HafenCity Universität Hamburg

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

Aufbau eines photogrammetrischen Messsystems unter Verwendung von Rasberry-Pi-Kameras als Low-Cost-Sensoren

Geodäsie und Geoinformatik

Masterthesis

Sommersemester 2024

Florian Timm

Abgabedatum: 01. August 2024

Verfasser

Florian Timm
Matrikelnummer: 6028121
Gaiserstraße 2, 21073 Hamburg

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

Erstprüfer

Prof. Dr.-Ing. Thomas Kersten
HafenCity Universität Hamburg
Überseeallee 16, 20457 Hamburg

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

Zweitprüfer

Dipl.-Ing. Kay Zobel
HafenCity Universität Hamburg
Überseeallee 16, 20457 Hamburg

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

Kurzzusammenfassung

Die Photogrammetrie bietet die Möglichkeit, mit relativ einfacher Technik 3D-Modelle zu erstellen. Die Aufnahme der Bilder ist jedoch sehr zeitaufwendig und daher für die Erfassung vieler Objekte, z.B. bei der Digitalisierung von Museumsobjekten, nicht praktikabel. Systeme mit mehreren fest installierten Kameras setzen in der Regel auf hochwertige Kameras, die jedoch die Hardwarekosten stark in die Höhe treiben.

In dieser Arbeit soll der Lösungsansatz untersucht werden, mehrere kostengünstige Kameras, die fest auf einem Rahmen montiert sind, zu verwenden. Mit Kameras für den Raspberry Pi soll ein photogrammetrisches Messsystem für kleine Objekte aufgebaut werden. Dazu soll eine Schnittstelle zur Synchronisation der Kameras programmiert und eine Möglichkeit zur Kalibrierung der Kameras entwickelt werden. Das Endergebnis soll es im Idealfall auch einem photogrammetrischen Laien ermöglichen, schnell und ohne lange Einarbeitungszeit 3D-Modelle in akzeptabler Auflösung und Qualität zu erzeugen.

Abstract

Photogrammetry offers the possibility of creating 3D models with relatively simple technology. However, capturing the images is very time consuming and therefore not practical for many objects, such as museum digitisation. Systems using multiple fixed cameras usually rely on high quality cameras, which significantly increases hardware costs.

This paper explores the solution of using multiple low-cost cameras mounted on a frame. Cameras for the Raspberry Pi are used to build a photogrammetric measurement system for small objects. This involves programming an interface to synchronise the cameras and developing a way to calibrate the cameras. The end result should ideally enable even a photogrammetric layman to produce 3D models of acceptable resolution and quality quickly and without a long training period.

Inhaltsverzeichnis

1. Konzept

In Museen besteht vielfach der Wunsch, ihre Exponate zu digitalisieren. Entsprechende Handreichungen des Deutschen Museumsbundes legen auch die Digitalisierung als 3D-Modelle nahe, verweisen aber auf Aufwand und Format-Probleme (Deutscher Museumsbund e. V., 2022, S. 43). Auch bei Ausgrabungen aber auch in anderen Bereichen besteht der Bedarf dreidimensionale Modelle einfach und kostengünstig zu erfassen.

Diese Möglichkeit soll das im Rahmen dieser Arbeit entwickelte Kamerasystem bieten. Es soll mittels Photogrammetrie ermöglichen, mit geringen personellen Aufwand kleine Objekte bis etwa 40 cm Durchmesser zu erfassen. Die Bedienung soll dabei laiensicher und mit nur kurzer Einarbeitungszeit möglich sein, dass System also die meisten Schritte selbstständig durchführen. Auch der Nachbau soll mit etwas handwerklichen Geschick möglich sein. Um Lizenzkosten zu sparen, soll die Möglichkeit auf OpenSource-Software zu setzen geprüft werden.

Neben der eigentlichen Entwicklung eines funktionsfähigen Systemes soll abschließend die Anzahl der Kameras und die Nutzung eines Drehtellers evaluiert werden, um hiermit gegebenenfalls die Hardwarekosten zu senken oder die Auflösung und Genauigkeit zu steigern.

2. Photogrammmetrischer Hintergrund

Das Erzeugen des 3D-Modelles basiert auf der Verknüpfung von Bildern aus verschiedenen Positionen über identische Punkte. Hieraus können dann die Orientierung der Kameras zueinander und die Koordinaten der Punkte in einem lokalen Koordinatensystem ohne bekannten Maßstab berechnet werden. Durch die Nutzung von bekannten Größen beispielsweise durch Maßstäbe kann dieses System transformiert werden.

Dieses Kapitel beschreibt die hierfür notwendigen Bedingungen und die Grundlagen der Rekonstruktion des Objektes als 3D-Modell.

2.1. Innere Orientierung

Aus der Position eines Punktes in einem Bild kann vereinfacht gedacht ähnlich einer Messung mit einem Theodolit die Richtung des Punktes in Relation zu der Kamera bestimmt werden. Damit diese Berechnung möglich wird, müssen die Parameter der Kamera bekannt sein, die sogenannte innere Orientierung. Sie beschreibt die Abbildung der Kamera mathematisch. Wichtigste Parameter sind hierbei die Lage des Bildhauptpunktes und die Kamerakonstante. Außerdem zählen hierzu auch die Parameter, die die Bildfehler wie die Verzeichnung, beschreiben. (Luhmann, 2023, S. 179f)

Die innere Orientierung kann während der Messung beispielsweise mittels Bündelblockausgleichung bestimmt werden. Bedingung hierfür ist jedoch, dass die innere Orientierung stabil ist und sich nicht während der Messung ändert (Luhmann, 2023, S. 181f).

Jede Einstellung der Kameraoptik verändert die innere Orientierung und auch jede Kamera,auch einer Modellreihe, kann je nach Genauigkeitsanspruch als unterschiedlich angesehen werden. Änderungen können sich beispielsweise durch Umfokussierung oder die Nutzung eines optischen Zoom ergeben, aber auch durch einen mechanisch instabilen Aufbau der Kameras. Daher sollten die Bilder möglichst mit einer Kamera

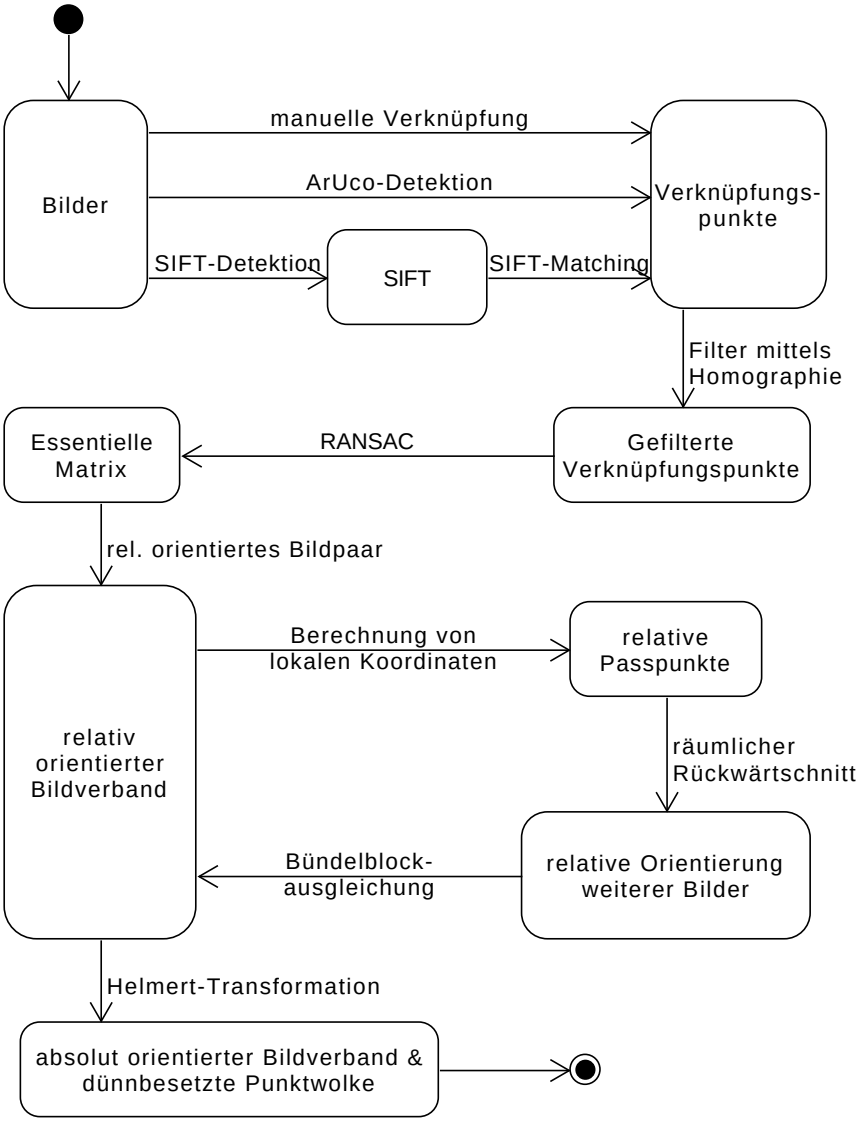

Abbildung 2.1.: Ablauf der Bildverknüpfung, nach Luhmann 2023, S. 492

mit festen Einstellungen (Brennweite, Fokus, Blende, Objektiv) aufgenommen werden. Änderungen der Empfindlichkeit (ISO-Zahl) oder Belichtungszeit sind unproblematisch für die innere Orientierung (Luhmann, 2023, S. 176).

2.2. Bilder

Die Berechnung der Tiefeninformationen 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 Schnitte zur Verfügung zu haben und die innere und äußere Orientierung möglichst gut berechnen zu können, müssen diese Bilder einige Bedingungen erfüllen.

Überlappung und Bildinhalte

Da die Bilder durch identische Punkte verbunden werden, müssen die Bildinhalte sich überlappen. Die automatische Identifikation von identischen Punkten ist auf verschiedene Weisen möglich: Entweder durch die Nutzung von codierten Passpunkten wie ArUco-Markern oder konzentrischen Passpunkten nach Schneider & Sinnreich (1993). Alternativ können auch eine Merkmalsextraktion zur Indentifikation von gemeinsamen Punkten genutzt werden, beispielsweise durch die SIFT-Methode. Hierfür muss die oberfläche aber genügend Texturen aufweisen. (Luhmann, 2023, S. 478)

Belichtung

und bei sich stark ändernden Helligkeitsverhältnissen auch hilfreich, jedoch wird hierdurch auch die Helligkeit der Verknüpfungspunkte verändert, was wiederum problematisch sein kann. Entsprechend ist es empfehlenswert bei gleichmäßiger Beleuchtung, die sich auch nicht ändern sollte, die Bilder zu erstellen - also beispielsweise bei bedeckten Himmel.

Position und Ausrichtung der Kamera

Bilder, die vom gleichen Standpunkt aufgenommen wurden, sind oft nur ungenau verknüpfbar. Daher empfiehlt es sich, eher um Objekte herum zu gehen, statt beispielsweise bei Innenräumen sich nur in die Mitte zu stellen und sich zu drehen. Während der Aufnahmen sollte die Kamera natürlich möglichst ruhig gehalten werden, um möglichst scharfe Bilder zu generieren. Dies ist vor allem bei UAV-Flügen relevant, die auch aus hoher Geschwindigkeit Bilder aufnehmen könnten. Jedoch weisen die meisten Digital-

kamera einen Rolling-Shutter-Effekt auf, dass heißt die Bildreihen werden nicht alle zeitgleich aufgenommen, sondern die oberen vor den unteren. Daher ist es sinnvoll, das UAV kurz schweben zu lassen und dann ein Bild zu machen. (Toffanin, 2019, S. 147)

Umso weiter die Kamera vom Objekt entfernt ist, umso mehr Inhalte und entsprechend mehr Verknüpfungsmöglichkeiten sind im Bild enthalten. Daher sollte sich immer klar gemacht werden, welche Auflösung benötigt wird und entsprechend die Entfernung bzw. bei UAV-Aufnahmen die Höhe zu wählen. Der Abstand sollte außerdem variiert werden, da dieses bei der Ausgleichung der inneren Orientierung hilft. (Toffanin, 2019, S. 144f)

Schärfentiefe und Fokussierung

2.3. Verknüpfungspunkte

Um die einzelnen Bilder verknüpfen zu können, werden identische Punkte zwischen zwei oder mehr Bildern benötigt. Diese können klassisch per Hand erfasst werden, jedoch ist dieses schon bei kleineren Projekten sehr zeitaufwändig. Daher wurde zusätzlich die Möglichkeit genutzt, automatisch Verknüpfungspunkte zu erzeugen.

2.3.1. ArUco-Marker

Eine Variante der automatischen Verknüpfungspunkte sind die sogenannten ArUco-Marker. Diese werden häufig für die Orientierung bei Augmented-Reality-Anwendungen genutzt. OpenCV unterstützt die Erkennung dieser Marker. Sie werden als codierte Messmarken verwendet und können automatisch im Subpixelbereich erkannt werden. Jede Ecke kann hier einzeln identifiziert werden, sodass ein erkannter Marker vier Verknüpfungspunkte liefern kann.

2.3.2. SIFT

Die SIFT-Methode 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.) oder in Bewegung ist.

Zur Erkennung von Merkmalen setzt das Verfahren auf die Detektion von Kanten. Diese werden in verschiedenen Stufen einer Bildpyramide erkannt und ihre Extrema berechnet. Es werden diese Merkmale weiter ausgedünnt, beispielsweise über den Kon-

trast. 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 diesen kann dann die Übereinstimmung von zwei Markern in zwei Photos bestimmt werden, auch wenn die Bilder zueinander gekippt oder gedreht sind. (Luhmann, 2018, S. 483)

2.4. Verknüpfung von Bildern

Durch die drei beschriebenen Verfahren und die hieraus entstandenen Verknüpfungspunkte können die Bilder miteinander verknüpft werden. Da durch GNSS nur eine sehr grobe und ggf. auch falsche Vorausrichtung besteht, kann diese nur als sehr grobes Hilfsmittel genutzt werden. In diesen Ansatz wird es nur für die Beschränkung der SIFT-Detektion auf Bilder, die nahe beieinander sind oder sich ArUco- oder manuelle Punkte teilen, genutzt. Für die eigentliche Verknüpfung werden dann nur photogrammmetrische Verfahren genutzt. Über diese wird im Folgenden ein kurzer Überblick gegeben.

Abbildungsgleichung

Die Abbildung eines Punktes auf einem Bild wird durch die Abbildungsgleichung beschrieben. In der Matrizenrechnung ergibt sich dieser aus der Multiplikation mit der Projektionsmatrix P. Diese ergibt sich aus der Kameramatrix K, der Rotation R und dem Projektionszentrum X_0. (siehe Gleichung 2.1, nach Hartley & Zisserman, 2003, S. 244 und Luhmann, 2018, S. 288)

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

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

$$P = \begin{bmatrix} c_x & 0 & x'_0 \\ 0 & c_y & y'_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_1 1 & r_2 1 & r_3 1 & X_0 \\ r_1 2 & r_2 2 & r_3 2 & Y_0 \\ r_1 3 & r_2 3 & r_3 3 & Z_0 \end{bmatrix} \tag{2.3}$$

Um die Beziehung zwischen zwei Bildern aufzustellen, kann man die Abbildungsgleichung nutzen. Da es hier nur um die Beziehung zwischen zwei Bildern geht, kann die Rotation und Translation des ersten Bildes auf 0 gesetzt werden (R ist dann eine 3x3-Einheitsmatrix und X_0 ein Nullvektor). X_0 des zweiten Bildes wird zur Translation zwischen den beiden Bildern. (Luhmann, 2018, S. 326)

Fundamental-/Essentielle Matrix

Die Fundamentalmatrix beschreibt das Verhältnis von identischen Punkten in zwei Bildern zueinander. Sie kann genutzt werden, um die Drehung und Verschiebung zwischen zwei Aufnahmeorten zu bestimmen. Die Formel für die Fundamentalmatrix lautet:

$$x'^T \cdot F \cdot x'' = 0 \tag{2.4}$$

Sie unterscheidet sich nur von der Essentiellen Matrix in der Eigenschaft, dass bei der Essentiellen Matrix die Kameraparameter bekannt sein müssen. Diese werden dann auf die Bildkoordinaten angewendet und x' und x'' werden so unabhängig (normalisiert) von den Kameraeinstellungen. Mittels Singulärwertzerlegung lässt sich dann die Rotation und Translation aus der Essentiellen Matrix bestimmen. Im Falle von Bildern die mit Digitalkameras aufgenommen wurden sind die Kameraparameter zumindest näherungsweise aus den EXIF-Daten der Bilder bekannt, sodass dieser Weg hier genutzt werden kann. (Hartley & Zisserman, 2003, S. 257)

In der entstandenen Software wurde die Methode verwendet, um die Verknüpfung des Startpaares zu errechnen.

2.4.1. Fundamental-/Essentielle Matrix

Die Fundamentalmatrix beschreibt das Verhältnis von identischen Punkten in zwei Bildern zueinander. Sie kann genutzt werden, um die Drehung und Verschiebung zwischen zwei Aufnahmeorten zu bestimmen. Die Formel für die Fundamentalmatrix lautet:

$$x'^T \cdot F \cdot x'' = 0 \tag{2.5}$$

Sie unterscheidet sich nur von der Essentiellen Matrix in der Eigenschaft, dass bei der Essentiellen Matrix die Kameraparameter bekannt sein müssen. Diese werden dann auf die Bildkoordinaten angewendet und x' und x'' werden so unabhängig (normalisiert) von den Kameraeinstellungen. Mittels Singulärwertzerlegung lässt sich dann die Rotation und Translation aus der Essentiellen Matrix bestimmen. Im Falle von Bildern

die mit Digitalkameras aufgenommen wurden sind die Kameraparameter zumindest näherungsweise aus den EXIF-Daten der Bilder bekannt, sodass dieser Weg hier genutzt werden kann. (Hartley & Zisserman, 2003, S. 257)

In der entstandenen Software wurde die Methode verwendet, um die Verknüpfung des Startpaares zu errechnen.

2.4.2. Vorwärtsschnitt

Aus den zwei Projektionsmatrizen zweier Bilder und der Position eines identischen Punktes in beiden Bildern lassen sich dann lokale (Modell-)Koordinaten des Punktes berechnen. Da auch dieses nicht fehlerfrei ist, wurde hierfür die Methode der linearen Triangulation verwendet. Hierbei wird der entstehende Fehler ausgeglichen. (Hartley & Zisserman, 2003, S.312)

2.4.3. Rückwärtsschnitt

Entsprechend lässt sich auch aus der Position von Punkten mit bekannten lokalen Koordinaten die Position und Drehung eines Bildes berechnen. Dieses wurde genutzt, um weitere Bilder an das Startpaar heranzuknüpfen. Hierfür werden mindestens 5 Punkte benötigt. (Hartley & Zisserman, 2003, S. 533ff)

2.5. 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 Unterunterabschnitt 2.4 beschriebene Abbildungsgleichung. Als Ergebnis erhält man die ausgeglichenen Parameter und Genauigkeitsangaben für diese. (Luhmann, 2018, S. 340)

3. Aufbau des Messsystemes

Die Kameras sollten eine höhe 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 unverändert bleibt. Daher ist ein stabiler Rahmen notwendig, an welchem die Kameras verdrehsicher angebracht werden können. Kleinere Restfehler in den Orientierungen können mit 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 dann auf das Steuerungssystem übertragen werden, hierfür wir eine Form der Datenübertragung benötigt. Damit die Bilder möglichst schattenfrei ausgeleuchtet werden, muss Beleuchtung mit eingeplant werden. Außerdem muss die Stromversorgung der einzelnen Kameras sichergestellt sein.

Aus diesen Anforderungen ergeben sich die einzelnen Abschnitte dieses Kapitels.

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 $1,4\ \mu m$ relativ große Pixel (Raspberry Pi Foundation, 2023), was im subjektiven Eindruck eine sehr gute Bildqualität ergibt.

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 (vgl. Abschnitt 2.1) weiter und wurde daher auch besonders im analysiert. Da die Bilder aber Nahbereich zwischen 20 und 70 cm benötigt werden, ist hier die Schärfentiefe niedrig. Der elektronische Fokus, eine wiederholgenaue und damit mathematisch modellierbare Fokussierung vorausgesetzt, ermöglicht hier, Fokusstacking zu nutzen um den Schärfebereich zu vergrößern.

Verweis zu Fokusexperimenten

Weiterer Vorteil der Lösung mit einzelnen Raspberry Pis ist es, dass hierdurch bereits die einzelnen Kameraeinheiten Berechnungen wie das Identifizieren von Passpunkten übernehmen könnten und auch durch die Nutzung von Netzwerkverbindungen für die Steuerung das System skalierbar im Sinne der Anzahl der Kameras aber auch der Größenordnung der Abstände zwischen den Kameras.

3.2. Rahmen

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

Als mögliche Materialien kamen Holz, Stahl und Aluminum in Frage. Aufgrund der einfachen Bearbeitung und der genormten Profile, wurde sich für Aluminiumprofile entschieden. Diese gibt es in verschiedenen Ausführungen mit Nuten an den Seitenflächen, so dass eine einfache Montage, aber auch eine Demontage zu Transportzwecken, möglich wird. Außerdem sind diese sehr stabil bei leichtem Gewicht.

Durch eine Konstruktion mit Eckwürfeln sowie dem Einbau von dreieckigen Strukturen und Platten, die Scheibenwirkung haben, wurde die Stabilität der Verbindungen erhöht.

3.3. Beleuchtung

Um möglichst gute Bilder zu erzeugen, sollte das Objekt gut ausgeleuchtet sein. Eine dunkle Umgebung verlängert die Belichtungszeit, wodurch die Gefahr von unscharfen Aufnahmen steigt und dunklere Bereiche (ungleichmäßige Ausleuchtung) verursacht Rauschen in diesen Bildbereichen. Daher soll das System eine gleichmäßige Ausleuchtung ermöglichen. Problematisch ist hierbei, dass die Kameras ggf. auch die Lichtquellen mit im Bildbereich haben können, wodurch Linsenreflexionen oder Ausbrennen der Bildbereiche möglich sind.

Es wurde sich zur Beleuchtung für einzeln steuerbare LED-Lichtstreifen entschieden. Diese können einfach an den Aluprofilen montiert werden und ermöglichen es, einzelne Bereiche und so ggf. blendende Bereiche abzuschalten.

3.4. Stromversorgung

Die Raspberry Pis werden mit 5 Volt betrieben. Der Raspberry Pi Zero 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 Ampere. Für den Raspbery Pi 4 wurde ein eigenes Netzteil eingeplant und für die Zero W ein gemeinsames 35 Watt-Netzteil. Versuche zeigten jedoch, dass der Stromverbrauch kurzfristig höher ausfallen konnte, so dass die Zero W, die am meisten von Spannungsabfällen betroffen sind zum Absturz gebracht wurden, wenn alle Kameras gleichzeitig auslösten. Nach dem die Last auf sicherheitshalber auf zwei weitere Netzteile verteilt wurde, lief das System zuverlässig.

Als Kabelmaterial wurde Klingeldraht mit $0,75$ mm^2 verwendet. Der relativ hohe Kabelquerschnitt 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 auf Seiten der Zero W verlötet und in den Verteilerdosen mit Federkraftklemmen verbunden.

Die Stromversorgung der Beleuchtung erfolgt über ein 12-V-Netzteil mit $3,5$ A Ausgangsleistung. Auch hier wurde Klingeldraht zur Verteilung zwischen den einzelnen Holmen genutzt.

Beim WLAN-Router war ein entsprechendes USB-Netzteil mitgeliefert.

3.5. Kommunikation und Datenübertragung

Die Kommunikation zwischen den Raspberry Pis erfolgt über WLAN. Vorteil dieser Lösung ist, dass hier keine weiteren Leitungen außer der Stromversorgung zu den einzelnen Raspberry Pi Zero W benötigt wird und es auch möglich wäre, die gleiche Hard- und Software auch 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 Datenleitungen wird eine Steuerleitung für die LED-Streifen benötigt. Über diese erfolgt die Steuerung der einzelnen LED-Gruppen. Auch hier wurde wieder Klingeldraht verwendet.

4. Software-Entwicklung

Für die Steuerung der Kameras und die anschließende Berechnung des 3D-Modelles muss ein Betriebssystem für das Kamerasystem und eine entsprechende Schnittstelle zu einer SfM-Software geschaffen werden. Diese Entwicklung erfolgte hauptsächlich in Python in Form von Prototyping. Das Kapitel beschreibt die Anforderungen an die Software in Abschnitt 4.1. Anschließend werden hieraus die Anwendungsfälle (Abschnitt 4.2) erarbeitet und abschließend die Implementation (Abschnitt 4.3) beschrieben.

4.1. Anforderungsanalyse

4.1.1. Funktionale Anforderungen

- Die Kameras sollen zeitgleich auslösbar sein. Die Auslösung soll möglichst verzögerungsfrei erfolgen.
- Die Steuerung soll auch unabhängig von anderen Geräten möglich sein, beispielsweise per Tastensteuerung.
- Der Status des Systemes soll für den Nutzer erkennbar sein - auch ohne Anschluss eines Computers etc..
- Es sollen Passpunkte automatisch gefunden und und für die Bestimmung der äußeren Orientierung genutzt werden.
- Die Bilder sollen scharf und fokussiert sein.

4.1.2. Schnittstellen

- Die Daten sollen intern gespeichert werden.
- Eine Speicherung auf tragbaren Speichermedien wie USB-Sticks soll möglich sein.
- Eine direkte Übertragung an SfM-Software soll möglich sein.

4.1.3. Nicht-funktionale Anforderungen

- Die Erfassung soll ohne weitere Hardware möglich sein. Das System soll unabhängig von Netzwerkanschlüssen etc. sein.
- Alle Kommunikation soll über WLAN erfolgen.

4.2. Anwendungsfallmodellierung

Entsprechend der benötigten Schritte aus Kapitel 2 und Abbildung 2.1 wurde die Anwendungsfälle, die die Benutzeroberfläche ermöglichen soll, im Anwendungsfall-Diagramm in Abbildung 4.1 zusammengetragen.

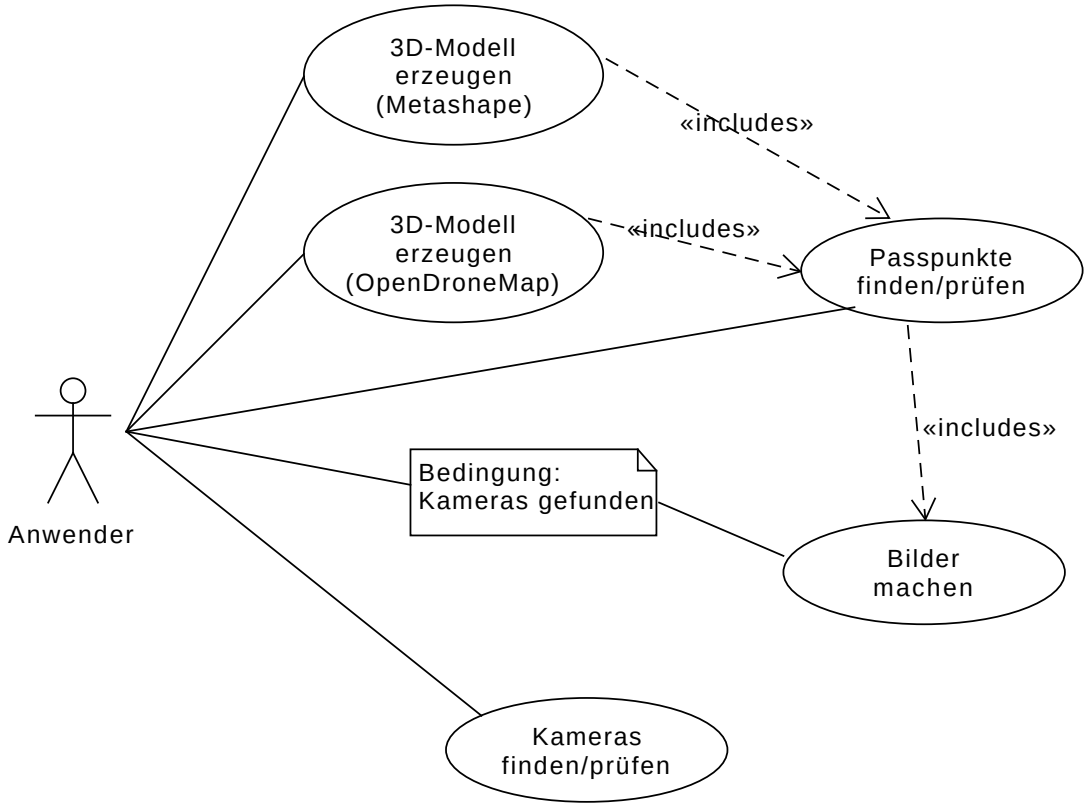

Abbildung 4.1.: Anwendungsfall-Diagramm

Aus den benötigten Daten wurde das Domänen-Klassendiagramm aus Abbildung 4.2 erzeugt. Dieses zeigt vor allem die Abhängigkeiten der einzelnen Datensätze untereinander.

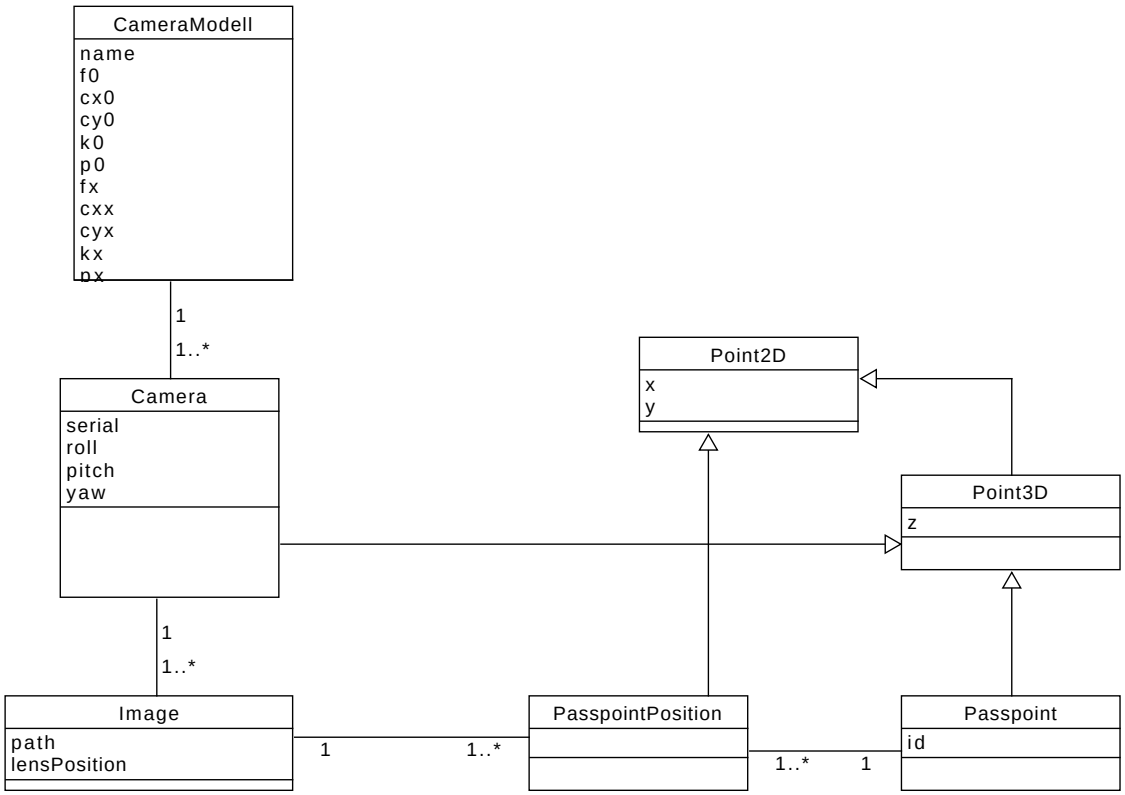

Abbildung 4.2.: Domänen-Klassendiagramm

4.3. Implementierung

Dieses ist dann auch Grundlage für die Implementierung der SQLite-Datenbank. Der Datenbankaufbau ist Abbildung 4.3 zu entnehmen. Die Datenbank dient der Zwischenspeicherung der Passpunkte und der durchgeführten Berechnungen.

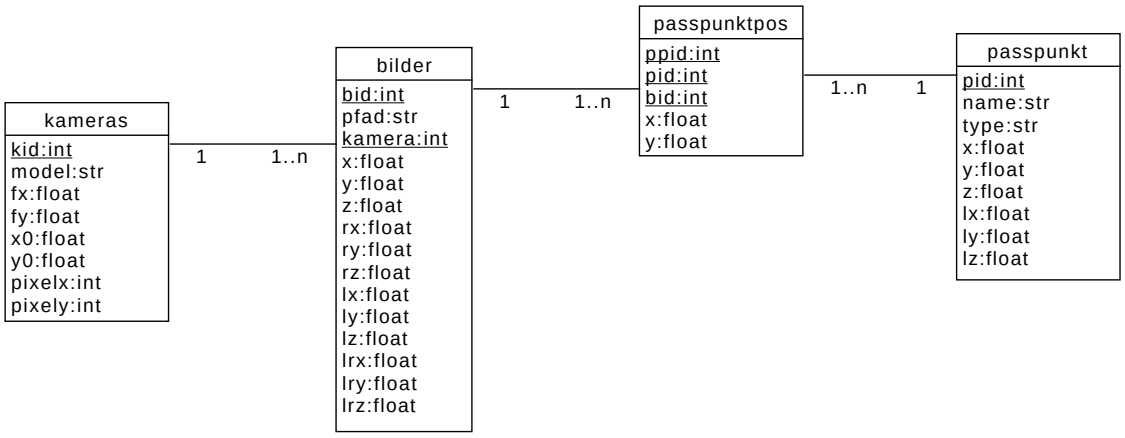

Abbildung 4.3.: Datenbank-Struktur

Das Backend wurde in Python entwickelt. Die Pakete orientieren sich an den Arbeitsschritten aus dem Ablaufdiagramm (Abbildung 2.1). Die einzelnen Python-Module greifen auf die oben beschriebene SQLite-Datenbank zu. Die einzelnen Module sind der Übersicht in Abbildung 4.4 zu entnehmen.

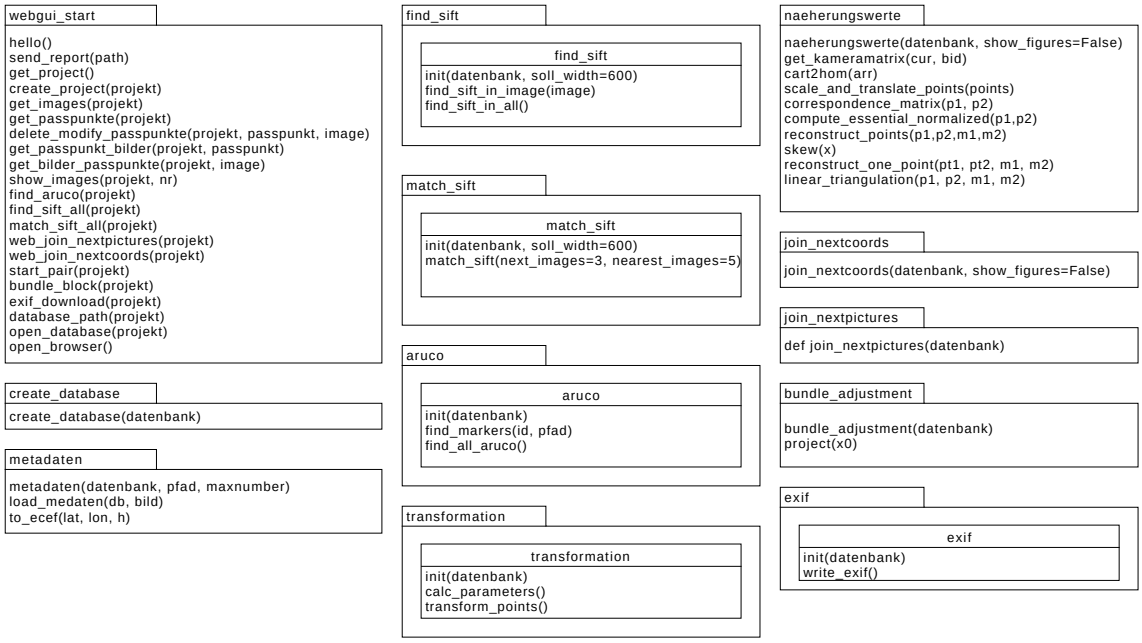

Abbildung 4.4.: Implementierung der Python-Pakete

Über das Webstart-Modul werden dann die einzelnen Module über eine Website mit TypeScript angesprochen. Die einzelnen TypeScript-Klassen sind der Abbildung 4.5 zu entnehmen. Hierbei wurde darauf Wert gelegt, die einzelnen Funktionen modular zu halten, sodass die Möglichkeit besteht, das System mit weiteren Modulen zu erweitern.

4.4. Untersuchungen

4.4.1. 3D-Modell aus Fokusstacking

- Automatisierter Fokusstacking
- keine Beachtung der festen Ausrichtung
- Transformation über SIRF und Homographie

4.4.2. Brennweitenänderung durch Fokussierung

- Charuco-Kalibriermuster

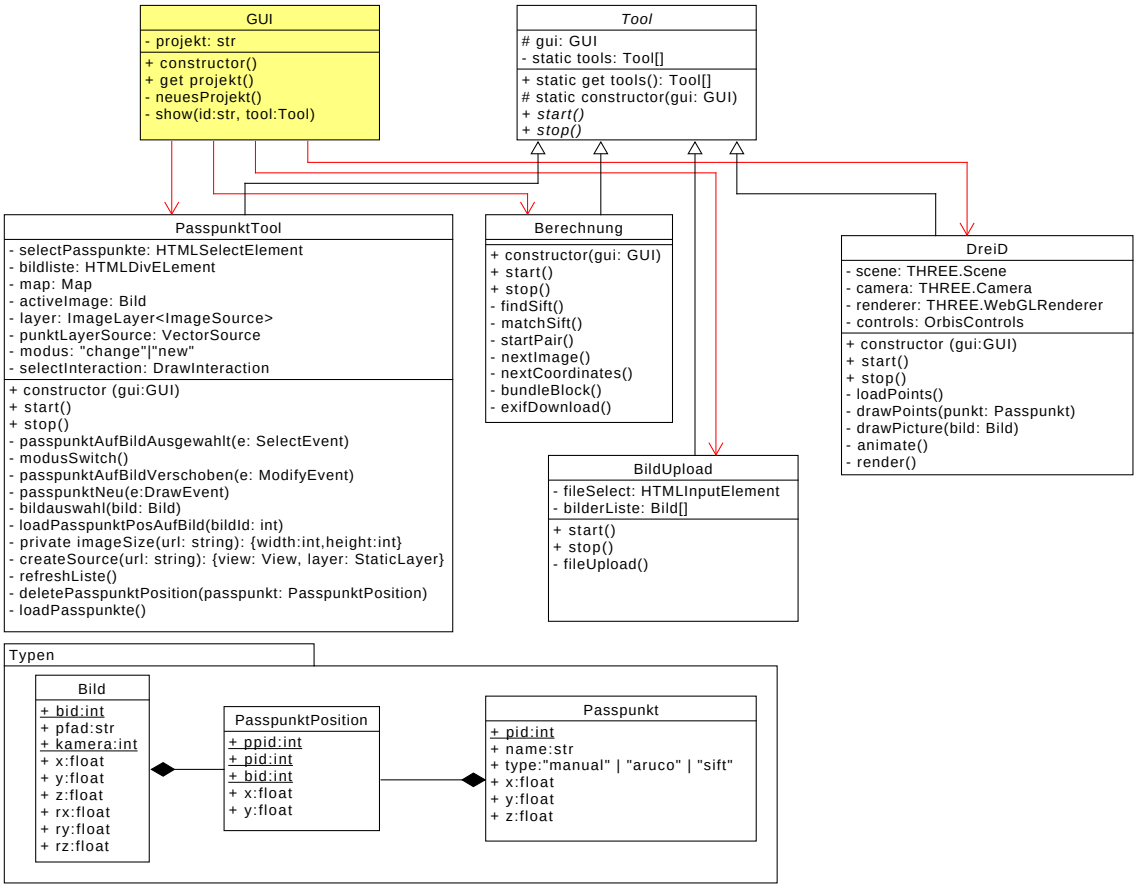

Abbildung 4.5.: Implementierung der TypeScript-Klassen

- Kamera und Board unbewegt
- 11 verschiedene Fokussierungen [0,5; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10]
- Fokussierung nacheinander und 4 Wiederhoungen
- Erkennung des Musters
- Analyse der Unterschiede/Vergrößerung

4.4.3. Änderung der Verzeichnung

- Kamera fest positioniert (keine Änderungen durch Schwerkraft)
- jeweils festen Fokus
- ChAruco-Board wird bewegt

4.5. Vorgehen

Die Programmierung des Systemes erfolgte iterativ. Einzelne Arbeitspakete wurden in einem Jupyter-Notebook ausprobiert und dann, wenn dieser Schritt erfolgreich war, in den Gesamtworkflow integriert. Größtenteils wurden der Python-Code objektorientiert und typisiert geschrieben, einzelne Module sind jedoch noch aus der Prototyp-Phase funktionsbasierend programmiert.

4.5.1. Python-Bibliotheken

Es wurde, wenn möglich, auf Python-Bibliotheken zurückgegriffen. Hierdurch sollte der Programmieraufwand verringert und auf bereits getesteten Code gesetzt werden. Wo dieses nicht möglich war, wurden Funktionen entsprechend Code-Beispielen aus GitHub oder „Rezepten“ aus dem Werk von Hartley & Zisserman programmiert.

OpenCV

ist eine Bibliothek für Bildbearbeitung und maschinelles Sehen. Sie ist weit verbreitet und bietet viele photogrammetrische Funktionen. Viele der unter Abschnitt 2.4 beschriebenen Schritte wurden mit dieser Bibliothek durchgeführt. (OpenCV team, 2023) Es zeigten sich jedoch Probleme bei der Berechnung der Essentiellen Matrix, sodass diese mit Hilfe eines Codebeispieles von Quek (2021) auf Grundlagen von Hartley & Zisserman (2003) berechnet wurde.

NumPy

bietet neben vielen weiteren Funktionen die Möglichkeit der Matrizenrechnung. Diese wurde für viele Berechnungen benötigt.

SciPy

wurde für die Berechnung der Bündelblockausgleichung verwendet. Der manuelle Ansatz mit den Formeln aus Luhmann (2018) unter Nutzung von NumPy war sehr ressourcenlastig. Unter Verwendung von SciPy und der Projektionsgleichung konnte die Berechnungsdauer stark dezimiert werden.

Flask

wurde genutzt um die Weboberfläche bereitzustellen und die Kommunikation zwischen dem Python und dem HTML/TypeScript-Modulen sicherzustellen. Die Weboberfläche selber muss nur einmalig kompiliert werden und wird dann von einen Flask-Webserver zur Verfügung gestellt. Per REST-Abfragen werden dann Daten zwischen TypeScript (bzw. nach dem Kompilieren eigentlich JavaScript) und Python ausgetauscht.

OpenSFM

wurde zwischenzeitlich verwendet um photogrammetrische Berechnung durchzuführen. Jedoch wurde dieses aufgrund mangelnder Anpassungsmöglichkeiten wieder verworfen. OpenSFM basiert aber auch unter anderem auf OpenCV. Der Code von OpenSFM wurde in der Prototyping-Phase zur Ideenfindung genutzt.

Erstaunlicherweise wurde kein brauchbares Paket zur Durchführung einer Helmert-Transformation mittels identischer Punkte gefunden. Daher wurde hier eine Funktion geschrieben, die eine Abbildungsmatrix mit Hilfe der Methode der kleinsten Quadrate errechnet. Nachteilig ist bei dieser Lösung jedoch, dass diese Ausgleichung auch Scherungen und unterschiedliche Maßstäbe für die einzelnen Koordinatenkomponenten unterstützt. Eigentlich wäre jedoch eine Transformation mit einem festen Maßstab sinnvoller, da die errechnete Struktur durch die Bündelblockausgleichung bereits deutlich besser der Realität entspricht als die GNSS-Koordinaten, die zur Transformation genutzt wurden. So werden die Daten aktuell an dieser Stelle wieder verschlechtert.

4.5.2. TypeScript/JavaScript-Module

Wie bereits erläutert wurde die GUI in Form einer Webseite entwickelt. Entsprechend wurden hier JavaScript-Module verwendet. Da TypeScript durch seinen Kompiler wieder zu JavaScript umgeformt wird, kann hier auch auf die große Auswahl von JavaScript-Bibliotheken aus dem Node Package Manager zurückgriffen werden.

OpenLayers

ist eigentlich zur Anzeige von Online-Karten gedacht. In diesem Fall wurde es zur Darstellung der Passpunkte auf den Bildern genutzt. Es bietet einfache und weitläufig bekannte Funktionen zum Zoomen und Digitalisieren. Statt einer Karte wurde in dem entsprechenden Fenster das Bild und die Passpunkte als Marker auf diesem visualisiert. Auch die Editierfunktionen wurden hieraus genutzt.

Three.JS

ist eine Bibliothek zur Darstellung von 3D-Inhalten. Diese wurde hier genutzt um eine Vorschau der errechneten 3D-Koordinaten bereitzustellen.

5. Systemkalibrierung

- Kameramodellierung
- Kamerakalibrierung
- Kameraausrichtung

6. Anwendungsversuche

- Kameraanzahl
- Drehteller
- Vergleichsmessung
-

7. Ausblick und Fazit

Vor allem das Erzeugen der Näherungswerte in Vorbereitung der Bündelblockausgleichung benötigte deutlich mehr Zeit und Theorieverständnis als gedacht. Daher wurde leider nicht alle ursprünglich geplanten Features umgesetzt. Aufgrund von Krankheit und anderen Uni-Projekten konnte dann zusätzlich auch nicht so viel Zeit in der zweiten Semesterhälfte in das Projekt gesteckt werden, wie eigentlich ursprünglich gedacht. Es sind bisher beispielsweise keine Nebenbedingungen möglich - ein Festlegen eines Maßstabes aufgrund einer bekannten Strecke ist so nicht möglich und auch nicht die Optimierung der Ausrichtung durch die Angabe gleich hoher Punkte. Auch wird aktuell nur die Position, nicht jedoch die bereits errechnete Drehung in den EXIF-Daten gespeichert. Hierfür müsste noch eine Umrechnung der Drehung aus dem System der ECEF-Koordinaten in die für EXIF-Daten übliche Ausrichtung an der Lotrichtung der Ortes des Bildes erfolgen. Ein weiteres offenes Problem ist die bereits erwähnte Transformation des lokalen Koordinatensystemes. Hier müsste noch die Ausgleichung so optimiert werden, dass nur ein Maßstab und keine Scherung verwendet wird.

Neben den erwähnten fehlenden Funktionen wäre als weitere Erweiterungen eine Berechnung einer dichten Punktwolke denkbar. Entsprechende Bibliotheken wurden während der Entwicklung entdeckt und schienen relativ leicht einbaubar. So würde die Software zu einer Komplettlösung für Structure-from-Motion-Punktwolken aus Bildern werden.

Im Gesamten sorgte das Projekt dafür, ein tiefergehendes Verständnis von Photogrammmetrie im Allgemeinen und Structure-from-Motion im Speziellen zu erarbeiten sowie vor allem die Probleme und Schwierigkeiten kennenzulernen.

Literaturverzeichnis

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

Hartley, Richard; Zisserman, Andrew (2003): Multiple View Geometry in Computer Vision. Cambridge University Press, Cambridge, Vereinigtes Königreich.

Luhmann, Thomas (2018): Nahbereichsphotogrammetrie Grundlagen - Methoden - Beispiele. 4. Auflage, Wichmann.

Luhmann, Thomas (2023): Nahbereichsphotogrammetrie Grundlagen - Methoden - Beispiele. 5. Auflage, Wichmann.

OpenCV team (2023): OpenCV. *https://opencv.org/*. (letzter Aufruf: 20. März 2023).

Quek, Alyssa (2021): 3D reconstruction. GitHub, *https://github.com/alyssaq/3Dreconstruction*. (letzter Aufruf: 20. März 2023).

Raspberry Pi Foundation (2023): Raspberry pi documentation - camera. Cambridge, Vereinigtes Königreich, *https://www.raspberrypi.com/documentation/accessories/camera.html*. (letzter Aufruf: 22. Februar 2024).

Schneider, Carl-Thomas; Sinnreich, Kurt (1993): Optical 3-D measurement systems for quality control in industry. *International Archives of Photogrammetry and Remote Sensing*, Band 29: S. 56–56, International Society for Photogrammetry and Remote Sensing.

Toffanin, Piero (2019): OpenDroneMap: the missing guide. MasseranoLabs LLC.

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang

A. Bedienungsanleitung

A.1. Zweck

Ziel des Systemes 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. Inbetriebnahme

Beim Aufstellen ist darauf zu achten, keine starke, seitliche Lichtquellen um das System herum zu haben, wie beispielsweise auch Fensterflächen. Diese könnten die Belichtung der Bilder beeinflussen und so die Qualität der 3D-Modelle negativ beeinflussen. Gegebenenfalls muss für Verschattung gesorgt werden.

Die Berechnung des 3D-Modelles erfolgt auf einem externen Rechner. Hier kann wahlweise Agisoft Metashape oder OpenDroneMap genutzt werden. Die Software muss auf dem Rechner installiert sein und die Bilder müssen auf diesen übertragen werden. Die Übertragung kann über USB-Sticks oder automatisch über eine Netzwerkverbindungen erfolgen. Hierfür muss das System Java installiert haben und die mitgelieferte Verbindungssoftware auf dem Rechner gestartet sein (siehe Abschnitt A.3).

Das System startet bei Anschluss an eine Stromversorgung selbstständig. Da die Gefahr besteht, dass die kamerasteuernden Raspberry Pi Zero Daten verlieren, wenn die Stromversorgung unterbrochen wird, sollte das System immer ordnungsgemäß heruntergefahren werden und auf eine zuverlässige Stromversorgung 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 Stromversorgung kann getrennt werden.

A.3. 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 Java installiert sein. Für die Nutzung von Metashape muss eine Lizenz vorhanden sein und neben der ausführbaren jar-Datei abgelegt werden. Für OpenDroneMap muss die Software in Form von NodeODM auf dem Rechner installiert sein. Die Verbindung zu einem Server ist nicht implementiert.

A.4. Kalibrierung

Die letzten Koordinaten der Passpunkte wird 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, wo durch dann die Koordinaten der Passpunkte neu bestimmt werden können.

Eine Kalibrierung mit Bordmitteln ist bisher nicht umgesetzt.

A.5. Durchführung

A.6. Auswertung

A.7. Wartung

A.8. Fehlerbehebung

B. Teileliste

B.1. Mechanische Bauteile

Bezeichnung	Anzahl	Einheit	pro Einheit	Gesamtpreis
Aluminium Strebenprofil Nut 6 Typ B	5,8 m	0,1 m	0,44 EUR	25,52 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	40	1	1,40 EUR	56,00 EUR
Zylinderschraube M4	40	100	3,80 EUR	3,80 EUR
Scheibe M4	40	100	1,85 EUR	1,85 EUR
Winkelprofil 30 x 500mm	2 m	2 m	23,96 EUR	23,96 EUR
				219,53 EUR

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

B.2. Elektronische Bauteile

Bezeichnung	Anzahl	Einheit	pro Einheit	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 + Flachbandkabel	24	1	3,60 EUR	86,40EUR
Speicherkarte 32 GB	25	1	5,95 EUR	142,80 EUR
LED-Streifen	1	1	31,10 EUR	31,10 EUR
Stromversorgung 12V 3,5 A	1	1	11,70 EUR	11,70 EUR
Stromversorgung 5V 7 A	2	1	18,20 EUR	36,40 EUR
Buchse für Netzteile	3	1	2,30 EUR	6,90 EUR
Litze 2*0,75		10	2,50 EUR	2,50 EUR
				1447,01 EUR

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

Erklärung

Hiermit versichere ich, dass ich die beiliegende Master-Thesis ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe.

Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Hamburg, den 01. April. 2024

Ort, Datum Florian Timm