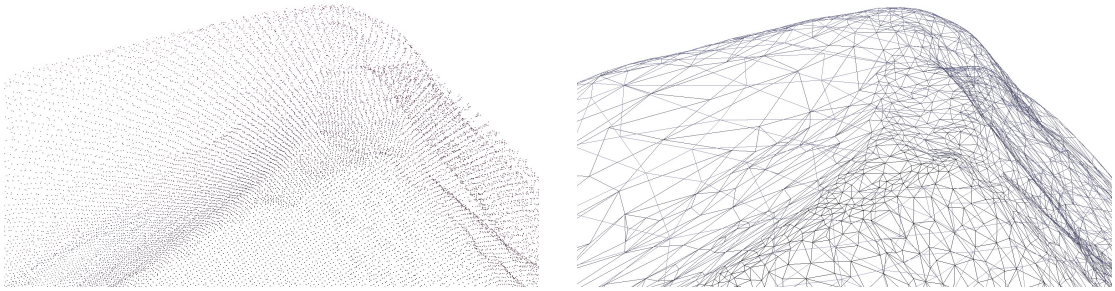
2.7. Multi-View-Stereo

Die dünne Punktwolke aus dem Vorwärtsschnitt und der anschließenden Bündelblockausgleichung kann durch das Multi-View-Stereo-Verfahren zu einem 3D-Modell erweitert werden. Hierbei werden jeweils Bildpaare gebildet und die Disparitäten, also die Verschiebung des Objektes in der Abbildung, bestimmt. Diese sind abhängig von der Entfernung des Objektes. Bei einem unendlich weit entfernten Objekt tritt keine Disparität auf (vgl. Luhmann, 2023, S. 313). Die Disparitäten können dann in Tiefeninformationen umgerechnet, gemittelt und zu einem Tiefenbild zusammengefasst werden. (vgl. Luhmann, 2023, S. 505)

2.8. Mesh-Generierung

Bis zu diesem Schritt besteht das Modell nur aus einzelnen Punkten, die keine Oberfläche ergeben. Um ein geschlossenes 3D-Modell zu erhalten, muss eine Oberfläche generiert werden. Hierfür wird ein Mesh-Generierungsverfahren genutzt, welches die Punktwolke in Dreiecke unterteilt. Es gibt verschiedene Verfahren, die sich in der Art der Unterteilung und ihrem Umgang mit Einzelpunkten unterscheiden. Meistens wird hierbei die Punktwolke auch gefiltert, um Ausreißer und nicht notwendige Punkte zu entfernen. OpenDroneMap nutzt beispielsweise die Screened Poisson Surface Reconstruction zur Vermaschung und Filterung. (vgl. Toffanin, 2019, S. 52f)

Abbildung 2.6 zeigt beispielhaft eine Ecke aus einer Punktwolke vor und nach der Vermaschung. Gut zu erkennen ist, dass die Anzahl der Punkte reduziert wurde und die Oberfläche durch Dreiecke angenähert wurde, jedoch sich die äußeren Konturen nicht verändert haben.



(a) Ungefilterte Punktwolke (b) 3D-Mesh mit gefilterten Punkten

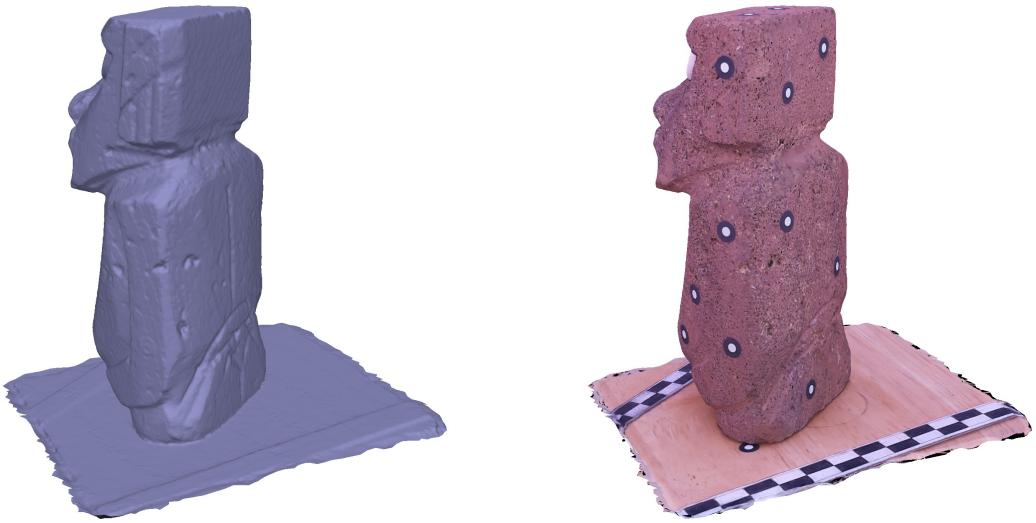
Abbildung 2.6.: Vermaschung der Punktwolke

Die Screened Poisson Surface Reconstruction ist ein Verfahren, das auf der Poisson-Gleichung basiert. Diese Gleichung beschreibt die Beziehung zwischen einer Funktion und ihrer Ableitung. Bei der Screened Poisson Surface Reconstruction wird diese verwendet, um die Oberfläche zu glätten und zu interpolieren. Dabei werden die Punkte in eine Gitterstruktur überführt und die Oberfläche durch die Lösung der Poisson-Gleichung bestimmt. Dieser Ansatz berücksichtigt auch die Ausrichtung der Punkte, um eine genauere Rekonstruktion der Oberfläche zu erzielen. (vgl. Kazhdan et al., 2006)

2.9. Texturierung

Abschließend wird das Modell texturiert. Hierfür werden die Bilder, die zur Erstellung des Modells genutzt wurden, auf das Modell projiziert. Außerdem werden Helligkeits- und Farbunterschiede der Bilder ausgeglichen. (vgl. Toffanin, 2019, S. 54f)

Vorteil der Texturierung ist, dass das Modell meist realistischer wirkt und fehlende Informationen, beispielsweise fehlende Punkte, überdeckt werden können. Nachteilig ist, dass die Texturierung hierdurch auch Informationen verdeckt. So kann bei einem nur schattierten Modell die Struktur des 3D-Modells deutlich besser erkannt werden (vgl. Luhmann, 2023, S. 702). Beispielhaft ist dieses in Abbildung 2.7 an dem später auch für die Überprüfungen (siehe Kapitel 7) verwendeten Testkörper dargestellt. Dieser hat auf der Rückseite (rechts im Bild) einige Einkerbungen, welche im texturierten Modell (siehe Abbildung 2.7b) kaum erkennbar sind. Im ausschließlich schattierten Modell (siehe Abbildung 2.7a) sind diese hingegen deutlich sichtbar.



(a) 3D-Modell ohne Textur (b) 3D-Modell mit Textur

Abbildung 2.7.: 3D-Modell ohne und mit Textur

3. Aufbau des Messsystems

Entsprechend der im Kapitel 2 beschriebenen Anforderungen an ein photogrammetrisches Messsystem wurden die Komponenten ausgewählt.

Die Kameras sollten eine hohe geometrische Auflösung und möglichst stabile innere Orientierung aufweisen. Außerdem sollen sie während einer Messkampagne nicht in ihrer Lage zueinander verändert werden, damit die äußere Orientierung größtenteils gleich bleibt. Daher ist ein stabiler Rahmen notwendig, an welchem die Kameras verdrehsicher angebracht werden können. Kleinere Restfehler in den Orientierungen können mit der Bündelblockausgleichung ausgeglichen werden. Um Ungenauigkeiten durch Bewegungen zu verhindern, müssen die Kameras möglichst zeitgleich auslösen. Daher ist eine gemeinsame Steuerung und Kommunikation zwischen den Kameras notwendig. Außerdem sollen alle Bilder auf das Steuerungssystem übertragen werden, hierfür wird eine Form der Datenübertragung benötigt. Damit die Bilder möglichst schattenfrei ausgeleuchtet werden, sollte Beleuchtung mit eingeplant werden. Außerdem muss die Energieversorgung der einzelnen Kameras sichergestellt sein.

In diesem Kapitel wird der Aufbau des Messsystems beschrieben und die Auswahl der Komponenten begründet.

3.1. Kameras

Als Kameras wurde das Raspberry Pi Camera Module 3 verwendet, welches jeweils von einem Raspberry Pi Zero W gesteuert wird. Im Vergleich zu anderen günstigen Kameras wie Webcams oder der ESP32 CAM haben die Kameras eine hohe geometrische Auflösung von 12 Megapixeln und dennoch mit 1,4 µm relativ große Pixel (px) (Raspberry Pi Foundation, 2023), was im subjektiven Eindruck eine sehr gute Bildqualität ergibt.

Andere Kameramodule für den Raspberry Pi wurden in Tabelle 3.1 verglichen. Das Camera Module v1 entfiel als Möglichkeit, da die Kamera einen Mindestabstand von einem Meter benötigt. Hiermit müsste der Aufnahmebereich für die Objekte auf über zwei Meter vergrößert werden. Die HQ- und GS-Kameras haben kein Objektiv mitgeliefert, sodass hier die Kosten für ein Objektiv hinzukommen würden. Das Camera Module v2 hat eine geringere Auflösung und auch eine geringere Bildqualität bei

gleichem Preis wie das Module 3. Außerdem ist die Fokussierung nur manuell möglich, was die Automatisierung derselben verhindert. Das Camera Module 3 Wide hat zwar ein größeres Sichtfeld, jedoch damit auch eine geringere Auflösung auf dem Objekt. Vorteil ist der geringe Mindestabstand. Da das Camera Module 3 besser verfügbar und etwas günstiger war, wurde sich für dieses Modell entschieden, was einen guten Kompromiss aus Auflösung, Bildqualität und Preis darstellt.

Tabelle 3.1.: Vergleich der möglichen Kameramodule für den Raspberry Pi (Raspberry Pi Foundation, 2023)

	Preis	**Sensor-auflösung**	**Pixel [µm]**	**Fokus**		**Brennweite [mm]**	**Sichtfeld**	**Blende**
Camera Module v1	$25	2592 × 1944	1,40	fix	1 m - ∞	3,60	54° x 41°	F2.9
Camera Module v2	$25	3280 × 2464	1,12	manuell	10 cm - ∞	3,04	62° x 49°	F2.0
Camera Module 3	$25	4608 x 2592	1,40	motorisiert	10 cm - ∞	4,74	66° x 41°	F1.8
Camera Module 3 Wide	$35	4608 x 2592	1,40	motorisiert	5 cm - ∞	2,75	102° x 67°	F2.2
HQ Camera	$50	4056 x 3040	1,55	manuell				
GS Camera	$50	1456 x 1088	3,45	manuell				

Nachteil und Vorteil zugleich ist, dass die Kamera über einen Autofokus verfügt, der aber auch elektronisch gesteuert manuell fokussieren kann. Dieser verschlechtert die Stabilität der inneren Orientierung und wurde daher im Kapitel 4 analysiert. Da die Bilder im Makrobereich zwischen 0,1 und 1 m aufgenommen werden und die Kameras keine Veränderung der Blende ermöglichen, ist die Schärfentiefe vergleichsweise niedrig. Hierauf wurde in den Untersuchungen in Abschnitt 4.4 genauer eingegangen.

Ein weiterer Vorteil der Lösung mit einzelnen Raspberry-Pi-Computern besteht darin, dass hierdurch bereits die einzelnen Kameraeinheiten parallel zur Datenübertragung Aufgaben wie das Identifizieren von Passpunkten übernehmen können. Durch die Parallelisierung dieses Schrittes ist eine Reduktion der Berechnungszeit zwischen den Aufnahmen zu erwarten. Zudem ermöglicht die Nutzung von leitungslosen Netzwerkverbindungen zur Steuerung eine Skalierung des Systems sowohl hinsichtlich der Anzahl der Kameras als auch der Abstände zwischen den Kameras.

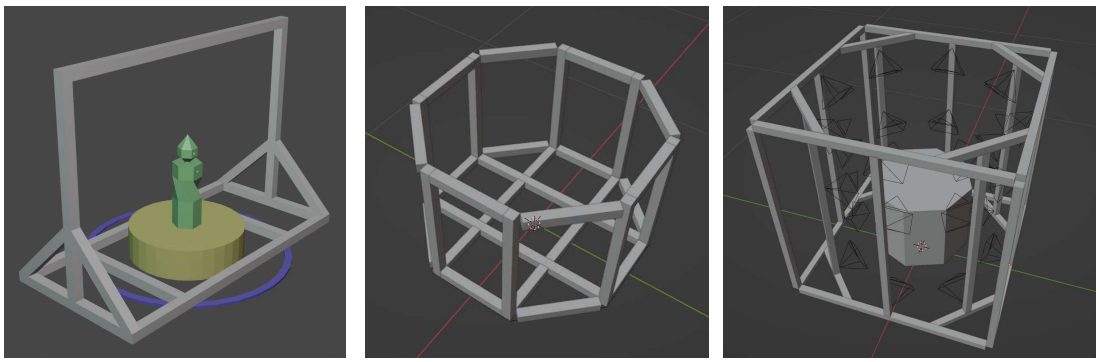
Die Anzahl der Kameras für den Prototyp wurde zusammen mit der Art des Rahmens (siehe Abschnitt 3.2) definiert. Hierzu wurde der Rahmen mit seinen Kameras in der 3D-Visualisierungssoftware Blender modelliert und einzelne Bilder der möglichen Kamera-Positionen gerendert. Im Anschluss wurden die Bilder in Agisoft Metashape zur Berechnung eines 3D-Modells verwendet. Dabei wurde evaluiert, ob die Berechnung möglich ist und wie gut die Abdeckung des Testobjektes ist. Das virtuelle Modell des

Systems wurde darüber hinaus genutzt, um Visualisierungen für die Bedienungsanleitung zu erstellen (siehe Anhang A). Eine Verwendung in der Websoftware zur visuellen Überprüfung der berechneten Kamera-Positionen wurde geprüft, aber nicht umgesetzt, da die Erkennbarkeit schlechter als erwartet und somit der Mehrwert gering war.

3.2. Rahmen

Der Rahmen muss möglichst stabil sein, damit die Kameras sich nicht in ihrer Lage verändern können. Jedoch sollte das System weiterhin transportabel - also nicht zu schwer - und veränderbar bleiben, um beispielsweise Kameras für Messreihen in ihrer Lage zu verändern. Der Aufbau aus genormten Bauteilen bietet sich an, um hier ggf. den Nachbau einfach ermöglichen zu können. Außerdem sollte der Rahmen auch demontierbar sein, damit er transportiert werden kann.

Als mögliche Materialien kamen Holz, Stahl und Aluminium infrage. Aufgrund der einfachen Bearbeitung und der Standardisierung wurde sich für Aluminiumprofile entschieden. Diese gibt es in verschiedenen Ausführungen mit Nuten an den Seitenflächen, sodass eine einfache Montage, aber ebenso eine Demontage zu Transportzwecken, möglich wird. Außerdem sind diese sehr stabil bei leichtem Gewicht.



(a) Unter Nutzung eines Drehtellers (b) Rahmen mit Boden-Versteifung (c) Endgültiger Entwurf mit seitlichen Versteifungen

Abbildung 3.1.: Verschiedene Entwurfsideen, modelliert mit Blender

Es wurden verschiedene Varianten geprüft, die in Abbildung 3.1 dargestellt sind. Der erste Entwurf (siehe Abbildung 3.1a) sah weniger Kameras und die zwingende Nutzung eines Drehtellers vor, dies wurde jedoch verworfen, weil dieser nicht der Anforderung einer schnellen Erfassung entsprochen hätte und auch der Anforderung der kurzen Einarbeitungszeit widersprechen würde. Der zweite Entwurf (siehe Abbildung 3.1b) sah bereits die Nutzung von 8 Kamera-Hauptrichtungen vor, was für gute Schnittwinkel sorgt. Für die Aussteifung wurde hier ein Boden vorgesehen. Dies ist jedoch nachteilig, da das System dann beispielsweise nicht mehr über große, nicht bewegliche Objekte

herüber gestülpt werden könnte. Gerade bei empfindlichen Objekten wäre dies aber vorteilhaft. In der dritten Variante (siehe Abbildung 3.1c) wurde dann die Versteifung in den Seitenwänden umgesetzt. Durch die Konstruktion mit Eckwürfeln sowie der Bildung von dreieckigen Strukturen und dem Einbau von eckaussteifenden Platten mit Scheibenwirkung, wurde die Stabilität der Verbindungen erhöht. Abbildung 3.2b zeigt den fertigen Rahmen vor Einbau der Platten und der Technik.

(a) Kamera-Winkel

(b) Aluminium-Rahmen

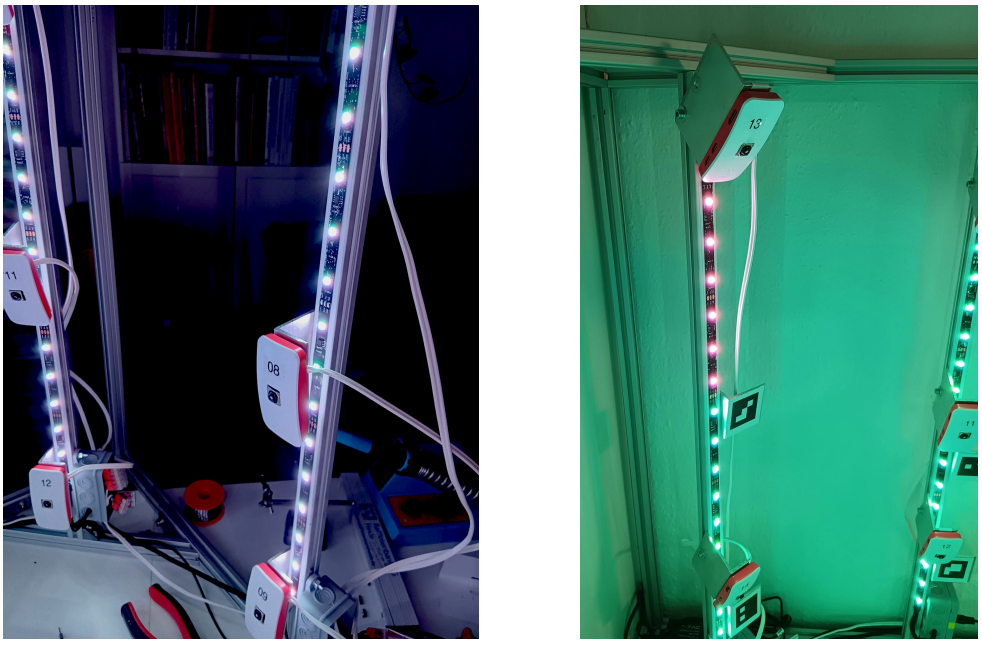
Abbildung 3.2.: Alu-Bauteile des Rahmens

Die Kameras wurden mit einem 90°-Winkel am Rahmen montiert, um diese weiterhin noch vertikal schwenken zu können. Abbildung 3.2a zeigt einen der Winkel, der an den Aluprofilen befestigt wird (noch ohne entsprechende Befestigungsbohrungen).

3.3. Beleuchtung

Um möglichst gute Aufnahmen zu erzeugen, sollte das Objekt ausreichend und gleichmäßig ausgeleuchtet sein. Eine dunkle Umgebung verlängert die Belichtungszeit, wodurch die Gefahr von unscharfen Aufnahmen steigt. Schlecht ausgeleuchtete Bereiche (ungleichmäßige Ausleuchtung) verursachen verstärktes Rauschen in diesen Bildbereichen. Problematisch ist bei der Beleuchtung, dass die Kameras ggf. auch die Lichtquellen mit aufnehmen, wodurch Linsenreflexionen oder ein Ausbrennen der Bildbereiche möglich ist. Außerdem störend sind fremde Lichtquellen, die Schatten werfen oder die Belichtung der Kameras beeinflussen können.

Es wurde sich für einzeln steuerbare LED-Lichtstreifen als Lichtquelle entschieden (siehe Abbildung 3.3a). Diese können einfach an den Aluprofilen montiert werden und ermöglichen es, einzelne Bereiche abzuschalten, beispielsweise um Blendwirkungen zu vermindern. Außerdem können hiermit verschiedene Lichtfarben eingestellt werden, um



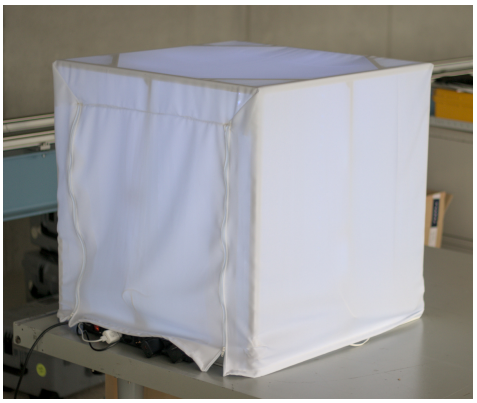
(a) Schattenarme Beleuchtung durch LED-Streifen

(b) Farbige Beleuchtung zur Statusmeldung

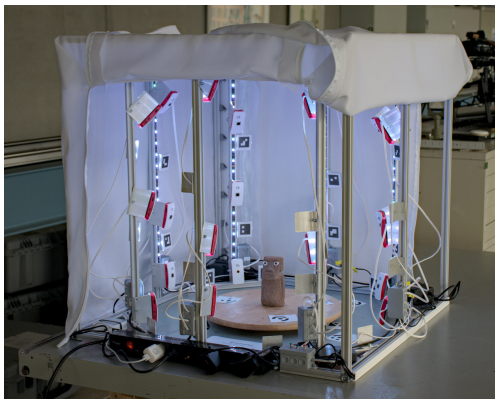
Abbildung 3.3.: Beleuchtung

Statusmeldungen zu ermöglichen (siehe Abbildung 3.3b) oder ggf. die Farbgebung des Objektes zu beeinflussen. Die Steuerung erfolgt über einen Raspberry Pi 4, der auch die Steuerung der Kameras übernimmt. Die LED-Streifen verfügen hierfür pro drei LEDs über einen integrierten Schaltkreis vom Typ WS2811. Dieser ermöglicht es, jedem Dreier-Verbund über ein proprietäres Steuerprotokoll eine eigene Farbe und Helligkeit zuzuweisen (vgl. Worldsemi, 2012). Für die Implementierung der Steuerung wurde die Bibliothek `rpi-ws281x` verwendet, die eine einfache Ansteuerung der LED-Streifen ermöglicht (siehe auch Absatz 5.3.1).

Für diffuseres Licht von außen kann ein halbtransparenter, weißer Stoff über den Rahmen gespannt werden. Hierfür wurde aus weißem Baumwollstoff eine entsprechende Haube genäht (siehe Abbildung 3.4a), welche durch zwei mittels Reißverschluss verschließbaren Eingriffmöglichkeiten weiterhin das Einlegen von Objekten durch die Seite oder das Verstellen von Kameras ermöglicht (siehe Abbildung 3.4b). Diese sorgt für eine gleichmäßigere Ausleuchtung durch Reflexion im Inneren, verhindert Reflexionen an Glasscheiben etc. und vermindert Blendwirkungen durch externe Lichtquellen.



(a) geschlossene Stoffhülle



(b) geöffnete Stoffhülle

Abbildung 3.4.: Stoffhülle zur Verminderung von Reflexionen und Blendwirkungen

3.4. Kommunikation und Datenübertragung

Die Kommunikation zwischen den Raspberry-Pi-Computern erfolgt über WLAN. Hierfür ist ein Mini-WLAN-Router mit im System verbaut worden. Vorteil dieser Lösung ist, dass hier keine weiteren Leitungen außer der Energieversorgung zu den einzelnen Raspberry Pi Zero W benötigt werden und es auch möglich wäre, die gleiche Hard- und Software für ein größeres System ohne Änderungen zu nutzen. Nachteilig ist die Verbindungsgeschwindigkeit, gerade im Hinblick auf die Synchronisierung der Kameras. Diese Problematik soll aber durch entsprechende Programmierung der Software möglichst klein gehalten werden.

Als weitere Datenleitung wird eine Steuerleitung für die LED-Streifen benötigt. Über diese werden die einzelnen LED-Gruppen angesteuert. Hier wird die gleiche Zwillingslitze wie für die Energieverteilung (siehe Abschnitt 3.5) verwendet. Die Leitung ist in Abbildung 3.6 in Abschnitt 3.5 grün dargestellt.

3.5. Energieverteilung

Alle Raspberry-Pi-Computer werden mit 5 V betrieben. Der Raspberry Pi Zero W mit Kamera hatte dabei in Messungen einen maximalen Stromverbrauch von 270 mA aufgezeigt, der Raspberry Pi 4 kann bis zu 1,5 A unter Last verbrauchen. Hieraus ergibt sich ein Gesamtstromverbrauch von maximal rund 8 A. Für den Raspberry Pi 4 wurde ein eigenes 15-Watt-Netzteil eingeplant und für die 24 Raspberry Pi Zero W ein gemeinsames 35-Watt-Netzteil. Versuche zeigten jedoch, dass der Leistungsbedarf

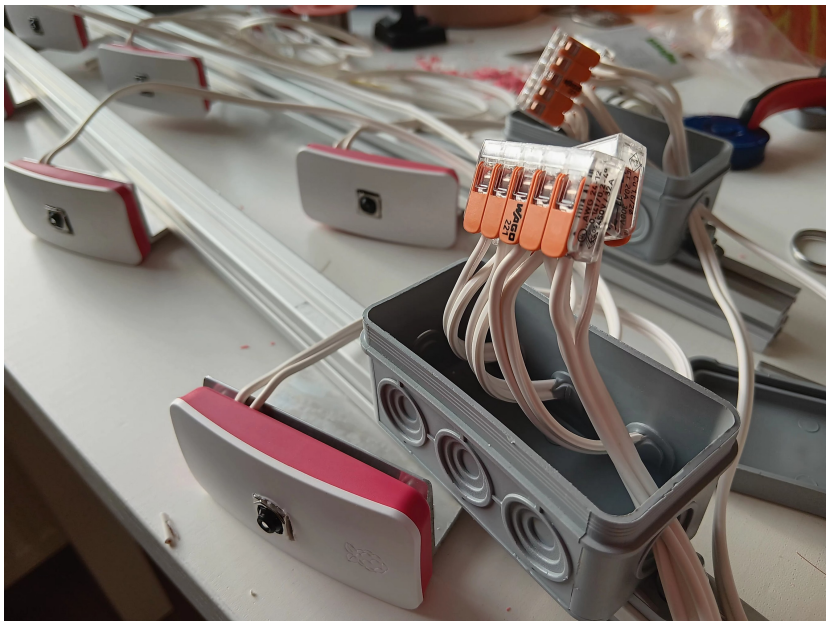


Abbildung 3.5.: Energieverteilung zu den einzelnen Raspberry Pi Zero

kurzfristig höher ausfallen kann, sodass die Raspberry Pi Zero W, die am meisten von Spannungsabfällen betroffen sind, zum Absturz gebracht wurden, wenn alle Kameras gleichzeitig auslösten. Nachdem die Last sicherheitshalber auf zwei weitere Netzteile verteilt wurde, lief das System zuverlässig.

Als Leitungsmaterial wurde Zwillingslitze mit $0{,}75\,\mathrm{mm}^2$ verwendet. Der relativ hohe Leitungsquerschnitt soll für einen geringen Spannungsabfall sorgen. Durch die Verwendung von mehreren Netzteilen ist dieser jedoch nun nicht mehr notwendig. Hier würde sich nun ein geringerer Querschnitt anbieten, auch um eine einfachere Verbindung zu den Raspberry Pi Zero W zu ermöglichen. Diese wurden aufseiten der Zero W verlötet und in den Verteilerdosen mit Federkraftklemmen verbunden (siehe Abbildung 3.5).

Die Energieversorgung der Beleuchtung erfolgt über ein 12-Volt-Netzteil mit 3,5 A Ausgangsleistung. Auch hier wurde Zwillingslitze zur Verteilung zwischen den einzelnen Holmen genutzt.

Der WLAN-Router wird ebenfalls mittels 5-Volt-Gleichspannung betrieben. Hier war ein entsprechendes USB-Netzteil mitgeliefert.

Alle Netzteile werden von einer zentralen Steckdosenleiste mit 230-Volt-Netzspannung versorgt. Sie sind für eine Wechselspannung zwischen 100 und 230 V ausgelegt, sodass mit einem entsprechenden Adapter eine Nutzung in anderen Ländern möglich wäre. Die gesamte Energieverteilung ist der Abbildung 3.6 zu entnehmen. Rote Verbindungen stellen hierbei 12-Volt-Leitungen dar, blaue 5-Volt-Leitungen und schwarze die 230-Volt-Leitungen. Grüne Verbindungen sind Datenleitungen (Steuerleitung LED-Streifen).

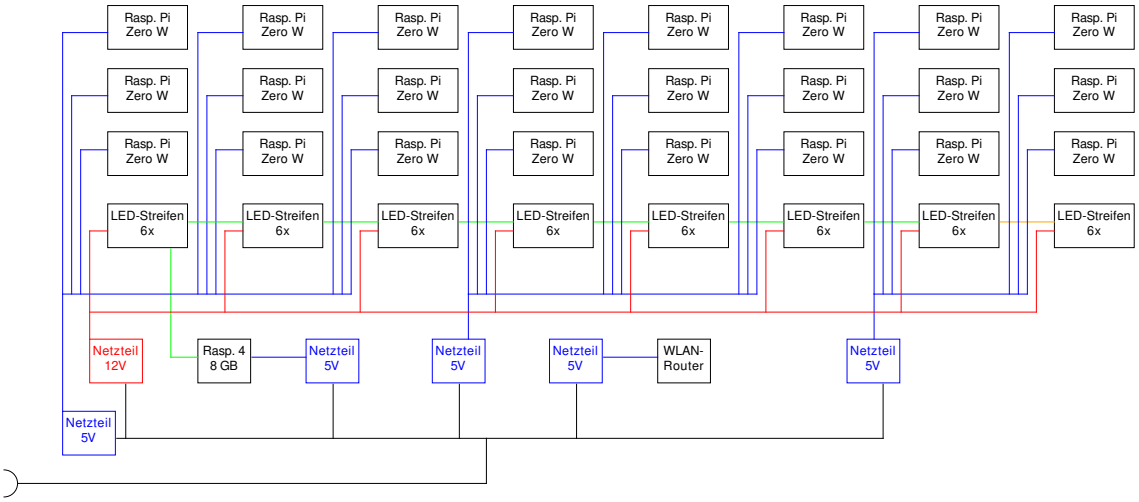


Abbildung 3.6.: Schematische Darstellung der Energieverteilung (blau: 5 V; rot: 12 V; schwarz: 230 V; grün: Daten)

3.6. Kostenbetrachtung

Der Titel dieser Arbeit verweist bereits darauf, dass eine kostengünstige Lösung gesucht werden sollte. Während der Konzeption und des Baus des Systems wurde entsprechend auf ein möglichst gutes Preis-Leistungs-Verhältnis geachtet. Eine Auflistung der Einzelposten ist in Anhang D aufgeführt. Die Gesamtkosten betrugen rund 1900 Euro. Die Kosten für die Softwareentwicklung bzw. der Zeitaufwand dafür sind hier nicht berücksichtigt.

Den größten Teil der Kosten machen die Kameras und die zugehörigen Raspberry-Pi-Computer aus. Wie schon in Abschnitt 3.1 erwähnt, kostet ein Kameramodul etwa 30 Euro. Hierzu kommen noch 18 Euro für den Raspberry Pi Zero W. Mit Gehäuse und Speicherkarte ergeben sich so rund 60 Euro pro Kamera. Für den Prototypen wurden 24 Kameras verwendet, somit entfallen hierauf grob 1400 Euro und damit über drei Viertel der Gesamtkosten. Die Anzahl der Kameras wird daher in Abschnitt 7.5 auf ihr Einsparpotenzial hin untersucht.

Andere elektronische Bauteile verursachten Kosten von rund 200 Euro. Hierzu zählen der Raspberry Pi 4, die Netzteile, der WLAN-Router, die LED-Streifen und die Leitungen. Der Rahmen aus Aluminiumprofilen mit Schrauben und Leitungshalterungen kostete rund 250 Euro. Für die Stoffhülle wurden etwa 20 Euro ausgegeben.

Ein nicht zu unterschätzender weiterer Kostenfaktor ist die für den Bau benötigte Zeit. Für den Prototypen wurden, zusammengerechnet rund 2-3 Wochen benötigt. Diese Zeit könnte durch Nutzung von mehr fertigen Bauteilen oder Automatisierung verkürzt werden, dieses würde jedoch wieder höhere Materialkosten zur Folge haben. Auch die Softwareentwicklung, die hier nicht berücksichtigt wurde, verursacht Kosten, die jedoch nicht direkt in die Hardwarekosten einfließen und - Pflegeaufwand nicht berücksichtigt - eher einmaliger Natur sind.

4. Voruntersuchungen

Vor und während des Aufbaus des eigentlichen Messsystems wurden einige Voruntersuchungen durchgeführt. Diese dienten dazu, die Machbarkeit des Systems zu prüfen und die notwendigen Schritte zu ermitteln und zu optimieren. Hauptsächlich ging es hierbei um die Ermittlung der Kamerakonstanten und der Verzeichnung der Kamera. Auch die Möglichkeit der Erstellung eines 3D-Modells durch Fokusstacking wurde untersucht. Die einzelnen Untersuchungen werden im Folgenden kurz vorgestellt.

4.1. Überprüfung der Kameraauflösung

These Die Angabe der Auflösung in Pixeln täuscht bei günstigen Kameras über die tatsächliche Auflösung hinweg.

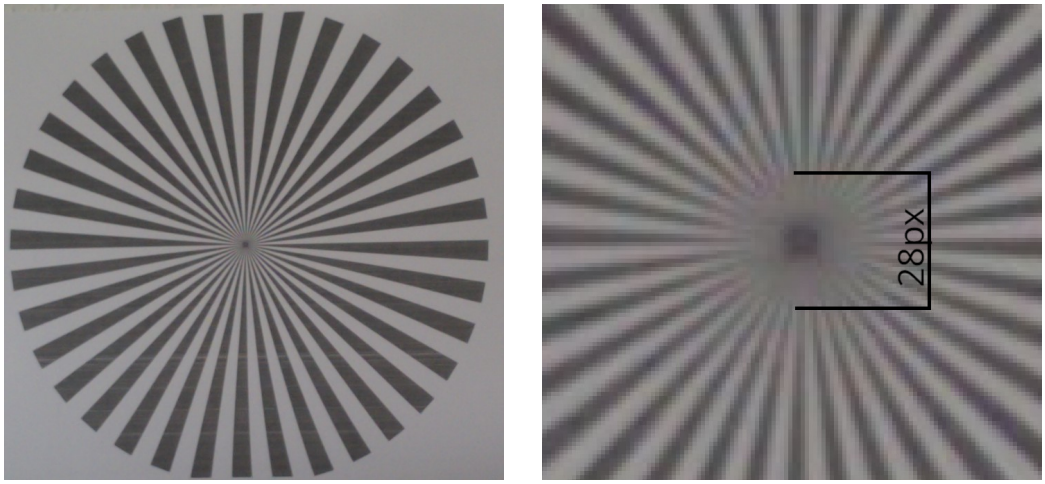
Ziel Die tatsächliche Auflösung der Kamera soll ermittelt werden. Hierbei spielt neben der reinen Anzahl der Pixel (px) auch die Qualität des Objektivs der Kameras eine Rolle.

Vorgehen Es wurden Bilder von einem Siemensstern unter verschiedenen Belichtungssituationen und Entfernungen aufgenommen (Beispielaufnahme siehe Abbildung 4.1a). Anschließend wurde die Größe des Unschärfekreises ermittelt und die Linienauflösung und -größe hieraus nach Luhmann (2023, S. 161) berechnet. Zum Vergleich wurden die Bilder auch mit einer hochwertigeren Kamera aufgenommen (Spiegelreflexkamera Canon EOS 77D mit EF-S 18-55 mm 1:4-5.6 IS STM).

$$[\text{Auflösungsvermögen}] = \frac{[\text{Anzahl Segmente}]}{\pi \cdot [\text{Durchmesser Unschärfekreis}]}$$

$$[\text{Liniengröße}] = \frac{1}{[\text{Auflösungsvermögen}]}$$

Ergebnis Der Unschärfekreis war durchschnittlich 28 Pixel (px) groß (siehe Abbildung 4.1b). Dies entspricht einer Linienauflösung von 292 Linien pro Millimeter, die Linienbreite beträgt 0,0034 mm beziehungsweise 2,44 px. Die Linienauflösung ist mit etwa 80 % der Sensorauflösung sehr hoch. Die Linienbreite in Pixeln entsprach der zum Vergleich genutzten Spiegelreflexkamera. Sie kann im Maximalfall (beste Auflösung) 2 px betragen, entsprechend je eines weißen und schwarzen Pixels pro Linie.



(a) 100 % (b) 800 %

Abbildung 4.1.: Siemensstern

4.2. Änderung der Kamerakonstante durch Fokussierung

These Die Kamerakonstante einer Kamera ändert sich durch die Fokussierung. Bei gleich eingestellter Objektdistanz ist die Kamerakonstante näherungsweise gleich.

Ziel Kraus (2004, S. 59) gibt die Gleichung 4.1 für die Bildweite b in Abhängigkeit der Gegenstandsweite g an. Näherungsweise entspricht die Bildweite der Kamerakonstante (vgl. Kraus, 2004, S. 59). Die Nutzung der Formel als Näherungswert soll überprüft werden und eine optimierte Formel für die Raspberry-Pi-Kameras ermittelt werden.

$$\frac{1}{f} = \frac{1}{g} + \frac{1}{b} \tag{4.1}$$

mit f : Brennweite der Kamera

g : Gegenstandsweite

b : Bildweite

Vorgehen Die Änderungen der Parameter wurden in einem Versuch beobachtet. Hierzu wurde der Raspberry Pi Zero mit montierter Kamera fest vor einem ChArUco-Kalibriermuster platziert. Das Kalibriermuster besteht aus einer Kombination von ArUco-Markern und einem Schachbrettmuster (siehe Abbildung 4.2). Durch das Schachbrettmuster ermöglicht es auch bei unscharfen Bildern eine gute automatische Erkennung der zu beobachtenden Punkte. Es wurden je 11 Bilder mit unterschiedlichen Fokussierungen von 2 m bis 10 cm aufgenommen. Dieser Vorgang wurde insgesamt viermal wiederholt, um auch die Wiederholungsgenauigkeit zu ermitteln. Die Bilder wurden anschließend mit einem Python-Skript unter Nutzung von OpenCV ausgewertet. Hierbei wurde die relative Veränderung der Kamerakonstante ermittelt und mit dem erwarteten Wert verglichen. Die relativen Änderungen wurden auf eine Fokusdistanz von 20 cm, entsprechend einer Fokussierung von 5 Dioptrien (dpt), normiert. Es wurden relative Angaben genutzt, da noch keine genaue Kenntnis über die tatsächliche Kamerakonstante bestand.

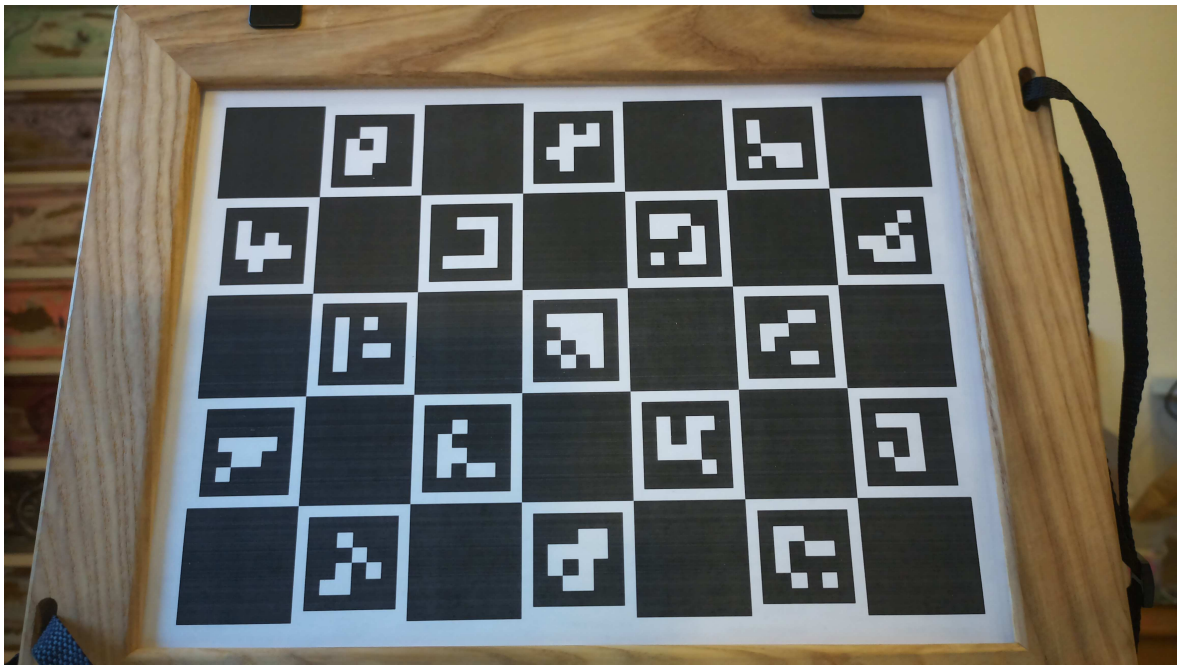


Abbildung 4.2.: ChArUco-Board mit Fokus auf 5 dpt

Ergebnis Die Ergebnisse sind in einem Box-Whisker-Plot in Abbildung 4.3 dargestellt. Es zeigt sich, dass die Änderungen der Kamerakonstante durch die Fokussierung linear zu der Dioptrienzahl (Kehrwert der Gegenstandsweite) sind. Ein lineares Verhältnis ergibt sich auch mit der im Datenblatt angegebenen Brennweite von 4,74 mm, jedoch eine etwas flachere Gerade (blau). Die ausgleichende Gerade (rot) ergab eine Brennweite

von 6,97 mm (Kamerakonstante bei Fokussierung auf unendlich bzw. 0 dpt). Die Abweichung von 2,23 mm entspricht einer Abweichung von 47 %, was als unrealistisch hoch eingeschätzt wird. Hier scheinen sich weitere Effekte bemerkbar zu machen, die noch nicht berücksichtigt wurden. Es wurde daher auch die Verzeichnung weiter untersucht.

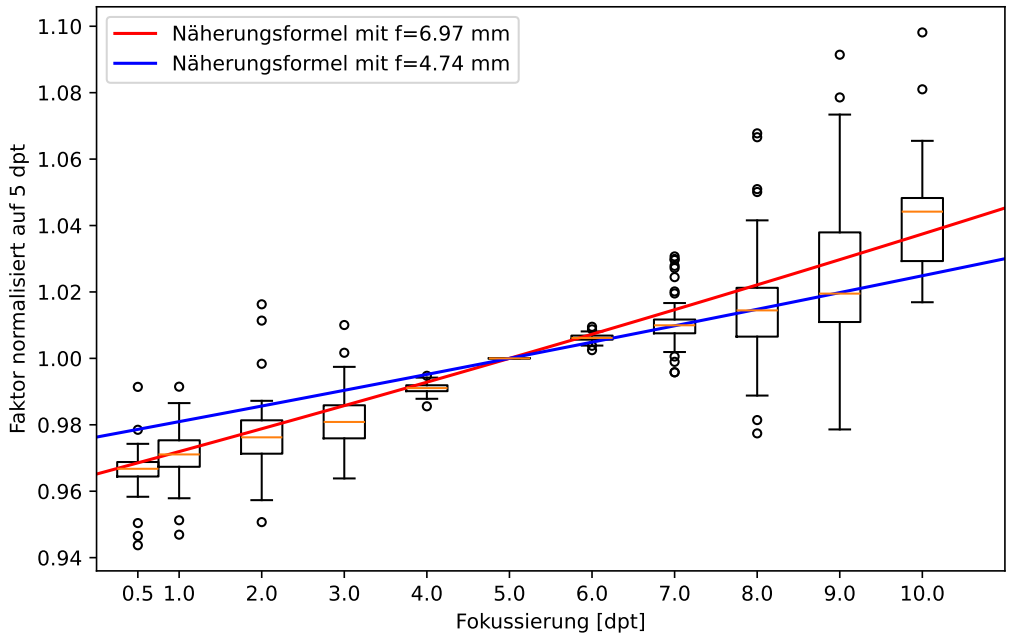


Abbildung 4.3.: Box-Whisker-Plot der relativen Veränderung der Kamerakonstante normalisiert auf eine Fokusdistanz von 20 cm (5 dpt)

4.3. Änderung der Verzeichnung durch Fokussierung

These Durch die Verschiebung der Linsen bei der Fokussierung verändert sich auch die Verzeichnung der Kamera.

Ziel Die Veränderung der Verzeichnung und eine entsprechende Korrekturformel soll ermittelt werden.

Vorgehen Der Versuchsaufbau aus Abschnitt 4.2 zur Bestimmung der relativen Änderung der Kamerakonstante blieb bestehen. Es wurden jedoch zusätzlich die Verzeichnungen der Bilder ermittelt. Die Verzeichnung wurde mit OpenCV ermittelt und mit der erwarteten Verzeichnung verglichen.

Ergebnis Die Ergebnisse waren nicht zufriedenstellend. Es zeigte sich, dass die Verzeichnung mit nur einer Aufnahme pro Fokussierung nicht ausreichend genau modelliert werden konnte. Dieses wurde daher erst bei der Kamerakalibrierung weiter untersucht (siehe Kapitel 6).

4.4. 3D-Modell aus Fokusstacking

These Die Schärfentiefe der Kameras ist gering. Durch Fokusstacking kann ein besseres 3D-Modell erstellt werden.

Ziel Es wird vermutet, dass die geringe Schärfentiefe (siehe Unterabschnitt 2.2.4) die Qualität der 3D-Modelle begrenzt. Daher soll geprüft werden, ob durch Fokusstacking die Qualität des 3D-Modells verbessert werden kann.

Vorgehen Mittels eines Python-Skriptes wurden die notwendigen Fokusschritte berechnet, damit der Bereich von 10 cm bis 1 m scharf abgebildet wird. Hierbei wurden die Werte so bestimmt, dass jeweils die hintere Schärfegrenze a_h einer Aufnahme der vorderen a_v des nächsten Bildes entspricht. Im Gegensatz zu den Berechnungen in Unterabschnitt 2.2.4 wurde hier ein Unschärfekreis von 6 px akzeptiert, da ansonsten die Anzahl der notwendigen Aufnahmen zu groß geworden wäre. Die berechneten Werte mit ihren Grenzen sind in Tabelle 4.1 aufgelistet und in Abbildung 4.4 dargestellt – die blauen Striche stellen hierbei die Abdeckung einer Aufnahme dar.

Tabelle 4.1.: Notwendige Fokusschritte für den Bereich von 0,1 bis 1 m

Fokussierung		**Nahgrenze**	**Ferngrenze**
[m]	**[dpt]**	**[m]**	**[m]**
0,11	9,31	0,10	0,12
0,13	7,93	0,12	0,14
0,15	6,53	0,14	0,17
0,19	5,13	0,17	0,23
0,27	3,72	0,23	0,33
0,43	2,30	0,33	0,63
1,15	0,87	0,63	6,66

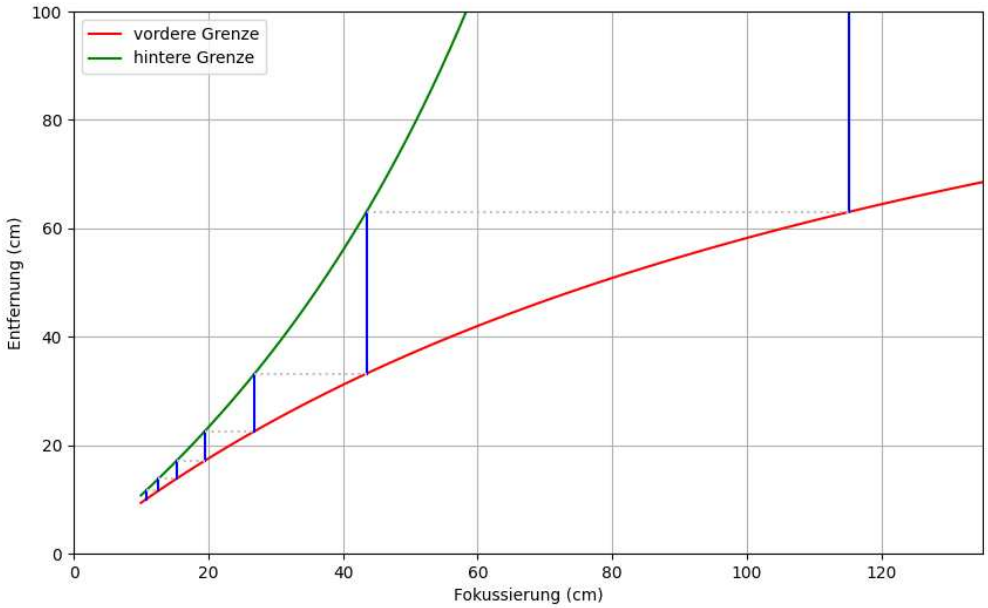


Abbildung 4.4.: Notwendige Fokusschritte, um den Bereich von 10 cm bis 1 m scharf abzubilden

Anschließend wurden die Bilder mit einem Python-Skript unter Verwendung von OpenCV automatisiert zu je einem Bild pro Kamera, bestehend aus allen scharf dargestellten Bereichen, zusammengerechnet. Dazu wurde jeweils die notwendige Transformation mittels SIFT (siehe Unterabschnitt 2.4.2) und Homografie berechnet. Die fertigen Bilder wurden in Agisoft Metashape zur Berechnung eines 3D-Modells genutzt. Wie bereits in den vorherigen Untersuchungen beschrieben, wurden die Daten mit denen des Streifenprojektionssystems verglichen.

Ergebnis Die Genauigkeit der Passpunkte bei der Bildverknüpfung in Agisoft Metashape ließ zuerst ein gutes Ergebnis vermuten. Jedoch waren die Ergebnisse des 3D-Modells nicht so wie erhofft, die Qualität des 3D-Modells war schlechter als bei einer einzelnen Aufnahme. Die Flächen des Testobjektes (Testy, siehe Abschnitt 7.1) waren kaum mehr vorhanden. Nur mit sehr viel manueller Nacharbeit wurde überhaupt ein erkennbares Ergebnis erzielt. Die Ursache hierfür ist nicht bekannt, aber es wird vermutet, dass die automatische Zusammenrechnung der Bilder zu vielen Artefakten in den Bildern geführt hat, sodass die Bilder dann von der SfM-Software nicht genau genug verknüpft werden konnten. Auch unter Nutzung von kommerziellen Programmen wie Helicon Focus (verwendet von Clini et al., 2016) und Zerene Stacker waren die Ergebnisse nicht viel besser. Es wurde daher entschieden, den Ansatz hier nicht weiterzuverfolgen.

4.5. Überprüfung der Kamerasynchronität

Eine Anforderung war die simultane Auslösung aller Kameras. Um dies zu überprüfen, wurden die Kameras auf eine Stoppuhr mit Anzeige von hundertstel Sekunden gerichtet und eine Aufnahme entsprechend der automatischen Aufnahme eines 3D-Modells durchgeführt. Die Bilder wurden anschließend ausgewertet und hieraus das erste und letzte Bild bestimmt. Dieser Vorgang wurde dreimal wiederholt.

Die Messung zeigte eine Abweichung zwischen dem ersten und letzten Bild von maximal 0,2 Sekunden. Die Synchronität der Kameras ist damit für den Anwendungsfall gegeben. Eine Aufnahme von bewegten Objekten ist damit jedoch nicht möglich. Die Abweichung ist auf die unterschiedlichen Reaktionszeiten der Kameras zurückzuführen. Ursächlich hierfür ist unter anderem, dass es sich bei dem Betriebssystem der Raspberry-Pi-Computer nicht um ein Echtzeitbetriebssystem handelt und durch das Multitasking des Prozessors Ausführungen um eine unbestimmte Zeit verzögert werden können. Außerdem kann die Netzwerkverbindung das Signal weiter verzögern. Die Daten werden zwar gleichzeitig an die verschiedenen Kameras abgesendet, aber da auch das WLAN ein Zeitschlitzverfahren nutzt, werden diese nicht gleichzeitig übertragen.

5. Software-Entwicklung

Für die Steuerung der Kameras und die anschließende Berechnung des 3D-Modells muss eine Steuerungssoftware für das Kamerasystem und eine entsprechende Schnittstelle zu einer SfM-Software geschaffen werden. Die Entwicklung erfolgte hauptsächlich in Python in Form von Prototyping. Dieses Kapitel beschreibt die Anforderungen an die Software (siehe Abschnitt 5.1) und die zu implementierenden Anwendungsfälle (siehe Unterabschnitt 5.1.1). Abschließend wird die Implementation (siehe Abschnitt 5.3) erarbeitet.

Der Quellcode ist dem Git-Repository unter `https://github.com/FlorianTimm/PhotoBox` zu entnehmen. Die Software wurde unter der MIT-Lizenz veröffentlicht, sodass sie frei genutzt und weiterentwickelt werden kann.

5.1. Anforderungsanalyse

Die notwendigen Anforderungen an das System wurden aus Kapitel 1 und 2 abgeleitet und in diesen Abschnitt in Form von Anwendungsfällen und eines Lastenheftes aufgezeichnet. Sie wurden in funktionale und nicht-funktionale Anforderungen unterteilt. Die funktionalen Anforderungen beschreiben, was das System leisten soll, die nicht-funktionalen Anforderungen beschreiben, wie das System arbeiten soll.

5.1.1. Anwendungsfallmodellierung

Entsprechend der benötigten Schritte aus Kapitel 2 und Abbildung 2.1 wurden die Anwendungsfälle, die die Benutzeroberfläche ermöglichen soll, im Anwendungsfall-Diagramm in Abbildung 5.1 zusammengetragen. Entsprechend der Anforderung einer einfach gehaltenen Bedienung des Systems, ist die Anzahl der verschiedenen Anwendungsfälle gering. Hauptanwendungsfall ist die Erzeugung eines 3D-Modells, welches in mehrere Unterfälle unterteilt ist, je nach verwendeter SfM-Software. Es soll aber auch möglich sein, Teilschritte dieser Anwendungsfälle wie das Aufnehmen der Bilder, das Identifizieren der Passpunkte oder das Verbinden der Kameras manuell auszulösen.

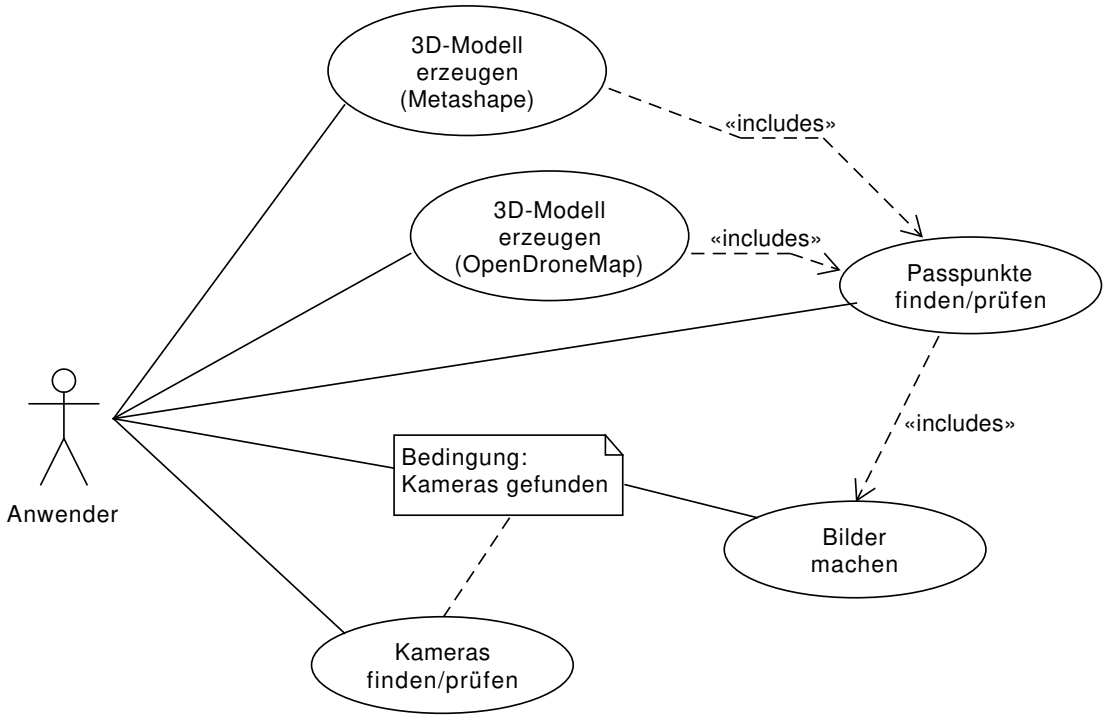


Abbildung 5.1.: Anwendungsfall-Diagramm

5.1.2. Funktionale Anforderungen

F1 Die Kameras sollen zeitgleich und möglichst verzögerungsfrei auslösbar sein.

F2 Die Steuerung soll unabhängig von anderen Geräten möglich sein, beispielsweise per Tastensteuerung.

F3 Der Status des Systems soll einfach erkennbar sein - auch ohne Anschluss eines Computers.

F4 Es sollen Passpunkte automatisch gefunden und für die Bestimmung der äußeren Orientierung genutzt werden.

F5 Die Bilder sollen scharf und fokussiert sein.

F6 Die Belichtung soll automatisch erfolgen, jedoch die Helligkeit der Bilder identisch sein.

5.1.3. Schnittstellen

S1 Die Daten sollen intern gespeichert werden.

S2 Eine Speicherung auf tragbaren Speichermedien wie USB-Sticks soll möglich sein.

S3 Eine direkte Übertragung an eine SfM-Software soll möglich sein.

5.1.4. Nicht-funktionale Anforderungen

N1 Die Erfassung soll ohne weitere Hardware möglich sein. Das System soll unabhängig von Netzwerkanschlüssen etc. sein.

N2 Jegliche (Daten-)Kommunikation soll über WLAN erfolgen.

5.2. Anwendungsentwurf

Der Anwendungsentwurf legt die grundlegenden Strukturen der Software fest. Hierbei wird festgelegt, welche Module benötigt werden und wie diese miteinander kommunizieren. Diese Struktur wird einmalig festgelegt und bildet die Grundlage für die Implementierung.

5.2.1. Domänen-Klassendiagramm

Aus den für die Berechnungen des 3D-Modells benötigten Daten wurde das Domänen-Klassendiagramm aus Abbildung 5.2 erzeugt. Dieses zeigt vor allem die Abhängigkeiten der einzelnen Datensätze untereinander. Es bildet hiermit die Grundlage für die Datenschnittstellen der einzelnen Komponenten zueinander und den jeweils benötigten Datensätzen.

Zur Berechnung eines 3D-Modells werden die innere und äußere Orientierung der Kameras benötigt (vgl. Kapitel 2). Da die innere Orientierung der Kameras instabil ist, reichen hier Näherungswerte. Entsprechend wurde diese nicht wie üblich pro Kamera, sondern nur pro Modell festgehalten. Die Klasse `CameraModel` hat daher neben dem Namen Attribute für die Parameter der inneren Orientierung in Form einer Geradengleichung abhängig von der Fokussierung. Jede Kamera hat dann (pro Messung) eine Position (`Point3D`) und drei Drehwinkel, hierdurch wird die äußere Orientierung im Modell abgebildet. Um die Passpunktpositionen festzuhalten, wird das Objekt `PasspointPosition` verwendet. Dieses speichert die Position des Passpunktes in den Bildern der Kameras als `Point2D` und die Position im Raum als `Passpoint`. Außerdem verweist die Klasse auf das entsprechende Bild (`Image`), welches wiederum auf die Kamera (`Camera`) verweist. Die Klasse `Image` hält des Weiteren die Attribute für den Dateinamen und die genutzte Fokussierung.

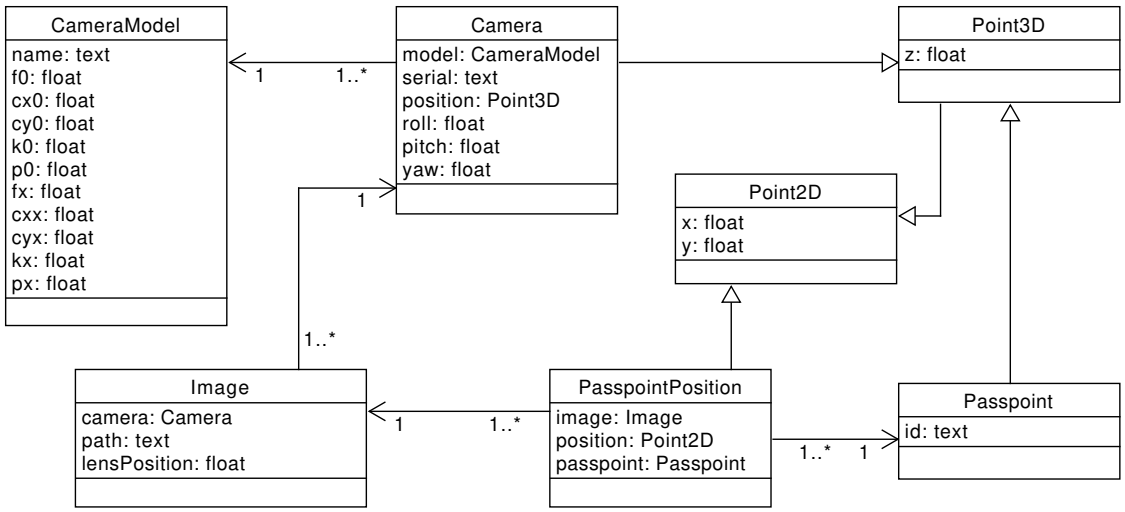


Abbildung 5.2.: Domänen-Klassendiagramm

5.2.2. Programmablauf

Die Kommunikation innerhalb des Systems ist in dem Ablaufdiagramm in Abbildung 5.3 dargestellt. Eine Auslösung der Kamera erfolgt über den Software-Button in der Desktop-Software, dem Webinterface oder über einen Taster. Optional werden die Belichtungen der Kameras synchronisiert und anschließend die Aufnahmen erzeugt. Die Raspberry Pi Zero W senden die Aufnahmen und die gefundenen ArUco-Marker an den Raspberry Pi 4. Dieser berechnet die Kamerapositionen und speichert die Daten. Die Desktop-Software kann die Daten dann herunterladen und an die SfM-Software übergeben, welche ein 3D-Modell erzeugt. Auf die Details der einzelnen Module wird in Abschnitt 5.3 eingegangen.

5.3. Implementierung

Die Programmierung des Systems erfolgte iterativ. Einzelne Arbeitspakete wurden in einem Jupyter-Notebook ausprobiert und dann, wenn dieser Schritt erfolgreich war, in den Gesamtworkflow integriert - teilweise sind diese im Kapitel 4 beschrieben. Größtenteils wurde der Python-Code objektorientiert und typisiert geschrieben. Die Teile, die auf einem Desktoprechner ausgeführt werden sollen, wurden in Java geschrieben, da hier später die Einrichtung auf verschiedenen Rechnern und Plattformen einfacher ist.

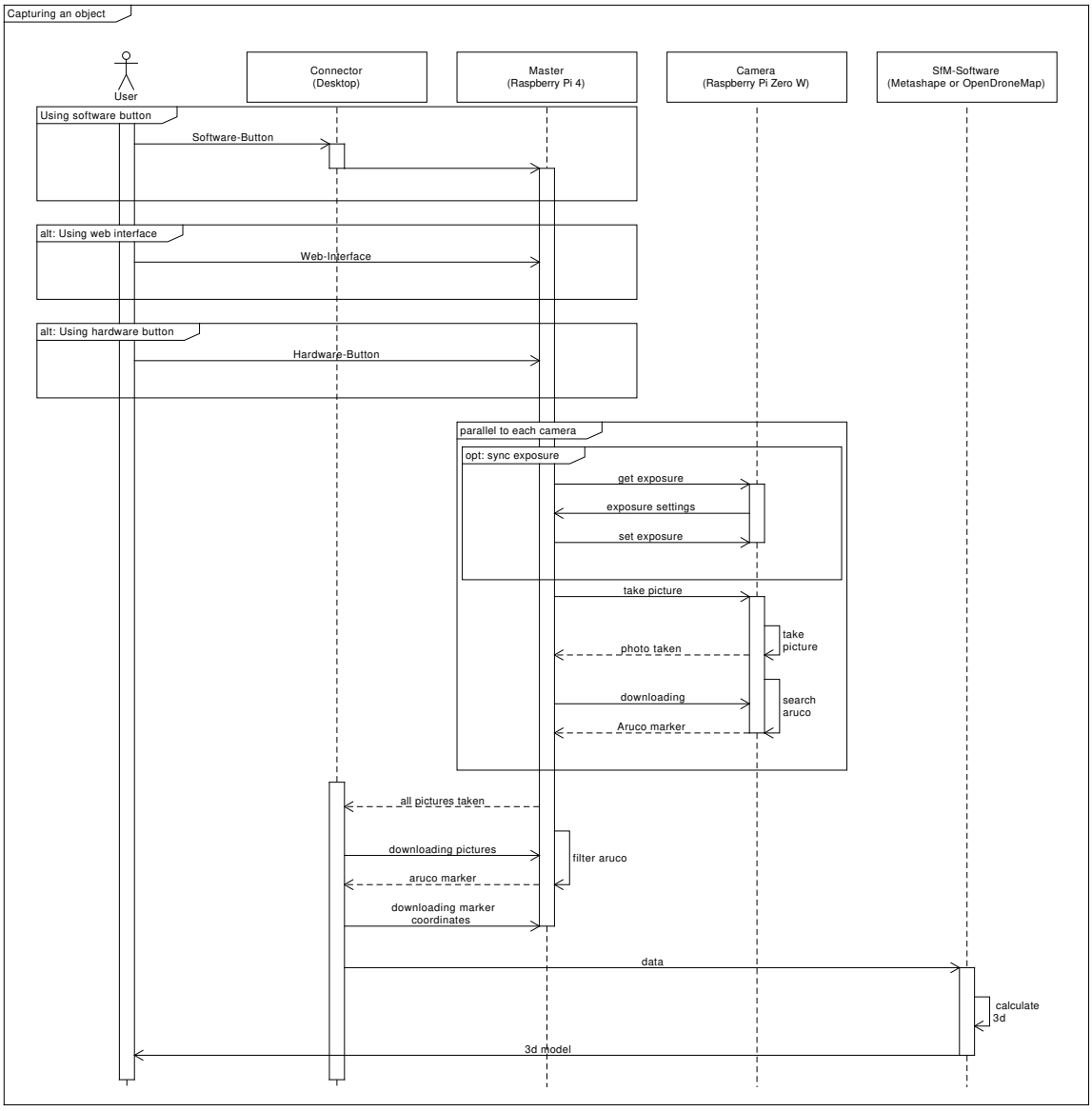


Abbildung 5.3.: Ablaufdiagramm zur Kommunikation bei der Aufnahme eines 3D-Modells

5.3.1. Module auf den Raspberry-Pi-Computern (Python)

Die Software auf den Raspberry-Pi-Computern wurde in Python geschrieben. Hierbei wurde darauf geachtet, dass die Module möglichst unabhängig voneinander sind und nur über definierte Schnittstellen kommunizieren.

Bibliotheken

Es wurde, wenn möglich, auf fertige Python-Bibliotheken zurückgegriffen. Hierdurch sollte der Programmieraufwand verringert und auf bereits getesteten Code gesetzt werden. Außerdem greifen viele der Bibliotheken wie OpenCV oder NumPy auf hardwarenahe Berechnungen zurück, sodass der Geschwindigkeitsnachteil von Python nicht weiter ins Gewicht fällt. Die wichtigsten Bibliotheken sind:

OpenCV ist eine Bibliothek für Bildbearbeitung und maschinelles Sehen. Sie ist weit verbreitet und bietet viele photogrammetrische Funktionen. Hiermit wurde beispielsweise die Detektion von Markern durchgeführt und die Näherungswerte der Kameras berechnet.

SciPy stellt Methoden für wissenschaftliche Berechnungen bereit, beispielsweise für verschiedene Formen von Ausgleichungsrechnungen. Sie wurde für die Berechnung der Bündelblockausgleichung verwendet. Der manuelle Ansatz mit Formeln aus Luhmann (2023, S. 349ff) unter Nutzung von NumPy alleine war sehr ressourcenlastig. Mit Unterstützung von SciPy und unter Verwendung der Projektionsgleichung konnte die Berechnungsdauer stark dezimiert werden.

NumPy bietet Datenstrukturen für Matrizen und Vektoren sowie effiziente Berechnungen mit diesen. Sie wurde beispielsweise für die Berechnung der Kamerapositionen und die Bündelblockausgleichung genutzt. Außerdem ist NumPy Grundlage für OpenCV und SciPy.

Flask ist ein Webframework für Python. Es wurde genutzt, um die Weboberfläche und die Datendownloads bereitzustellen. Hiermit wurde ein Webserver aufgesetzt, der die Daten der Kameras anzeigt und die Steuerung ermöglicht.

rpi-ws281x ermöglicht die Steuerung der RGB-LEDs. Hierzu wird das in Worldsemi (2012) beschriebene Steuerprotokoll von der Bibliothek implementiert. Die Steuerung kann dann mittels einfacher Farbcodes erfolgen.

Allgemeine Module

In Abbildung 5.4 werden Klassen dargestellt, die in allen Modulen genutzt werden. Diese sind verantwortlich für allgemeine Funktionen wie das Logging und das Auslesen der Konfiguration. Die Klasse `Config` liest die Konfiguration aus einer Datei aus und stellt diese zur Verfügung. `Logger` ermöglicht das Loggen von Informationen und Fehlern in einer Textdatei, in den Systemprotokollen und/oder die Ausgabe auf dem Terminal bei manuellem Start. Die Klassen `CamSettings` und `ArucoMarkerPos` sind Datenklassen, die die Einstellungen der Kameras und die Position der ArUco-Marker speichern. Diese Daten werden zwischen der Master-Steuerung und der Kamera-Steuerung ausgetauscht – hierdurch wird sichergestellt, dass das erwartete Datenmodell auf beiden Seiten gleich ist. `CamSettings` werden dabei vom Raspberry Pi 4 an die Raspberry Pi Zero W gesendet, um die Kameras zu konfigurieren. `ArucoMarkerPos` werden von den Raspberry Pi Zero W an den Raspberry Pi 4 gesendet, um die Position der identifizierten ArUco-Marker zu übertragen.

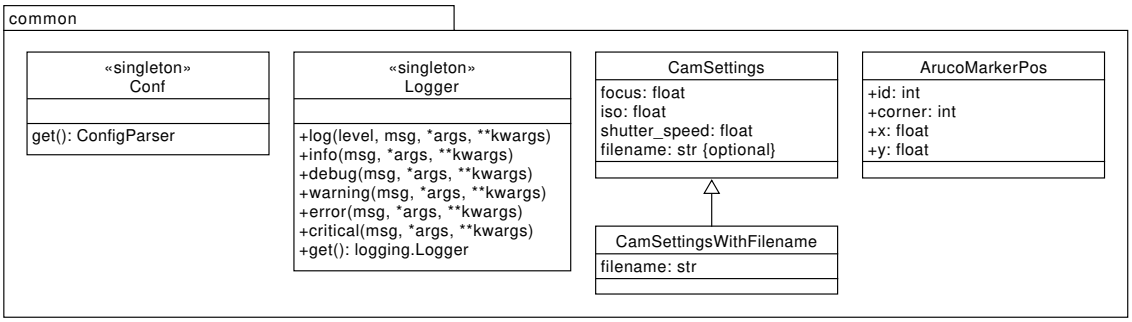


Abbildung 5.4.: Klassen des Common-Package

Master-Steuerung

Auf einem Raspberry Pi 4 läuft die Gesamtsteuerung des Systems. Dieses stellt Schnittstellen zur Steuerung auf drei verschiedenen Wegen bereit: per Taster, per Weboberfläche und per Socket-Verbindung. Außerdem stellt es die Daten per Representational-State-Transfer (REST)-Schnittstelle zur Verfügung. Die Klassen sind in Abbildung 5.5 dargestellt.

Das Skript `Master` wird automatisch bei Systemstart über `systemd` als Daemon – als im Hintergrund und dauerhaft laufender Service (vgl. Negus, 2020, S. 369) – gestartet. Hier ist die Webschnittstelle implementiert. Sie ermöglicht die komplette Steuerung, das Konfigurieren sowie das Anzeigen und Herunterladen der Bilder. Das Skript instanziiert außerdem die Klasse `Control`, welche die hauptsächliche Steuerung übernimmt und den Kern des Systems darstellt.

Die Steuerung per Taster implementiert die Klasse `ButtonControl`. Sie ermöglicht einfache Aufgaben wie das Suchen von Kameras, das Aufnehmen von Bildern und die Aktivierung des Standbys bzw. das Herunterfahren des Systems auch ohne PC durchzuführen (Anforderung F2). Die Klasse `DesktopControlThread` startet, wie der Name bereits vermuten lässt, als eigenständiger Thread und stellt eine Socket-Verbindung zur Kommunikation mit der Desktop-Software und die Steuerung hierüber bereit. Einen weiteren Thread stellt `CameraControlThread` zur Verfügung. Dieser überwacht das Netzwerk auf Nachrichten der Kameras und initiiert die entsprechenden Schritte. Die eigentlichen Verarbeitungsschritte werden in der Klasse `Control` durchgeführt. Auch für diese Aufgaben werden zum Teil eigene Threads gestartet, damit die Verarbeitung parallel erfolgen kann. Beispielsweise wird der Download der Bilder und die Berechnung von Kamerapositionen in eigenen Threads durchgeführt.

Die `Control`-Klasse steuert die Aufnahme, sammelt die Daten der einzelnen Kameras und stellt diese zur Weiterverarbeitung zur Verfügung. Sie sendet eingehende Aufträge als Broadcast-Nachrichten in das Netzwerk an die Raspberry Pi Zero W mit den Kameras, die dann beispielsweise Bilder aufnehmen. Hierdurch wird die nahezu zeitgleiche Auslösung gewährleistet (Anforderung F1), die bei einer einzelnen Ansteuerung als Schleife nicht möglich wäre. Zur Vereinheitlichung der Belichtung kann eine Funktion aktiviert werden, sodass erst einmal die Belichtung von jeder Kamera berechnet und diese dann von der `Control`-Klasse gemittelt wird. Dadurch werden alle Bilder mit der gleichen Einstellung aufgenommen, sodass keine Helligkeitsunterschiede zwischen den Aufnahmen bestehen (Anforderung F6). Nach der Aufnahme der Bilder und der Erkennung der ArUco-Marker werden die Daten von den Raspberry Pi Zero W an den Raspberry Pi 4 übertragen. Aus den ArUco-Markern wird die äußere Orientierung der Kameras als Näherungswerte bestimmt (Anforderung F4). Die Daten werden anschließend in einem Archiv gespeichert (Anforderung S1).

Die Datenübertragung an die Desktop-Software erfolgt über die REST-Schnittstelle, die wieder vom `Master`-Skript bereitgestellt wird. Alternativ kann auch ein USB-Stick an den Raspberry Pi angesteckt werden (Anforderung S2) oder die Daten über die Weboberfläche manuell heruntergeladen werden. Dazu wird bei der Verarbeitung der Daten geprüft, ob ein USB-Stick angeschlossen ist. Falls dies der Fall ist, wird dieser gemountet, die Daten kopiert und der Stick wieder ausgehängt, damit dieser (fast) jederzeit getrennt werden kann. Das Datenformat auf dem USB-Stick und in den Archiv-Dateien aus der Weboberfläche entspricht dem, welches die Desktop-Software anlegt. So kann diese die Daten dann auch einladen und weiterverarbeiten.

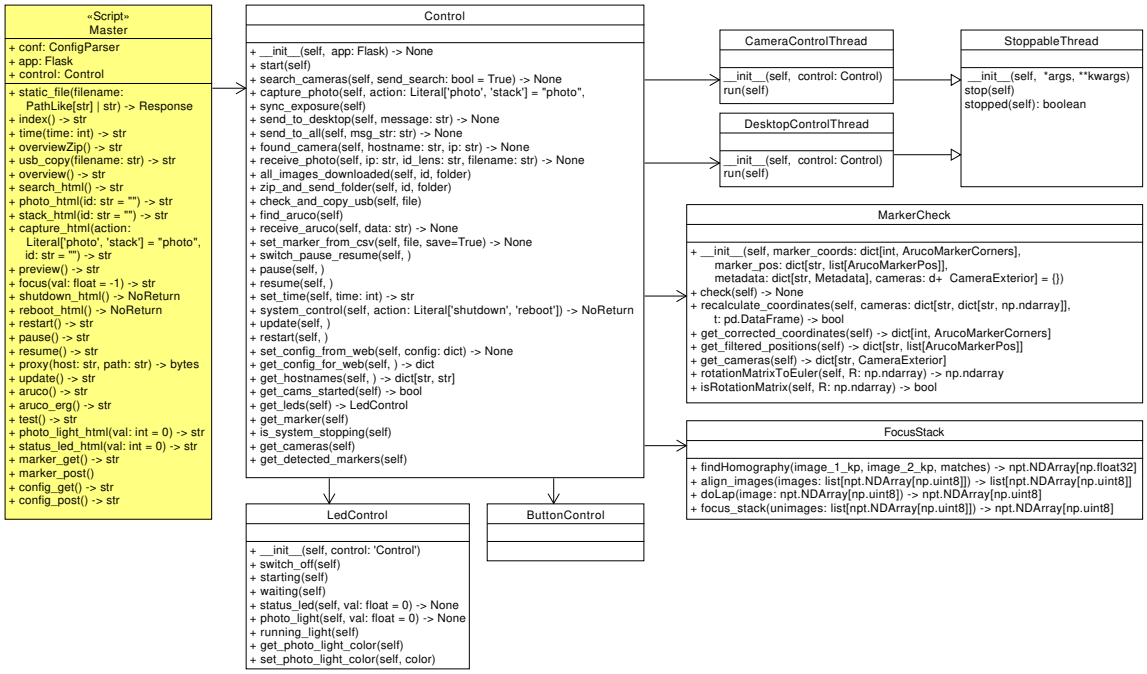


Abbildung 5.5.: Klassen des Master-Package

Kamera-Steuerung

Die Raspberry Pi Zero W übernehmen die Steuerung der Kameras. Hierfür wurde ein Modul entwickelt, dass die Kameras steuert, die Bilder aufnimmt und anschließend dem steuernden Raspberry zur Verfügung stellt. Die Klassen sind in Abbildung 5.6 dargestellt.

Das Skript `Camera` wird automatisch bei Systemstart über `systemd` als Daemon gestartet. Es erzeugt dann über Flask eine einfache Weboberfläche, über die die Kameras konfiguriert und die Bilder betrachtet werden können. Außerdem stellt das Skript auch die REST-Schnittstelle bereit, über die die Bilder und gefundenen Passpunkte abgerufen werden können. Die Klasse `CameraControl` wird vom Skript instanziiert und übernimmt die eigentliche Steuerung. Sie nimmt die Konfiguration entgegen und setzt diese um. Sie überwacht das Netzwerk auf Broadcast-Nachrichten der Master-Steuerung und initiiert die entsprechenden Schritte. Über die gleiche Socket-Verbindung meldet sie auch den Status an die Master-Steuerung, beispielsweise wenn Bilder aufgenommen wurden oder die Erkennung der Passpunkte abgeschlossen ist. Die Klasse `CameraInterface` wird von `CameraControl` aufgerufen, bildet einen Adapter für das Kameramodul und kapselt so die Hardwaresteuerung. Sie nimmt die Bilder auf und speichert diese. Gegebenenfalls leitet sie die Bilder an die `CameraAruco`-Klasse weiter, damit diese in einem eigenen Thread die ArUco-Marker identifizieren kann (Anforderung F4).

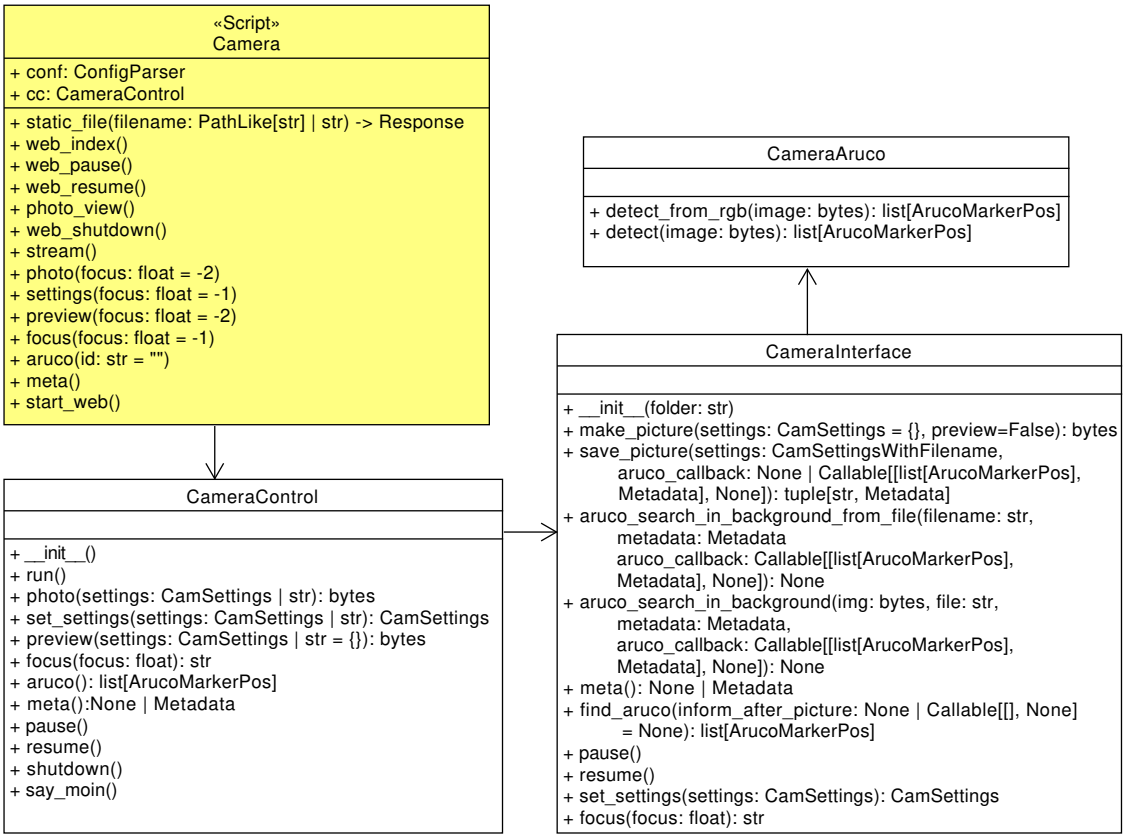


Abbildung 5.6.: Klassen des Camera-Package

5.3.2. Desktop-Schnittstelle (Java)

Die Desktop-Schnittstelle dient zur Steuerung des Systems und zur automatischen Übertragung der Daten an die SfM-Software (Anforderung S3). Sie wurde in Java geschrieben, um eine einfache Nutzung auf verschiedenen Betriebssystemen zu ermöglichen. Das Application Programming Interface (API) für Java der SfM-Software Agisoft Metashape kann im Gegensatz zum Python-API ohne Installation verwendet werden. Die Software kommuniziert mit dem Raspberry Pi 4 hauptsächlich über eine Socket-Verbindung, wobei jedoch die Datenübertragung der Bilder vom Raspberry Pi 4 zur Desktop-Software über die REST-Schnittstelle erfolgt. Neben der Übertragung der Bilder an die SfM-Software ermöglicht die Software das Starten von Aufnahmen. Ein Screenshot der Benutzeroberfläche ist in Abbildung 5.7 dargestellt. Die Schnittstellen-Software unterstützt als SfM-Software aktuell Agisoft Metashape und OpenDroneMap, jedoch ist die Schnittstelle so gestaltet, dass auch andere Software ergänzt werden könnte.

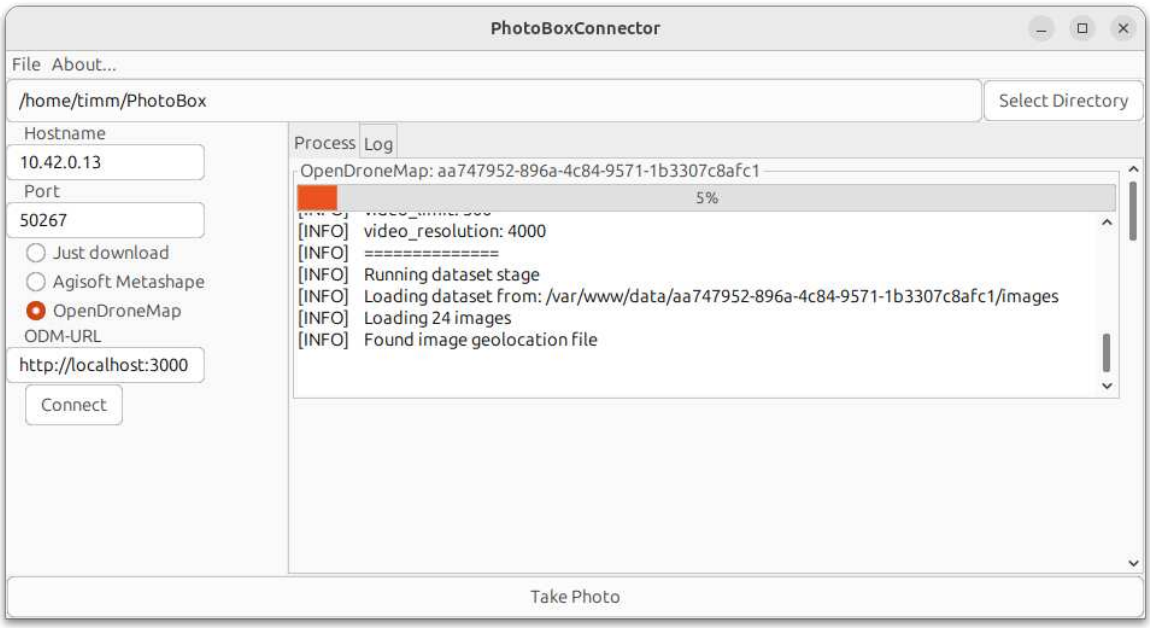


Abbildung 5.7.: Screenshot der Connector-Software beispielhaft unter Ubuntu 24.04

Die Klassen sind in Abbildung 5.8 dargestellt. `Connector` stellt hierbei den Einstiegspunkt dar. Die Klasse instanziiert die Benutzeroberfläche `ConnectorGui` und übernimmt die Steuerung der Software. Die Klasse `PhotoBoxClient` wird beim Klick auf Verbinden von `Connector` gestartet und übernimmt die Kommunikation mit dem Raspberry Pi 4. Sie sendet die Steuerbefehle und empfängt die Daten. Außerdem wird eine Klasse, die das Interface `SfmClient` implementiert, aufgerufen. Diese Klasse stellt die Schnittstelle zur SfM-Software dar. Hierbei wird je nach Auswahl der

Software `MetashapeClient`, `OpenDroneMapClient` oder `DownloadClient` instanziiert. Diese Klasse übernimmt die Kommunikation mit der entsprechenden SfM-Software und übergibt die Daten. `DownloadClient` stellt einen Sonderfall dar, hier werden die Daten nur gespeichert, um sie später weiterverarbeiten zu können.

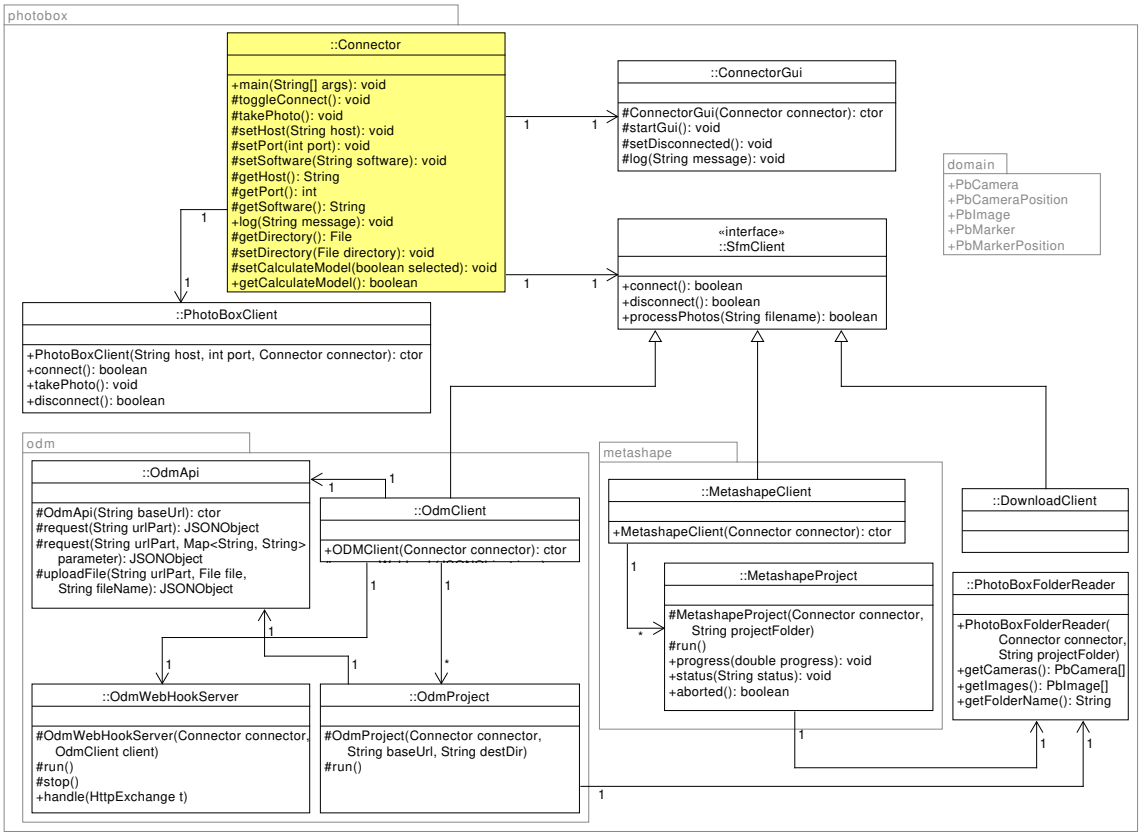


Abbildung 5.8.: Connector-Package

5.3.3. Konfiguration

Neben der eigentlichen Programmierung erfolgten noch verschiedene Konfigurationen, um das System zu betreiben. Diese sollen hier kurz vorgestellt werden.

WLAN-Router Der WLAN-Router wurde so konfiguriert, dass er ein WPA2-verschlüsseltes WLAN-Netzwerk aussendet. Außerdem wurde er als DHCP-Server eingerichtet, der den angeschlossenen Geräten automatisch eine IP-Adresse zuweist. Die IP-Adressen der Raspberry-Pi-Computer wurden fest vergeben, um die Kommunikation zu erleichtern. Der Router wurde so konfiguriert, dass er optional die Kommunikation zwischen dem Raspberry Pi 4 und einem per Netzwerkleitung angeschlossenen Desktop-PC zulässt, aber auch eine angeschlossene Internetverbindung durchleitet.

Raspberry Pi 4 Der Raspberry Pi 4 wurde so konfiguriert, dass er automatisch das Skript `Master` startet. Hierzu wurde ein `systemd`-Service erstellt, der das Skript beim Systemstart ausführt. Vorher wird bei Vorliegen einer Internetverbindung geprüft, ob das Git-Repository eine neue Version der Software zur Verfügung stellt. Falls dies der Fall ist, wird dieses heruntergeladen und installiert.

Zur Beschleunigung von Softwareupdates des Betriebssystems, wurde ein entsprechender apt-Proxyservice eingerichtet. Dieser speichert die heruntergeladenen Pakete und stellt diese bei Bedarf den Raspberry Pi Zero W zur Verfügung. Hierdurch wird die Bandbreite des Internetanschlusses geschont und die Updates können schneller durchgeführt werden.

Raspberry Pi Zero W Die Raspberry Pi Zero W wurden so konfiguriert, dass sie automatisch das Skript `Camera` starten. Hierzu wurde ebenfalls ein `systemd`-Service erstellt, der das Skript beim Systemstart ausführt. Auch hier wurde der bereits beschriebene Update-Mechanismus eingebaut. Entsprechend der Cache-Konfiguration des Raspberry Pi 4 wurde hier der apt-Proxyservice als Paketquelle eingerichtet.

6. Systemkalibrierung

Für die Berechnung von 3D-Modellen mit korrekter Skalierung sind einige Parameter zu bestimmen. Neben der Bestimmung der inneren Orientierung der Kameras ist auch eine Realisierung eines Maßstabes notwendig. Auf die notwendigen Schritte und die möglichen Fehlerquellen wird in diesem Kapitel eingegangen.

6.1. Maßstab und Passpunkte

Zur Bestimmung der Skalierung (siehe Abschnitt 2.3) wurde sich für eine Kombination aus Maßstäben und Passpunktkoordinaten entschieden. Die Möglichkeit der festen und bekannten äußeren Orientierung der Kameras entfiel, da diese projektabhängig bewegt werden sollen.

Die Passpunkte in Form von ArUco-Markern wurden fest am Rahmen montiert. Diese sollen später als dauerhafte Realisierung des Maßstabes dienen. Außerdem wurden kalibrierte Maßstäbe im Objektraum verteilt, welche für die erstmalige Bestimmung der Passpunktkoordinaten den Maßstab bilden. Zur Unterstützung der Bildverknüpfung wurden weitere Punkte in Form von Schneider-Markern im Bildbereich verteilt (ähnlich dem Aufbau in Abbildung 7.3). Anschließend wurden Bilder vom gesamten System mit einer externen Kamera mit festen Einstellungen aufgenommen. Diese Bilder wurden dann in Agisoft Metashape verarbeitet und Koordinaten der Passpunkte bestimmt. Über manuell in Agisoft bestimmte Punkte am Boden des Systems wurde das System so transformiert, dass der Boden die XY-Ebene darstellt.

6.2. Kamerakalibrierung

Die verwendeten Kameras weisen keine stabile innere Orientierung auf. Daher ist eine Kalibrierung in Form von festen Parametern nicht möglich. Es wurde die Annahme verfolgt, dass die innere Orientierung linear von der Fokussierung abhängig ist. Daraus wurde eine Formel zur Bestimmung von Näherungswerten für die innere Orientierung in Abhängigkeit der Fokussierung ermittelt.

Die eigentliche Bestimmung der inneren Orientierung erfolgt im Betrieb in Form einer Simultankalibrierung (siehe Unterabschnitt 2.1.1). Die Näherungswerte sollen dabei als Startwerte dienen.

Vorgehen Es wurden mit fünf verschiedenen Fokussierungen mit jeweils 24 Raspberry-Pi-Kameras Bilder aufgenommen und die Bilder in Agisoft Metashape mittels ArUco-Markern orientiert, dessen Position aus Abschnitt 6.1 bekannt ist. Außerdem wurden etwa 100 Schneider-Marker im Bildbereich der Kameras verteilt und als Verknüpfungspunkte benutzt. Es wurden in Metashape jeweils alle Bilder mit der gleichen Fokussierung als eine Kamera angenommen und die innere Orientierung bestimmt. Anschließend wurden alle Kameras nochmal einzeln ausgeglichen. Dieses zweistufige Vorgehen bewirkt, dass die Näherungswerte schrittweise verbessert werden. Ein einstufiges Vorgehen ohne Näherungswerte führte zu falschen Werten. Die innere Orientierung (vgl. Unterabschnitt 2.1.1) wurde in Form von Brennweite, Bildhauptpunktverschiebung und Verzeichnung ermittelt, die Ergebnisse in einem Box-Whisker-Plot dargestellt und eine ausgleichende Gerade berechnet.

Ergebnis Die Ergebnisse sind in Abbildung 6.1 dargestellt. Wie auch schon in der Voruntersuchung in Abschnitt 4.2 zeigt sich, dass die Kamerakonstante c linear zur Fokussierung ist. Bei der Bildhauptpunktverschiebung x'_0, y'_0 und der radial-symmetrischen Verzeichnung k_1,k_2,k_3 ist die Abhängigkeit nicht eindeutig. Tabelle 6.1 zeigt die Korrelationsmatrix der Näherungswerte. Die Fokussierung und die Kamerakonstante sind stark korreliert. Die Korrelation der restlichen Parameter ist nur schwach. Es zeigt sich aber, dass die innere Orientierung durch die Fokussierung stark beeinflusst wird.

Tabelle 6.1.: Korrelationsmatrix der Näherungswerte

	Fokus [dpt]	c	x'_0	y'_0	k_1	k_2	k_3
Fokus [dpt]	1,000	0,901	0,032	-0,128	-0,069	-0,181	0,177
c		1,000	-0,010	-0,211	-0,220	-0,071	0,092
x'_0			1,000	-0,230	0,005	0,005	0,005
y'_0				1,000	0,031	0,023	-0,051
k_1					1,000	-0,925	0,866
k_2						1,000	-0,985
k_3							1,000

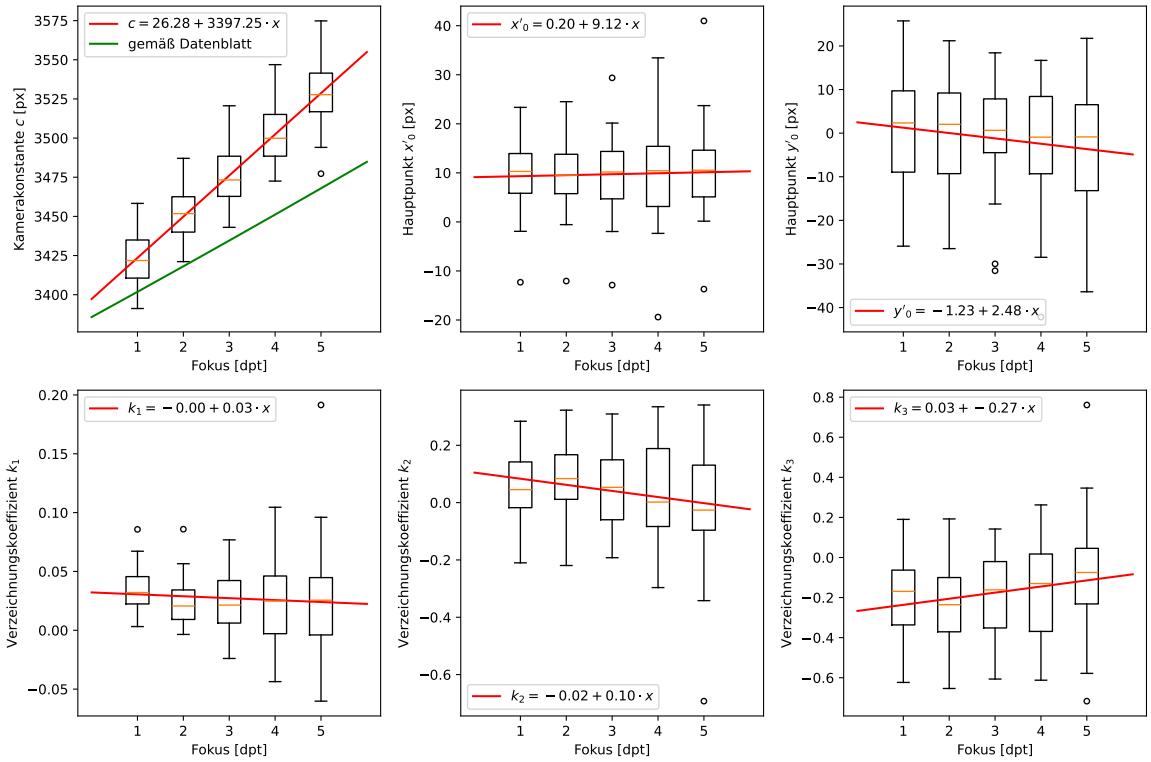


Abbildung 6.1.: Box-Whisker-Plots und ausgleichende Gerade der inneren Orientierung in Abhängigkeit von der Fokussierung [dpt]

7. Untersuchungen zur Genauigkeit und Systemaufbau

Nach Abschluss der Konstruktion und des Aufbaus des Prototyps, wurden verschiedene Untersuchungen durchgeführt, um die Genauigkeit des Systems zu überprüfen und die Anzahl der Kameras zu evaluieren. Hierzu wurden unter anderem Vergleichsmessungen mit verschiedenen Systemen durchgeführt.

7.1. Referenzdaten

Zum Vergleich standen verschiedene Testobjekte zur Verfügung. Hauptsächlich wurde ein etwa 14 cm hohes Modell einer Moai-Statue der Osterinsel verwendet, da dieses Objekt eine gute Textur und viele Details aufweist (siehe Abbildung 7.1a). Neben einem texturierten Spielzeugdino standen noch zwei weiße Gipsmodelle zur Verfügung: Ein Abdruck einer Büste Einsteins (siehe Abbildung 7.1b, 15 cm) und ein kerzenartiges Testobjekt, im folgenden Testy genannt (siehe Abbildung 7.1c, 38 cm). Alle Objekte wurden bereits in anderen Projekten an der HafenCity Universität verwendet (vgl. z. B. Kersten et al., 2016a) und sind mit mehreren verschiedenen Systemen vermessen worden.

Als Referenzwerte wurden in dieser Arbeit die Messungen mit einem Zeiss GOM ATOS 5 genutzt. Hierbei handelt es sich um ein Streifenprojektionssystem, welches hauptsächlich in der Industrie zur Vermessung von Bauteilen eingesetzt wird. Streifenprojektionssysteme arbeiten ebenfalls photogrammetrisch, haben aber den Vorteil, das sie durch das Projektionssystem auch texturarme Objekte erfassen können. Wie der Name bereits andeutet, wird ein Streifenmuster auf das Objekt projiziert, welches dann von mindestens einer Kamera aufgenommen wird. Projektor und Kamera sind auf einer festen Basis montiert. (vgl. Luhmann, 2023, S. 581f)

(a) Moai (b) Einstein (c) Testy

Abbildung 7.1.: Verschiedene Testobjekte

Bei dem verwendeten System werden zwei Kameras eingesetzt, die sich ein massives Gehäuse mit dem mittig angeordneten Projektor teilen. Je nach verwendeten Messvolumen und Objektentfernung beträgt die Genauigkeit etwa 0,01 - 0,03 mm (GDV-Systems GmbH, 2020). Aufgrund der deutlich höheren erwarteten Genauigkeit des Streifenprojektionssystems, kann davon ausgegangen werden, dass die Messungen deutlich genauer sind als die des Prototyps und als quasi-wahre Werte angenommen werden können.

Ausnahmen hiervon stellen die Positionen der Verknüpfungspunkte durch kreisförmige Marker dar. Diese wurden zur Messung mit dem ATOS 5 angebracht, um die verschiedenen Aufnahmen zu verknüpfen. Sie werden vom Streifenprojektionssystem automatisch erkannt, für die Verknüpfung genutzt und die Bereiche im erzeugten 3D-Modell interpoliert. Dadurch können hier größere Abweichungen entstehen, sofern die Oberfläche nicht eben ist oder die Dicke der Passpunktmarken nicht korrekt in der Software des Streifenprojektionssystems angegeben ist.

Die Moai-Figur wurde auch bereits mit einer Spiegelreflexkamera Nikon D90 vermessen. Hierbei wurden manuell Bilder aufgenommen, diese mit Agisoft Metashape ausgewertet und ein klassisches photogrammetrisches Modell erzeugt.

7.2. Vorgehen zur Genauigkeitsüberprüfung

Das Vorgehen zur Bewertung der erzeugten Daten unterscheidet sich zwischen den verschiedenen Untersuchungen kaum. Dieser Abschnitt beschreibt daher allgemein das Vorgehen und die verwendeten Parameter zur Bewertung der Punktwolken. Außerdem wird eine Abschätzung der erwarteten Genauigkeit des Systems vorgenommen.

7.2.1. Erwartete Genauigkeit

Die erwartete Genauigkeit kann auf verschiedenen Wegen berechnet werden. Grundlage ist meistens der Bildmaßstab. Dieser unterscheidet sich je nach Entfernung. Die Entfernung zum Objekt beträgt im Normalfall zwischen 10 und 50 cm. Der Bildmaßstab m berechnet sich nach Luhmann (2023, S. 171) wie folgt:

$$
\begin{aligned}
m &= \frac{h}{c} \\
m_{min} &= \frac{100\,\text{mm}}{4{,}7\,\text{mm}} = 21{,}28 \\
m_{max} &= \frac{500\,\text{mm}}{4{,}7\,\text{mm}} = 106{,}38
\end{aligned}
\tag{7.1}
$$

mit m : Bildmaßstab

h : Abstand zur Kamera

c : Kamerakonstante (4,7 mm)

Für die Bildmessgenauigkeit werden verschiedene Werte in der Literatur erwähnt, sie liegen je nach Messmethode zwischen 0,05 und 3 px. Für Messung von CCCTs wird zum Beispiel von Soot et al. (2015) eine Genauigkeit von 0,1 px angegeben. Eigene, manuelle Messungen ergaben eine Genauigkeit von etwa 1,5 px. Vereinfacht wurde mit 1 px Genauigkeit gerechnet.

Die Bildmessgenauigkeit dx' wird dann mit dem Bildmaßstab multipliziert und ergibt die vorläufige Lagegenauigkeit dX (Gleichung 7.2):

$$\begin{aligned} dx' &= 1\,\text{px} \cdot 0{,}0014\,\text{mm/px} = 0{,}0014\,\text{mm} \\ dX &= m \cdot dx' \\ dX_{min} &= 21{,}28 \cdot 0{,}0014\,\text{mm} = 0{,}03\,\text{mm} \\ dX_{max} &= 106{,}38 \cdot 0{,}0014\,\text{mm} = 0{,}15\,\text{mm} \end{aligned} \tag{7.2}$$

mit dx' : Bildmessgenauigkeit

dX : Lage-Genauigkeit (Objektraum)

An diesem Wert muss dann noch der sogenannte Design-Faktor q angebracht werden. Luhmann (2023, S. 174) beschreibt diesen als Parameter der Aufnahmegeometrie und den auftretenden Schnitten. Dieser liege bei Rundumverbänden zwischen 0,4 und 0,8 und bei Stereoaufnahmen bei $1{,}5 - 3{,}0$. Daher wird hier vereinfacht für eine grobe Abschätzung der Faktor weggelassen. Die erwartete Genauigkeit liegt damit deutlich unter einem Millimeter in der Lage.

$$\begin{aligned} s'_{px;min} &= dX_{min} \\ s'_{px;max} &= dX_{max} \end{aligned} \tag{7.3}$$

Die Genauigkeit der Tiefeninformationen ist von dem Verhältnis des Abstandes der Kameras zur Entfernung abhängig. Zwischen zwei Kamerareihen sind etwa 25 cm Abstand. Die Genauigkeit der Tiefeninformationen kann daher wie folgt berechnet werden (Luhmann, 2023, S. 174):

$$\begin{aligned} s_Z &= m \frac{h}{b} s'_{px} \\ s_{Z;min} &= 21{,}28 \cdot \frac{10\,\text{cm}}{30\,\text{cm}} \cdot 0{,}03\,\text{mm} = 0{,}02\,\text{mm} \\ s_{Z;max} &= 106{,}38 \cdot \frac{50\,\text{cm}}{30\,\text{cm}} \cdot 0{,}15\,\text{mm} = 0{,}25\,\text{mm} \end{aligned} \tag{7.4}$$

mit s_Z : Genauigkeit der Tiefeninformationen

b : Basislinie (hier 30 cm)

Die erwartete Genauigkeit und auch die Auflösung der Kameras liegen demnach im Bereich eines fünftel Millimeters.

7.2.2. Verwendete Parameter

Zum Vergleich der Ergebnisse wurden verschiedene Parameter in den Auswertungen betrachtet. Diese sind im Folgenden kurz erläutert.

Quasi-wahrer Wert/Fehler

Die Messung mit dem Streifenprojektionssystem ist nach der Genauigkeitsabschätzung des Prototyps etwa zehnmal genauer. Daher kann das Ergebnis des Streifenprojektionssystems hier als quasi-wahrer Wert angenommen werden (vgl. Höpcke, 1980, S. 43). Der quasi-wahre Fehler ist nach Höpcke (1980, S. 44, Formel 2-2, siehe Gleichung 7.5) die Differenz zwischen quasi-wahren Wert und der Beobachtung. Im Fall dieser dreidimensionalen Messung wurde der Abstand zwischen einem Punkt der Punktwolke des Prototyps und der Oberfläche aus der Messung des Streifenprojektionssystems als quasi-wahrer Fehler angenommen. Punkte innerhalb des Objektes wurden mit einem negativen Vorzeichen versehen.

$$l + \varepsilon' = \lambda' \tag{7.5}$$

mit l : Beobachtung

ε' : quasi-wahrer Fehler

λ' : quasi-wahrer Wert

Durchschnittlicher Fehler / Mittelwert des quasi-wahren Fehlers

Der durchschnittliche Fehler ist der Mittelwert des Betrages des quasi-wahren Fehlers (Höpcke, 1980, S. 44, Gleichung 2-4, siehe Gleichung 7.6). Er gibt an, wie stark die Punktwolke von der Referenz abweicht. Der Mittelwert des quasi-wahren Fehlers (Gleichung 7.7) ist eher unüblich, da sich hier positive und negative Abweichungen aufheben. Da das Vorzeichen mitberücksichtigt wird, zeigt dieser, ob die Punktwolke systematisch zu groß oder zu klein ist, also ob die Realisierung des Maßstabes ungenau war.

$$t = \frac{1}{n} \sum_{i=1}^{n} |\varepsilon'_i| \tag{7.6}$$

$$\overline{\varepsilon'} = \frac{1}{n} \sum_{i=1}^{n} \varepsilon'_i \tag{7.7}$$

Mittlerer Fehler

Der mittlere Fehler ergibt sich aus der Quadratsumme des quasi-wahren Fehlers, geteilt durch die Anzahl der Messungen (Höpcke, 1980, S. 45, Gleichung 2-5b, siehe Gleichung 7.8). Er ist das wichtigste Maß für die Genauigkeit der Punktwolke.

$$m = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \varepsilon_i^2} \tag{7.8}$$

Maximaler Fehler

Der maximale Fehler gibt den größten Fehler an, der in der Punktwolke auftritt. Er zeigt, wie stark die Punktwolke von der Referenz abweicht und ob es Ausreißer gibt.

Punktanzahl

Die Anzahl der Punkte einer Punktwolke alleine ist kein Maß für die Qualität der Punktwolke. Sie kann zwar als Indikator für die Abdeckung der Oberfläche genutzt werden, jedoch nicht für die Genauigkeit. Die Anzahl der Punkte wurde daher nur als zusätzlicher Parameter betrachtet bzw. aufgrund der Einfachheit der Bestimmung als Hilfsparameter genutzt, welcher durch die beiden folgenden Parameter ergänzt wird.

Abdeckung

Als weiteres Kriterium wurde die Abdeckung der Objektoberfläche mit Punkten bewertet. Dazu wurden die Punktwolken auf diejenigen gefiltert, die weniger als einen Millimeter vom Modell des Streifenprojektionssystems entfernt waren. Diese wurden dann auf eine Punktdichte von 1 mm Punktabstand reduziert und die Anzahl der Punkte gezählt. Die Anzahl dieser Punkte wurde mit der genauso bestimmten Anzahl des Referenzdatensatzes in Beziehung gesetzt. Dadurch konnte die Abdeckung der Oberfläche bewertet werden. Eine Abdeckung von 100 % bedeutet, dass beide ausgedünnten Punktwolken die gleiche Anzahl an Punkten aufweisen und der Datensatz

die Oberfläche des Referenzdatensatzes ähnlich gut abdeckt. Kleinere Werte zeigen an, dass die Oberfläche nicht vollständig erfasst wurde. Durch den Ansatz kann auch eine Abdeckung von mehr als 100 % erreicht werden, wenn die Punktwolke mehr Punkte aufweist, als die Referenzdaten.

Richtigkeit

Darüber hinaus wurde die Anzahl der Punkte nach der Ausdünnung bestimmt ohne Nutzung des zuvor beschriebenen Abstandsfilter. Dieser wurde dann mit der Anzahl der Punkte unter Nutzung des Abstandsfilters in Beziehung gesetzt. Dadurch konnte der Anteil der richtig liegenden Punkte unabhängig von der Anzahl der Punkte bewertet werden. Auch hier bedeutet ein Wert von 100 %, dass alle Punkte richtig bzw. maximal 1 mm von der Oberfläche der Referenzdaten entfernt liegen. Kleinere Werte zeigen an, dass die Punktwolke Ausreißer enthält.

7.3. Genauigkeitsüberprüfung des 3D-Modells

Um die Genauigkeit der 3D-Modell-Erzeugung zu überprüfen, wurden mehrere Prüfkörper mit dem Prototyp vermessen. Die Ergebnisse wurden anschließend im Vergleich zu dem Streifenprojektionssystem und der Aufnahme des Moais mit einer Spiegelreflexkamera betrachtet.

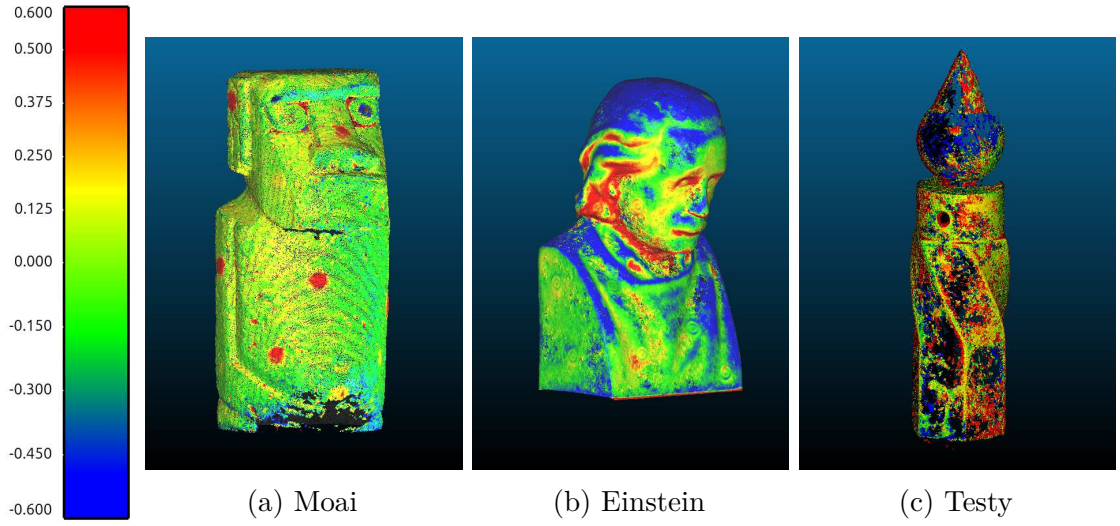
7.3.1. Durchführung

Alle drei Testobjekte wurden mit dem Prototyp aufgenommen und die Daten in Agisoft Metashape verarbeitet. Die modellierte Oberfläche aus dem Streifenprojektionssystem und die Punktwolke aus dem Prototyp wurden in CloudCompare aufeinandergelegt. Hieraus wurde anschließend die Differenz und die Genauigkeit berechnet.

7.3.2. Ergebnisse

Die Visualisierung der Differenzen ist in Abbildung 7.2 dargestellt. Es ist zu erkennen, dass je nach Modell einige Bereiche gar nicht erfasst bzw. deren Genauigkeit deutlich von der erwarteten abweichen.

Moai Beim Moai (siehe Abbildung 7.2a) sind die größten Fehlstellen im Bereich des Kinns und unterhalb des Bauchs zu erkennen. Abweichungen im Bereich des restlichen Körpers sind deutlich geringer und liegen im Bereich von 0,21 mm. Sie sind auf die schlechtere Erfassung der nach unten gerichteten Flächen zurückzuführen. Diese Bildbereiche sind nur in wenigen Bildern sichtbar und können daher nicht so gut rekonstruiert werden. Weitere Abweichungen sind im Bereich der Passpunktmarken zu erkennen - hier ist unklar, welche Daten korrekt sind (vgl. Abschnitt 7.1).



(a) Moai (b) Einstein (c) Testy

Abbildung 7.2.: Differenzbilder verschiedener 3D-Modelle

Einstein Das Modell von Einstein (siehe Abbildung 7.2b) zeigt ebenfalls größere Abweichungen im Bereich des Kinnes und des Halses. Die Passpunktmarken sind zwar auch in der Differenzdarstellung erkennbar, jedoch nicht mit so großen Abweichungen wie beim Moai. Im Gegensatz zum Moai sind die Abweichungen im Bereich des Kopfes und der Schultern deutlich großflächiger. Gerade die großen weißen, texturarmen Flächen führen zu schlechteren Ergebnissen.

Testy Das Modell des Testy (siehe Abbildung 7.2c) zeigt ebenfalls die Problematik der weißen, texturarmen Flächen. Der Körper weist im Gegensatz zur Einstein-Büste keine Verschmutzungen auf, sodass auch hier keine zusätzliche Textur entsteht. Da dieser Testkörper allseitig ähnlich aussieht, ist die Zuordnung der Punkte schwierig. Die Abweichungen sind daher deutlich größer, als bei den anderen Modellen. Ein weiteres Problem ist die Größe des Körpers, der mit einer Höhe von 38 cm fast das Maximum dessen darstellt, was mit dem System erfasst werden kann.

Die Abweichungen des Moai und der Einstein-Büste liegen in der Größenordnung der in Unterabschnitt 7.2.1 berechneten Werte und sind nur minimal schlechter, als die Genauigkeit eines 3D-Modells des Moai, welches photogrammetrisch mit einer Spiegelreflexkamera Nikon D90 manuell aufgenommen wurde. Durch Kombinationen von anderen Perspektiven bei der manuellen Aufnahme ist hier die Abdeckung subjektiv betrachtet besser. Die Ergebnisse sind in Tabelle 7.1 zusammengefasst.

Tabelle 7.1.: Ergebnisse der Genauigkeitsüberprüfung

	Durchschnittlicher Fehler	Mittlerer Fehler	Maximaler Fehler
Moai	−0,024 mm	0,159 mm	2,57 mm
...ohne Bereiche der Marker	−0,038 mm	0,140 mm	2,59 mm
...manuell mit Nikon D90	0,255 mm	0,154 mm	1,33 mm
Einstein	0,001 mm	0,356 mm	−3,64 mm
Testy	−0,047 mm	0,644 mm	11,56 mm

7.4. Nutzung eines Drehtellers

Statt die reelle Zahl der Kameras zu erhöhen, kann auch ein Drehteller genutzt werden, sodass jede Kamera mehr als ein Bild zur Erfassung des Objektes liefert.

7.4.1. Durchführung

Hierzu wurde ein einfacher manueller Drehteller genutzt (IKEA SNUDDA). Dieser wurde mit Passpunkten beklebt und weist ansonsten eine texturreiche Naturholzoberfläche auf (siehe Abbildung 7.3). Die Daten wurden in Metashape weiterverarbeitet und die einzelnen Aufnahmeschritte über die Passpunkte und die Punktwolke zusammengerechnet. Die Ergebnisse wurden mit denen des Streifenprojektionssystems und den Aufnahmen ohne Drehteller verglichen.

Abbildung 7.3.: Drehteller mit Moai-Figur im Prototypen (aufgenommen von einem der verbauten Raspberry Camera Module 3)

7.4.2. Ergebnisse

In Abbildung 7.4 ist das Differenzbild des Moai zu sehen. Es zeigt sich, dass die Genauigkeit des Modells mit Drehteller deutlich besser ist, als die des Modells ohne Drehteller (siehe Abbildung 7.2a). Auch ist die visuelle Vollständigkeit deutlich höher, die nach unten gerichteten Flächen sind ebenfalls erfasst und passen gut zu dem Modell des Streifenprojektionssystems. Die Anzahl der Punkte hat sich nicht relevant verändert – im Vergleich zu dem vollständigen Modell der Spiegelreflexkamera zeigt sich, dass über die Punktanzahl selbst keine sinnvollen Aussagen zur Abdeckung getroffen werden können.

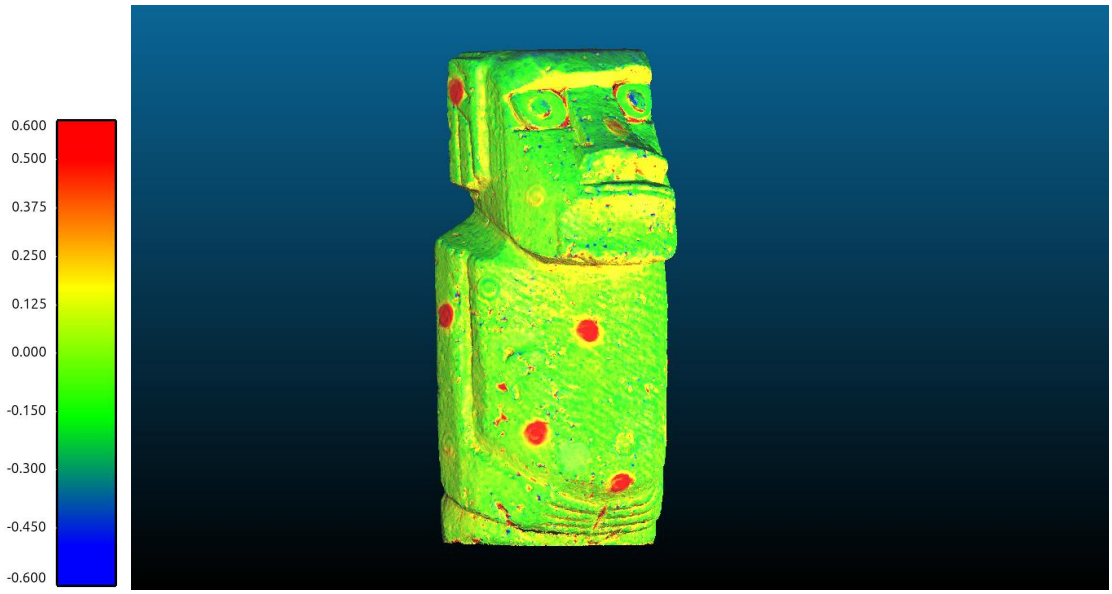


Abbildung 7.4.: Differenzbild des Moai

Der geringe durchschnittliche Fehler ist auf die automatische Anpassung des Maßstabes zurückzuführen. Die Ergebnisse sind in Tabelle 7.2 zusammengefasst.

Tabelle 7.2.: Ergebnisse der unter Anpassung des Maßstabes durchgeführten Genauigkeitsüberprüfung

	Durchschnittlicher Fehler	Mittlerer Fehler	Maximaler Fehler	Punktanzahl
ohne Drehteller	−0,03 mm	0,22 mm	2,0 mm	3,3 Mio.
mit Drehteller	−0,03 mm	0,15 mm	1,6 mm	3,4 Mio.
Nikon D90	0,03 mm	0,17 mm	1,2 mm	0,05 Mio.

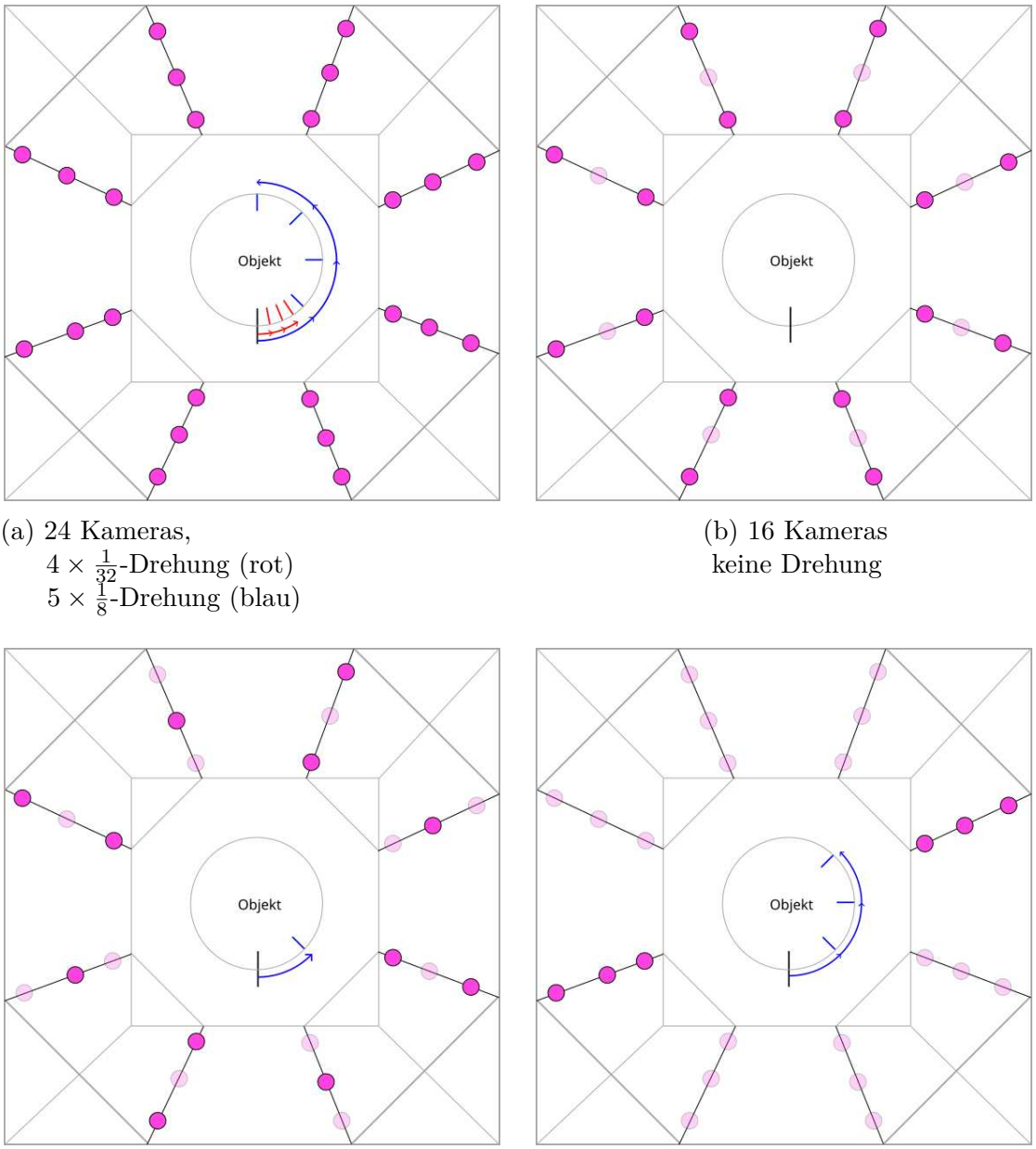
7.5. Evaluation der Kameraanzahl

Die Kameras machen einen großen Teil des (Kosten-)Aufwands zur Realisierung des Projektes aus. Es soll daher in diesem Schritt geprüft werden, ob die Anzahl der Kameras auch reduziert werden kann und die mindestens für die Testobjekte notwendige Anzahl bestimmt werden. Entsprechend Abschnitt 7.4 wird auch der Einsatz des Drehtellers statt mehr Kameras geprüft.

7.5.1. Durchführung

Um abzuschätzen, wie viele Kameras für eine gute Genauigkeit notwendig sind, wurden verschiedene Konfigurationen getestet. Dazu wurde der Moai aus Abschnitt 7.3 verwendet und in Agisoft Metashape verschiedene Kameras deaktiviert und das 3D-Modell neu berechnet. Die Ergebnisse wurden wieder mit dem Streifenprojektionssystem ATOS 5 verglichen. Als weitere Parameter wurden die Abdeckung und die Richtigkeit der Punktwolke bestimmt. Die Verteilung der jeweils genutzten Kameras ist in Abbildung 7.6 dargestellt. Die pink dargestellten Punkte stellen jeweils die aktiven Kameras dar.

Als Ausgleich für fehlende Kameras wurde auch nochmal der Drehteller aus der vorherigen Untersuchung verwendet und geprüft, wie viele Kameras durch den Drehteller ersetzt werden können. Die verwendeten Drehungen sind der Tabelle und den Bildern jeweils in der Form [Anzahl Aufnahmen]×[Betrag der Drehung] zu entnehmen. Die Drehungen sind zusätzlich auch in Abbildung 7.5 dargestellt. Jeder Strich steht für eine Aufnahme mit den aktiven Kameras. Beispielsweise zeigt die Abbildung Abbildung 7.5c zwei Striche auf dem angedeuteten Drehteller. Diese entsprechen zwei Aufnahmen mit einer Drehung zwischen den beiden Aufnahmen von $\frac{1}{8}$, also hier 45°.



(a) 24 Kameras,
$4 \times \frac{1}{32}$-Drehung (rot)
$5 \times \frac{1}{8}$-Drehung (blau)

(b) 16 Kameras
keine Drehung

(c) 12 Kameras,
$2 \times \frac{1}{8}$-Drehung (blau)

(d) 6 Kameras,
$4 \times \frac{1}{8}$-Drehung (blau)

Abbildung 7.5.: Positionierung der Kameras und verwendete Positionen des Drehtellers, Blick von oben auf das System

Zur Bestimmung des Maßstabes wurden die Daten automatisiert bestmöglich an die Referenzdaten angepasst. Die Passpunkte wurden daher hier nur als Verknüpfungspunkte genutzt und nicht zur Bestimmung des Maßstabes.

7.5.2. Ergebnisse

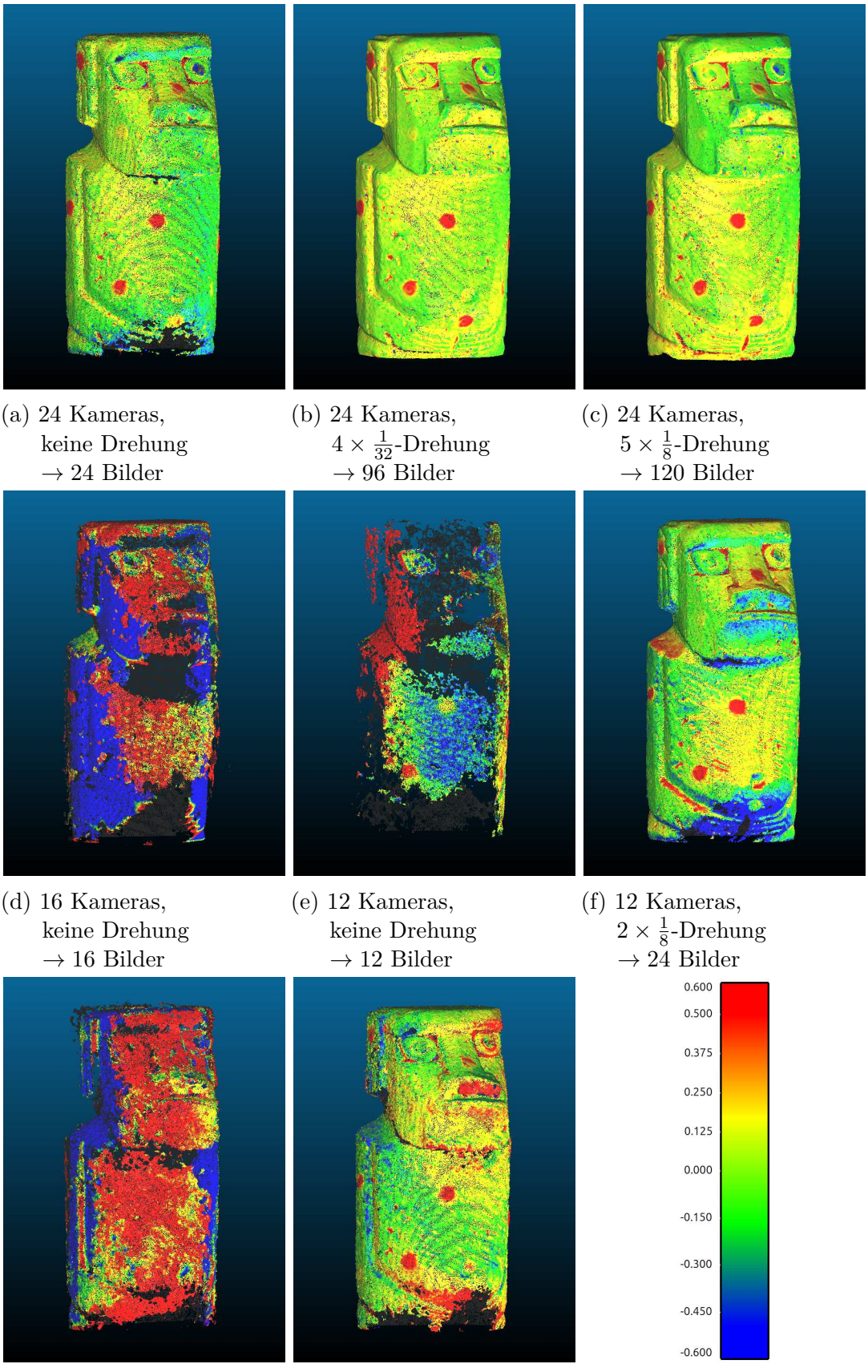
Wie zu erwarten nahm die Abdeckung und die Qualität der Punktwolke ab, je weniger Kameras verwendet wurden. Bei einer Aufnahme mit allen 24 Kameras wurde die Oberfläche (ohne Boden) zu 93 % abgedeckt (siehe Abbildung 7.6a). Bei einer Aufnahme mit 18 Kameras (siehe Abbildung 7.6d) waren es nur noch 68 % und bei 12 Kameras (siehe Abbildung 7.6e) noch 54 %. Schon bei 18 Kameras war der Moai kaum mehr zu erkennen, bei 12 war die Erkennbarkeit nicht mehr gegeben. Die Anzahl der Kameras scheint also für dieses Objekt angebracht zu sein, sofern kein Drehteller genutzt wird.

Unter Nutzung des Drehtellers mit einer Drehung um eine Achtel-Drehung konnten sogar mit nur der Hälfte der Kameras (siehe Abbildung 7.6f) ein ähnliches bzw. sogar minimal besseres Ergebnis erzielt werden, als bei der einfachen Aufnahme mit 24 Kameras. Bei der Verwendung von nur 6 Kameras in einer Ebene, so wie es eine Option bei der Planung des Rahmens war, brachte auch die Verwendung des Drehtellers mit 4 Positionen kein brauchbares Ergebnis im direkten Anlauf (siehe Abbildung 7.6g). Erst durch manuelles mehrfaches Verarbeiten der Daten und die Nutzung weiterer Passpunkte konnte die Genauigkeit und Abdeckung auf ein ähnliches Niveau wie bei 12 Kameras und zwei Aufnahmen gebracht werden (siehe Abbildung 7.6h) - dafür aber mit einem deutlich höheren personellen und technischen Aufwand.

Im Vergleich des Parameters der Richtigkeit zeigt sich ein ähnliches Bild: Auch hier schneiden die Aufnahmen mit wenig Bildern und ohne Drehteller deutlich schlechter ab. Der Parameter zeigt, dass zwar die Punktwolken immer ähnlich viele Punkte haben, jedoch der Großteil davon außerhalb der 1-mm-Genauigkeit liegt. Die Aufnahmen mit Drehteller schneiden hier deutlich besser ab, als die ohne, erreichen aber nicht die Qualität der Aufnahmen mit 24 Kameras, vor allem nicht der Aufnahmen mit 24 Kameras und Nutzung des Drehtellers.

Die Ergebnisse sind in Tabelle 7.3 zusammengefasst. Zum Vergleich sind auch die Ergebnisse der Nutzung des Drehtellers mit 24 Kameras aufgeführt - einmal mit feinen Drehungen von $\frac{1}{32}$ (siehe Abbildung 7.6b) und einmal mit groben Drehungen von $\frac{1}{8}$ (siehe Abbildung 7.6c). Abbildung 7.6 zeigt die Abweichungen der jeweiligen Punktwolken von den Referenzdaten in grafischer Form.

[1]Details hierzu in Unterunterabschnitt 7.2.2



(a) 24 Kameras, keine Drehung → 24 Bilder

(b) 24 Kameras, $4 \times \frac{1}{32}$-Drehung → 96 Bilder

(c) 24 Kameras, $5 \times \frac{1}{8}$-Drehung → 120 Bilder

(d) 16 Kameras, keine Drehung → 16 Bilder

(e) 12 Kameras, keine Drehung → 12 Bilder

(f) 12 Kameras, $2 \times \frac{1}{8}$-Drehung → 24 Bilder

(g) 6 Kameras, $4 \times \frac{1}{8}$-Drehung → 24 Bilder

(h) 6 Kameras, $4 \times \frac{1}{8}$-Drehung → 24 Bilder + Marker

(i) Skala

Abbildung 7.6.: Abweichungen der jeweiligen Punktwolken von den Referenzdaten

Tabelle 7.3.: Ergebnisse mit unterschiedlicher Kameraanzahl und teilweise Nutzung eines Drehtellers

Art	Kamera-anzahl	Drehungen	Bilder	Mittlerer Fehler	Maximaler Fehler	Punktanzahl	Abdeckung	Richtigkeit
ATOS 5						346830	100,0 %	100,0 %
Automatik	24	1	24	0,18 mm	2,52 mm	628727	93,1 %	99,9 %
Feinschritt	24	4 x $\frac{1}{32}$	96	0,14 mm	1,42 mm	758364	100,2 %[1]	100,0 %
Grobschritt	24	5 x $\frac{1}{8}$	120	0,16 mm	1,42 mm	777533	98,8 %	100,0 %
2 von 3	16	1	16	1,20 mm	7,74 mm	596979	68,4 %	70,4 %
jede zweite K.	12	1	12	0,49 mm	4,01 mm	346830	54,9 %	94,1 %
... mit Drehung	12	2 x $\frac{1}{8}$	24	0,25 mm	1,44 mm	760538	95,7 %	99,8 %
eine Ebene	6	4 x $\frac{1}{8}$	24	1,57 mm	1,44 mm	753062	77,8 %	68,6 %
... mit Markern	6	5 x $\frac{1}{8}$	30	0,29 mm	2,50 mm	640810	94,2 %	98,9 %

7.6. Zusammenfassung

Das System erreichte die vorgesehene Genauigkeit - sie lag je nach Testobjekt bei 0,1 mm - 0,5 mm. Die Anzahl von 24 Kameras war für die meisten getesteten Objekte ausreichend. Experimente mit weniger Kameras führten schnell zu einer schwachen Abdeckung des Objektes und somit zu fehlenden Bereichen in der Punktwolke. Der Einsatz eines Drehtellers kann die Anzahl der notwendigen Kameras reduzieren, jedoch ist die Verarbeitung der Daten deutlich aufwändiger. Der Einsatz des Drehtellers zur weiteren Genauigkeitssteigerung ist aber durchaus sinnvoll, wenn die Anzahl der Kameras nicht erhöht werden kann oder soll.

8. Fazit und Ausblick

In dieser Arbeit wurde die Entwicklung und Implementierung eines Systems zur Erfassung und Erstellung von 3D-Modellen kleiner Objekte untersucht. Ausgangspunkt war die Frage, wie man mit geringem finanziellen und personellen Aufwand hochwertige 3D-Modelle erstellen kann, die für den Einsatz in Museen, Archiven oder Bildungseinrichtungen geeignet sind.

Es hat sich gezeigt, dass es möglich ist ein photogrammetrisches Messsystem auf Basis von Raspberry-Pi-Kameras zu entwickeln. Die Kameras sind in der Lage, Bilder in hoher Qualität aufzunehmen und diese an eine zentrale Steuereinheit zu übertragen. Die Steuereinheit kann die Kameras synchronisieren und die Bilder an eine SfM-Software übertragen. Durch die 24 Kameras ist eine schnelle Erfassung möglich, ohne das zum Beispiel ein Drehteller notwendig wird. Untersuchungen der Genauigkeit der 3D-Modelle zeigten, dass diese bis zu 0,1 mm genau sein können und damit im Bereich von manuellen photogrammetrischen Aufnahmen mit einer hochwertigen Amateurkamera liegen. Die Kosten des Systems belaufen sich auf ca. 1900 Euro, was im Vergleich zu professionellen Systemen sehr günstig ist und nur der Anschaffung einer entsprechenden Kamera zur manuellen Aufnahme entspricht.

Die Arbeit zeigt wie auch andere Arbeiten in diesem Themengebiet das Potenzial der Photogrammetrie zur automatischen Erzeugung von 3D-Modellen. Gerade durch die stetige Weiterentwicklung der Auswertesoftware und der Hardware wird es in Zukunft möglich sein, noch genauere und schnellere 3D-Modelle zu erstellen. Dadurch werden sich sehr wahrscheinlich noch weitere Anwendungsfelder für die Photogrammetrie ergeben. Projekte wie KulturGutRetter (vgl. Deutsches Archäologisches Institut, 2024) nutzen bereits ähnliche Ansätze zur schnellen fotografischen Dokumentation von mobilen Kulturgütern. Hier wäre es nur noch ein kleiner Schritt, statt eines einzelnen Bildes gleich Daten für die dreidimensionale Rekonstruktion zu erfassen.

Der in dieser Arbeit entwickelte Prototyp zeigt zwar das Potenzial der Photogrammetrie mit einem solchen Kamera- und Passpunktrahmen, jedoch gibt es noch einige Optimierungsmöglichkeiten. Im aktuellen Entwicklungsstand ist das System aber schon gut für Ausbildungszwecke geeignet, da es die Funktionsweise der Photogrammetrie gut veranschaulicht und viele Eingriffsmöglichkeiten bietet. Durch die Verwendung von Python sollte eine einfache Anpassung des Systems an spezielle Anforderungen möglich sein.

Versuche mit einem Drehteller haben gezeigt, dass gerade komplexere Objekte von ein oder zwei zusätzlichen Aufnahmen stark profitieren. Hier könnte die Vorausgleichung der Kameras so angepasst werden, dass nach einer Drehung die vorberechneten Koordinaten der Kameras sich auf den Drehteller beziehen, sich also mit dem Teller drehen. Ein weiterer Schritt in diese Richtung wäre die Nutzung eines motorisierten Drehtellers, der die Drehung automatisch durchführt. Im einfachsten Fall würde die Drehung ein an den Raspberry Pi angeschlossener Schrittmotor übernehmen. Der genaue Betrag der Drehung könnte dann über kodierte Passpunkte photogrammetrisch bestimmt werden.

Die Verwendung von WLAN zur Datenübertragung hat sich als nicht optimal herausgestellt. Die Übertragungsgeschwindigkeit ist bei dieser Anzahl an Teilnehmern im Netzwerk gering, sodass die Übertragung der Bilder lange dauert. Eine Übertragung über LAN wäre hier effektiver, jedoch wäre der Verdrahtungsaufwand deutlich höher und die Mobilität des Systems eingeschränkt. Da die Raspberry Pi Zero W über keinen RJ45-Anschluss verfügt, müsste ein zusätzlicher Adapter verwendet werden.

Die Verwendung des Raspberry Pi 4 als Schnittstelle vereinfacht zwar die Bedienung und ermöglicht die Verwendung des Systems ohne externe Hardware, jedoch ist die Leistung des Raspberry Pi 4 für die Verarbeitung der Bilder zum 3D-Modell nicht ausreichend. Bei der Verwendung eines externen Rechners könnte hier eine Desktop-Software alle Aufgaben übernehmen, die dieser ausführt. Neben der Kostenersparnis würde dies auch die räumliche Skalierbarkeit des Systems erhöhen und die Komplexität verringern. Eine andere Variante hiervon wäre die direkte Kommunikation des Systems mit einem Server mit OpenDroneMap, sodass die Bilder direkt auf dem Server verarbeitet werden. Hier wäre dann kein leistungsstarker Rechner vor Ort notwendig, dafür jedoch eine stabile Internetverbindung.

Ein weiterer Punkt ist die Verwendung des Module 3 Wide statt des klassischen Module 3. Dieses hat einen geringeren Mindestabstand zum Objekt und könnte so bei gleichem Montagerahmen größere Objekte erfassen. Nachteilig wäre hierbei jedoch, dass ein noch größerer Schärfentiefebereich notwendig wäre - also dann auch Fokusstacking notwendig wäre. Zu Beginn des Projekts war die Verfügbarkeit des Module 3 Wide noch nicht gegeben, sodass das Module 3 verwendet wurde. Außerdem hatte die Nutzung von Fokusstacking bisher nicht zu den gewünschten Ergebnissen geführt.

Zur besseren Erfassung von Objekten mit einheitlicher Textur könnte die Verwendung eines LED-Beamers mit Musterprojektion helfen. Hierdurch wäre es mit der Software möglich, auch Punkte auf der Oberfläche dieser Objekte zu bestimmen. Die Steuerung des Beamers könnte über den Raspberry Pi 4 erfolgen.

Das System bietet an einigen Stellen noch Optimierungsmöglichkeiten, die beispielsweise zur besseren Erfassung von größeren oder texturarmen Objekten führen können. Insgesamt zeigt sich, dass die Nutzung der Raspberry-Pi-Kameramodule an einem festen Rahmen schon in dieser Entwicklungsstufe gute Ergebnisse liefert, bei sehr kurzem personellen Aufwand bei der Erfassung.

Glossar

Bildhauptpunkt Punkt, an dem die optische Achse die Bildebene schneidet. 3, 6, 8–10, 52, 72

Bildweite Abstand zwischen der Bildebene und dem Hauptpunkt der Linse, entspricht der Kamerakonstante, bei Fokussierung auf unendlich ist die Bildweite gleich der Brennweite. 32

Brennweite Abstand zwischen dem Hauptpunkt der Linse und dem Brennpunkt. 32, 33, 72

Homografie projektive Abbildung einer Ebene auf eine andere, z. B. in zwei Bildern, siehe Unterabschnitt 2.5.3. 36

innere Orientierung Kalibrierung der Kamera, beinhaltet die Kamerakonstante, Lage des Bildhauptpunkt und der Verzeichnungskoeffizienten, siehe Unterabschnitt 2.1.1. 6, 7, 12, 14, 16–18, 22, 40, 51, 52

Kamerakonstante auch Kammerkonstante, photogrammetrische Begriff für die Bildweite. 6, 8, 9, 32–34, 52, 56, 72, 78

Socket Kommunikationsendpunkt, ermöglicht die Kommunikation zwischen zwei Prozessen. 44–46, 48

Verzeichnung Verzeichnung ist die Abweichung der Abbildung eines Objektes von der idealen Abbildung. Es gibt verschiedene Arten von Verzeichnungen, siehe Unterabschnitt 2.1.1 . 6, 7, 9, 10, 31, 34, 35, 52

ähnlich Zwei Objekte sind ähnlich, wenn sie sich in Größe und Ausrichtung unterscheiden, sich aber durch Drehung, Skalierung und Translation ineinander überführen lassen. 13

äußere Orientierung Lage und Ausrichtung der Kamera im Raum, siehe Unterabschnitt 2.1.2. 6, 8, 11, 13, 14, 16, 17, 22, 40, 45

Akronyme

API Application Programming Interface. 48

CCCT concentric circular coded target. 14, 15, 56

dpt Dioptrien. 33, 34, 78

HTTP Hypertext Transfer Protocol. 73

px Pixel. 22, 31, 32, 35, 56, 57

REST Representational State Transfer, Architekturstil für verteilte Systeme, oft in Verbindung mit Webanwendungen verwendet, basiert auf dem Hypertext Transfer Protocol (HTTP)-Protokoll, Zustand wird mit übertragen, sodass dieser nicht gespeichert werden muss. 44–46, 48

SfM Structure-from-Motion. 3–5, 36, 38, 39, 41, 48, 49, 69, 78

SIFT Scale-Invariant Feature Transform, Algorithmus zur Bestimmung von Merkmalen in Bildern, die invariant gegenüber Skalierung, Rotation und Beleuchtung sind, siehe Unterabschnitt 2.4.2. 15, 36

VPE Verpackungseinheit. 94, 95

Literaturverzeichnis

Brown, Duane C. (1971): Close-range camera calibration. *Photogrammetric Engineering*, Band 37(8): S. 855–866, American Society for Photogrammetry and Remote Sensing, Bethesda, Vereinigte Staaten von Amerika, *https://www.asprs.org/wp-content/uploads/pers/1971journal/aug/1971_aug_855-866.pdf*. (letzter Aufruf: 09. Feb. 2026).

Busen, Tobias; Wedekind, Wanja (2023): KulturGutRetter – Der Umgang mit gebautem Kulturerbe in Krisensituationen. In: Ziesemer, John (Hg.), Baudenkmale in Konflikten und Katastrophen - Prävention / Intervention / Nachsorge: Internationale Tagung auf der denkmal 2022, S. 46–53, Anton H. Konrad Verlag, Weißenhorn, *https://www.icomos.de/data/pdf/icomos-katastrophen-internet-240208-0221-0934-28.pdf*. (letzter Aufruf: 09. Feb. 2026).

Bösemann, Werner (1996): The Optical Tube Measurement System OLM Photogrammetric Methods used for Industrial Automation and Process Control. In: Kraus, Karl; Waldhäusl, Peter (Hg.), XVIIIth ISPRS Congress - Technical Commission V: Close Range Techniques and Machine Vision, S. 55–58, Washington, D.C., Vereinigte Staaten von Amerika, *https://www.isprs.org/proceedings/xxxi/congress/part5/55_XXXI-part5.pdf*. (letzter Aufruf: 09. Feb. 2026).

Clini, Paolo; Frapiccini, Nicoletta; Mengoni, Maura; Nespeca, Romina; Ruggeri, Ludovico (2016): SfM technique and focus stacking for digital documentation of archaeological artifacts. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Band XLI-B5: S. 229–236, Copernicus GmbH, Göttingen, *doi:10.5194/isprsarchives-XLI-B5-229-2016*.

Deutscher Museumsbund e. V. (2022): Handreichung Digitale Grunderfassung. Berlin, *https://www.museumsbund.de/wp-content/uploads/2022/12/handreichung-digitale-grunderfassung.pdf*. (letzter Aufruf: 09. Feb. 2026).

Deutsches Archäologisches Institut (2024): KulturGutRetter. Deutsches Archäologisches Institut, Berlin, *https://www.kulturgutretter.org/projekt/*. (letzter Aufruf: 09. Feb. 2026).

Garsthagen, Richard (2021): Raspberry Pi 3D Scanner. Zoetermeer, Niederlande, *https://www.pi3dscan.com/*. (letzter Aufruf: 09. Feb. 2026).

GDV-Systems GmbH (2020): 3D-Scanning. GDV-Systems GmbH, Bad Schwartau, *https://gdv-systems.de/dienstleistungen/messsysteme/3d-scanning.html*. (letzter Aufruf: 09. Feb. 2026).

Hartley, Richard; Zisserman, Andrew (2004): Multiple View Geometry in Computer Vision. Cambridge University Press, Cambridge, Vereinigtes Königreich, *doi:10.1017/CBO9780511811685*.

Höpcke, Walter (1980): Fehlerlehre und Ausgleichsrechnung. Walter de Gruyter, Berlin, *doi:10.1515/9783110838206*.

Kazhdan, Michael; Bolitho, Matthew; Hoppe, Hugues (2006): Poisson Surface Reconstruction. In: Polthier, Konrad; Sheffer, Alla (Hg.), Eurographics Symposium on Geometry Processing, S. 61–70, The Eurographics Association, Eindhoven, Niederlande, ISBN 3-905673-24-X, *doi:10.2312/SGP/SGP06/061-070*.

Kersten, Thomas; Lindstaedt, Maren; Mechelke, Klaus; Zobel, Kay (2012): Automatische 3D-Objektrekonstruktion aus unstrukturierten digitalen Bilddaten für Anwendungen in Architektur, Denkmalpflege und Archäologie. In: Seyfert, Eckart (Hg.), 32. Wissenschaftlich-Technische Jahrestagung der DGPF, S. 137–148, Deutsche Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation e.V., Potsdam, *https://www.dgpf.de/src/pub/DGPF2012.pdf*. (letzter Aufruf: 09. Feb. 2026).

Kersten, Thomas; Przybilla, Heinz-Jürgen; Lindstaedt, Maren; Tschirschwitz, Felix; Misgaiski-Hass, Martin (2016a): Genauigkeitsuntersuchungen handgeführter Scannersysteme. In: Kersten, Thomas (Hg.), Dreiländertagung der DGPF, der OVG und der SGPF, Band 25, S. 271–287, Deutsche Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation e.V., Bern, Schweiz, *https://www.dgpf.de/src/tagung/jt2016/proceedings/band_25/dgpf_tagungsband_2016.pdf*. (letzter Aufruf: 09. Feb. 2026).

Kersten, Thomas; Stallmann, Dirk; Tschirschwitz, Felix (2016b): Development of a new low-cost indoor mapping system – system design, system calibration and first results. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Band XLI-B5: S. 55–62, Copernicus GmbH, Göttingen, *doi:10.5194/isprs-archives-XLI-B5-55-2016*.

Kraus, Karl (2004): Photogrammetrie: Geometrische Informationen aus Photographien und Laserscanneraufnahmen, Band 1. 7. Auflage, Walter de Gruyter, Berlin, ISBN 978-3-11-017708-4, *doi:10.1515/9783110908039*.

Lepetit, Vincent; Moreno-Noguer, Francesc; Fua, Pascal (2008): EPnP: An Accurate O(n) Solution to the PnP Problem. *International Journal of Computer Vision*, Band 81(2): S. 155–166, Springer Science and Business Media LLC, New York, Vereinigte Staaten von Amerika, *doi:10.1007/s11263-008-0152-6*.

Liu, Yan; Su, Xin; Guo, Xiang; Suo, Tao; Yu, Qifeng (2021): A Novel Concentric Circular Coded Target, and Its Positioning and Identifying Method for Vision Measurement under Challenging Conditions. *Sensors*, Band 21(3), MDPI, Basel, Schweiz, *doi:10.3390/s21030855*.

Luhmann, Thomas (1990): An integrated system for real-time and on-line applications in industrial photogrammetry. In: Gruen, A.; Baltsavias, E. (Hg.), ISPRS Commission V Symposium, S. 488–495, Zürich, *doi:10.14463/KXP:1662442289*.

Luhmann, Thomas (2023): Nahbereichsphotogrammetrie Grundlagen - Methoden - Beispiele. 5. Auflage, Wichmann, Berlin, ISBN 9783879077335, *https://content-select.com/de/portal/media/view/640ef3b8-246c-49ab-ad94-48e1ac1b0014*.

Malis, Ezio; Vargas, Manuel (2007): Deeper understanding of the homography decomposition for vision-based control. Research Report RR-6303, INRIA, Sophia Antipolis, Frankreich, *https://inria.hal.science/inria-00174036*.

Negus, Christopher (2020): Linux Bible. Bible, 10. Auflage, John Wiley & Sons, Hoboken, New Jersey, Vereinigte Staaten von Amerika, ISBN 9781119578888, *doi:10.1002/9781119578956*.

Nikolov, Ivan; Madsen, Claus (2016): Benchmarking Close-range Structure from Motion 3D Reconstruction Software Under Varying Capturing Conditions. In: Ioannides, Marinos; Fink, Eleanor; Moropoulou, Antonia; Hagedorn-Saupe, Monika; Fresa, Antonella; Liestøl, Gunnar; Rajcic, Vlatka; Grussenmeyer, Pierre (Hg.), Digital Heritage. Progress in Cultural Heritage: Documentation, Preservation, and Protection, S. 15–26, Springer International Publishing, Cham, ISBN 978-3-319-48496-9, *doi:10.1007/978-3-319-48496-9_2*.

OpenCV (2023): OpenCV. OpenCV team, Palo Alto, Kalifornien, Vereinigte Staaten von Amerika, *https://opencv.org/*. (letzter Aufruf: 09. Feb. 2026).

Raspberry Pi Foundation (2023): Raspberry Pi Documentation - Camera. Raspberry Pi Foundation, Cambridge, Vereinigtes Königreich, *https://www.raspberrypi.com/documentation/accessories/camera.html*. (letzter Aufruf: 09. Feb. 2026).

Reinhard, Jochen (2013): Structure from Motion, Drohnen & Co. Neue Wege in der Dokumentation archäologischer Ausgrabungen. *TUGIUM - Jahrbuch des Staatsarchivs des Kantons Zug, des Amtes für Denkmalpflege und Archäologie, des Kantonalen Museums für Urgeschichte Zug und der Burg Zug*, Band 29: S. 177–188, Regierungsrat des Kantons Zug, Zug, Schweiz, *doi:10.5169/seals-526824*.

Schneider, Carl-Thomas; Sinnreich, Kurt (1992): Optical 3-D measurement systems for quality control in industry. *International Archives of Photogrammetry and Remote Sensing*, Band 29: S. 56–59, International Society for Photogrammetry and Remote Sensing, Washington, D.C., Vereinigte Staaten von Amerika, *https://www.isprs.org/proceedings/xxix/congress/part5/56_xxix-part5.pdf*.

Soot, Matthias; Schulze, Marc; Mulsow, Christian (2015): Untersuchungen zur Genauigkeit der Kamerakalibrierung über Merkmalspunkte in SfM-Werkzeugen. In: Luhmann, Thomas; Müller, Christian (Hg.), Photogrammetrie - Laserscanning - Optische 3D-Messtechnik: Beiträge der Oldenburger 3D-Tage 2015, S. 269–277, Wichmann, Berlin, *https://tu-dresden.de/bu/umwelt/geo/ipf/photogrammetrie/ressourcen/dateien/forschung/publikationen/pubdocs/2015/O3D2015Soot*. (letzter Aufruf: 09. Feb. 2026).

Toffanin, Piero (2019): OpenDroneMap: the missing guide. MasseranoLabs LLC, Saint Petersburg, Florida, Vereinigte Staaten, *https://odmbook.com/1/*. (letzter Aufruf: 09. Feb. 2026).

Worldsemi (2012): WS2811 Datasheet. Dalingshan, Dongguan, Guangdong, China, *https://cdn-shop.adafruit.com/datasheets/WS2811.pdf*. (letzter Aufruf: 09. Feb. 2026).

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang

A. Bedienungsanleitung

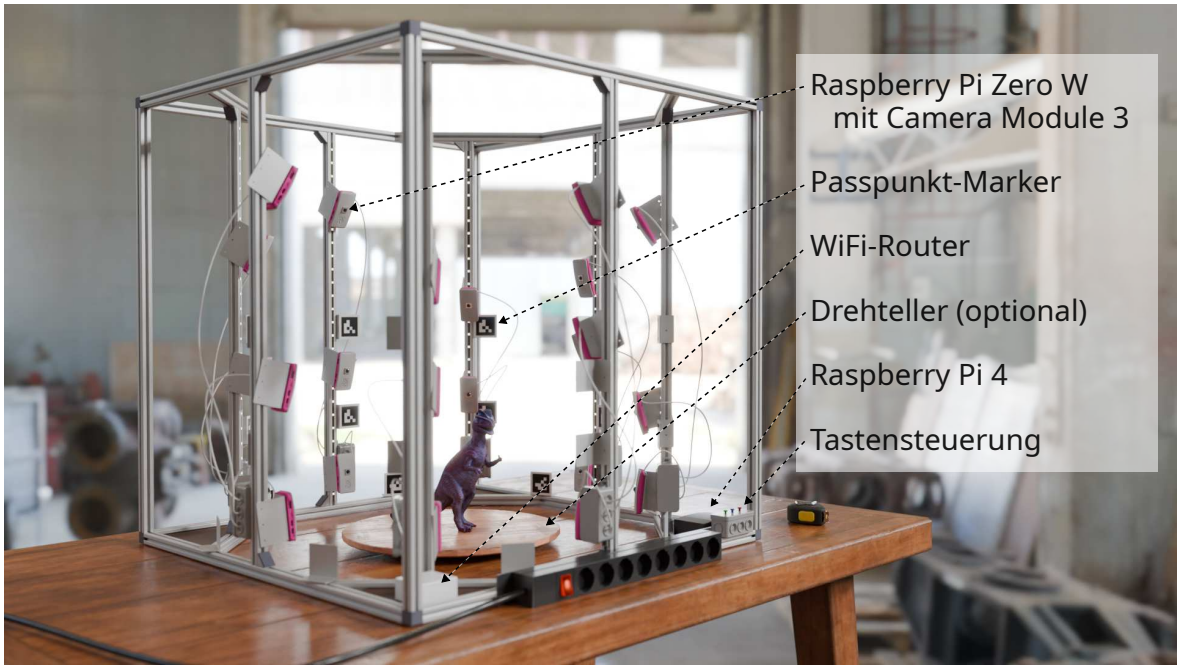


Abbildung A.1.: Überblick über das System und seine Komponenten (vereinfachtes Rendering)

A.1. Anwendungsbereich

Ziel des Systems ist es, 3D-Modelle von Objekten bis zu einer Größe von 40 cm Durchmesser zu erstellen. Die Bedienung soll dabei möglichst einfach und selbsterklärend sein, um auch Laien die Möglichkeit zu geben, das System zu bedienen.

A.2. Menü-Struktur der Weboberfläche

A.2.1. Capture Data

- **Photo** - automatische Aufnahme eines Bildes pro Kamera
- **Focus-Stack** - automatischer Fokus-Stack mit fünf Bildern pro Kamera

- **Detect ArUco** - automatische Erkennung von ArUco-Markern
- **Autofocus** - automatische Fokussierung auslösen

A.2.2. View Data

- **Overview** - Übersicht über alle Kameras
- **Preview** - Vorschau eines Bildes und Möglichkeit, Einstellungen auszuprobieren
- **Photo-Download** - Download der Bilder
- **ArUco-Marker-Coordinates** - Anzeige und Eingabe der Koordinaten der ArUco-Marker
- **Detected ArUco-Marker** - Anzeige der Bildkoordinaten der erkannten ArUco-Marker

A.2.3. Status

- **Search** - Anzeige der System-Informationen
- **Test** - Anzeige der Kamerastatus
- **Light/Status** - Umschalten zwischen Beleuchtung und Statusmeldung

A.2.4. System-Control

- **Configuration** - Einstellungen für Belichtung und Beleuchtung
- **Pause/Resume** - Pausieren und Starten der Kameras
- **Update** - Update der Software
- **Restart** - Neustart der Software
- **Shutdown** - Herunterfahren des Systems
- **Reboot** - Neustart des Systems

A.3. Inbetriebnahme

Beim Aufstellen ist darauf zu achten, dass sich keine stärkeren seitlichen Lichtquellen um das System herum befinden, wie beispielsweise Fensterflächen. Diese könnten die Belichtung der Bilder beeinflussen und so die Qualität der 3D-Modelle negativ beeinflussen. Gegebenenfalls muss durch die Stoffhülle oder andere Maßnahmen für Verschattung oder Streuung des Lichtes gesorgt werden.

Die Berechnung des 3D-Modells erfolgt auf einem externen Rechner. Dafür kann wahlweise Agisoft Metashape oder OpenDroneMap (NodeODM) genutzt werden. Die entsprechende Software sowie eine Java-Laufzeitumgebung müssen auf dem Rechner installiert sein und die Bilder müssen auf diesen übertragen werden. Die Übertragung kann automatisch über eine Netzwerkverbindung oder manuell per USB-Stick erfolgen. Für die automatische Übertragung muss die mitgelieferte Verbindungssoftware auf dem Rechner gestartet sein (siehe Abschnitt A.4).

Das System startet bei Anschluss an eine elektrische Energieversorgung selbstständig. Da die Gefahr besteht, dass die kamerasteuernden Raspberry Pi Zero Daten verlieren, wenn die Energieversorgung unterbrochen wird, sollte das System immer ordnungsgemäß heruntergefahren werden und auf eine zuverlässige Versorgung geachtet werden. Das Abschalten erfolgt durch langes Drücken auf den roten Taster. Das System fährt dann selbstständig herunter - erkennbar an dem Erlöschen der LEDs der Raspberry Pi Zero und der Beleuchtung - und die Energieversorgung kann getrennt werden.

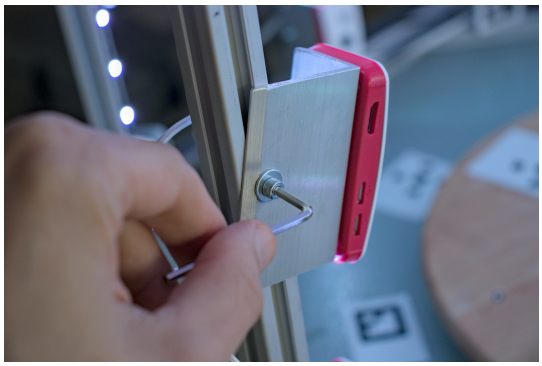
A.4. Software-Einrichtung

Die Software zur Steuerung der Kameras und zur Übertragung der Bilder auf den Rechner ist in Java geschrieben. Sie kann unter Linux, Windows und MacOS genutzt werden. Auf dem Rechner muss entsprechend eine Java-Laufzeitumgebung installiert sein. Für die Nutzung von Metashape muss eine Lizenz vorhanden sein und im selben Ordner wie die ausführbare jar-Datei abgelegt werden. Für OpenDroneMap muss die Software in Form von NodeODM auf dem Rechner installiert sein. Alternativ kann OpenDroneMap auch auf einem externen Rechner installiert sein, der über das Netzwerk erreichbar ist.

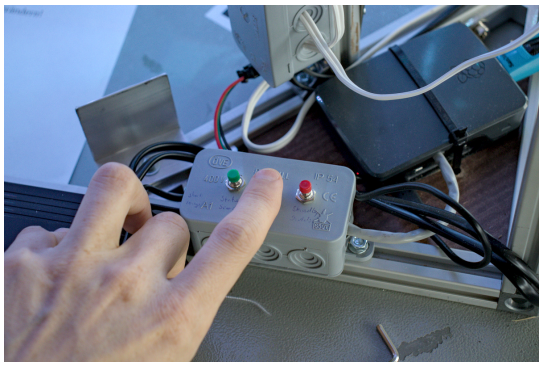
A.5. Kalibrierung

Die letzten Koordinaten der Passpunkte werden im System gespeichert - daher sollten diese möglichst nicht verändert werden. Falls diese dennoch verändert werden, kann das System einzelne Veränderungen berechnen und nutzen. Bei Änderung einer Vielzahl muss das System jedoch extern neu kalibriert werden, beispielsweise durch Bilder mit einer externen Kamera, wodurch dann die Koordinaten der Passpunkte neu bestimmt werden können. Eine Kalibrierung mit Bordmitteln ist nicht möglich.

A.6. Durchführung



(a) Nutzung eines Innensechskantschlüssels zur Ausrichtung der Kameras



(b) Drucktaster zur Steuerung des Systems, z. B. für die Bildaufnahme

Abbildung A.2.: Bedienelemente des Systems

Das System wird gestartet, indem die Energieversorgung hergestellt wird. Die Kameras starten selbstständig und die Beleuchtung wird eingeschaltet. Nach kurzer Zeit sollten kurz alle LEDs grün leuchten. Falls nicht, sind einige Kameras nicht erreichbar. Problemlösungen werden im Abschnitt A.9 behandelt. Die Kameras können ggf. durch Nutzung einen Innensechskantschlüssels neu ausgerichtet werden (siehe Abbildung A.2a). Eine Bestimmung der neuen Positionen erfolgt selbstständig aus den Passpunkten. Das Objekt wird mittig, ggf. auf einer Erhöhung im Rahmen positioniert. Ab hier trennen sich die Wege je nach verwendetem System. Die Bildaufnahme erfolgt in allen Fällen durch kurzen Druck auf den grünen Taster.

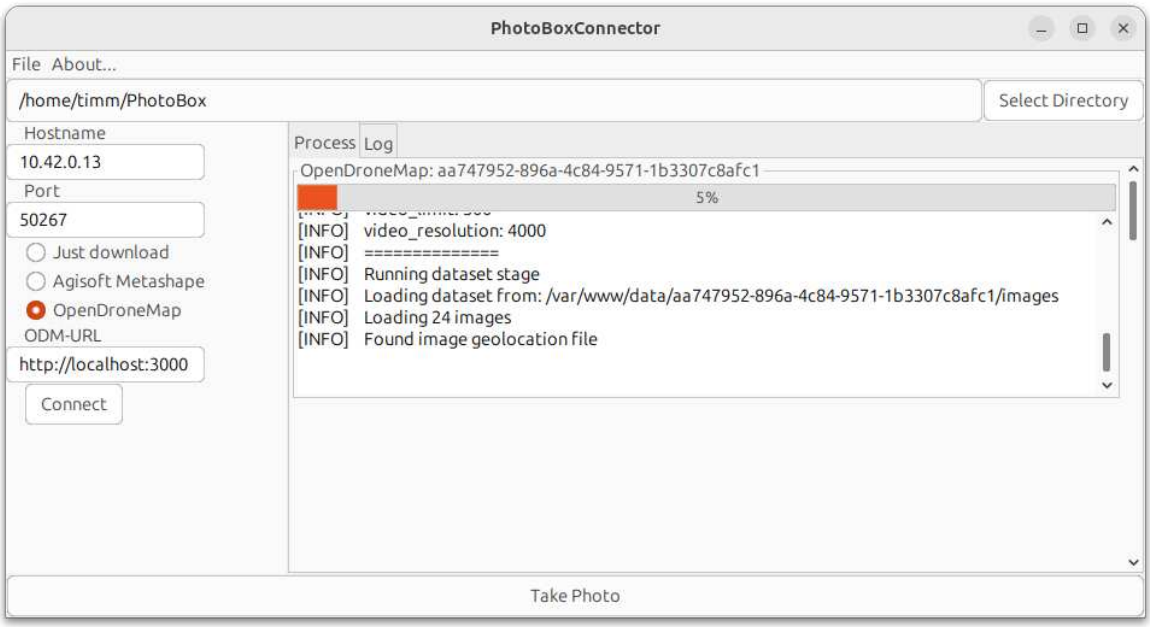


Abbildung A.3.: Screenshot der Connector-Software unter Ubuntu 24.04

A.6.1. ... mit Netzwerkverbindung

Der zu verwendende Rechner wird mit dem System per WLAN (bevorzugt) oder Netzwerkleitung verbunden. Die Verbindungssoftware wird gestartet und die IP-Adresse des Systems angegeben. Standardmäßig lautet diese im WLAN 10.0.1.1, per Netzwerkleitung muss die IP von einem DHCP-Server festgelegt werden, beispielsweise in dem der angeschlossene Rechner als DHCP-Server konfiguriert wird. Unter Gnome ist dies beispielsweise möglich, indem in den Netzwerkeinstellungen die Internetverbindung des PCs freigegeben wird. Anschließend wird die zu nutzende Software ausgewählt und die Verbindung hergestellt. Nun können die Bilder aufgenommen werden (grüne Taste, Abbildung A.2b). Die Bilder werden auf den Rechner übertragen und dort in der ausgewählten Software weiterverarbeitet. Der Fortschritt ist in der rechten Hälfte der Verbindungssoftware zu erkennen.

A.6.2. ... ohne Netzwerkverbindung

Alternativ können die Bilder ohne angeschlossenen PC aufgenommen werden und später weiter verarbeitet werden. Die Bilder werden in allen Fällen auf dem Raspberry Pi 4, der die gesamte Steuerung übernimmt, gespeichert. Die Bilder können dann von der Website des Raspberry Pi heruntergeladen werden. Auch ist es möglich, einen USB-Stick in den USB-Port des Raspberry Pi 4 vor der Bildaufnahme zu stecken (siehe Abbildung A.4b). Die Bilder werden dann zusätzlich auf den USB-Stick kopiert.

Die Verbindungssoftware unterstützt auch das Laden der Bilder aus einem lokalen Ordner, beispielsweise aus der ZIP der Website oder dem Ordner auf dem USB-Stick. Somit kann dann der weitere Workflow analog zum Workflow mit Netzwerkverbindung durchgeführt und die Bilder weiter verarbeitet werden.

A.7. Weiterverarbeitung

Die Bilder können in der ausgewählten Software weiterverarbeitet werden. Die genaue Vorgehensweise hängt von der Software ab und wird in diesem Abschnitt grob erläutert. Für weitere Informationen hierzu ist die entsprechende Bedienungsanleitung der Software zu konsultieren.

A.7.1. Agisoft Metashape

Sofern in der Schnittstellen-Software aktiviert, wird in Metashape ein Workflow durchgeführt, in dem die Bilder importiert, die Kameras kalibriert, die Passpunkte gesetzt, die Modelle berechnet und exportiert werden. Sofern die Funktion nicht aktiviert wird, wird nur ein Projekt angelegt und die Bilder und Passpunkte importiert. Die weitere Verarbeitung kann dann manuell erfolgen. Hierbei ist darauf zu achten, die Passpunkte zu aktivieren, damit deren Koordinaten übernommen werden und der korrekte Maßstab des Modells gewährleistet ist. Anschließend müssten die entsprechenden Schritte manuell durchgeführt werden.

1. Haken bei allen Passpunkten und Kameras setzen
2. *Ablauf / Fotos ausrichten...*
3. Bereich des Modells festlegen
4. *Ablauf / Punktwolke erzeugen...*
5. ggf. *Ablauf / Mesh erzeugen...*
6. ggf. *Ablauf / Textur erzeugen...*
7. *Exportieren / Modell exportieren...* oder *Punktwolke exportieren...*

A.7.2. OpenDroneMap

In OpenDroneMap wird ein Prozess gestartet, der die Bilder importiert, die Modelle berechnet und die Modelle exportiert. Hier sind keine weiteren Eingriffe notwendig und möglich.

A.8. Wartung

Software-Updates können zu Inkompatibilitäten führen, daher sollten diese erstmal auf einem zusätzlichen Raspberry Pi ausprobiert werden, bevor diese auf dem System ausgerollt werden. Sofern ein Software-Update durchgeführt werden soll, kann das System per RJ45-Stecker über den Router (siehe Abbildung A.4a) mit dem Internet verbunden werden. Anschließend kann das Update der Software über die Weboberfläche durchgeführt werden. Ein Betriebssystemupdate kann per SSH-Verbindung zum Raspberry Pi 4 durchgeführt werden. Hierfür sind die Zugangsdaten am Ende des Dokuments zu finden.

(a) Anschluss eines RJ45-Steckers zur Verbindung mit dem Internet oder einem PC

(b) Anschluss eines USB-Sticks zur Datenübertragung

Abbildung A.4.: Anschlüsse des Systems

A.9. Fehlerbehebung

A.9.1. Kameras sind nicht erreichbar

Die Kameras sind nicht erreichbar, wenn die LEDs nicht grün leuchten. Dies kann verschiedene Ursachen haben. Als erste Maßnahme sollte das System nochmal heruntergefahren werden durch einen langen Druck auf die rote Taste. Anschließend wird die Energieversorgung für einige Sekunden vollständig getrennt und wieder hergestellt. Das System sollte nun neu starten und die Kameras erreichbar sein.

Falls das noch nicht der Fall ist, hilft ein Blick in das Menü des WLAN-Routers. Hier sollten in der Übersicht aller Netzwerkgeräte die 24 Kameras, der steuernde Raspberry Pi 4 und das Gerät, über das der Zugriff erfolgt, aufgezählt sein. Falls das nicht der Fall ist, muss der fehlerhafte Raspberry Pi Zero an einem Display und einer Tastatur angeschlossen werden, um den Fehler zu finden.

Falls der Raspberry Pi im Netzwerk zu finden ist, kann sich per SSH mit dem Raspberry Pi verbunden werden. Zugangsdaten sind die Übersicht am Ende zu entnehmen.

A.9.2. Bilder zu hell/zu dunkel

Je nach Umgebungslicht können die Bilder zu dunkel oder zu hell sein. Im Normalfall sollte die Grundeinstellung mit einem Belichtungswert von +1 zu guten Ergebnissen führen. Falls dies nicht der Fall ist, kann dieses unter *Configuration* umgestellt werden. Hier können der Belichtungswert oder die Helligkeit und Lichtfarbe verändert werden.

A.10. Zugangsdaten

Gerät	**Zugang**	**Benutzer**	**Passwort**
WLAN-Netzwerk	photobox		photogrammetry
Raspberry Pi 4	192.168.1.1:8080	photo	box
Raspberry Pi Zero	192.168.2.1-24:8080	photo	box
WLAN-Router	192.168.0.1:80		photobox1

B. Kurzbedienungsanleitung

Für detaillierte Erklärungen und Problemlösungen lesen Sie bitte die Bedienungsanleitung. Hintergrundinformationen sind der Thesis zu entnehmen.

1. Energieversorgung herstellen, ggf. Steckdosenleiste einschalten
2. Warten bis alle LEDs grün aufleuchten, evtl. mittels kurzem Druck auf Status-Taste überprüfen
3. Gewünschte Ausgabe anschließen
   - Computer per WLAN verbinden
   - USB-Stick an Raspberry Pi 4 anschließen
4. Aufnahmeknopf drücken
5. Warten bis alle LEDs grün aufleuchten, Bilder werden übertragen
6. Weiterverarbeitung auf Computer
   - automatisch, wenn Computer per WLAN verbunden
   - manuell, wenn USB-Stick verwendet
7. System herunterfahren
   - Langen Druck auf roten Taster
   - Warten bis alle LEDs erlöschen
   - Energieversorgung trennen

B.1. Agisoft Metashape

1. Metashape starten
2. Projekt öffnen
3. Haken bei allen Passpunkten und Kameras setzen
4. *Ablauf / Fotos ausrichten...*
5. Bereich des Modells festlegen

6. *Ablauf / Punktwolke erzeugen...*
7. ggf. *Ablauf / Mesh erzeugen...*
8. ggf. *Ablauf / Textur erzeugen...*
9. *Datei / Exportieren / Modell exportieren...* oder *Punktwolke exportieren...*

C. Quick Guide

For detailed explanations and troubleshooting, please refer to the user manual (German). Background information can be found in the thesis (also German).

1. Connect power supply
2. Wait until all LEDs light up green, possibly check by pressing the status button briefly
3. Connect desired output
    - Connect computer via WLAN
    - Connect USB stick to Raspberry Pi 4
4. Press the record button
5. Wait until the recording is finished, images are transferred
6. Post-processing on computer
    - automatically, if computer is connected via WLAN
    - manually, if USB stick is used
7. Shut down system
    - Long press on red button
    - Wait until all LEDs go out
    - Disconnect power supply

C.1. Agisoft Metashape

1. Start Metashape
2. Open project
3. Set checkbox on all control points and cameras
4. *Workflow / Align Photos...*
5. Define region of the model

6. *Workflow / Build Point Cloud...*
7. If necessary *Workflow / Build Mesh...*
8. If necessary *Workflow / Build Texture...*
9. *File / Export / Export Model...* or *Export Point Cloud...*

D. Teileliste

D.1. Mechanische Bauteile

Bezeichnung	Anzahl	VPE	Preis/VPE	Gesamtpreis
Alu-Strebenprofil Nut 6 Typ B	17,5 m	0,1 m	0,44 EUR	77,00 EUR
Eckwürfel	8	1	5,00 EUR	40,00 EUR
90-Grad-Winkel	16	1	1,40 EUR	22,40 EUR
45-Grad-Winkel	16	10	23,00 EUR	46,00 EUR
Hammerkopf-Mutter M4	46	50	11,90 EUR	11,90 EUR
Zylinderschraube M4	46	100	3,80 EUR	3,80 EUR
Scheibe M4	76	100	1,85 EUR	1,85 EUR
Senkkopfschraube M4	30	30	2,45 EUR	2,00 EUR
Mutter M4	24	24	0,59 EUR	0,59 EUR
Linsenkopfschraube M5	18	18	2,32 EUR	2,32 EUR
Mutter M5	18	18	0,88 EUR	0,88 EUR
Scheibe M5	18	18	0,45 EUR	0,45 EUR
Isolierband	1	5	2,98 EUR	2,98 EUR
Schrumpfschlauch	1	1	1,00 EUR	1,00 EUR
Winkelprofil 30 x 50mm	2 m	2 m	23,96 EUR	23,96 EUR
Verteilerdose klein	9	1	0,75 EUR	6,75 EUR
Leitungskanal 10x20	2	2	2,30 EUR	2,30 EUR
Baumwollstoff	3	1	6,00 EUR	18,00 EUR
Klettband	0,4	0,4	0,80 EUR	0,80 EUR
Reißverschluss	3	1	2,00 EUR	2,00 EUR
				266,98 EUR

Tabelle D.1.: Mechanische Bauteile mit Preisen (Stand: September 2023)

D.2. Elektronische Bauteile

Bezeichnung	Anzahl	VPE	Preis/VPE	Gesamtpreis
Raspberry Pi Zero W	24	1	17,90 EUR	429,60 EUR
Raspberry Pi Camera 3	24	1	29,15 EUR	699,61 EUR
Raspberry Pi 4	1	1	66,80 EUR	66,80 EUR
RPi Zero Gehäuse + Leitung	24	1	3,60 EUR	86,40 EUR
RPi 4 Gehäuse	1	1	4,60 EUR	4,60 EUR
RPi Netzteil 5V 3A	1	1	8,65 EUR	8,65 EUR
Speicherkarte 32 GB	25	1	5,95 EUR	142,80 EUR
LED-Streifen	1	1	31,10 EUR	31,10 EUR
WLAN-Access-Point	1	1	31,90 EUR	31,90 EUR
Netzteil 12V 3,5 A	1	1	11,70 EUR	11,70 EUR
Netzteil 5V 7 A	3	1	18,20 EUR	54,60 EUR
Buchse für Netzteile	4	1	2,30 EUR	9,20 EUR
Taster	3	1	0,67 EUR	2,01 EUR
Aderendhülsen	1	1	1,20 EUR	1,20 EUR
Hebelklemmen 2 Anschlüsse	10	1	0,47 EUR	4,70 EUR
Hebelklemmen 5 Anschlüsse	18	1	0,85 EUR	15,30 EUR
Hebelklemmen 3 Anschlüsse	16	1	0,50 EUR	8,00 EUR
Litze 2*0,75 mm^2	3	10	2,50 EUR	7,50 EUR
Steckdosenleiste	1	1	14,95 EUR	14,95 EUR
				1630,62 EUR

Tabelle D.2.: Elektronische Bauteile mit Preisen (Stand: September 2023)