HafenCity Universität Hamburg

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

Semi-automatische Bildorientierung für Structure-from-Motion

Geodäsie und Geoinformatik

GIS-Programmierung

Wintersemester 2022/2023

Florian Timm (6028121)

Abgabedatum: 24. März 2023

Inhaltsverzeichnis

1 Konzept

Structure-from-Motion bietet die Möglichkeit mit einfachsten (Hardware-)Mitteln 3D-Modelle bzw. farbige Punktwolken zu erzeugen - beispielsweise aus Luftbildern von unbemannten Luftfahrzeugen (Unmanned Aerial Vehicles, UAV). Die meisten, voralem die Open-Source-Lösungen, nutzen hierbei die automatische Detektion von Verknüpfungspunkten. Das Setzen des Maßstabes oder anderen festen Bedingungen ist hier nicht möglich. Es werden maximal die GNSS-Koordinaten des Bildes und ggf. die Ausrichtung aus den EXIF-Daten der Bilder genutzt.

Zur Verbesserung des Matchings von Punkten ist es daher geplant, ein Tool zu entwickeln, mit dem das manuelle Erzeugen von Passpunkten möglich ist. Außerdem sollen auch vor Ort angebrachte ArUco-Marker erkannt und verwendet werden, um den Aufwand der Nachbearbeitung zu verkleinern. Eine Angabe von Abständen zwischen zwei Verknüpfungspunkten bzw. das Setzen von Nebenbedingungen, wie z.B. das zwei Punkte übereinander liegen, die gleiche Höhe haben oder die X-Achse bilden, sind weitere Erweiterungsideen. Die ausgeglichenen Daten sollen dann den Eingangsbildern als EXIF-Tags angehängt werden. So wird eine Weiterverarbeitung in verschiedensten Softwarepaketen (OpenDroneMap, VisualSFM, Agisoft MetaShape) ermöglicht.

2 Nutzung

2.1 Installation und Start

Zum Betrieb wird ein Webbrowser, Python und eine Reihe von Python-Paketen benötigt. Die genauen Pythonpakete sind in einer beigefügten *requirements.txt* aufgelistet. Auf die Pakete wird auch nochmal im Unterabschnitt 4.3 eingegangen. Das Programm ist außerdem unter https://github.com/FlorianTimm/SFMBuendelBlock downloadbar.

Zum Start muss im Programmverzeichnis das Programm mit

```
python ./webGUI/webgui_start.py
```

gestartet werden. Der Browser sollte sich automatisch öffnen. Alternativ kann der Link aus dem nächsten Absatz verwendet werden.

2.2 Bedienung über die Weboberfläche

Alle Funktionen können über eine Weboberfläche gesteuert werden. Hierzu wird ein aktueller und standardkonformer Webbrowser benötigt. Die Bedienoberfläche ist unter http://127.0.0.1:2000 erreichbar.

2.2.1 Anlegen/Auswählen eines Projektes

Mit dem Button *Neu...* kann ein neues Projekt angelegt werden. Der Name muss einem gültigen Ordner-Namen entsprechen. Im Projektverzeichnis wird dann ein entsprechender Ordner und eine SQLite-Datenbank angelegt. Nach dem Anlegen muss die Weboberfläche einmal manuell neu geladen werden. Alternativ kann aus dem Drop-Down-Menü am oberen Bildrand ein bestehendes Projekt ausgewählt werden.

2.2.2 Hinzufügen von Bildern

In der Ansicht *Bilder* besteht die Möglichkeit, neue Bilder dem Projekt hinzuzufügen. Hier können Bilder aus dem Datei-System über den Button *Dateien auswählen* gewählt und dem Projektverzeichnis hinzugefügt werden. Aus Performancegründen sollten hier nicht allzu viele Bilder hinzugefügt werden.

2.2.3 Erkennen von ArUco-Markern

Über die Schaltfläche *ArUco-Marker in allen Bildern suchen* in der Ansicht *Bilder* werden diese Marker in allen Bildern gesucht. Ein PDF mit den Druckvorlagen der Marker liegt dem Programm bei.

2.2.4 Anlegen von Verknüpfungspunkten

Über die Ansicht *Passpunkte* können manuelle Passpunkte festgelegt werden und automatisch erzeugte Punkte überprüft werden. Die Bilder auf der linken Seite ermöglichen den Wechsel zu dem entsprechenden Bild. In der Displaymitte wird das Bild dann in groß mit der Überlagerung der Passpunkte angezeigt. Es handelt sich hierbei technisch um eine Webkarte, entsprechend kann hier navigiert und vergrößert werden. Am rechten Rand sind im Drop-Down-Menü die aktuell vorhandenen Passpunkte auswählbar. Darunter werden die aktuellen Markierungen dieses Passpunktes in den Bildern angezeigt. Beim Linksklick auf eines dieser Ausschnitte wird das entsprechende Bild geöffnet. Mit einem Rechtsklick kann die Verknüpfung entfernt werden, beispielweise bei Punktverwechslungen. Alternativ kann der Punkt auch in der großen Bildansicht ausgewählt

und verschoben werden - hierfür muss der Radiobutton oben rechts auf *Ändern* stehen. Durch die Auswahl von *Neu* können neue Verknüpfungen erzeugt werden. Falls der Punkt noch nicht im Drop-Down-Menü vorhanden ist, kann hier *Neu...* ausgewählt werden. Dann wird nach dem Setzen durch einen Klick auf das Bild ein Name abgefragt. Beim nächsten Laden des Passpunkt-Modules wird dieser dann mit in der Auswahl angezeigt.

2.2.5 SIFT-Marker

Falls nicht ausreichend ArUco-Marker vorhanden sind und keine manuellen Punkte gesetzt werden sollen, besteht die Möglichkeit, eine Feature-Erkennung zu nutzen. Hierfür können unter der Ansicht *Berechnung* die Buttons *SIFT finden* und *SIFT matchen* genutzt werden. Hierdurch werden alle Bilder nach entsprechenden Features durchsucht und ähnliche Features miteinander verknüpft.

2.2.6 Verknüpfen von Bildern

Um die eigentliche Verknüpfung der Bilder zu starten, wird die Schaltfläche *Startpaar* in der Ansicht *Berechnung* genutzt. Hiermit wird das Bildpaar ausgewählt, dass am meisten gemeinsame Punkte hat. Eine manuelle Auswahl ist im Moment leider noch nicht möglich.

Um weitere Bilder anzuknüpfen, können dann mit dem Button *Neue Koordinaten berechnen* lokale Koordinaten von Passpunkten berechnet werden. Mit Druck auf den Button *Neue Bilder mergen* werden über diese Passpunkte Position und Drehung von neuen Bildern mittels Rückwärtsschnitt berechnet. Zwischendurch und zum Abschluss sollte dann eine Bündelblockausgleichung (siehe Unterunterabschnitt 2.2.7) durchgeführt werden.

Meldungen werden nur in der Kommandozeile angezeigt - gegebenenfalls können hier Hinweise auf Fehler angezeigt werden.

2.2.7 Bündelblockausgleichung

Durch die Bündelblockausgleichung werden die Fehler in der Verknüpfung der Bilder minimiert und verteilt. Diese ist mit der Schaltfläche *Bündelblock* in der Ansicht *Berechnung* möglich. Auch hier werden Meldungen, wie zum Beispiel die erreichte Genauigkeit nur in der Kommandozeile angezeigt.

2.2.8 3D-Ansicht

Die Ansicht *3D* bietet die Möglichkeit, sich die bereits berechneten 3D-Daten anzusehen. Kamerapositionen entsprechen hierbei einer roten Box, manuelle Verknüpfungspunkte sind grün, ArUco-Marker sind blau und SIFT-Features sind grau.

2.2.9 Schreiben von EXIF-Daten

Um die Bilder in einer anderen Software nutzen zu können, können die Bilder mit dem Button *Bilder mit EXIF speichern* in der Ansicht *Berechnung* im Projektverzeichnis mit den neuen Koordinaten gespeichert werden. Hierfür wird das lokale Ausgleichungssystem über die GNSS-Koordinaten der Ursprungsbilder transformiert. Dieser Schritt ist entsprechend nur möglich, wenn die Ausgangsbilder Koordinatenangaben hatten.

2.3 Nutzung als Bibliothek

Die einzelnen Funktionen wurden so angelegt, dass Sie auch aus anderen Python-Skripten heraus nutzbar sind, beispielsweise in einem Jupyter-Notebook. Ein Beispiel-Notebook liegt dem Quellcode bei.

3 Technischer Hintergrund

Das Programm stellt einen Teil einer Structure-from-Motion-Pipeline dar (siehe Abbildung 1, vgl. Luhmann 2018, S. 491). Auf die einzelnen Schritte wird in den folgenden Absätzen eingegangen.

3.1 Bilder

Die Berechnung der Tiefeninformationen ist bei Structure-from-Motion nur möglich aufgrund der Bewegung der Kamera zwischen den Bildern. 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.

Kameraeinstellungen Jede Einstellung der Kameraoptik verändert die innere Orientierung und auch jede Kamera (auch einer Modellreihe) ist unterschiedlich. Daher sollten die Bilder möglichst mit einer Kamera mit festen Einstellungen (Brennweite, Blende, Objektiv) aufgenommen werden. Bei der Empfindlichkeit (ISO-Zahl) oder Belichtungszeit wären Änderungen unproblematisch für die innere Orientierung und

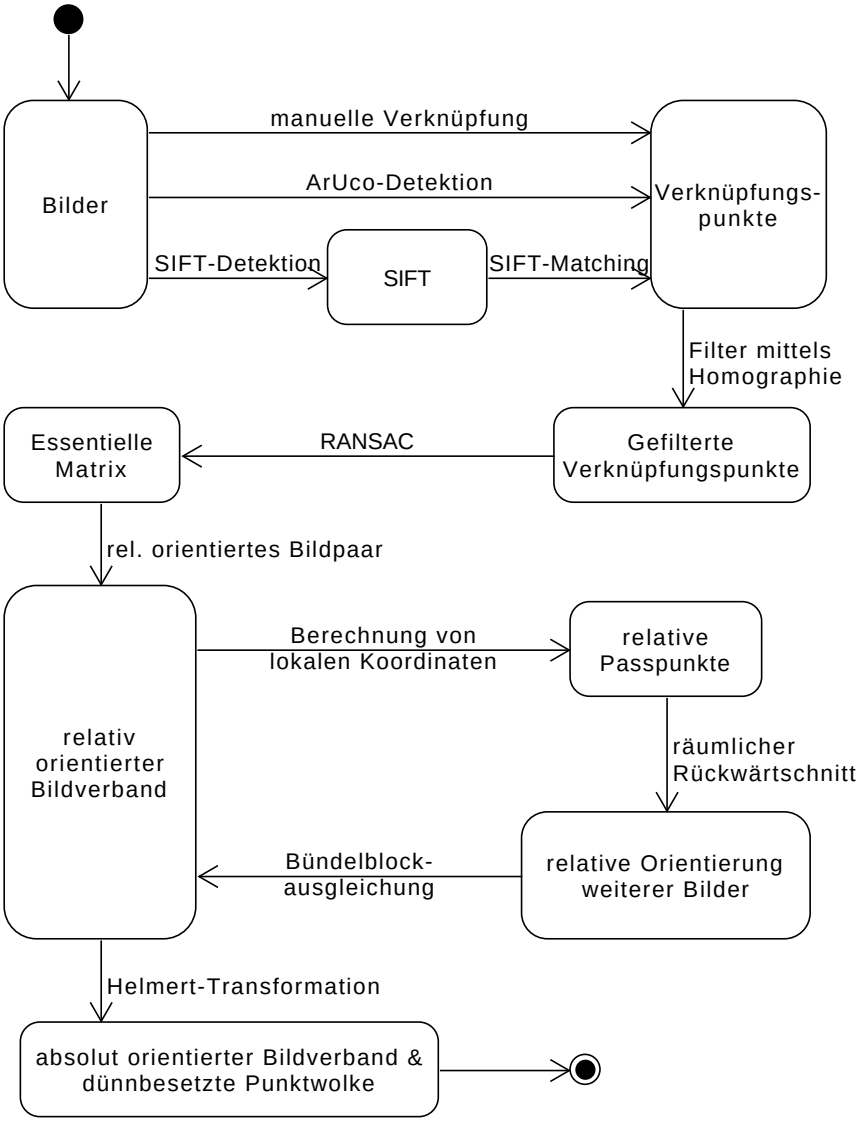

Abbildung 1: Ablauf der Bildverknüpfung

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.

Überlappung und Bildinhalte Da die Bilder durch identische Punkte verbunden werden, müssen die Bildinhalte sich überlappen. Außerdem sollten alle Bilder strukturierte Bereiche aufweisen. (Toffanin, 2019, S. 146f)

Bewegungen 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 Digitalkamera 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)

Abstand bzw. Höhe 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)

3.2 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.

3.2.1 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.

3.2.2 SIFT

Die SIFT-Methode liefert Verknüpfungspunkte aus Mustern auf den fotografierten 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 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 diesen kann dann die Übereinstimmung von zwei Markern in zwei Bildern bestimmt werden, auch wenn die Bilder zueinander gekippt oder gedreht sind. (Luhmann, 2018, S. 483)

3.3 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.

3.3.1 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 1, nach Hartley & Zisserman, 2003, S. 244 und Luhmann, 2018, S. 288)

$$x' = P \cdot X \tag{1}$$

$$P = K \cdot [R|X_0] \tag{2}$$

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

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)

3.3.2 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:

$$x'^T \cdot F \cdot x'' = 0 \tag{4}$$

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.

3.3.3 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)

3.3.4 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)

3.4 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 3.3.1 beschriebene Abbildungsgleichung. Als Ergebnis erhält man die ausgeglichenen Parameter und Genauigkeitsangaben für diese. (Luhmann, 2018, S. 340)

3.5 Transformation

Die Berechnungen erfolgen alle in einem lokalen Koordinatensystem. Die GNSS-Koordinaten der Bilder werden nur genutzt, um potentielle Nachbarbilder zu identifizieren. Daher muss zum Schluss eine 3D-Transformation der lokalen Daten in ein weltweites Koordinatensystem erfolgen. Diese Transformation erfolgt erst in ein geozentrisches, kartesisches Koordinatensystem (ECEF), welches dann in Längen- und Breitengrade sowie Höhen über dem Ellipsoid umgerechnet werden. Die letztgenannten Angaben sind für die EXIF-Daten der Bilder notwendig.

4 Entwicklung

Die Entwicklung erfolgte mittels Prototyping. Die einzelnen Arbeitspakete wurden als eigenständige Jupyter-Notebooks programmiert und jeweils immer weiter bis zu der gewünschten Funktionalität entwickelt. Anschließend wurden die einzelnen Module zu einem Gesamtprogramm zusammengeführt und eine Weboberfläche zur Steuerung und Kontrolle erzeugt. Auf die einzelnen Problemstellungen wird im Unterabschnitt 4.3 eingegangen.

4.1 Konzeption

Entsprechend der benötigten Schritte aus Abschnitt 3 und Abbildung 1 wurde die Anwendungsfälle, die die Benutzeroberfläche ermöglichen soll, im Anwendungsfall-Diagramm in Abbildung 2 zusammengetragen.

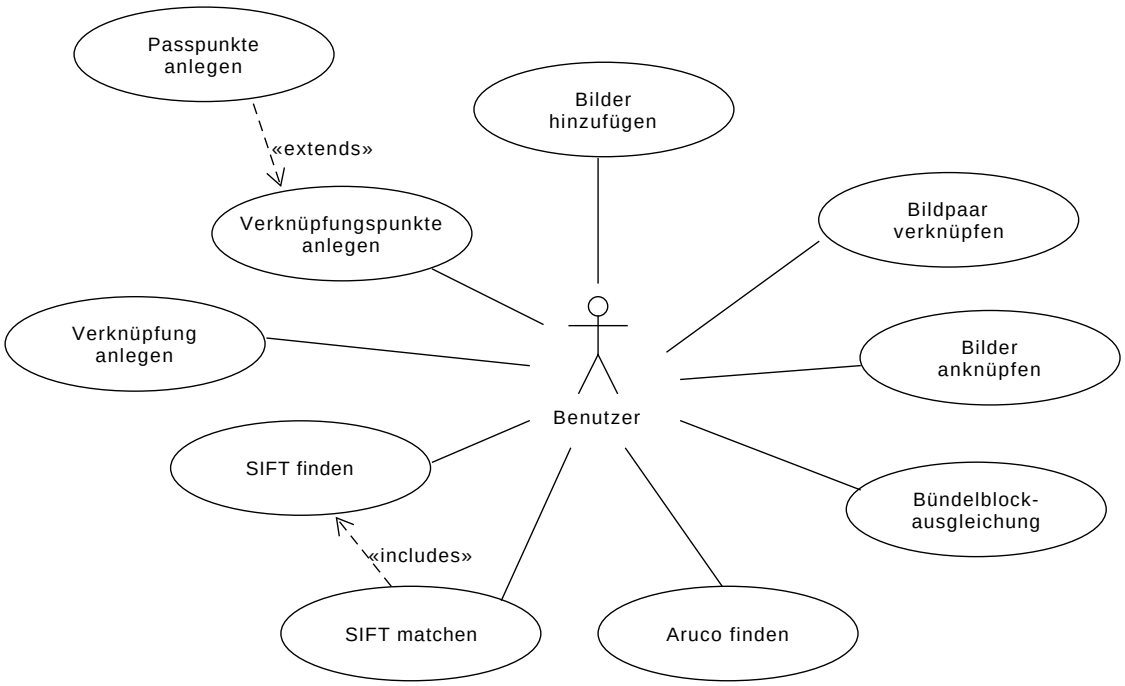

Abbildung 2: Anwendungsfall-Diagramm

Aus den benötigten Daten wurde das Domänen-Klassendiagramm aus Abbildung 3 erzeugt. Dieses zeigt vor allem die Abhängigkeiten der einzelnen Datensätze untereinander.

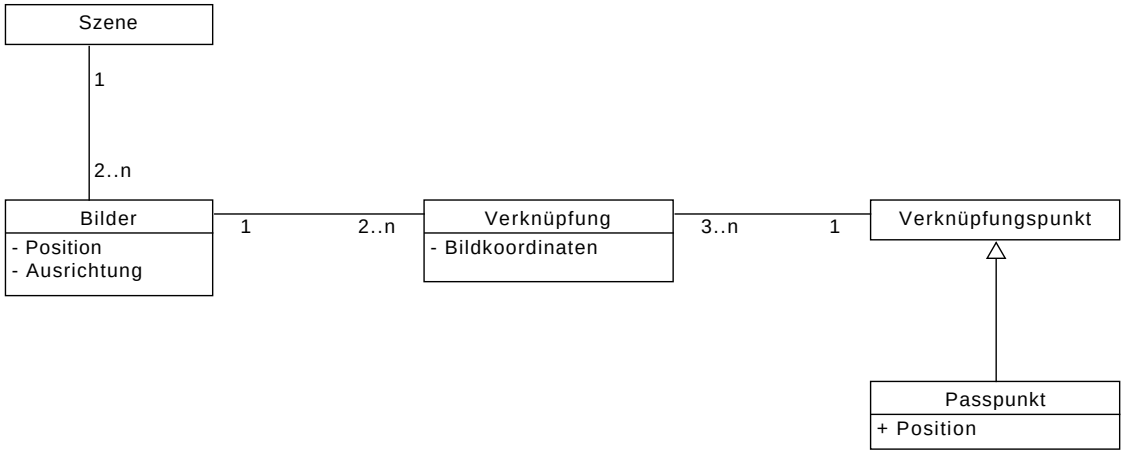

Abbildung 3: Domänen-Klassendiagramm

4.2 Implementierung

Dieses ist dann auch Grundlage für die Implementierung der SQLite-Datenbank. Der Datenbankaufbau ist Abbildung 4 zu entnehmen. Die Datenbank dient der Zwischenspeicherung der Passpunkte und der durchgeführten Berechnungen.

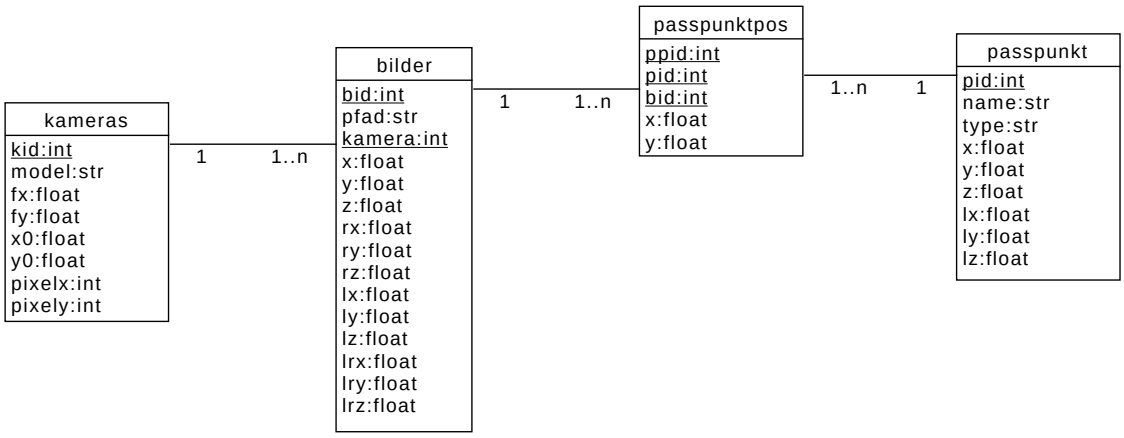

Abbildung 4: Datenbank-Struktur

Das Backend wurde in Python entwickelt. Die Pakete orientieren sich an den Arbeitsschritten aus dem Ablaufdiagramm (Abbildung 1). Die einzelnen Python-Module greifen auf die oben beschriebene SQLite-Datenbank zu. Die einzelnen Module sind der Übersicht in Abbildung 5 zu entnehmen.

Über das Webstart-Modul werden dann die einzelnen Module über eine Website mit TypeScript angesprochen. Die einzelnen TypeScript-Klassen sind der Abbildung 6 zu entnehmen. Hierbei wurde darauf Wert gelegt, die einzelnen Funktionen modular zu halten, sodass die Möglichkeit besteht, das System mit weiteren Modulen zu erweitern.

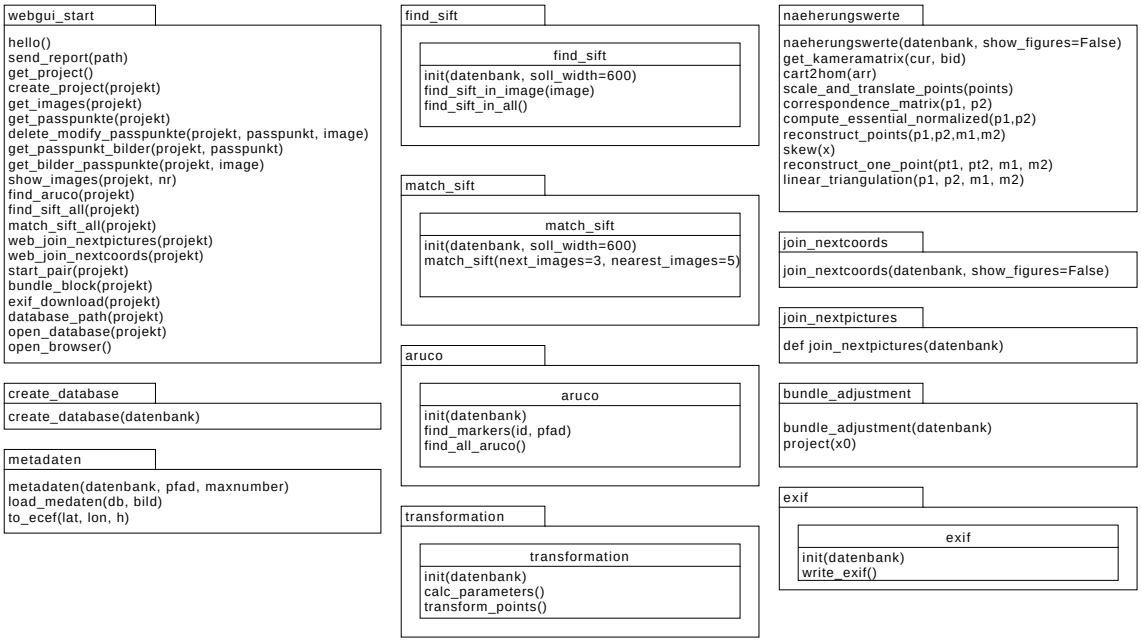

Abbildung 5: Implementierung der Python-Pakete

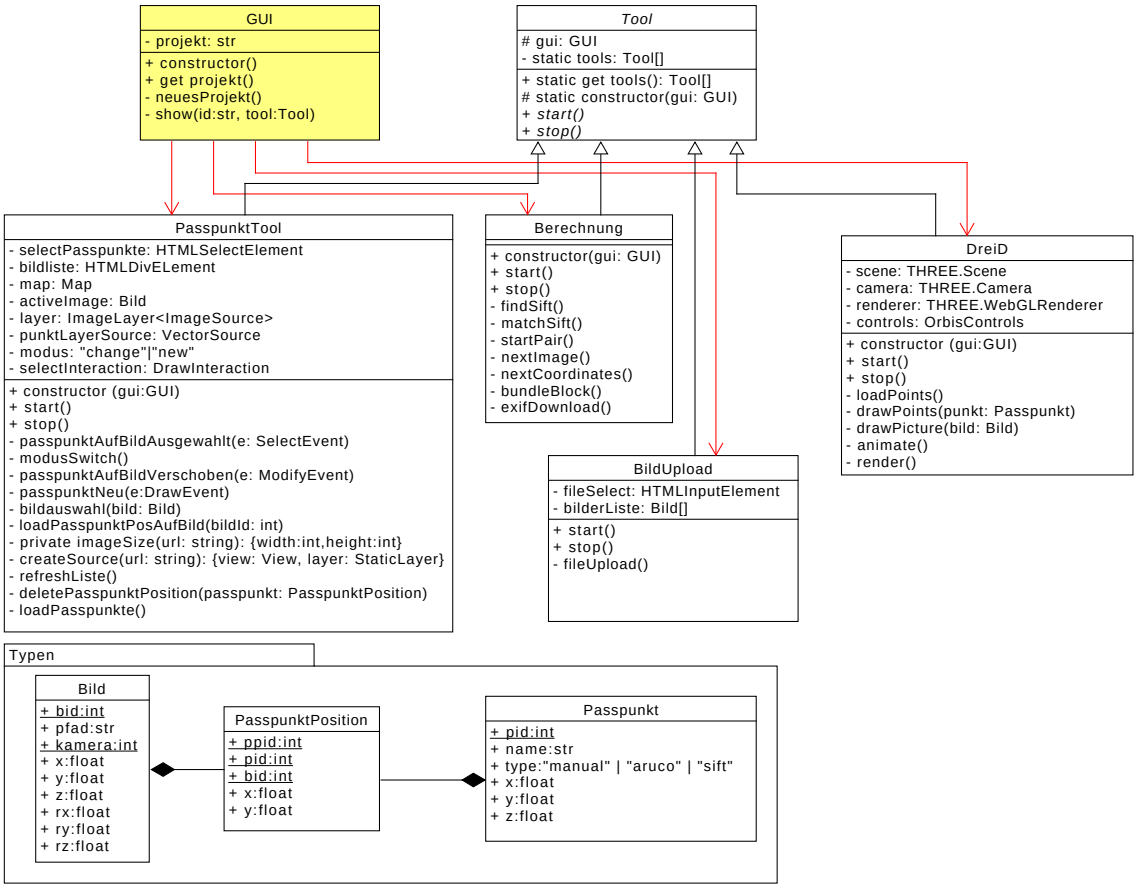

Abbildung 6: Implementierung der TypeScript-Klassen

4.3 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.

4.3.1 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 Unterabschnitt 3.3 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.

4.3.2 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 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

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, *https://www.content-select.com/index.php?id=bib_view&ean=9783879076413*.

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).

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

Abbildungsverzeichnis