Vorauswahl von Daten für die photogrammetrische Baufortschrittskontrolle

Wissenschaftliche Arbeit zur Erlangung des Grades
Bachelor of Science
an der Munich School of Engineering der Technischen Universität München.

Betreut von Prof. Dr.-Ing. André Borrmann
Felix Eickeler, M.Sc.
Lehrstuhl für Computergestützte Modellierung und Simulation

Eingereicht von Felix Nothdurft
[redacted]

Eingereicht am München, den 10. Oktober 2018

Anhang I

Erklärung

Ich versichere hiermit, dass ich die von mir eingereichte Abschlussarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

München, 10. Oktober 2018, Unterschrift

Inhaltsverzeichnis

1. Einleitung

Mit Building Information Modelling ist auch in die Baubranche die Digitalisierung eingezogen. Dabei ist das Ziel, alle Planungs-, Bau- und Betriebsschritte digital zu dokumentieren und zu verknüpfen. Hierfür ist insbesondere die Rückkopplung der Ist-Zustände fundamental. Dadurch kann der reale Baufortschritt mit der Planung verglichen werden.
Um den aktuellen Stand zu erfassen, wird mithilfe einer optischen Drohne die Baustelle aus verschiedenen Perspektiven aufgenommen. Durch Analyse der erzeugten Bilddaten kann im Nachgang eine Punktwolke der Baustelle erzeugt und diese mit dem Planungsmodell verglichen werden. Hierbei können Übereinstimmungen sowie Abweichungen im aktuellen Bauprozess detektiert werden.
Zur Erzeugung einer möglichst präzisen Punktwolke, sind in erster Linie große Mengen an Bildern gewünscht. Deshalb werden bei Flügen eher mehr, als weniger Bilder aufgenommen. Diese werden im Nachgang jedoch nicht sortiert oder anhand ihrer Güte kategorisiert. Problem dabei ist, dass insbesondere die Berechnung der Punktwolke extrem abhängig von der Anzahl der Aufnahmen ist. Aus rechen-technischer Sicht wäre es also interessant, das Datenset möglichst klein zu halten.
Um das Set zu verkleinern, aber trotzdem die Qualität der Punktwolke (nahezu) konstant zu halten, müssen die Bilder kategorisiert werden. Hierzu wird im Rahmen dieser Arbeit einerseits eine Analyse der Bildschärfe, andererseits eine räumliche Anordnung der Drohne zum Zeitpunkt der Aufnahme untersucht. Auf Basis dieser Informationen, werden die Bilder bewertet und für eine weitere Punktwolkenberechnung aussortiert.

2. Stand der Technik

2.1. Building Information Modell

Die Idee des Building Information Modell (BIM) wurde in den 1990er Jahren zum ersten Mal entwickelt. Mithilfe vereinheitlichter Datenformate soll es gelingen, die Baubranche zu digitalisieren und einen Standard zu schaffen, der für den ganzen Lebenszyklus des Gebäudes genutzt werden kann. Dies bedeutet, dass von der Planung über die Konstruktion und statische Auslegung bis hin zur aktiven Nutzungsplanung alles innerhalb dieses Modells dargestellt werden kann. Durch eine zentrale Verwaltung der Datensätze werden anfallende Änderungen zeitnah an alle Beteiligten weitergegeben und die kooperative Zusammenarbeit gestärkt [Borrmann et al., 2015].
Zur Überprüfung der Baufortschritte, ist es besonders wichtig, dass nicht nur der finale Stand des Gebäudes modelliert wurde, sondern auch zu den Zeitpunkten der Überprüfung ein Referenzmodell existiert.

2.2. Generierung der Punktwolke aus Bilddaten

Im Gegensatz zum BIM wird die Punktwolke aus Messdaten entwickelt und versucht dadurch den aktuellen Stand des Bauvorhabens abzubilden [Braun et al., 2015]. Dabei kann dieser Prozess in vier Phasen eingeteilt werden:

1. data acquisition (I) [Datengenerierung],
2. orientation of the images (II) [Bildorientierung],
3. image matching (III) [Bildabgleich] and
4. co-registration (IV) [Braun et al., 2015, S. 71]

Datengenerierung

Zur Generierung der Daten werden zum einen Laserscanner zum anderen photogrammetrische Verfahren genutzt, wobei das Hauptaugenmerk auf der Photogrammetrie liegt.
Allgemein betrachtet handelt es sich bei der Photogrammetrie um „die berührungslose Aufnahme“ [Eipke, 2017, S. 3]. Im Rahmen dieser Arbeit sind dabei Kameras gemeint, welche zumeist in Verbindung mit Drohnen genutzt werden. Die Nutzung von Drohnen ermöglicht es dabei, Baustellen aus vielen verschiedenen Blickwinkeln verwacklungsarm aufzunehmen. Darüber hinaus gewährleisten Drohnen insbesondere durch die zusätzliche Flexibilität in der Höhe die Dreidimensionalität der Aufnahmepositionen.

Bildorientierung

Hierbei handelt es sich um die Suche nach Bildmerkmalen, welche in allen Bildern gesucht werden. Auf Basis dieser Merkmale können die Bilder und deren Aufnahmepunkt korrekt im Raum eingeordnet werden.
Spezielle Software (z.B. VisualSFM oder COLMAP) verarbeitet mithilfe des Structure from Motion (SFM) Prozesses die Bilder. Ziel ist es, alle Bilder in eine räumliche Abhängigkeit zueinander stellen zu können. Dies geschieht anhand von Merkmalen, wie Ecken oder Strukturübergängen die wiederholt in verschiedenen Bildern auftreten und durch deren Perspektive Rückschlüsse auf die Kameraposition zum Zeitpunkt der Aufnahme zulassen. Der Prozess ist stark abhängig von der Anzahl der Bilder und es konnte bewiesen werden, dass der Zeitaufwand in der Theorie quadratisch $O(n^2)$ ansteigt. In der Praxis hat sich jedoch gezeigt, dass der Zeitaufwand besonders für große Modelle linear zunimmt $O(n)$ [Wu, 2013]. In der Abbildung 1 ist diese Entwicklung sehr gut zu erkennen. Besonders interessant ist dabei die rote Linie, welche ein kreisförmig aufgenommenes Objekt darstellt. Aufgrund der hohen Zahl an wiederkehrenden Merkmalen in verschiedenen Bildern steigt hierbei die Rechenzeit stärker als bei anderen Rekonstruktionen an, jedoch weiterhin linear.

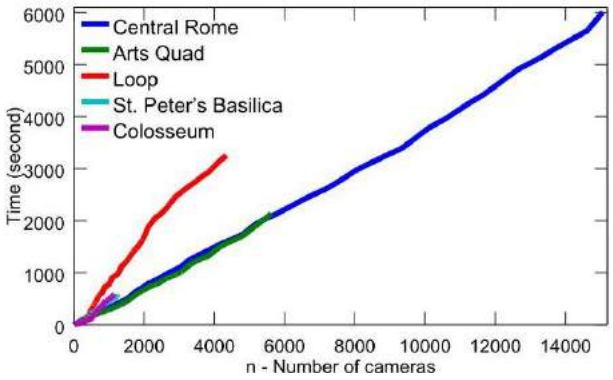


Abbildung 1 Verhältnis zwischen Aufnahmen und Rechenaufwand, [Wu, 2013, S. 7]

Bildabgleich

Nachdem eine genaue Positionierung der Bilder im Raum vorgenommen wurde, geht es in diesem Schritt um die dreidimensionale Rekonstruktion. Mithilfe der Semi-Global Matching (SGM) Technik wird aus vielen zweidimensionalen Aufnahmen, ein dreidimensionales Modell erzeugt [Hirschmüller, 2011]. Rein mathematisch betrachtet würden hierfür, ähnlich unserer Augen, zwei Aufnahmen reichen, was jedoch zu Rauschen und Ungenauigkeiten führen würde. Deshalb werden ausschließlich Punkte in das Modell aufgenommen, die in mindestens drei verschiedenen Bildern auftauchen [Braun et al., 2015]. Der Rechenaufwand steigt in diesem Schritt ..... an. (Hier fehlen noch verlässliche Daten über den Anstieg der Rechenzeiten

Co-Registrierung

Im Rahmen der Rekonstruktion kann es passieren, dass das Modell entweder falsch im Raum angeordnet oder falsch skaliert ist. Dies geschieht, da die Rekonstruktion zumeist keine Information über die absolute Ausrichtung oder Größe des Modells enthält. Um dem vorzubeugen, können mithilfe von vordefinierten Markern Längen umdefiniert werden. Dabei wird in der Realität die Position oder der Abstand zwischen mehreren Markern erfasst und dann als Parameter in das Modell übertragen. Anhand der vorhandenen Abhängigkeiten innerhalb des Modells, ist es nun möglich das gesamte Modell daran auszurichten.
Optimal ist es, wenn die Marker in allen Bauphasen sichtbar sind, sodass sie als Validierung in allen Modellen genutzt werden können. Dabei unterscheidet man zwischen zwei Typen von Markern: „Künstlich“ angelegte Marker, welche speziell für diese Baustelle installiert wurden, sowie „natürliche“ Marker, welche unabhängig von der Baustelle existieren z.B. Häuserfronten (siehe Abbildung 2). Damit eine gute Rekonstruktion im Modell möglich ist, dürfen die Marker nicht alleine fotografiert werden, sondern müssen stets mit Teilen der Baustelle zu sehen sein. Dadurch wird sichergestellt, dass die vordefinierten Längenverhältnisse auf das Modell angewendet werden können [Tuttas et al., 2017].

(a) (b)

Abbildung 2 Kontrollpunktarten: **(2a)** Photogrammetrische Marker für Nadir Aufnahmen. **(2b)** „Natürliche“ Kontrollpunkte in einer Innenstadt markiert durch rote Kreise (Fensterecken) [Tuttas et al., 2017, S. 7].

2.2.1. Qualitätssicherung der Punktwolke

Um eine möglichst hohe Präzision der Punktwolke zu erhalten sollten einige Regeln bei der Bildgenerierung beachtet werden.
Jedes Merkmal, welches in die Punktwolke übernommen wird, sollte in mindestens drei Aufnahmen auftauchen. Dadurch kann einerseits eine gewisse Robustheit gegenüber Unschärfen, zum anderen eine höhere Präzision der Punkte gewährleistet werden [Braun et al., 2015]. Um eine Konsistenz zwischen den verschiedenen Punktwolken erreichen zu können, sollten in allen Baufortschritte die selben Markerträger (siehe Kapitel 2.2) genutzt werden. Aus diesem Grund sind vegetative Bereiche, wie Bäume oder Sträucher, als natürliche Kontrollpunkte schlecht, umliegende Gebäude hingegen gut geeignet.
Bei der Bildkomposition sollte darauf geachtet werden, dass nicht ausschließlich die Bau-

stelle, sondern auch die Umgebung mit aufgenommen wird. Während dadurch einerseits die Rekonstruktion und positionelle Einordnung ermöglicht wird, kann andererseits der abfallenden Aufnahmequalität zum Rand hin vorgebeugt werden (Bildsensoren erzeugen in deren Mitte das beste Ergebnis [Helmers, 2013]).
Spiegelnde, transparente oder besonders gleichmäßige Oberflächen können im Laufe des Prozesses ebenfalls nicht erkannt werden, da diese nahezu keine Merkmale für die Rekonstruktion beinhalten, bzw. diese sich aus verschiedenen Blickwinkeln verändern. Selbiges gilt auch für Wasser, sodass größere Wasseransammlungen auf der Baustelle (z.B. Pfützen auf dem Dach) zu Fehlern im Modell führen können.
Darüber hinaus kann mit dem aktuellen Rekonstruktionsprozess ausschließlich bestimmt werden, ob sich an einer Stelle im Modell ein Festkörper befindet oder nicht. Dies kann insbesondere bei Schalungen oder Baugerüsten zu Problemen führen, da der Algorithmus lediglich erkennt, dass sich dort etwas befindet. Er ist jedoch nicht in der Lage zwischen Wänden und temporären Strukturen zu differenzieren, dies muss händisch vorgenommen werden.

2.2.2. Datenformat der Aufnahmen

Um eine fehlerfreie Verarbeitung der Bilder zu ermöglichen, ist ein durchgängiges Datenformat nötig. Deshalb wird sich in diesem Rahmen auf das am weitesten verbreitete Format fokussiert: JPEG [Furht, 2008]. Dies gewährleistet zum einen eine hohe Kompatibilität mit unterschiedlichsten Kameramodellen, zum anderen können dadurch viele vor-implementierte Methoden und Klassen der Datenverarbeitung genutzt werden. Alle anderen Datenformate werden im Rahmen dieser Arbeit nicht betrachtet, könnten jedoch durch eine Formartkonvertierung ebenfalls analysiert werden. Darüber hinaus wird angenommen, dass alle Bilder einen GPS-Stempel tragen, welcher für die spätere Selektion der Bilder nötig ist.

2.3. Vergleich Punktwolke zu BIM

Nach Erstellung der Punktwolke und Auswahl der korrekten Bauphase des BIM werden diese beiden Modelle verglichen. Dabei werden alle Bereiche des Modells entsprechend dreier Zustände klassifiziert:

- **As-planned:** Enthält die Informationen, ob ein Bereich zu dem entsprechenden Zeitpunkt schon gebaut sein soll oder nicht. Es handelt sich um das Referenzmodell an welchem die Punktwolke gemessen wird.
- **As-built:** Repräsentiert den „perfekten“ Scan des Modells und enthält alle Informationen über den aktuellen Stand der Baustelle. Da es ein Optimalzustand ist, wird dieser niemals so in der Realität vorkommen sondern dient hier lediglich als Referenz.
- **Detected:** Beschreibt, ob ein Element des As-built Zustand detektiert wurde oder nicht. Perfekterweise sollte dieser Zustand dem As-built Zustand entsprechen, in der Realität wird dieser jedoch immer nur angenähert [Braun et al., 2016, S. 3].

Mathematisch gesehen kann es aus diesen drei Zuständen mit dem jeweiligen Wertebereich (Ja/Nein) insgesamt $2^3 = 8$ Möglichkeiten geben:

#	as-planned	as-built	detected	color
1	-	-	-	
2	x	-	-	
3	x	x	-	
4	-	x	-	
5	-	x	x	
6	x	x	x	
7	x	-	x	
8	-	-	x	

Tabelle 1 Farbschema zur Klassifizierung der detektierten Elemente [Braun et al., 2016, S. 4]

Fall 1

Das Element ist zum Aufnahmezeitpunkt weder geplant noch gebaut oder detektiert, dies trifft auf alle Elemente zu, die später gebaut werden.

Fall 2

Das Element sollte laut Zeitplan existieren, wurde jedoch noch nicht gebaut und ebenfalls nicht detektiert. Dies deutet auf eine Verzögerung im Zeitplan hin.

Fall 3

Das Element wurde geplant und gebaut entsprechend dem Prozessplan, es konnte jedoch nicht detektiert werden. Mögliche Gründe hierfür werden in Unterabschnitt 2.2.1 „Qualitätssicherung der Punktwolke“ dargelegt.

Fall 4

In diesem Fall ist eine Komponente errichtet, aber weder geplant, noch detektiert. Dies kann lediglich auftreten, wenn eine Komponente schon früher gebaut wurde als im Zeitplan angedacht, sowie die Detektion gestört wurde. Dieser Fall ist ein rein theoretischer. In der Praxis ist er schwer bis gar nicht von Fall 1 zu unterscheiden.

Fall 5

Auch hier wurde, ähnlich Fall 4 das Element zu früh errichtet, es wurde jedoch zusätzlich detektiert.

Fall 6

Alle Kategorien stimmen überein, sodass ein Bauteil im entsprechenden Zeitplan rechtzeitig errichtet und detektiert wurde. Dies ist der Optimalfall.

Fall 7

Hierbei handelt es sich um einen kritischen Fall, da das Element geplant und detektiert ist, jedoch nicht gebaut wurde. Etwaiges Verhalten kann die Folge von Strukturen sein, die innerhalb der nächsten Tage fertig gestellt werden. Es kann aber auch aufgrund von schlechten matching Algorithmen oder schwach definierter Grenzen geschehen.

Fall 8

Dieser Fall ähnelt Fall 7 sehr stark und kann durch temporäre Stützstrukturen, wie Schalungen, ausgelöst werden. [Braun et al., 2016, S. 3–4]

Besonders gefährlich sind die Fälle, bei denen der as-built und der detected Zustand voneinander abweichen. Dabei verdeutlichen Fall 3 & 4 eine fehlende Detektion welche durch abgeleitete Bauteile gelöst werden können.
Die anderen beiden Fälle 7 & 8 bedeuten, dass Elemente zwar detektiert, diese jedoch noch nicht errichtet wurden. Ein solches Szenario ist oftmals die Folge von Stützstrukturen, die im Bauprozess temporär errichtet werden. Eine mögliche Lösung für dieses Problem ist die neue Elementklasse „In-Konstrukution“.

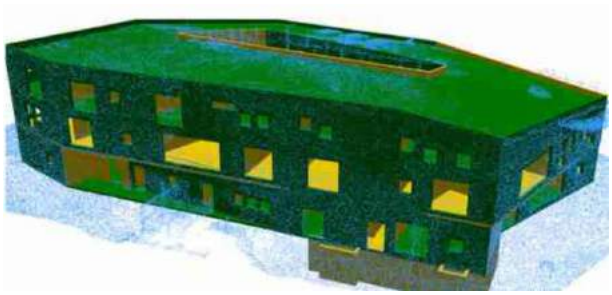


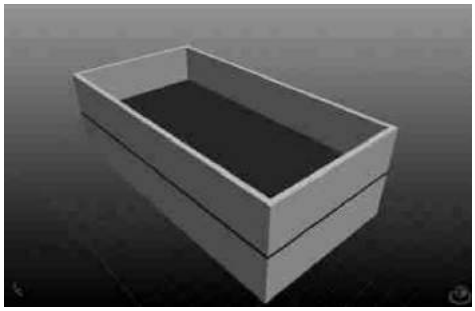
Abbildung 3 Das entwickelte Farbschema anhand einer Modellbaustelle, [Braun et al., 2016, S. 5]

2.3.1. Möglichkeiten zur Fehlervermeidung

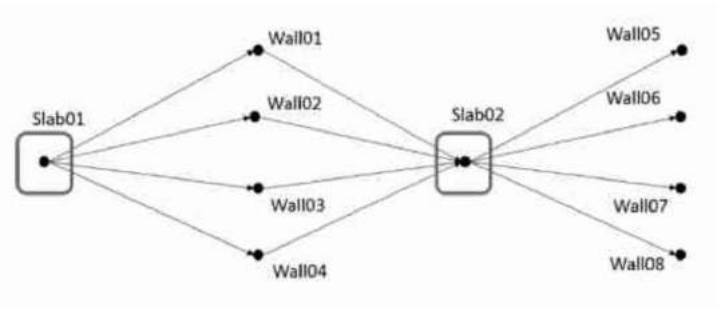
Um möglichen Falschdetektionen vorzubeugen kann mithilfe verschiedener Verfahren versucht werden diese zu vermeiden.

- **In-Konstruktion:** durch Einführung einer weiteren Elementklasse, In-Konstruktion, können entsprechende Stützstrukturen klassifiziert und modelliert werden.
- **Abgeleitete Bauteile:** Durch sogenannte Abhängigkeitsgraphen (siehe Abb. 4b) werden die Bauteile untereinander in einen Kausalzusammenhang gestellt. Diese Kausalkette leitet sich aus dem Modell 4a her. Beispielhaft gesprochen, wenn „Wall05“ detektiert wurde, so müssen alle vorangegangenen Elemente, „Wall01–04“ sowie „Slab01/02“ ebenfalls vollendet worden sein.
- **Vorangegangene Detektionen:** Elemente, die in einem früheren Bauabschnitt schon einmal detektiert wurden, sind auch in späteren weiterhin gültig, da diese nicht wieder „verschwinden“ können. Ein Problem dabei können fälschlich detektierte Elemente (z.B. Stütz-

strukturen) aus dem vorherigen Schritt darstellen, da sich diese Fehler somit durch das Modell fortpflanzen können [Braun et al., 2015, Braun et al., 2016].



(a) Beispielmodell zur Veranschaulichung der Abhängigkeiten



(b) Graph zur Visualisierung der Abhängigkeiten der Abbildung 4a (Slab=Bodenplatte, Wall=Wand)

Abbildung 4 Beispiel zur Visualisierung eines Abhängigkeitsgraphen, [Braun et al., 2015, S. 73–74]

3. Methoden

Wie im vorherigen Kapitel gezeigt, ist der Prozess der Punktwolkengenerierung sehr rechenintensiv. Es wäre also wünschenswert, die Rechenzeit bei gleichbleibender Qualität der Punktwolke zu reduzieren. Hierfür muss es gelingen, informationsarme sowie redundante Bilder zu detektieren und auszusortieren. Ausgewählte Ansätze werden in diesem Kapitel dargelegt, bevor sie im „Kapitel 4 „Beschreibung Programmcode““ anhand von Code dargelegt werden.

3.1. Bildschärfeanalyse

Ein Kriterium, Bilder zu bewerten, ist die Betrachtung der Bildschärfe. Je unschärfer ein Bild ist, desto schwieriger ist es, Bildinhalte zu detektieren (siehe Abb. 5). Um eine optimale Punktwolke zu erzeugen müssen die Bilder möglichst viele Informationen enthalten, sprich möglichst scharf sein. Im Umkehrschluss nimmt mit abnehmender Kantenzahl der Nutzen eines Bildes für das Gesamtkonstrukt ab [Schönberger and Frahm, 2016].

(a) Scharfe Kanten führen zu hohem Informationsgehalt

(b) Verschwommene Kanten führen zu Informationsverlust

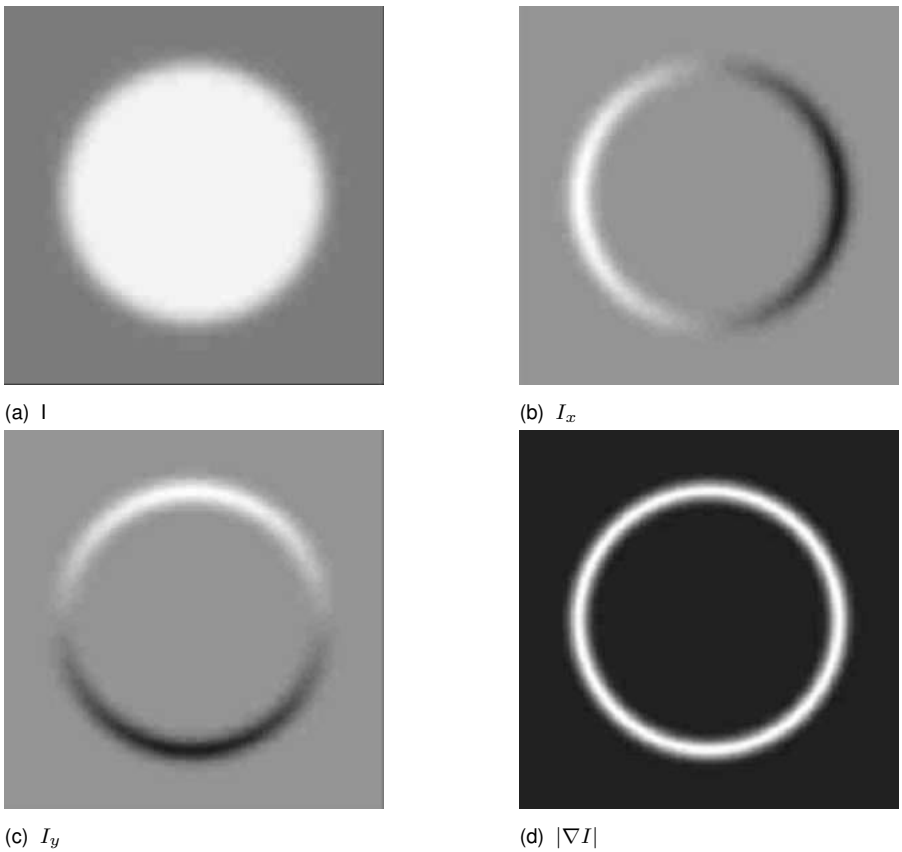
Abbildung 5 Visualisierung des Informationsgehalt aufgrund von (un)schärfe [Veguillas, 2014]

Anschaulich gesprochen ist die Schärfe der Bilder durch die Anzahl der Kanten beschrieben, wobei Kanten Orte im Bild sind, an denen sich die Bildinhalte abrupt ändern. Zur Detektion dieser Änderungen können verschiedene mathematische Prinzipien genutzt werden. In diesem Rahmen sollen eine Gradienten und eine Laplace basierte Methode näher erläutert werden.
Zur Verarbeitung der Bilder werden diese als Matrizen interpretiert [Burger and Burge, 2015]. Das bedeutet, dass jedem Pixel im Bild ein Zahlenwert in der Matrize zugeordnet werden kann, ein Bild der Größe 3000×4000 Pixel also einer Matrize mit 3000×4000 Einträgen entspricht. Da Bilder jedoch klassischer Weise im Farbraum RGB, Rot-Gelb-Blau, abgespeichert werden, besteht jedes Bild nicht aus einer, sondern aus drei Matrizen. Dabei entspricht jede Matrix einem Farbwert. Innerhalb dieser Matrizen werden den einzelnen Pixeln Intensitäten von 0 bis 255 zugeordnet.

Eine Besonderheit in der Anzahl der Matrizen nehmen hierbei Graustufenbilder ein. Sie benötigen lediglich eine Matrix zur Darstellung der Intensität (schwarz=0, weiß=255). Diesen Vorteil machen sich die beiden Analyseverfahren zu Nutze, da beide unabhängig von der Aufnahmefarbtiefe sind. Hierdurch kann die Datenmenge um den Faktor $1/3$ verringert werden und somit die Verarbeitung beschleunigt werden.

3.1.1. Gradienten basierte Methode

Änderungen in Matrizen können mathematisch als Gradient beschrieben werden [Burger and Burge, 2015]. Hierbei wird richtungsabhängig die Ableitung eines jeden Punktes gebildet, sodass die Dimension des Gesamtsystems um eins erweitert wird. Somit ergibt sich bei einem zweidimensionalen Bild ein dreidimensionaler Gradient (siehe Abbildung 6). Um dieses Problem zu kompensieren, kann mithilfe der Betragsfunktion die Richtungsabhängigkeit aufgelöst werden und eine allgemeine Aussage über die Änderungen im Bild getroffen werden. Dies wird in Abbildung 6d verdeutlicht.



(a) I (b) I_x

(c) I_y (d) $|\nabla I|$

Abbildung 6 Partielle erste Ableitungen.Synthetische Bildfunktion I (6a), erste Ableitung in horizontaler Richtung $I_x = {}^{\partial I}/_{\partial x}$ (6b) und in vertikaler Richtung $I_y = {}^{\partial I}/_{\partial y}$ (6c). Betrag des Gradienten $|\nabla I|$ (6d). In (6b, 6c) sind maximal negative Werte *schwarz*, maximal positive Werte *weiß* und Nullwerte *grau* dargestellt [Burger and Burge, 2015, S. 128].

Zur Analyse der Betragsmatrix ist im weiteren Schritt die Einbeziehung der Varianz fundamental. Mithilfe dieser kann eine Aussage über die Homogenität der Matrix, bzw. des Bildes getroffen werden. Allgemein gesprochen beschreibt die Varianz, wie stark Werte um den Mittelwert gestreut sind, je höher die Varianz also ausfällt, desto stärker sind die Werte gestreut.

Nehmen alle Matrixeinträge den selben Wert an, so ist die Varianz 0. Auf die Gradienten angewendet bedeutet dies, je kleiner die Varianz ist, desto gleichmäßiger ist das Bild, desto weniger Kanten enthält es. Ein Bild mit einer kleinen Varianz, ist also ein unscharfes Bild. Im Umkehrschluss enthalten Bilder mit einer hohen Varianz besonders viele Informationen.

3.1.2. Laplace basierte Methode

Auch bei dieser Methode wird mit Differentialen gearbeitet, es wird jedoch statt dem Gradienten, das Laplace Differential gebildet. Bei diesem handelt es sich um die Divergenz des Gradienten: $\Delta I = \nabla \cdot (\nabla I)$
Die Nullstellen des Laplace stellen die steilsten Stellen der Bildmatrix dar, sprich die Kanten. Äquivalent zum Gradienten interessiert uns auch hier wieder die Inhomogenität des Bildes, sprich die Varianz. Deshalb gelten auch alle Annahmen bezüglich der Varianz, sodass eine hohe Varianz viele Kanten, eine kleine Varianz, Unschärfe bedeutet.

3.1.3. Vergleich Gradient und Laplace

Zur besseren Vorstellung der verschiedenen Vorgehen wurde die Abbildung 7 mithilfe der verschiedenen Verfahren getestet und das Ergebnis dreidimensional visualisiert. Das Ergebnis ist in Abbildung 8 dargestellt.
Hierbei ist deutlich zu erkennen, dass die Kanten beim Gradientenbetrag (Abb. 8b & 8e) durch Maxima dargestellt werden. Dahingegen sind sie im Laplaceverfahren (Abb. 8c & 8f) durch einen starken Wechsel zwischen positivem und negativem Vorzeichen gekennzeichnet. Bei beiden ist jedoch klar zu erkennen, dass in kantenreichen Bereichen hohe Varianz (Schachbrettmuster) und in kantenarmen Bereichen (Farbverlauf) eine geringe Varianz existiert.

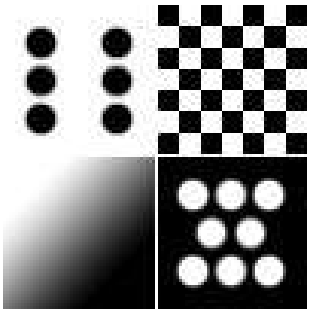
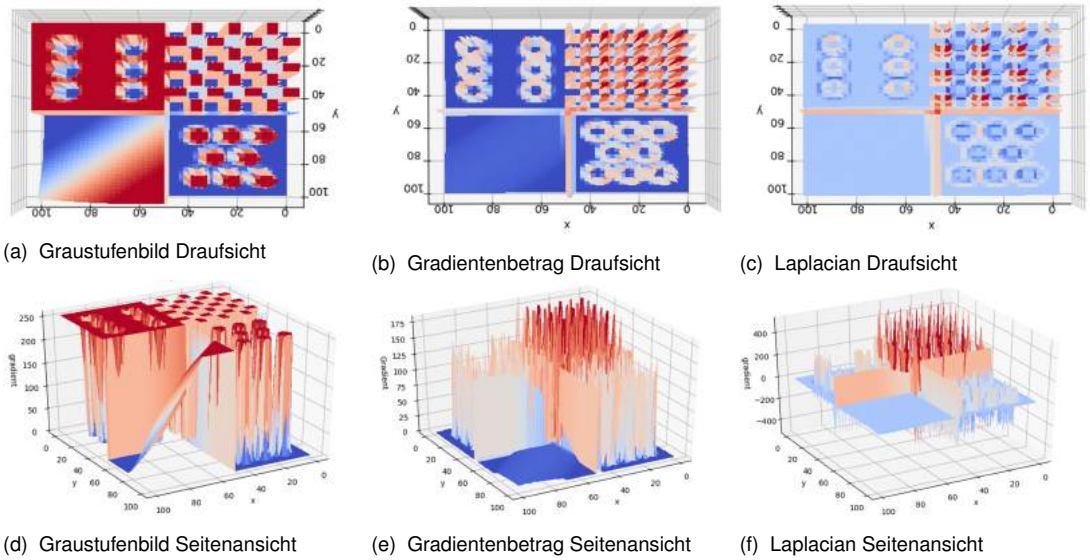


Abbildung 7 Graustufenbild als Ausgang zur Visualisierung verschiedener Bildschärfematrizen

Des weiteren wurden beide Verfahren auf drei verschiedene Datensets (siehe Abb. 9a) angewendet. Das erste Set enthält das selbe Bild mit zunehmender Weichzeichnung, das zweite ein Bild mit zunehmender Auflösung und das dritte ein gemischtes Datenset. Bei dem gemischten Datenset handelt es sich hierbei um eine ausgewählte Anzahl von Bildern, bei denen variierende Ausprägungen der Schärfe und Auflösung vorliegen (siehe Abb. 10). In allen Graphen (Anordnung von oben nach unten: Schärfe, Auflösung, gemischt) zeigt sich, dass der jeweilige Wertebereich der beiden Verfahren unterschiedlich ausfällt. So ergeben „schar-



(a) Graustufenbild Draufsicht (b) Gradientenbetrag Draufsicht (c) Laplacian Draufsicht

(d) Graustufenbild Seitenansicht (e) Gradientenbetrag Seitenansicht (f) Laplacian Seitenansicht

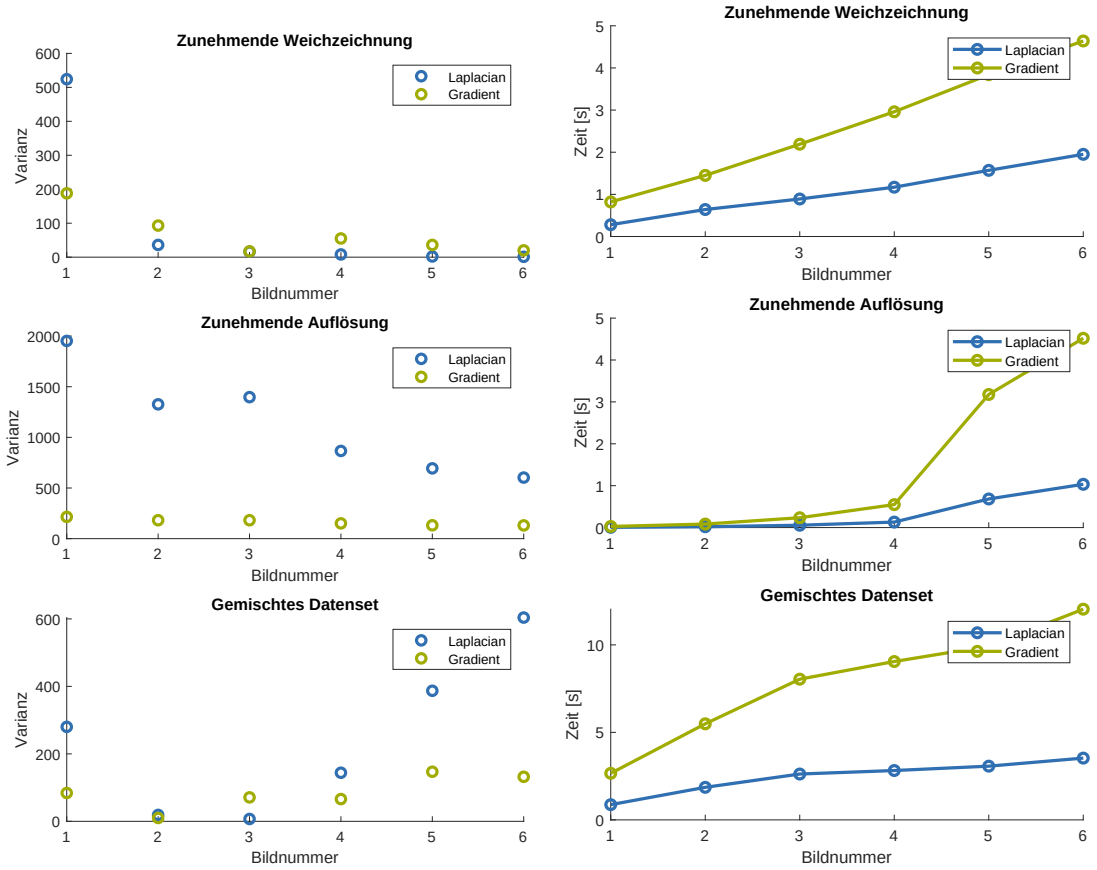
Abbildung 8 Analyseergebnisse des Ausgangsbildes (siehe Abb. 7) mithilfe verschiedener Verfahren sowie dreidimensional visualisiert.

fe“ Bilder im Laplace Verfahren Werte bis zu 2000, wohingegen das Maxima des Gradienten Verfahrens bei 215 liegt. Dadurch können sie nicht quantitativ, sehr wohl aber qualitativ verglichen werden.

Im obersten Graphen zeigen beide Verfahren, dass sie mit zunehmendem Weichzeichner abfallen und somit die ansteigende Unschärfe zuverlässig detektieren. Im zweiten Set schwankt der Wertebereich des Laplace Verfahrens sehr stark (zwischen 603 und 1952), trotzdem werden alle Bilder zuverlässig als scharf detektiert. Selbiges gilt für das Gradienten Verfahren. Auch hier schwanken die Werte, befinden sich jedoch immer in einem ausreichend scharfen Bereich, sodass Bilder nie falsch detektiert werden. Das letzte Datenset (genutzte Bilder siehe Abb. 10) enthält drei scharfe Bilder (10a, 10e & 10f), sowie drei unscharfe (10b, 10c & 10d). Besonders bei Bild 10d sieht man jedoch, dass der enge Wertebereich des Gradienten bei unsicheren Bildern keine deutliche Abstufung erreicht. Im Gegenteil hierzu erkennt das Laplace Verfahren, dass das Bild zwar „schärfer“ als die beiden vorhergegangen ist, jedoch immer noch Unscharf. Für eine spätere Festlegung eines Schwellenwertes, ab dem Bilder als scharf detektiert werden sollen, ist deshalb der größere Wertebereich der Varianz beim Laplace Verfahren vorteilhafter.

Darüber hinaus wurde die benötigte Rechenzeit pro Bild bei beiden Ansätzen verglichen. Hierfür wurden verschiedene Datensets analysiert und währenddessen die Zeit gestoppt. Alle Rechnungen wurden auf einem Intel i5-8250U Prozessor, 8 GB Arbeitsspeicher und ohne Grafikkarte durchgeführt. Auf einem anderen System würden die Zeiten dementsprechend anders ausfallen, sie sind deshalb nicht quantitativ zu interpretieren, sondern sollen lediglich eine qualitative Aussage ermöglichen.

Die Graphen 9b visualisieren die benötigte Zeit für die entsprechenden Datensets aus der Varianzanalyse. Dabei ist die benötigte Rechenzeit beim Gradientenverfahren immer größer als beim Laplace Verfahren. Besonders bei hochauflösenden Bildern (siehe Abb. 9b „Ge-



(a) Varianz (b) Zeitlicher Rechenaufwand

Abbildung 9 Untersuchung verschiedener Datensets mit verschiedenen Schwerpunkten.

mischtes Datenset“) sind die benötigten Rechenzeiten sehr unterschiedlich. Dies hat mit der Implementierung des Codes zu tun.

Zur Berechnung mathematischer Operatoren wurde im Rahmen dieser Arbeit vorimplementierte Funktionen der Bibliothek „OpenCV“ genutzt. Das Gradienten Verfahren nutzt mehrere Matrixoperationen, so wird zuerst der Gradient bestimmt und im Anschluss der Betrag. Diese einzelnen Schritte sind jeder für sich sehr effektiv, wurden jedoch nicht für das Zusammenspiel optimiert. Dahingegen wird beim Laplace Verfahren lediglich eine Matrixoperation, die Berechnung des Laplace, durchgeführt. Diese ist aufgrund von Multi-Core-Prozessen möglich.

Zusammenfassend kann man sagen, dass beide Verfahren zuverlässig Schärfe detektieren und aus mathematischer Sicht gut geeignet sind die Fragestellung vollständig zu beantworten. Aufgrund der großen Unterschiede in der benötigten Laufzeit pro Bild ist jedoch das Laplace Verfahren besser geeignet, die Problemstellung zu lösen. Darüber hinaus bietet der größere Wertebereich bessere Möglichkeiten einen Schwellenwerte zu formulieren.

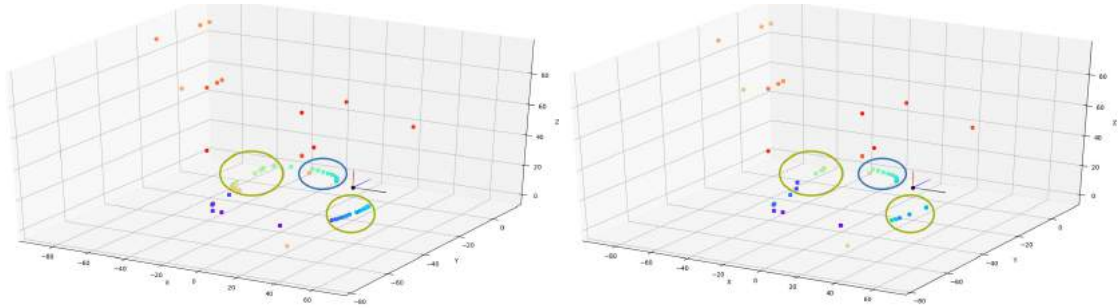
(a) Bild Nr. 1 (b) Bild Nr. 2 (c) Bild Nr. 3

(d) Bild Nr. 4 (e) Bild Nr. 5 (f) Bild Nr. 6

Abbildung 10 Bilder des „Gemischten Datensets“ in Abb. 9

3.2. räumliche Bildorientierung

Eine weitere Möglichkeit die Bilder zu selektieren, bietet deren Aufnahmeort. Hierbei wird der Ort an dem die Bilder aufgenommen wurden, in einem virtuellen Modell nachgebildet und mithilfe der Verteilung im Raum die Auswahl verkleinert. Ein entsprechendes Beispiel wird hierbei in der Abbildung 11 visualisiert. Dabei stellt jeder Punkt eine Drohnenposition zum Zeitpunkt einer Aufnahme dar. Abbildung 11a zeigt hierbei das Ausgangsset, Abbildung 11b das Set nach Anwendung der Methode.



(a) Ausgangsdatenset (b) aussortiertes Datenset

Abbildung 11 dreidimensionale Visualisierung der Bildanordnung

JPEG-Bilder bieten durch das „Exchange image file format“ (EXIF) die Möglichkeit, zusätzlich zum eigentlichen Bild weitere Informationen zu speichern [Camera & Imaging Products Association, 2016]. Diese enthalten unter anderem Angaben über das Bild, die Kamera aber auch GPS-Informationen. Die Positionsdaten bestehen hierbei aus drei verschiedenen Angaben: Breitengrad, Längengrad, sowie Höhe über Normal-Null. Mithilfe dieser Informationen kann ein lokales Modell der Bildanordnung entworfen werden (siehe Abb. 11a).

Da alle Positionsdaten in Grad, Minuten, Sekunden gegeben sind, werden sie zuallererst in ein lokales, metrisches Koordinatensystem konvertiert. Eine Koordinate, welche sich möglichst innerhalb des Modells befindet, wird hierfür als Null-Punkt definiert. Relativ zu diesem Ort, werden alle anderen Bilder mithilfe der Distanz ins metrische System konvertiert.
Im nächsten Schritt wird der Raum in Subräume (Voxel) unterteilt, innerhalb derer die maximale Bildanzahl definiert wird. Hierfür wird die Größe eines jeden Subraums absolut in Meter und nicht relativ in Prozent definiert, sodass Länge $L[m]$, Breite $B[m]$ und Höhe $H[m]$ festgelegt sind. Durch die Angabe der Kantenlänge in absoluten Maßen, kann eine gleichbleibende Auflösung der Punktwolke erreicht werden. Würde stattdessen eine relative Definition auf Basis der Gesamtgröße des abgebildeten Systems genutzt werden, kann es bei kleinen Baustellen zu einer zu großen Anzahl an Bildern, bei besonders großen Baustellen zu einer zu kleinen Anzahl an Bildern kommen. Durch die absolute Angabe der Längen kann dieses Problem umgangen werden und die Auflösung der Punktwolke erreicht immer vergleichbare Güte.
Des Weiteren wird die maximale Bilderzahl innerhalb eines Voxels definiert. Sie bestimmt, wie viele Bilder pro Voxel behalten werden, wobei eine Verteilung innerhalb des Voxels aktuell unbetrachtet bleibt.
Zur Bewertung der beiden Größen ist vor allen Dingen die theoretisch maximale Anzahl der Bilder $max(I)$ als Kennzahl zu beachten. Sie errechnet sich aus der gesamten Zahl an Voxeln $num(S_{voxel})$, sowie der Anzahl der Bilder pro Voxel $num(I_{pv})$. Mithilfe derer kann eine optimale Größe der Voxel bestimmt werden.

$$max(I) = num(S_{voxel}) * num(I_{pv}) \tag{3.1}$$

Für die maximale Anzahl Bilder pro Voxel, sowie die Anzahl der Voxel ergibt sich hierbei eine lineare Abhängigkeit. Da der Nutzer die Anzahl der Voxel jedoch nicht direkt steuern kann, sondern lediglich Einfluss auf die Kantenlänge eines einzelnen Voxels hat, ergibt sich hierbei bei einer linearen Änderungen der Kantenlängen, eine kubische Änderung der Bildzahl:

$$num(S_{voxel}) = \frac{S_{gesamt}}{S_{voxel}} = \frac{konst}{L * B * H} \tag{3.2}$$

$$\frac{S_{gesamt}}{2L * 2B * 2H} = \left(\frac{1}{2}\right)^3 * \frac{S_{gesamt}}{L * B * H} = \frac{1}{8} * num(S_{voxel}) \tag{3.3}$$

Aus diesem Grund muss bei Anpassung der beiden Kenngrößen besonders auf die maximale Zahl der Bilder geachtet werden, sodass die Bildanzahl nicht zu klein/ zu groß wird.
Eine mögliche Vorselektion ist in Abbildung 11 dargestellt. Der linke Graph stellt hierbei das vollständige Datenset, der rechte das aussortierte Datenset dar. Besonders in Bereichen, in denen es Häufungen bei der Anzahl der Bilder gibt, wurden hierbei Bilder aussortiert. Ersichtlich ist dies an den grün markierten Stellen. Im blau markierten Bereich sind hingegen sehr wenige Bilder aussortiert worden. Dies liegt daran, dass die Subräume sich genau in diesem Bereich schneiden und somit alle Bilder in unterschiedlichen Voxeln liegen. Diese

Methode zur Selektion bietet also nicht immer die optimale Lösung, schafft es aber sinnvoll, eine erste Sortierung der Bilder zu ermöglichen. Durch Verknüpfung der örtlichen Analyse mit der Bildschärfe-Analyse kann im nächsten Schritt eine gute Lösung für die Sortierung geschaffen werden.

3.3. Verknüpfung der Methoden

Im letzten Schritt werden die verschiedenen Methoden miteinander verknüpft. Hierzu wird jedes Bild mithilfe seines Pfades als Key in eine Datenbank aufgenommen und bekommt ein Value zugeordnet. Anhand dieses Values können die Bilder als gut oder als schlecht eingeordnet werden.
Die Analyse beginnt mit der Auswertung der Bildschärfen. Nach Bestimmung der absoluten Werte werden diese relativ zum ganzen Datenset bestimmt. Hierbei spielt der Schärfenschwellenwert eine Rolle. Werte oberhalb können als Scharf, Werte unterhalb als Unscharf bezeichnet werden. Zur mathematischen Unterscheidung werden die Values scharfer Bilder auf den Wertebereich 0 bis 1, die unscharfer Bilder auf den Bereich -1 bis 0 normiert. Im Anschluss werden die normierten Values in die Datenbank übernommen.
Im finalen Schritt werden die Bilder räumlich verordnet und anschließend aussortiert. Hierbei spielt die Option *seg_hard* eine Rolle. Sie gibt an, ob alle Bilder, die einen negativen Value haben, aussortiert werden sollen, oder ob eine Mindestzahl an Bildern pro Voxel behalten werden soll. Im ersten Fall spielen die Definition der Voxel und die Mindestzahl der Bilder pro Voxel keine weitere Rolle. Im zweiten Fall kann es vorkommen, dass Bilder die als Unscharf klassifiziert wurden, trotzdem weiter in dem Datenset verbleiben. Die zugrunde liegende Annahme ist hierbei: „besser schlechte Daten, als gar keine Daten“. Das Aussortieren der Bilder findet hierbei auf Basis der Values statt. Die Bilder innerhalb eines Voxels werden anhand ihres Values sortiert und diejenigen mit dem schlechtesten Value werden solange aussortiert, bis die Maximalzahl der Bilder pro Voxel erreicht ist.

4. Beschreibung Programmcode

Der Programmcode setzt die zuvor beschriebenen Methoden in die Praxis um. Dabei lassen sich die Files in drei Typen einteilen: Klassen, Libraries und Executables. Um eine Zuordnung zu den zuvor beschriebenen Methoden herzustellen, bietet die Abbildung 12 eine erste Übersicht. Dabei sind die entsprechenden Typen durch unterschiedliche Formen gekennzeichnet.

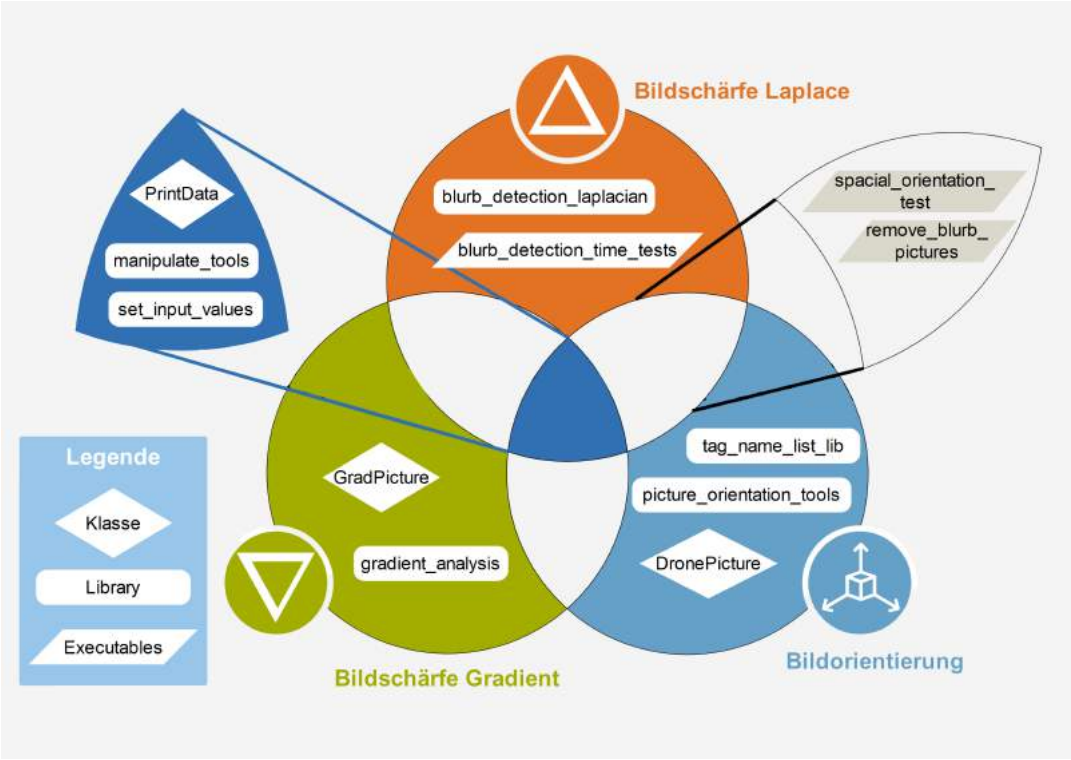


Abbildung 12 Zuordnung der Methoden zu den jeweiligen Programmbausteinen

4.1. Wahl der Programmiersprache

Zur Umsetzung des Codes wurde auf die Sprache Python zurückgegriffen. Diese bietet im Hinblick auf dieses Projekt mehrere Vorteile:

- Syntax und Semantik ermöglichen es, schnell und übersichtlich funktionellen Code zu schreiben.
- Python ermöglicht objektorientierte Programmierung, sodass für die verschiedenen Programmabschnitte Klassen angelegt werden können.
- Python wird mithilfe eines Interpreters übersetzt und ist somit sehr flexibel. Code kann direkt getestet werden und kleinere Anpassungen können ohne erneute Kompilierung getestet werden.

- Viele Funktionalitäten gibt es bereits in vorhandenen Bibliotheken welche lediglich eingebunden werden müssen.

Trotz all der Vorteile ist Python gegenüber C bzw. C++ laufzeittechnisch deutlich schlechter optimiert. Besonders bei rechenintensiven Vorgängen fällt dieser Nachteil ins Gewicht. Da jedes Bild als Matrize eingelesen wird (Dimension: ca. $R^{3000\times4000}$) ist bei deren Berechnung besonders die Geschwindigkeit der Analyse wichtig. Durch den geschickten Einsatz der externen Bibliothek „OpenCV“ kann die Laufzeit stark optimiert werden. Mithilfe seines Python Interfaces kann es in das bestehende Projekt eingebettet werden. Die Bibliothek selber ist jedoch in C implementiert und unterstützt darüber hinaus multi-core processing [OpenCV team, 2018]. Dadurch kann der Nachteil der ineffizienten Implementierung durch die hohe Kompatibilität mit third-party Bibliotheken ausgeglichen werden.

4.2. Code Aufbau und Funktionalität

Im Projektordner sind alle Dateien, die für eine vollständige Analyse gebraucht werden aufgeführt (Ordnerstruktur siehe Abb. 13a). Während der Großteil der Dateien in Ordner strukturiert ist, sind zwei Dateien aufgrund ihrer Relevanz außerhalb der Ordnerstruktur abgelegt. Das eine File „set_input_values.py“ enthält alle benötigten Konstanten. Das andere File „remove_blurb_pictures.py“ ist als Hauptprogramm zu verstehen und führt bei Start eine vollständige Analyse entsprechend der im Abschnitt 3.3 „Verknüpfung der Methoden“ beschriebenen Abläufe durch. Die Zuordnung der restlichen Dateien wurde entsprechend einer Kategorisierung in folgende Ordner durchgeführt:

- **Classes:** Enthält alle benötigten Klassen des Projekts.
- **grad_files:** Alle für die Gradientenanalyse benötigten Dateien wurden in diesem Ordner zusammengefasst. Diese Dateien sind nicht mehr Teil des Hauptprojektes, sondern werden lediglich zu Analysezwecken benötigt.
- **Libs:** Zur besseren Übersichtlichkeit wurden die Funktionen des Projektes in verschiedenen Bibliotheken kategorisiert ausgelagert. Alle benötigten Funktionen finden sich in diesem Ordner.
- **test_files:** Enthält Dateien, die für verschiedene Tests des Codes genutzt werden. Alle diese Files können weiterhin für verschiedene Analysen genutzt werden.
- **unused_files:** Dateien, die im Rahmen der Codeentwicklung geschrieben wurden, nun jedoch nicht mehr für die Funktionalität gebraucht werden, sind in diesen Ordner zusammengefasst. Diese können in einer release Version des Codes gelöscht werden. Da sie uninteressant für die Funktion des Programmes sind, werden sie in diesem Rahmen nicht weiter erläutert.

(a) Ordnerstruktur des vollständigen Python Projektes

(b) Einordnung der beschriebenen Files in die Codestruktur. Die speziellen Files werden durch einen gelb-roten Kasten hervorgehoben.

Abbildung 13 Ordnerstruktur des Projektes, sowie die Einordnung der beschriebenen Files in die Methodenabhängigkeit

4.2.1. Beschreibung relevanter Programmdateien

Der Aufbau des Codes ist sehr modular und deshalb in viele Bibliotheken und Funktionen eingeteilt. Da die Anzahl der Funktionen sehr umfangreich ist, wird auf die Beschreibung aller Funktionen in diesem Rahmen verzichtet. Bei der Durchführung einer Analyse kommt der Anwender auf jeden Fall mit zwei Dateien in Kontakt. Diese sollen in diesem Rahmen vorgestellt und erläutert werden. Deren Zuordnung zu den Methoden ist in Abb. 13b hervorgehoben.

File set_input_values.py

In diesem File werden alle Konstanten die im Laufe der Analyse gebraucht werden gesammelt. Hierdurch lässt sich eine Konsistenz der Variablen innerhalb des gesamten Projekts gewährleisten und eine versehentliche Doppelbelegung vermeiden. Die Variablen sind in vier verschiedene Kategorien eingeteilt. Dabei richten sich die Kategorien nach dem Anwendungsbereich der Variable im Programm, sprich wird eine Variable im Zusammenhang mit der Bestimmung der GPS-Position benötigt, so findet sich diese in der Kategorie GPS-values. Im Rahmen der Beschreibung aller vorhandenen Variablen, werden die wichtigen Variablen „**bold**“ markiert. Diese Variablen sollten vor jeder Ausführung überprüft und entsprechend angepasst werden.

GENERAL

Dieser Abschnitt definiert Konstanten, die im Laufe des Programmes regelmäßig benötigt werden und keinem Programmteil unmittelbar zugeordnet werden können.

- **source_dir:** Gibt den Pfad des Ordners an, in welchem nach Bildern gesucht wird. Es wird ausschließlich dieser Ordner untersucht, Bilder aus Unterordnern werden nicht betrachtet.
- search_endings: Alle Dateien mit einer in der Liste angegebenen Endung werden in der späteren Analyse betrachtet. Die angegebenen Endungen müssen dabei vollständig klein geschrieben werden, damit sie korrekt erkannt werden.
- folder_content: Berechnet aus den beiden vorhergegangenen Informationen eine Liste mit allen Dateien, die sich im Pfad *source_dir* mit den Dateiendungen *search_endings* befinden. Alle Dateien, die dieser Liste hinzugefügt werden, werden bei der Selektion ausgewertet.

BLURB-DETECTION

Dieser Abschnitt enthält alle Konstanten, die zur Berechnung der Unschärfe benötigt werden.

- blur_safe_name: Die im Laufe des Programms berechneten Unschärfewerte der Bilder werden in einer Textdatei abgelegt um die Rechenzeiten bei sich wiederholenden Berechnungen zu optimieren. *blur_safe_name* gibt den Namen dieses Files an, wobei dieser korrekterweise auch *blur_val* mit einbezieht, sodass sich der vollständige Name wie folgt zusammensetzt: [*blur_safe_name*]_[*blur_val*].txt. Diese Datei wird im Pfad *source_dir* gespeichert.
- force_recalculation_blur_vals: Falls eine entsprechende Textdatei mit dem Namen [*blur_safe_name*]_[*blur_val*].txt vorhanden ist, so wird überprüft, ob allen ausgewählten Bildern ein Unschärfewert aus der Datei zugeordnet werden kann. Falls dem so ist, werden die Werte aus diesem File ausgelesen und nicht neu berechnet. Wenn der Wert *force_recalculation_blur_vals* auf *True* gesetzt ist, so wird die Neuberechnung erzwungen und in jedem Fall ausgeführt.
- **blur_val:** Dieser Wert ist der Schwellenwert der Unschärfe. Unter diesem Wert werden Bilder als unscharf betrachtet.

GPS-VALUES

GPS-Values fässt alle Konstanten zur Bestimmung der Positionsdaten zusammen.

- error_handling: Nicht alle Bilder enthalten zwangsweise korrekte Metadaten für die Positionsbestimmung. Um trotzdem ein Ergebnis zu erhalten, kann mit dieser Konstante das Verhalten bei einem entsprechenden Bild festgelegt werden. Der Wert *move* verschiebt das Bild in den entsprechenden Pfad *error_dir*, ansonsten werden die Bilder ignoriert und einfach in dem Ordner belassen.
- error_dir: Falls eine Verschiebung von Bildern ohne korrekte Metadaten gewünscht ist,

kann hier ein spezieller Pfad angegeben werden. Falls dieser inkorrekt oder nicht angegeben wird, aber trotzdem eine Verschiebung gewünscht wird, so wird als Standardwert „[*source_dir*]/missing_location_information“ als Pfad angelegt und genutzt.

- tag: Jede benutzte Drohne kann unterschiedlich aufgebaute Metadaten enthalten. Zur Spezifikation wurde eine Datenbank in der Datei *tag_name_list_lib.py* für jeden genutzten Drohnentyp angelegt. Da bisher nur ein Typ Drohne genutzt wurde, wurde auch nur ein Wert für den Typ „dji-ph4“angelegt.
- **max_num_pic_per_voxel:** Gibt die maximal Anzahl der Bilder pro definiertem Voxel an.
- **voxel_size:** Defniert die Größe eines einzelnen Voxels. Alle Maße werden in Meter gegeben. Die Dimensionen in der Liste sind folgendermaßen: [x, y, z], wobei x den Längengrad, y den Breitengrad und z die Höhe relativ zum Drohnenstartpunkt (dieser ist als Höhe 0 definiert) beschreibt.
- **base_coordinate_gz:** Mithilfe dieser Konstante wird der Nullpunkt des späteren Koordinatensystems zur räumlichen Einordnung gesetzt. Je weiter weg dieser von den Bildern ist, desto länger dauert die Visualisierung, weshalb dieser in der Nähe, bzw. im Gebäude liegen sollte. Die erforderliche Einheit ist Dezimalgrad und in der Form: [*Breitengrad*, *Längengrad*]

SEGMENTATION
Variablen, die bei der Segmentierung, der Bilder genutzt werden. Sie definieren, wie, ob und wohin Bilder verschoben werden.

- segregation_path: Gibt an, wohin aussortierte Bilder gespeichert werden. Wenn dieser Wert auf *default* gesetzt ist, wird ein neuer Ordner in *source_dir* mit dem Namen *segregated* angelegt. Falls ein Pfad gewählt wird, so muss dieser zum Programmstart schon existieren. Falls er nicht existiert, wird der Default-Pfad gewählt.
- seg_hard: Falls auf *True* gesetzt, so werden alle Bilder, die unterhalb des Unschärfe Schwellenwertes liegen automatisch aussortiert. Dadurch kann es passieren, dass zu viele Bilder aussortiert werden und zu wenige Bilder für die Punktwolke verbleiben. Falls auf *False* gesetzt, so entfällt dieser erste Schritt der Bildselektion.

File remove_blurb_pictures.py

Dieses File führt eine vollständige Analyse aller per *source_dir* ausgewählten Bilder auf deren Unschärfe und räumliche Anordnung durch. Nach Initialisierung einiger Werte und der Berechnung der Unschärfewerte, werden die bildspezifischen Positionskennwerte berechnet. Der aktuell auskommentierte Bereich PLOT bietet die Möglichkeit, die erhaltenen Daten entsprechend der Abbildung 11 zu visualisieren. Aufgrund des hohen Rechenaufwands ist dieser Bereich aktuell jedoch auskommentiert. Im Abschnitt SPATIAL ORIENTATION werden die Bilder räumlich in die Voxel einsortiert, bevor sie in SEGMENTATION aussortiert werden.

Der letzte Teil MOVE FILES TO NEW FOLDER verschiebt die aussortierten Bilder in den Ordner *segregation_path*. Eine ausführlichere Beschreibung des Ablaufs wird im Abschnitt 4.3 „Ablauf der Bilderanalyse“ vorgenommen

4.2.2. Beschreibung der Ordnerstruktur

Zum besseren Verständnis der Ordnerstruktur wird in diesem Abschnitt jeder Ordner und die darin enthaltenen Files erläutert. Dabei wurden alle Files entsprechend ihrer Ordnerzugehörigkeit in der Abbildung 14 durch einen farbigen Rahmen markiert.

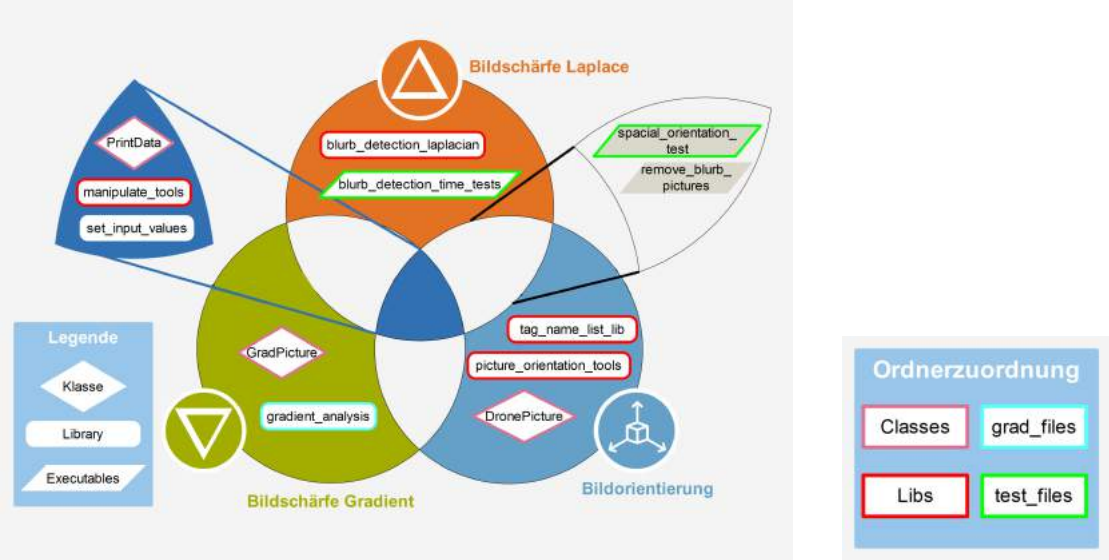


Abbildung 14 Kennzeichnung der verschiedenen Dateien entsprechend ihrer Ordnerzugehörigkeit durch einen unterschiedlich eingefärbten Rahmen.

Ordner Classes

Alle im Rahmen der Analyse benötigten Klassen werden in diesem Ordner zusammengefasst. Sie sind in Abbildung 14 durch einen pinken Rahmen dargestellt.

DronePicture.py

Diese Klasse enthält alle Kennwerte, die die Drohne betreffen. Sie definiert unter anderem, wie die von der Drohne erfassten Metadaten in jedem einzelnen Bild zu interpretieren sind. Wird mehr als eine Drohne zur Aufnahme der Baustelle eingesetzt, so muss sich angeschaut werden, wie diese weitere Drohne die Metadaten speichert. Gibt es hierbei Unterschiede zur bisher genutzten Drohne, so muss die Klasse modifiziert werden. Darüber hinaus muss im Laufe des Programmes eine weitere Instanz mit den neuen Drohnenspezifikationen erzeugt werden. Die Drohnen werden hierbei über die Variable „*tag*“ eindeutig klassifiziert, weshalb diese Variable auch zur Klasseninitialisierung essenziell ist.

GradPicture.py
Eine Klasse, um die Gradienten bezogenen Kennwerte zu berechnen. Da der letzte Stand des Programms jedoch mit der Laplace-Methode operiert, ist diese Klasse lediglich für Vergleiche beider Verfahren nötig.

PrintData.py
Zur Optimierung des Visualisierungsvorgangs für die in diesem Projekt typischen Datenstrukturen wurde diese Klasse angelegt. Dadurch können schnell und einfach verschiedene Informationen dargestellt werden. Es sind abgeleitete Klassen für scatter plots (siehe Abb. 11) oder für surface plots (siehe Abb. 8) vorhanden.

Ordner grad_files

Die hier verbliebene Datei „gradientanalysis.py“ wird zur Gradientenanalyse gebraucht. Da diese für die weiteren Abläufe nicht mehr interessant ist, wird der Code nicht weiter beschrieben (in Abb. 14 durch einen türkisen Rahmen visualisiert).

Ordner Libs

Der Ordner enthält Libraries zu verschiedenen Kategorien. Hierbei wurde versucht, die Funktionen thematisch zu sortieren, aufgrund thematischer Überlappungen, kann es jedoch vorkommen, dass eine Funktion in einer unintuitiven Bibliothek eingeordnet ist (Abb. 14 rote Rahmen).

blurb_detection_laplacian.py
Diese Bibliothek enthält alle Funktionen, die zur Unschärfeanalyse mithilfe der Laplace Methode benötigt werden.

manipulate_tools.py
Funktionen, die zur allgemeinen Pfadmanipulation gebraucht werden sind in dieser Datei aufgeführt. Dazu zählen Funktionen zur Pfad-Validierung aber auch das kopieren/ verschieben von Dateien von einem zum anderen Ordner findet sich hier.

picture_orientation_tools.py
Alle Funktionen die mit der Bildanalyse zusammenhängen finden sich in dieser Bibliothek. So befinden sich hier Funktionen zur Umrechnung von Koordinaten in Meter. Des Weiteren befindet sich hier die Funktion um die Bilder basierend auf eines Dictionaries aller Bilder und

deren Sortierungsvalues als (un-)brauchbar zu bewerten.

tag_name_list_lib.py
Die EXIF und XMP Dateien sind hochflexibel in der Bezeichnung der enthaltenen Daten. So muss die Information des GPS nicht immer gleich heißen. Um dieses Problem anpassbar zu lösen, bietet diese Datei die Möglichkeit, für alle benötigten Drohnen eine Datenbank anzulegen. Dazu müssen in dem Dictionary der Konstante *tag* die entsprechenden Values zugeordnet werden, sodass die richtige Liste ausgewählt werden kann. Für die bisher genutzte Drohne „DJI Phantom 4“ sind diese Informationen schon korrekt hinterlegt.

Ordner test_files

Dateien, die im Laufe des Entstehungsprozesses für verschiedene Tests benötigt wurden finden sich in diesem Ordner. Diese sind für die Funktionalität des Kerns nicht wichtig, bieten aber die Möglichkeit, die Daten auf verschiedene Dinge zu untersuchen (Abb. 14 grüner Rahmen).

blurb_detection_time_tests.py
Untersucht die benötigte Zeit pro Bild bei der Analyse mithilfe der Laplace Methode. Ist aufgrund dieser zusätzlichen Testeigenschaften für eine Gesamtanalyse nicht brauchbar. Darüber hinaus wird hier nur die Analyse, nicht aber die Interpretation der Daten durchgeführt.

spacial_orientation_test.py
Vorgänger der Datei „remove_blurb_pictures.py“. Beide Dateien sind sich sehr ähnlich, die Datei „spacial_orientation_test.py“ ist dabei jedoch ein bisschen experimenteller. Falls sich eine Funktion bewährt hat, so wird diese in das Hauptprogramm „remove_blurb_pictures.py“ übernommen.

4.3. Ablauf der Bilderanalyse

Das File *remove_blurb_pictures.py* selektiert Bilder unter Anwendung von zwei Methoden. Dabei handelt es sich um die Bildschärfeanalyse mithilfe der Laplace Methode sowie die räumliche Einordnung. Im ersten Schritt wird die absolute Bildschärfenvarianz pro Bild S_{abs_i} berechnet. Anhand des kleinsten $min(S_{abs})$ und größten $max(S_{abs})$ Schwellenwertes werden daraus relative Schwellenwerte S_{rel_i} für jedes Bild bestimmt:

$$S_{rel_i} = \begin{cases} \frac{S_{abs_i}}{max(S_{abs})}, & \text{wenn } S_{abs_i} > blur\_val \\ -\frac{min(S_{abs})}{S_{abs_i}}, & \text{andernfalls} \end{cases} \tag{4.1}$$

Durch diese Vorgehensweise werden alle Bilder auf einer Skala von -1 bis 1 bewertet und zueinander ins Verhältnis gesetzt. Darüber hinaus sind als unscharf deklarierte Bilder durch ein negatives Vorzeichen gekennzeichnet. Die sich ergebenden Werte werden als Values val bezeichnet. Sollen in einem späteren Verfahren noch weitere Eigenschaften des Bildes mit in Betracht gezogen werden, so kann durch die Formel 4.2 ein Gesamtvalue val_{ges} aus verschiedenen Values val_i errechnet werden:

$$val_{ges_i} = a_0 * val_{0_i} + a_1 * val_{1_i} + ... + a_j * val_{j_i} \quad , a_j = \text{Gewichtungsfaktor} \wedge val_{0_i} = S_{rel_i} \tag{4.2}$$

Zur weiteren Verarbeitung der Values werden diese in der Datenstruktur Dictionary gespeichert. Hierbei können einzigartigen Keys jeweils ein Value zugeordnet werden, sodass jeder Key nur einmal pro Dictionary vorkommen kann. In diesem Fall handelt es sich bei dem Key um den absoluten Pfad eines jeden Bildes. Da die Berechnung der Values für die Bildschärfe recht zeitintensiv ist, werden diese Ergebnisse darüber hinaus in einer Textdatei abgelegt. Dadurch kann bei einer wiederholten Durchführung der Analyse auf diese Werte zurückgegriffen und Zeit gespart werden.
Im nächsten Schritt werden die Koordinaten aus der EXIF-Datei ausgelesen und mithilfe der *base_coordinate_gz* in das metrische System konvertiert. Zur ersten Einschätzung der Bilder können diese nun in einem Scatter Plot visualisiert werden. Dieser Schritt ist jedoch sehr rechenintensiv und sollte deshalb standardmäßig ausgelassen werden. Mithilfe der Positionsinformationen können im Anschluss die Minima min und Maxima max jeder Koordinatenrichtung des Konstrukts berechnet werden. Hierdurch werden die Anzahl der benötigten Voxel num_{vox} in jeder Dimension bestimmt, wobei die Funktion $ceil$ aufrundet:

$$num_{vox_x} = ceil\left(\frac{max_x - min_x}{voxel\_size_x} + 1\right) \tag{4.3}$$

$$num_{vox_y} = ceil\left(\frac{max_y - min_y}{voxel\_size_y} + 1\right) \tag{4.4}$$

$$num_{vox_z} = ceil\left(\frac{max_z - min_z}{voxel\_size_z} + 1\right) \tag{4.5}$$

Die Gesamtzahl der Voxel ergibt sich somit zu:

$$num_{vox} = num_{vox_x} * num_{vox_y} * num_{vox_z} \tag{4.6}$$

Aus der Gesamtzahl der Voxel kann ein Netz bestimmt werden in welches die Bilder eingeordnet werden. Dadurch sind alle Bilder einem Voxel zugeordnet. Innerhalb dieser Voxel werden im Anschluss die Bilder der Größe ihres Values val_i nach geordnet.
Im nächsten Schritt wird abhängig von der Option *seg_hard* die Selektion vorgenommen. Ist $seg\_hard = True$, werden in einem ersten Schritt alle Bilder mit einer geringeren Varianz als dem Schwellenwert aussortiert. Bei der Option $seg\_hard = False$ wird dieser Schritt

übersprungen. Im nächsten Schritt werden pro Voxel die Anzahl Bilder behalten, die durch die Konstante *max_num_pic_per_voxel* festgelegt ist. Dadurch wird eine Mindestqualität der Punktwolke garantiert. Zur Sortierung werden in jedem Voxel diejenigen Bilder mit dem kleinsten Key solange aussortiert, bis die verbleibende Bildzahl $\leq$ *max_num_pic_per_voxel*. Alle aussortierten Bilder werden entsprechend des Pfades *segregation_path* verschoben.

5. Versuche

Um die Methoden und Annahmen zu überprüfen, werden im Rahmen dieser Arbeit zwei Datensets der selben Baustelle untersucht. Sie unterscheiden sich zum einen im Zeitpunkt der Aufnahme und dem damit verbundenen Baufortschritt, zum anderen in der Qualität der Bilddaten. Aus beiden Sets werden jeweils der vollständige Datensatz, sowie anhand von verschiedenen Selektionskriterien erstellte Auswahlen zu einer Punktwolke verrechnet. Im Anschluss werden die erhaltenen Ergebnisse in Bezug auf benötigte Rechenzeit und Präzision verglichen. Auf Basis dieser Informationen soll festgestellt werden ob das Verfahren einen Mehrwert bringt und falls ja, welche Parameter gute Richtwerte sind.

5.1. Versuchsziel

Die Versuche verfolgen zwei Hauptziele. Im ersten Schritt soll festgestellt werden, ob das Vorgehen generell gewinnbringend ist. Dabei kann es als erfolgreich definiert werden, wenn mindestens 20% der Zeit eingespart und mindestens 80% der Qualität der Ausgangspunktwolke erreicht werden.
Falls sich das Verfahren bewährt, so sollen in einem zweiten Schritt sinnvolle Ausgangswerte für verschiedene Konstanten der Datei *set_input_values.py* gefunden werden. Ein besonderes Augenmerk liegt hierbei auf folgenden Variablen:

- *blur_val*: Ab welcher Varianz hat ein Bild bei der Punktwolkenerzeugung einen signifikanten Mehrwert. Es werden drei Varianzwerte betrachtet: 100, 150 und 200.
- *max_num_pic_per_voxel*: Was ist eine gute Maximalzahl an Bildern pro Voxel.
- *voxel_size*: Die Voxelgröße spielt eine große Rolle bei der Zahl der aussortierten Bilder. Wie in Formel 3.1 ff. gezeigt, kann diese nicht unabhängig von der *max_num_pic_per_voxel* betrachtet werden, sondern muss stets im Verhältnis zu dieser überprüft werden. Zur Überprüfung wurden insgesamt drei verschiedene Kombinationen betrachtet:

$[voxel\_size]$	*max_num_pic_per_voxel*
$[5, 5, 3]$	3
$[10, 10, 5]$	5
$[20, 20, 10]$	10

- *seg_hard*: Untersucht, wie sich ein vollständiges Aussortieren aller unscharfen Bilder auswirkt. Hierbei interessiert, ob aus der verbleibenden Bildzahl eine sinnvolle Aussage getroffen werden kann. Verändert sich die Qualität der Punktwolke, wenn unscharfe Bilder im Zweifel behalten werden?

Besonders wichtig ist auch die Abhängigkeit der einzelnen Variablen voneinander. Dieses Zusammenspiel ist besonders schwer abzuschätzen und nur durch Versuche validierbar.

5.2. Testsamples

Insgesamt wurden zwei Aufnahmesamples von derselben Baustelle analysiert. Diese sind jeweils nach dem Datum der Aufnahmen benannt und werden durch folgende Kennwerte klassifiziert:

Name	Bildqualität	Speichergröße	Bildzahl
2017-05-17	schlecht	2,24 GB	480
2018-05-23	gut	4,88 GB	1377

Tabelle 2 Kennwerte der Ausgangsdatensets

Die Bildqualität ist dabei eine subjektive Aussage. Bei stichprobenartigem Sichten der Bilddaten werden wenige, bzw. viele Bilder als Unscharf wahrgenommen. Dieser Eindruck wird durch die Zahl der aussortierten Bilder bei reiner Unschärfebetrachtung unterstützt (siehe Tabelle 4, Set Nr. 2 & 6).
Im weiteren Verlauf wurden aus beiden Datensätzen jeweils drei aussortierte Samples erzeugt. Dabei wurden die im Abschnitt 5.1 beschriebenen Konstanten entsprechend der Tabelle 3 variiert. Die sich ergebenden Kennwerte werden in Tabelle 4 zusammengefasst.

Set Nr.	Ausgangs-sample	blur_val	max_num_pic_-per_voxel	voxel_size	seg_hard
1	2017-05-17	100	3	[5, 5, 3]	False
2	2017-05-17	100	3	[5, 5, 3]	True
3	2017-05-17	150	5	[10, 10, 5]	False
4	2018-05-23	200	5	[10, 10, 5]	False
5	2018-05-23	200	10	[20, 20, 10]	False
6	2018-05-23	200	10	[20, 20, 10]	True

Tabelle 3 Variation der Konstanten zur Generierung der Datensets

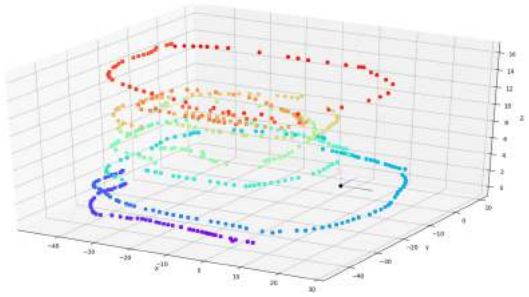
Set Nr.	Neue Speicher-größe	Rel. Speicher-größe	Neue Bild-zahl	Rel. Bild-zahl
1	1,88 GB	84 %	390	81 %
2	0,63 GB	28 %	127	26 %
3	1,45 GB	65 %	300	63 %
4	3,80 GB	79 %	1035	75 %
5	3,06 GB	63 %	837	61 %
6	2,94 GB	60 %	805	58 %

Tabelle 4 Kennwerte der sortierten Sets

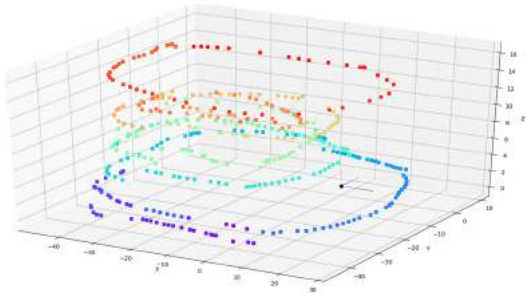
Zur Bewertung der Datensets bildet die rel. Bildzahl die wichtigste Kennzahl. Sie berechnet sich entsprechend der Formel 5.1. Anhand dieser Kennzahl kann die Abnahme der Bildzahl beschrieben werden. Durch eine spätere Betrachtung der benötigten Zeit zur Punktwolkengenerierung kann festgestellt werden, in welcher Ordnung die Laufzeit von der Bildzahl abhängt. Darüber hinaus ist sie für die Einordnung der Qualitätsveränderung maßgeblich.

$$\text{rel. Bildzahl} = \frac{\text{neue Bildzahl}}{\text{Ausgangsbildzahl}} \tag{5.1}$$

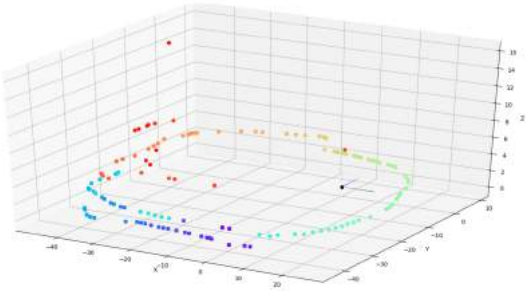
Zur Veranschaulichung der Datensets wurden diese in den Grafiken 15 & 16 visualisiert. Dabei stellt jeder Punkt in den Plots ein Bild dar. Der Farbverlauf von lila zu rot verkörpert die Reihenfolge der Bilder in der Ordnerstruktur. Diese korreliert in den meisten Fällen mit dem Aufnahmezeitpunkt der Bilder und kann somit auch als Trajektorie der Drohne interpretiert werden. Diese Annahme muss aber nicht ausnahmslos stimmen und kann nur als grobe Richtschnur gelten.
Bei allen aussortierten Sets fällt auf, dass die Gesamtzahl der Punkte und somit auch die Zahl der Bilder gesunken ist. Bei Set Nr. 2 erkennt man insbesondere, dass die radikale Entfernung unscharfer Bilder in Verbindung mit der schlechten Bildqualität zu großen Lücken in der Bildsammlung führen. Das die Option $seg\_hard =$ True jedoch nicht automatisch zu großen Informationsverlusten führen muss, sieht man bei Set Nr. 6, welches im Gegensatz zu Set Nr. 2 aus Bildern mit hoher Qualität besteht. Somit kann sichergestellt werden, dass trotz einer vollständigen Selektion der unscharfen Bilder, genügend Bilder verbleiben. Darüber hinaus erkennt man hierbei, dass dieses zusätzliche verschärfende Selektionskriterium bei guter Bildqualität lediglich wenig bis keinen Einfluss hat (vgl. Tabelle 4, Set Nr. 5 & 6). Durch die Versuche muss deshalb untersucht werden, ob bei der Punktwolkenerzeugung unscharfe Bilder besser sind als gar keine Informationen oder ihr erhöhter Rechenaufwand in keinem Verhältnis zu den zusätzlich generierten Informationen stehen.



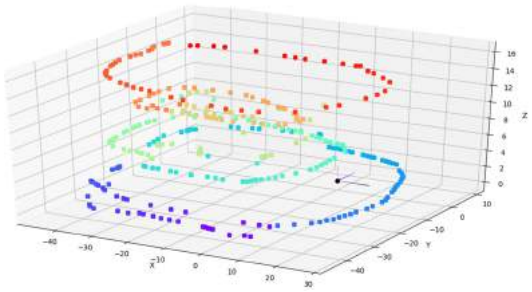
(a) Ausgangsset



(b) Set Nr. 1

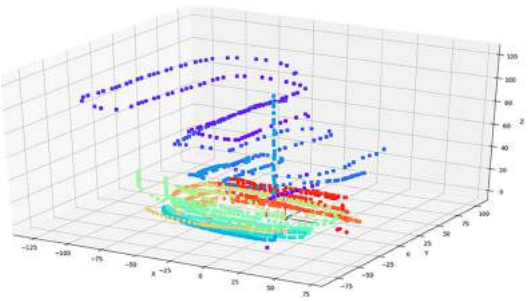


(c) Set Nr. 2

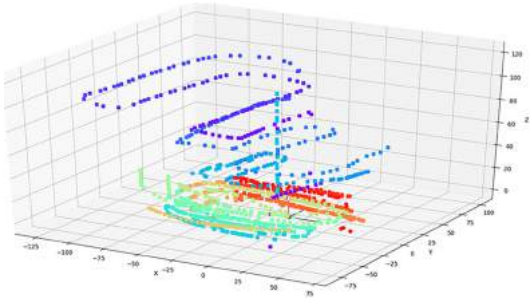


(d) Set Nr. 3

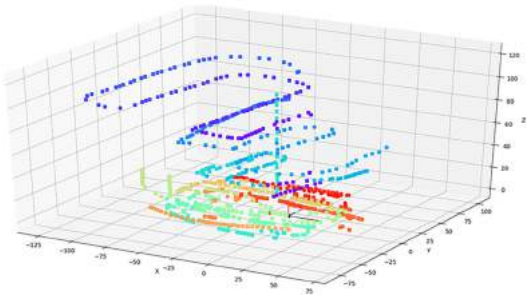
Abbildung 15 Visualisierung der verschiedenen Variationen des 2017er Datensets.



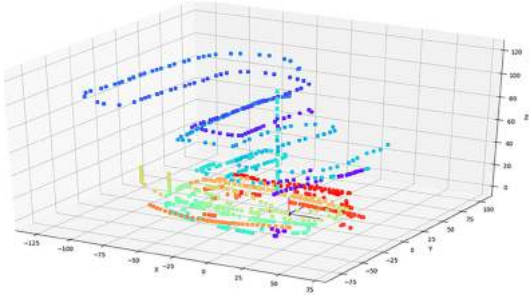
(a) Ausgangsset



(b) Set Nr. 4



(c) Set Nr. 5



(d) Set Nr. 6

Abbildung 16 Visualisierung der verschiedenen Variationen des 2018er Datensets.

5.3. Versuchsablauf

Aus den zuvor beschriebenen Sets wird jeweils mithilfe der Software COLMAP eine Punktwolke erzeugt. Dabei besteht jede Berechnung aus insgesamt sechs Schritten. Die einzelnen Schritte verfolgen dabei unterschiedliche Ziele und fallen deshalb auch in ihrem Rechenaufwand stark unterschiedlich aus. Folgende Schritte werden durchgeführt:

- **Feature extractor:** In jedem Bild wird einzeln nach Merkmalen gesucht. Diese werden gesammelt und zusammengefasst. Da alle Bilder einzeln, jedoch nicht in Zusammenhang mit den anderen Bildern untersucht werden, ist dieser Schritt relativ schnell.
- **Exhaustive match:** Alle zuvor gefundenen Merkmale werden in jedem einzelnen Bild gesucht. Mithilfe dieser Informationen kann der Algorithmus zurückrechnen an welcher Stelle sich die Kamera zum Zeitpunkt der Aufnahme befunden hat. Mit einem Anteil von 10-15% an der Gesamtzeit ist dies der zweit intensivste Rechenschritt.
- **Image undistorter:** Bei der Aufnahme der Bilder erzeugt jedes Objektiv eine charakteristische Verzerrung. Diese wird in diesem Schritt betrachtet um unter Anwendung dieser Informationen die Bilder zu orthonormalisieren.
- **Patch match stereo:** In diesem Schritt werden die Tiefeninformationen berechnet. Dabei werden sich jeweils überlappende Bilder zusammen betrachtet und mithilfe der Kameraposition zum Zeitpunkt der Aufnahme sowie sich wiederholender Merkmale kann eine Aussage über die Tiefe getroffen werden. Diese einzelnen Merkmale werden anschließend versucht durch weitere Bilder zu bestätigen und somit die Tiefeninformation verlässlicher zu berechnen. Dieser Schritt ist mit rund 80% Anteil an der Gesamtzeit der mit Abstand rechenintensivste Schritt.
- **Stereo fusion:** Die im vorherigen Schritt erzeugten Tiefeninformationen werden miteinander verknüpft und zu einem vollständigen Modell zusammengeführt.

Zur Analyse der Daten wird eine Initialisierungsdatei für COLMAP benötigt. Diese Datei gibt Aufschluss über die Kennwerte die zur Berechnung benötigt werden. Die für alle Punktwolken genutzte Datei findet sich im Anhang A. Wichtig hierbei ist, dass bis zum Schritt „Patch match stereo“ mit der maximalen Bildgröße gerechnet wurde. Zu Beginn des vorletzten Schritts wurde diese jedoch skaliert und auf eine maximale Größe von 2400 reduziert. Dies führt zu einer Beschleunigung der folgenden Rechenschritte.

5.4. Versuchsergebnisse

Die Berechnung der Datensets fand auf einem zentralen Computer des Lehrstuhls für „Computergestützte Modellierung und Simulation“ statt. Dabei wurden jeweils die benötigten Zeiten für die Berechnung der Datensets in verschiedene log-Files geschrieben. Hierbei handelt es sich um Dateien, die die Ausgabe des Codes mitschreiben und dokumentieren. Sie beinhalten Informationen über den Verlauf der Punktwolkenerzeugung, aufgetretene Fehler sowie die benötigte Zeit zur Berechnung.
Im Rahmen der Berechnungen kam es bei einigen log-Files zu Fehlern, sodass keine, bzw. unvollständige Laufzeitinformationen vorliegen. Dies gilt für das Set 2 (vollständig unbrauchbar) und das Ausgangsdatenset 2018 (fehlende Informationen zu den letzten drei Schritten). Um trotzdem eine sinnvolle Analyse der Daten vornehmen zu können werden ausschließlich die restlichen Informationen ausgewertet.
Zur Bewertung der Ergebnisse werden vorrangig zwei Kriterien betrachtet. Zum einen wird die benötigte Zeit pro Berechnung, zum anderen die Qualität der Punktwolke verglichen.

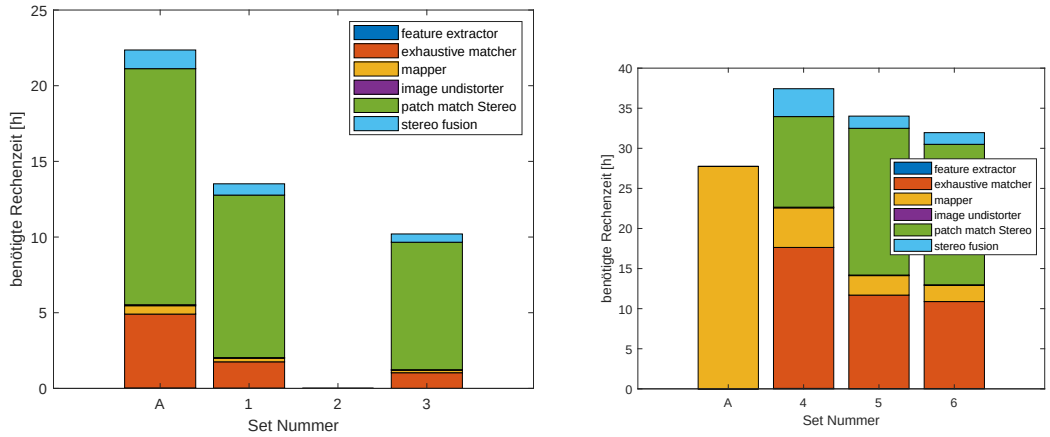
5.4.1. Ergebnisse Analyse Rechenaufwand

Die benötigten Rechenzeiten pro Datenset werden in der Abbildung 17 dargestellt. Die Sets werden jeweils mit ihrer Nummer, das Ausgangsset durch den Buchstaben „A“ gekennzeichnet. Während bei den Datensätzen zum Jahr 2017 alle Rechenschritte verglichen werden, werden bei den Datensätzen aus 2018 lediglich die ersten drei Schritte verglichen.
Bei allen Datensets ist zu beobachten, dass bei steigender Bildzahl die Laufzeiten deutlich länger werden. Verallgemeinert kann also festgehalten werden, dass eine höhere Bildzahl zu einer erhöhten Laufzeit führt. In diesem Zusammenhang ist besonders die Tabelle 5 zu beachten. Sie vergleicht die relative Bildzahl mit den relativen Laufzeiten (Definition siehe Formel 5.2).

$$\text{rel. Laufzeit} = \frac{\text{neue Laufzeit}}{\text{Ausgangsset Laufzeit}} \tag{5.2}$$

Hierbei lässt sich erkennen, dass der Rechenaufwand im Verhältnis zur Bildzahl nicht linear abnimmt. Eine erste Vermutung lässt auf eine logarithmische Abnahme schließen (siehe auch Abbildung 17a). Entsprechende Annahmen müssten jedoch durch weitere Versuche validiert werden.
Zu Beginn der Versuche wird die Aussage formuliert, dass der zeitliche Gewinn als erfolgreich zu betrachten ist, wenn mindestens 20 % Zeitersparnis geschafft wird. Bei Betrachtung der Sets wird dies in allen vollständig berechneten Sets geschafft. Auch bei den Sets aus dem Jahr 2018 zeichnet sich vorwiegend dieser Trend ab. Lediglich bei Datenset Nr. 4 wäre ein genauer Vergleich mit den endgültigen Rechenzeiten des Ausgangssets 2018 nötig um eine klare Aussage treffen zu können. Trotz dieser Unsicherheit kann insgesamt gesagt werden, dass eine Zeitoptimierung von mindestens 20 % eingehalten und sogar teilweise übertroffen

wird.s



(a) Set Jahr 2017 (b) Set Jahr 2018

Abbildung 17 Gesamtzeit der verschiedenen Datensets geordnet nach ihren jeweiligen Jahrgängen

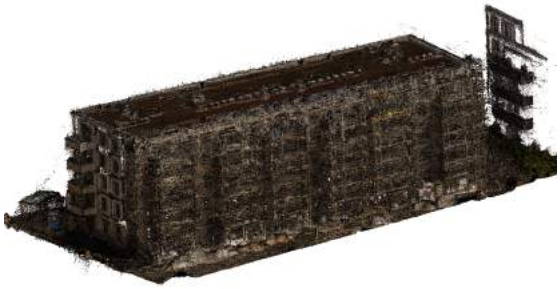
Set Nr.	Abs. Zeit	Rel. Bildzahl	Rel. Zeit Schritt 1-3	Rel. Zeit Gesamt
A17	22.36 h	-	-	-
1	13.52 h	81 %	36 %	60 %
2	-	26 %	-	-
3	10.20 h	63 %	22 %	46 %
A18	27.74 h	-	-	-
4	37.43 h	75 %	81 %	-
5	34.02 h	61 %	51 %	-
6	31.95 h	58 %	47 %	-

Tabelle 5 relative Zeiten der verschiedenen Datensets. Fehlende Informationen sind durch ein „-“ kenntlich gemacht.

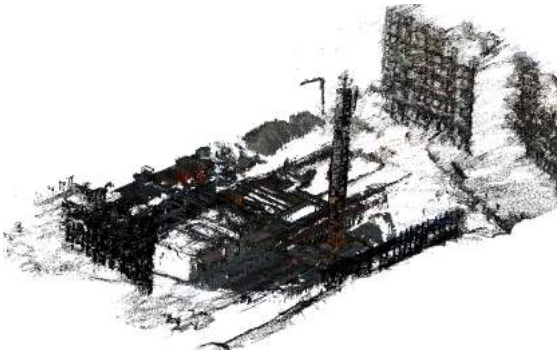
5.4.2. Ergebnisse Analyse Qualität

Zur Qualitätsprüfung wurden die Gesamtpunktwolken, sowie einzelne Details näher betrachtet. Darüber hinaus wurden die Datensätze von 2018 mit einem zusätzlichen Laserscan verglichen. Die Übersicht der sich ergebenden Punktwolken sind in Abbildung 18 dargestellt. Dabei fällt auf, dass sich die ergebenden Punktwolken für das 2018er Set äußerlich nicht unterscheiden. Im Gegensatz hierzu gibt es bei dem 2017er Set große Unterschiede zwischen Set 1 & 3 einerseits und 2 andererseits. Da beim zweiten Set alle unscharfen Bilder entfernt wurden, können deutlich weniger Informationen gewonnen werden. Dies zeigt sich besonders bei der Detailaufnahme 1 (siehe Abb. 19). Während die Abdeckung der Baustelle bei den $seg\_hard =$ False Datensets erkennbar ist, bleibt sie im zweiten Set fast vollständig undetektiert. Bei fehlenden Detailinformation, wie den Stützvorrichtungen im Detail 2 (Abb. 20), kommt es in allen Punktwolken zu Problemen. In diesem Fall wirkt sich die radikale Selektion weder positiv, noch negativ aus. Zusammenfassend kann man festhalten, dass trotz der geringen Bildqualität eine gute Rekonstruktion bei der ersten und dritten Punktwolke erreicht wird. Das zweite Set ist aufgrund der viel zu geringen Bildzahl nicht gut geeignet, den Zustand der Baustelle zum Aufnahmezeitpunkt verlässlich wiederzugeben.

(a) Set Nr. 1



(b) Set Nr. 4



(c) Set Nr. 2

(d) Set Nr. 5

(e) Set Nr. 3

(f) Set Nr. 6

Abbildung 18 Die erzeugten Punktwolken der verschiedenen Datensets

(a) Set Nr. 1

(b) Set Nr. 2

(c) Set Nr. 3

Abbildung 19 Detailansicht 1 des 2017er Datensets. Große Informationsverluste aufgrund entfernter Bilder.

(a) Set Nr. 1 (b) Set Nr. 2 (c) Set Nr. 3

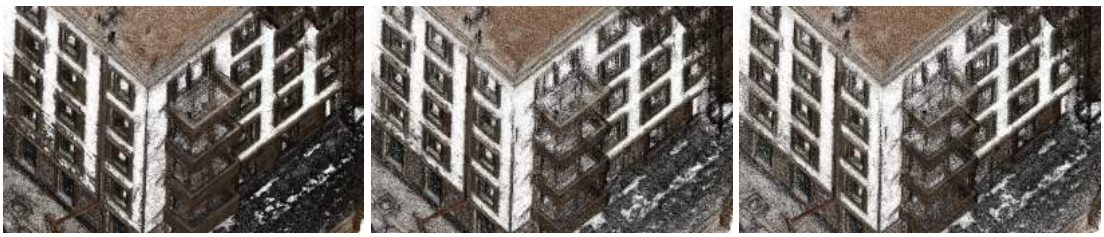
Abbildung 20 Detailansicht 2 des 2017er Datensets. Schlechte aufgelöste Stützstangen bleiben in allen drei Datensets von gleicher Qualität.

Mithilfe des 2018er Sets lässt sich insbesondere eine gute Aussage über die zu setzenden Variablen machen. Während für eine allgemeine Analyse, alle drei Punktwolken eine gute Qualität liefern, sieht man insbesondere bei den Details, wie sich die unterschiedliche Konfiguration der Variablen *max_num_pic_per_voxel* und *voxel_size* auf die Auflösung dünner Strukturen auswirkt (siehe Abb. 21). Während im Set Nr. 4 für die Stützstrukturen viele Punkte gefunden werden, fällt die Anzahl an Punkten bei 5 & 6 deutlich geringer aus. Dies hängt mit der höheren Rate an aussortierten Bildern zusammen. Der selbe Eindruck wird durch die Detailansicht 2 in Abbildung 22 bestätigt. Durch die abnehmende Bildzahl wird die Punktdichte verringert. Trotzdem wird in allen drei Sets eine hohe, bis sehr hohe Präzision erreicht. Um diesen ersten Eindruck zu belegen, wurde eine Punktwolke stellvertretend mit dem Laserscan verglichen. Dabei wird von jedem Punkt der Punktwolke, der nächste Punkt im Laserscan bestimmt. Die sich dabei ergebende Distanz ist dabei dimensionslos, wird aber in der weiteren Beschreibung zum einfacheren Verständnis mit der Einheit Punktwolkeneinheit [PE] bezeichnet.
Um die sich ergebenden Abweichungen in eine sinnvolle Maßeinheit, Meter, umzuwandeln wird in diesem Rahmen eine Abschätzung vorgenommen. Hierfür wird das Wissen genutzt, dass der Laserscan im Gegensatz zur Punktwolke keine Aufnahmen von oben behinhaltet und das Dach somit fehlt (Siehe Abb. 23a). Unter Anwendung dieser Information wird angenommen, dass in der Mitte des Dachs der größte Abstand zum nächsten Punkt existiert, sodass dieser Abstand dem Maximum von 7.411 PE entspricht (siehe Abb. 23b). In einem nächsten Schritt wird durch die Software Google Maps die Breite des Gebäudes auf 22 m vermessen. Der maximale Abstand in Metern ergibt sich daraus aus der halben Gebäudebreite zu 11 m, was somit 7.411 PE entspricht. Daraus folgt, dass ein 1 PE etwa 1.5 m sind. Bei genauerer Betrachtung des Vergleichs Punktwolke $\leftrightarrow$ Laserscan fällt auf, dass ein Großteil der Punkt einen Abstand kleiner 0.3 PE besitzt was einer Abweichung kleiner 50 cm entspricht. Diese mit dem Laserscanner verglichene Punktwolke wurde in einem weiteren Schritt mit den beiden verbleibenden Punktwolken verglichen. Auch hierbei ergaben sich geringe Abweichungen (mehr als 90 % der Punkte kleiner 0.3PE). Es wird also bei allen Punktwolken aus dem Jahr 2018 eine hohe Präzision erreicht.
Zusammenfassend kann man sagen, dass der Qualitätsverlust in den 2018er Sets objektiv, sowie subjektiv sehr gering ausfällt. Das bedeutet, dass die Qualitätsverluste kleiner 20% sind und somit die Eingangs geforderten 80% Qualität der Ausgangspunktwolke erreicht wer-

den. Die Selektion ist in Bezug auf die Qualität somit erfolgreich.

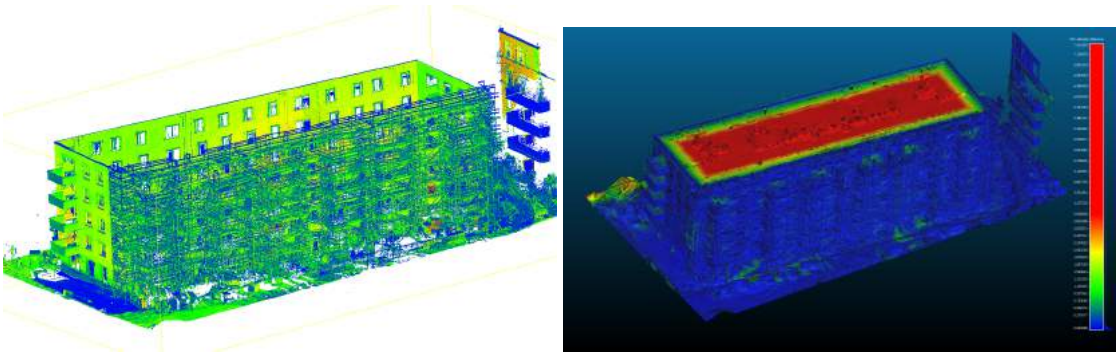
(a) Set Nr. 4 (b) Set Nr. 5 (c) Set Nr. 6

Abbildung 21 Detailansicht 1 des 2018er Datensets. Je höher die Bildzahl, desto besser werden Details dargestellt.



(a) Set Nr. 4 (b) Set Nr. 5 (c) Set Nr. 6

Abbildung 22 Detailansicht 2 des 2018er Datensets. Große, homogene Flächen können schwer bis gar nicht erfasst werden. Dahingegen werden die filigranen Balkonstrukturen zwar in allen Bildern detektiert, aber die Punktdichte ist stark abhängig von der Bildanzahl.



(a) Laserscan (b) Vergleich des Set Nr. 4 mit dem Laserscan

Abbildung 23 Die Laserscandaten, sowie die sich ergebende Abweichung vom Laserscan.

6. Fazit

Im Rahmen dieser Arbeit sollte ein Konzept, sowie eine Umsetzung zur Vorselektion von Bildern für die automatisierte Baufortschrittskontrolle entwickelt werden. Hierzu wurden drei Methoden entwickelt, wovon sich zwei als zielführend erwiesen haben. Unter Anwendung der Methoden wurden im Anschluss zwei verschiedene Datensets derselben Baustelle jeweils drei Mal selektiert, weshalb sich insgesamt sechs Sets ergaben. Nach Erzeugung der Punktwolken wurden die Sets auf Laufzeit und Qualität verglichen.
Die Analyse der Testergebnisse ergab dabei, dass bei minimal abnehmender, bzw. gleich bleibender Punktwolkenqualität die Laufzeit der Punktwolkenerzeugung deutlich optimiert werden kann. Es kann somit gesagt werden, dass die Vorselektion der Daten erfolgreich ist und auch bei weiteren Projekten angewendet werden kann.
Eine erfolgreiche Selektion hängt im Wesentlichen auch von den Selektionskonstanten ab. Um diese besser nutzen zu können wird nachfolgend, basierend auf den Versuchsergebnissen, eine Empfehlung gegeben.

6.1. Empfehlungen für ausgewählte Selektionskonstanten

Im Abschnitt 5.1 „Versuchsziel“ werden die an der Selektion maßgeblich beteiligten Konstanten vorgestellt. Um diese bei zukünftigen Projekten sinnvoll zu konfigurieren, wird in diesem Abschnitt für jede der Konstanten eine Standardeinstellung begründet empfohlen.

- *blur_val*: Bei durchschnittlicher bis guter Bildqualität sollte ein Wert von 200 für die Varianz verwendet werden. Lediglich bei extrem schlechten Datensets kann eine Korrektur nach unten vorgenommen werden. Bilder unter einer Varianz von 100 sind nur im Falle fehlender Alternativen dienlich (siehe Unterschied Set Nr. 1 & 2, Abb. 19).
- *voxel_size*: Es hat sich bewährt, dass die ebenen Komponenten, x und y, jeweils die selbe Länge besitzen, da die Grundfläche des Aufnahmegebiets etwa quadratisch ist. Da die absolute Höhe zumeist kleiner ausfällt, sollte diese etwa halb so groß gewählt werden. Grobe Richtwerte können hierbei abhängig von dem Fokus des Anwenders ausgesprochen werden. Liegt der Fokus auf einer guten Auflösung, so sollten etwa 5% der größten ebenen Ausdehnung als x und y Werte gewählt werden. Liegt er hingegen auf einer schnellen Berechnung so sollten 10% vorgezogen werden. Beispiel: Setgröße [200, 175, 50].
  - Voxelgröße bei guter Auflösung: [10, 10, 5]
  - Voxelgröße bei schneller Berechnung: [20, 20, 10]

  Bei besonders großen, bzw. besonders kleinen Modellen, sollte die Voxelgröße nicht an der prozentualen Ausdehnung ausgerichtet werden. Stattdessen kann eine der Ausgangskombinationen herangezogen werden (siehe 5.1, Absatz *voxel_size*).
- *max_num_pic_per_voxel*: Diese sollte ungefähr dem Wert der Höhe entsprechen. Für

unser Beispiel bedeutet dies im ersten Fall 5, im zweiten Fall 10.

- *seg_hard*: Es hat sich gezeigt, dass Bilder, die Unscharf und im Zweifel behalten werden, einen Einfluss auf die Qualität der Punktwolke haben (siehe Set Nr. 2, Abb. 19). Aus diesem Grund ist es besser, unscharfe Bilder einzubeziehen, als gar keine. Für *seg_hard* sollte deshalb stets der Wert $False$ als Standardkonfiguration gewählt werden. Lediglich bei genauer Kenntnis über die Bilddatenqualität kann eine Variation dieser Konstante vorgenommen werden.

6.2. Ausblick

Die Selektion der Bilder findet aktuell auf Basis der Unschärfe, sowie der Position der Drohne zum Zeitpunkt der Aufnahme statt. Um diese Selektion weiter zu verbessern und zu optimieren, können darüber hinaus weitere Selektionskriterien zu den bestehenden hinzugefügt werden. Dabei ist besonders die modulare Struktur des Codes und der Aufbau des Selektionsalgorithmus hilfreich (siehe Formel 4.2).
Eine mögliche Erweiterung bieten hierbei die XMP-Informationen der Drohne. Diese beinhalten insgesamt sechs verschiedene Winkel die der Verdrehung Drohne $\leftrightarrow$ Referenzsystem, sowie Kamera $\leftrightarrow$ Drohne entsprechen. Mit deren Hilfe kann die Ausrichtung der Kamera zum Zeitpunkt der Aufnahme bestimmt werden. Ist diese in Richtung des Aufnahmeziels, so wird das Bild behalten, geht es in eine andere Richtung wird das Bild entfernt. Bei allen als gut befundenen Bildern kann darüber hinaus approximiert werden, wie viel Prozent des Bildes die Baustelle darstellen um anhand dieser Prozentzahl die Bilder zu ranken.
Eine weitere Verbesserung, kann durch die Variation des Einsatzzeitpunktes der Software erreicht werden. Aktuell kann die Software lediglich nach Abschluss der Aufnahmen der Bilder „das Beste“ aus den vorliegenden Daten machen. Mithilfe einiger Anpassungen bietet diese Software jedoch die Möglichkeit, die Selektion und Betrachtung der Datengüte direkt vor Ort bei der Aufnahme der Bilder „live“ durchzuführen. Dadurch kann direkt ermittelt werden, welche Bereiche der Baustelle bereits sinnvoll abgelichtet wurden, und in welchen Bereichen noch wenig bzw. schlechte Informationen vorliegen. Die Selektion würde also nicht mehr im Nachhinein passiv stattfinden, sondern aktiv in den Prozess der Datenerhebung eingreifen und diesen optimieren. Durch dieses Vorgehen wird nicht nur sichergestellt, dass die Baustelle effizient und vollständig erfasst wird, es ermöglicht auch einem ungeübten Anwender brauchbare Daten zu erzeugen.

Literaturverzeichnis

[Borrmann et al., 2015] Borrmann, A., König, M., Koch, C., and Beetz, J., editors (2015). *Building Information Modeling: Technologische Grundlagen und industrielle Praxis*. VDI-Buch. Springer Vieweg, Wiesbaden.

[Braun et al., 2015] Braun, A., Tuttas, S., Borrmann, A., and Stilla, U. (2015). A concept for automated construction progress monitoring using bim-based geometric constraints and photogrammetric point clouds. *ITcon*, 20(8):68–79.

[Braun et al., 2016] Braun, A., Tuttas, S., Stilla, U., and Borrmann, A. (2016). Classification of detection states in construction progress monitoring. In *11th European Conference on Product and Process Modelling*, Limassol, Cyprus.

[Burger and Burge, 2015] Burger, W. and Burge, M. J. (2015). *Digitale Bildverarbeitung: Eine algorithmische Einführung mit Java*. X.media.press. Springer, Berlin [u.a.].

[Camera & Imaging Products Association, 2016] Camera & Imaging Products Association (2016). *Exchangeable image file format for digital still cameras: Exif Version 2.31: CIPA DC-008-Translation-2016*.

[Eipke, 2017] Eipke, C., editor (2017). *Photogrammetrie und Fernerkundung: Handbuch der Geodäsie*. Springer Reference Naturwissenschaften. Springer Berlin Heidelberg, Berlin, Heidelberg and s.l.

[Furht, 2008] Furht, B. (2008). Jpeg. In Furht, B., editor, *Encyclopedia of Multimedia*, pages 377–379. Springer, New York, NY.

[Helmers, 2013] Helmers, H. (2013). Randschärfe: Umfangreiches online nachschlagwerk zu (fast) allen fragen der digitalen fotografie. `http://www.henner.info/randscharf.htm` (abgerufen am 28.06.2018).

[Hirschmüller, 2011] Hirschmüller, H. (2011). Semi-global matching - motivation, developments and applications. In Dieter Fritsch, editor, *Photogrammetric Week*, pages 173–184. Wichmann.

[OpenCV team, 2018] OpenCV team (2018). Opencv documentation. `https://opencv.org/` (abgerufen am 15.08.2018).

[Schönberger and Frahm, 2016] Schönberger, J. and Frahm, J.-M. (2016). Structure-from-

motion revisited. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-January, pages 4104–4113.

[Tuttas et al., 2017] Tuttas, S., Braun, A., Borrmann, A., and Stilla, U. (2017). Acquisition and consecutive registration of photogrammetric point clouds for construction progress monitoring using a 4d bim. *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, 85(1):3–15.

[Veguillas, 2014] Veguillas, E. (2014). Paper napkins. `https://www.flickr.com/photos/la_nina_graphics/11779947584/` (abgerufen am 28.06.2018).

[Wu, 2013] Wu, C. (2013). Towards linear-time incremental structure from motion: 3d vision, 2013 international conference, seattle. In *Proceedings - 2013 International Conference on 3D Vision, 3DV 2013*, pages 127–134.

Abbildungsverzeichnis

Anhang

A. COLMAP Init-File

Diese Datei wird für die Versuchsdurchführung der Testsamples benötigt. Alle für COLMAP benötigten Kennwerte befinden sich in der Datei.

```
log_to_stderr=false
log_level=2
database_path=$project/database.db
image_path=$project/images
export_path=$project/sparse
output_path=$project/dense
[ImageReader]
single_camera=true
single_camera_per_folder=false
existing_camera_id=-1
default_focal_length_factor=1.2
camera_model=SIMPLE_RADIAL
camera_params=
[SiftExtraction]
use_gpu=true
estimate_affine_shape=false
upright=false
domain_size_pooling=false
num_threads=-1
max_image_size=4000
max_num_features=8192
first_octave=-1
num_octaves=4
octave_resolution=3
max_num_orientations=2
dsp_num_scales=10
peak_threshold=0.0066666666666666671
edge_threshold=10
dsp_min_scale=0.16666666666666666
dsp_max_scale=3
gpu_index=-1
[SiftMatching]
use_gpu=true
cross_check=true
multiple_models=false
guided_matching=false
num_threads=-1
max_num_matches=32768
max_num_trials=10000
min_num_inliers=15
max_ratio=0.80000000000000004
max_distance=0.69999999999999996
max_error=4
confidence=0.999
min_inlier_ratio=0.25
gpu_index=-1
[SequentialMatching]
quadratic_overlap=true
loop_detection=false
overlap=5
loop_detection_period=10
loop_detection_num_images=30
loop_detection_num_nearest_neighbors=1
loop_detection_num_checks=256
loop_detection_num_images_after_verification=0
loop_detection_max_num_features=-1
vocab_tree_path=
[SpatialMatching]
is_gps=true
ignore_z=true
max_num_neighbors=50
max_distance=100
[BundleAdjustment]
refine_focal_length=true
refine_principal_point=false
refine_extra_params=true
max_num_iterations=100
max_linear_solver_iterations=200
function_tolerance=0
gradient_tolerance=0
parameter_tolerance=0
[Mapper]
ignore_watermarks=false
multiple_models=true
extract_colors=true
ba_refine_focal_length=true
ba_refine_principal_point=false
ba_refine_extra_params=true
ba_global_use_pba=true
tri_ignore_two_view_tracks=true
min_num_matches=15
max_num_models=50
max_model_overlap=20
min_model_size=10
init_image_id1=-1
init_image_id2=-1
```

```
init_num_trials=200
num_threads=-1
ba_local_num_images=6
ba_local_max_num_iterations=25
ba_global_pba_gpu_index=-1
ba_global_images_freq=500
ba_global_points_freq=250000
ba_global_max_num_iterations=50
ba_global_max_refinements=5
ba_local_max_refinements=2
snapshot_images_freq=0
init_min_num_inliers=100
init_max_reg_trials=2
abs_pose_min_num_inliers=30
max_reg_trials=3
tri_max_transitivity=1
tri_complete_max_transitivity=5
tri_re_max_trials=1
min_focal_length_ratio=0.10000000000000001
max_focal_length_ratio=10
max_extra_param=1
ba_global_images_ratio=1.1000000000000001
ba_global_points_ratio=1.1000000000000001
ba_global_max_refinement_change=0.0005
ba_local_max_refinement_change=0.001
init_max_error=4
init_max_forward_motion=0.94999999999999996
init_min_tri_angle=16
abs_pose_max_error=12
abs_pose_min_inlier_ratio=0.25
filter_max_reproj_error=4
filter_min_tri_angle=1.5
tri_create_max_angle_error=2
tri_continue_max_angle_error=2
tri_merge_max_reproj_error=4
tri_complete_max_reproj_error=4
tri_re_max_angle_error=5
tri_re_min_ratio=0.20000000000000001
tri_min_angle=1.5
snapshot_path=
[PatchMatchStereo]
geom_consistency=true
filter=true
write_consistency_graph=false
max_image_size=2400
window_radius=5
window_step=1
num_samples=15
num_iterations=5
filter_min_num_consistent=2
depth_min=-1
depth_max=-1
sigma_spatial=-1
sigma_color=0.20000000298023224
ncc_sigma=0.60000002384185791
min_triangulation_angle=1
incident_angle_sigma=0.89999997615814209
geom_consistency_regularizer=0.30000001192092896
geom_consistency_max_cost=3
filter_min_ncc=0.10000000149011612
filter_min_triangulation_angle=3
filter_geom_consistency_max_cost=1
cache_size=56
gpu_index=-1
[Render]
adapt_refresh_rate=true
image_connections=false
min_track_len=3
refresh_rate=1
projection_type=0
max_error=2
[ExhaustiveMatching]
block_size=50
[VocabTreeMatching]
num_images=100
num_nearest_neighbors=5
num_checks=256
num_images_after_verification=0
max_num_features=-1
vocab_tree_path=
match_list_path=
[TransitiveMatching]
batch_size=1000
num_iterations=3
[StereoFusion]
max_image_size=-1
min_num_pixels=5
max_num_pixels=10000
max_traversal_depth=100
check_num_images=50
max_reproj_error=2
max_depth_error=0.0099999997764825821
max_normal_error=10
cache_size=48
[PoissonMeshing]
depth=13
num_threads=-1
point_weight=1
color=32
trim=10
[DelaunayMeshing]
num_threads=-1
max_proj_dist=20
max_depth_dist=0.050000000000000003
```

```
distance_sigma_factor=1
quality_regularization=1
max_side_length_factor=25
max_side_length_percentile=95
[ImageUndistorter]
output_type=COLMAP
blank_pixels=0
min_scale=0.2
max_scale=2
max_image_size=2400
roi_min_x=0
roi_min_y=0
roi_max_x=1
roi_max_y=1
```

B. Gesamtübersicht Testsample Kennwerte

Allgemeine Setinformationen

Set Nr.	Ausgangssample	blur_val	max_num_pic-_per_voxel	voxel_size	seg_hard	Neue Speichergröße	Rel. Speichergröße	Neue Bildzahl	Rel. Bildzahl
1	2017-05-17	100	3	5,5,3	False	1.88	84%	390	81%
2	2017-05-17	100	3	5,5,3	True	0.63	28%	127	26%
3	2017-05-17	150	5	10,10,5	False	1.45	65%	300	63%
4	2018-05-23	200	5	10,10,5	False	3.8	79%	1035	75%
5	2018-05-23	200	10	20,20,10	False	3.06	63%	837	61%
6	2018-05-23	200	10	20,20,10	True	2.94	60%	805	58%

Benötigte Rechenzeiten zur Punktwolkenerzeugung

Set Nr.	benötigte Gesamtzeit [h]	benötigte Gesamtzeit [min]	feature extractor [min]	exhaustive matcher [min]	mapper [min]	image undistorter [min]	patch match Stereo [min]	stereo fusion [min]
A17	22.36	1341.60	1.759	292.919	33.063	3.926	935.839	74.093
1	13.52	811.06	1.3	104.042	14.198	2.63	644.21	44.676
2	0.07	4.45	0	0.003	3.544	0.892	0.008	0
3	10.20	612.25	0.225	61.891	9.954	2.102	505.283	32.79
A18	27.74	1664.48	0.001	0.31	1664.171	0	0	0
4	37.43	2245.94	3.377	1054.844	296.004	5.327	677.85	208.535
5	34.02	2040.97	2.554	698.973	145.784	4.281	1098.11	91.264
6	31.95	1917.22	2.654	650.445	121.379	4.117	1051.275	87.346

A17 entspricht Ausgangsdatenset 2017
A18 entspricht Ausgangsdatenset 2018

C. Übersicht Inhalte Speichermedium

Das angehängte Speichermedium enthält verschiedene Inhalte. Alle Inhalte werden in diesem Rahmen aufgeführt und falls nötig erklärt.

- **01_WissenschaftlicheArbeit.pdf:** Enthält die vollständige Bachelorarbeit als digitales PDF.
- **02_Grafiken.zip:** Alle genutzten Grafiken sind hier in ihrer originalen Größe vorliegend, sodass sie in voller Auflösung betrachtet werden können.
- **03_PythonCode.zip:** Enthält den vollständigen Source Code der zur Selektion genutzt wurde (Stand: 10.10.2018). Darüber hinaus wird das Projekt in einem GIT gehostet. Dieses kann unter folgendem Link erreicht werden: git@gitlab.lrz.de:ga59mok/code_BA_felix_nothdurft.git