Institut für Geodäsie und Geoinformation

Professur für Theoretische Geodäsie

Effiziente und automatisierte Detektion von Kugel-Zielzeichen in 3D-Scan-Punktwolken

Masterarbeit

im Masterstudiengang Geodäsie und Geoinformation

an der Landwirtschaftlichen Fakultät

der Rheinischen Friedrich–Wilhelms–Universität

Bonn

Matthias Schreinemachers

aus Heinsberg

Bonn 2020

Erster Prüfer: Prof. Dr. techn. W.–D. Schuh

Zweiter Prüfer: Dr.-Ing. Jan Martin Brockmann

Institut of Geodesy and Geoinformation
Theoretical Geodesy Group
University of Bonn

Prof Dr. techn. Wolf-Dieter Schuh
Nußallee 17
53115 Bonn

☎ +49 228 73 26 26
+49 228 73 64 86
http://www.tg.uni-bonn.de

24. April 2020

Thema der Masterarbeit für Matthias Schreinemachers

Effiziente und automatisierte Detektion von Kugel-Zielzeichen in 3D-Scan-Punktwolken
(Efficient and automated detection of sphere targets in 3D scan point clouds)

Zur Registrierung bzw. Geo-Referenzierung lokaler 3D-Punktwolken aus Laserscans werden u.a. geometrische Zielzeichen genutzt. Sie bilden sogenannte Passpunkte, welche künstlich identische Punkte in mehreren Aufnahmen realisieren. Aus mindestens zwei identischen 3D-Punkten in beiden Aufnahmen, lassen sich im Registrierungsprozess die vier Transformationsparameter (t_x, t_y, t_z, r_z) einer 2.5D-Helmert-Transformation bestimmen, um beide Aufnahmen in ein gemeinsames Koordinatensystem zu überführen.
Eine besondere Art von Zielzeichen bildet das perspektiv invariante Kugel-Zielzeichen. Um die notwendigen Transformationsparameter redundant bestimmen zu können, sind in jeder Scanszene häufig mehr als fünf solcher Passpunkte platziert. Diese Zielzeichen werden in den 3D-Punktwolken bislang oftmals manuell bzw. semi-automatisch selektiert.
Ansätze aus der digitalen Bildverarbeitung und der Bildanalyse zeigen das Potential einer automatisierten Detektion, sodass die Effizienz der Registrierung signifikant gesteigert werden kann [1]. Die Anwendung in realistischen Szenerien führt auf die Analyse und Verarbeitung von Punktwolken aus mehreren Millionen Punkten pro 360°-Rundum-Scan, sodass eine hohe numerische Komplexität der Algorithmen entsteht. Eine effiziente Implementierung mit Methoden des Wissenschaftlichen Rechnens ist daher notwendig.
Innerhalb der Masterarbeit soll aufbauend auf [1] eine effiziente Methode zur Detektion von Kugel-Zielzeichen in 3D-Scan-Punktwolken ressourcenschonend und laufzeitoptimiert implementiert werden. In der Implementation soll ein spezielles Augenmerk auf die Robustheit des Verfahrens gegenüber unterschiedlichen Störeinflüssen sowie die Genauigkeit der Passpunkt-Koordinaten und der daraus abgeleiteten Orientierung gelegt werden. Dies umfasst unter anderem die folgenden Aspekte:

- Sichtung, Validierung und Vergleich alternativer Ansätze aus der Literatur
- Vergleich verschiedener Algorithmen aus der Bildverarbeitung zur Lösung von Teilaufgaben
- Einsatz und Bewertung verschiedener robuster Schätzverfahren zur effizienten Schätzung des Kugelmittelpunktes aus 3D-Koordinaten
- Effiziente und robuste Implementierung geeigneter Methoden und Datenstrukturen in C++
- Parallelisierung von geeigneten Teilproblemen zur Laufzeitoptimierung und Effizienzsteigerung

Diese Untersuchungen und Entwicklungen führen letztendlich auf zwei Ziele: Einerseits soll eine methodisch verbesserte Möglichkeit von automatischen Registrierungen bei Nutzung von Kugel-Zielzeichen realisiert werden. Anderseits sollen die Aspekte der Robustheit, Effizienz und Laufzeitoptimierung den Einsatz in der geodätischen Praxis weiter vorantreiben.

Vorarbeiten

[1] Schreinemachers, M. (2020). Automatisierte Erkennung von Kugel-Zielzeichen in 3D-Punktwolken und deren Registrierung bei vorhandenen Näherungskoordinaten, Seminararbeit im Rahmen des Moduls Ausgewählte Kapitel der Numerik, Geowissenschaftliches Rechnen, 3/2020, unveröffentlicht.

Aufgabensteller: Prof. Dr. techn. Wolf-Dieter Schuh
Betreuer: Dr.-Ing. Jan Martin Brockmann

Bonn, den 24. April 2020

Wolf-Dieter Schuh

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht.

Bonn, 15. Oktober 2020

Matthias Schreinemachers

Zusammenfassung

Im Rahmen dieser Masterarbeit wird eine effiziente und performante Implementation in C++ zur zielzeichen-basierten Registrierung von 3D-Scanpunktwolken mit Kugel-Zielzeichen entwickelt.
In der praktischen Ingenieur-Geodäsie können größere Objekte oftmals nicht durch Laserscanning von einem einzelnen Standpunkt abgebildet werden. Durch das Vermessen von mehreren Standpunkten, welche um das Objekt verteilt sind, gelingt die vollständige Aufnahme des Objektes. Die lokalen 3D-Punktwolken der einzelnen Standpunkte müssen anschließend im Rahmen einer Registrierung in ein gemeinsames Koordinatensystem überführt werden. Eine Methode stellt die zielzeichen-basierte Registrierung der einzelnen Standpunkte dar. Hierzu werden beispielsweise Kugel-Zielzeichen in den Umgebungen der lokalen Standpunkte platziert. Diese dienen als Passpunkte in einer Transformation zwischen identischen Referenzpunkten eines übergeordneten Koordinatensystems.
Die in mehreren realen Projekten, aus denen die Testdatensätze dieser Arbeit vorliegen, genutzten Kugel-Zielzeichen werden bisher halbautomatisch in Standardsoftwarepaketen verarbeitet. Die Registrierung bedarf der manuellen Suche aller Zielzeichen, der Segmentierung von Punkten auf den Kugel-Zielzeichen durch visuelle Betrachtung eines Nutzers sowie der Schätzung der Kugelmittelpunkte.

Die üblicherweise sehr großen Datenmengen der lokalen Punktwolken von über 40×10^6 Punkten ($P_i = [x_i, y_i, z_i, int_i]$) stellen im Rahmen der automatisierten Registrierung eine Herausforderung dar. Durch die Abwicklung der Punktwolke und die Darstellung in Intensitäts- und Distanzbildern und die damit verbundene Dimensionsreduktion der Daten bei Erhalt aller Informationen konnten in dieser Arbeit effiziente Algorithmen zur Zielzeichensuche in Punktwolken entwickelt und realisiert werden. Da die Kugeln in Bildern als Kreise erscheinen, vereinfacht sich die Suche der Zielzeichen-Kandidaten in den Bildern und konnte mittels eines Standardverfahrens der Computer Vision, der Circle Hough Transformation, gelöst werden. Schwachstellen des Verfahrens, wie das Detektieren vieler false-positive Kandidaten - Positionen im Bild, an denen sich keine Kugel-Zielzeichen befinden - sowie das aufwändige Validieren dieser falschen Kandidaten führten zu der Entwicklung eines alternativen Ansatzes. Dieser nutzt die bekannten geometrischen Soll-Eigenschaften der Kugel-Zielzeichen und die Distanz- sowie Intensitätsinformationen aus den abgewickelten Bildern.
Nach der erfolgreichen Validierung aller Zielzeichen-Kandidaten in den verschiedenen Bildern erfolgt das Segmentieren zielzeichen-spezifischer Subpunktwolken. In diesen werden 3D-Punkte, welche auf der Oberfläche des jeweiligen Zielzeichens liegen, mit einer robusten RANSAC Anwendung klassifiziert und Näherungsparameter der Kugel-Mittelpunkte errechnet, die zeitgleich die Koordinaten der Passpunkte realisieren. In den Bildern bisher nicht erkannte false-positiv Kandidaten werden ebenfalls in der RANSAC-Berechnung aufgedeckt. Nach Reduzierung der Anzahl klassifizierter Kugel-Punkte durch zufälliges Ausdünnen erfolgt die bestmögliche Schätzung der Koordinaten des Kugel-Mittelpunktes im Gauß-Helmert-Modell. Abschließend erfolgt die Berechnung der Transformationsparameter für die Transformation in ein übergeordnetes Koordinatensystem mit identischen Passpunkten. Ggf. in den Bildern nicht erkannte Kugel-Zielzeichen werden durch Rücktransformation der globalen Passpunkte in das lokale Koordinatensystem mit in die Auswertung integriert. Nach Berechnung der endgültigen Transformationsparameter erfolgt die Registrierung der lokalen Punktwolke in das globale Koordinatensystem.

Das Generieren der Bilder, das Suchen der Kugel-Kandidaten in Bildern, die teilweise in mehrere Distanzklassen aufgeteilt werden, erfolgt parallelisiert. Das Verarbeiten mehrerer Punktwolken eines Projektes ist ebenfalls parallelisiert implementiert und lässt für größere Projekte eine effiziente Umsetzung der gesamten Registrierung zu. Die entwickelte Software stellt eine performante und ganzheitliche Implementation einer vollautomatischen Registrierung mittels Kugel-Zielzeichen dar.

Abstract

In this master thesis an efficient and performant implementation for the target-based registration of 3D scan pointclouds with sphere targets is developed in C++.
In geodetic-engineering practice larger objects can often not be captured by the laserscanner from a single survey point. By measuring several positions distributed around the object, the complete recording of the object can be achieved. The local 3D pointclouds of the individual positions must then be transformed into a common reference system within the scope of the so called registration process. One approach is the target-based registration of the individual pointclouds. For this purpose, for example sphere targets are mounted in the surroundings of the local points of view. These serve as control points in a transformation between identical reference points of a global coordinate system.
The sphere targets used in several real projects which serve as test data sets in this thesis have so far been processed semi-automatically in standard software packages so far. The registration requires the manual search of all target marks, the segmentation of points on the sphere targets by visual observation of an operator as well as the estimation of the sphere centre points.

The usually very large data volumes of local pointclouds of more than $40{\times}10^6$ points ($P_i = [x_i, y_i, z_i, int_i]$) represent a challenge within the scope of an automated registration. By processing the pointcloud and reprocessing them in intensity and distance images, the associated dimension is reduced, with all information recoverable. They form the basis for the efficient algorithms for the target detection in the pointclouds developed in this thesis. Since the spheres in images appear as circles, the detection for sphere target candidates in the images is simplified and could be solved by a standard procedure from computer vision, the Circle Hough Transformation. Application of this firs approach resulted in the detection of many false-positive candidates - locations in the image where no sphere target is present. Intense validation processes are necessary to identify these false candidates. This led to the development of an alternative algorithm. This utilizes the known geometric properties of the sphere targets in addition to the distance and intensity information from the images.
After the successful validation of all target candidates in the different images, the segmentation of target-specific subpointclouds is performed. In these subpointclouds the 3D points, which are located on the surface of the respective target, are classified with a robust RANSAC application. This yields approximative parameters of the sphere centres as well. False-positive candidates, not yet recognised in the images, are also identified by the RANSAC method. After reducing the number of classified sphere points by random thinning, the best estimates of the coordinates of the sphere centre are derived by a Gauss-Helmert model. Finally, the estimation of the five parameters of the transformation into a global coordinate system with identical controlpoints is carried out. Sphere targets that may not be detected in the images are integrated into the evaluation by re-transforming the global controlpoints into the local coordinate system. After the final transformation parameters are determined, the local pointcloud is registered in the global coordinate system.

The generation of the images, the detection for sphere candidates in images, which are partly divided into several distance classes, is done in parallel processing. The analysis of several pointclouds of a project is also implemented in parallel and represents an efficient approach of the entire registration for larger surveying projects. The developed software represents a performant and holistic implementation of a fully automatic registration by means of geometrical sphere targets.

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Vorarbeiten . . . 2
1.2 Bestehende Ansätze & Stand der Auswertetechnik . . . 2
1.3 Spezifizierung der Aufgabenstellung und Ziel der Arbeit . . . 3
1.4 Gliederung der Arbeit . . . 4

2 Messtechnische Grundlagen **5**
2.1 Messprinzip & Zusammenhänge . . . 5
2.2 Instrumente zur Punktwolkenerfassung . . . 7
2.3 Kugel-Zielzeichen als Messobjekt . . . 8
2.4 Restriktionen an die Vermessung & Datengrundlage . . . 13

3 Überblick über den Programmablauf **17**

4 Datenfluss und Vorprozessierung **19**
4.1 Datenformat der Punktwolken . . . 19
4.2 Dimensionsreduktion durch Bildabwicklung Punktwolke . . . 21
4.3 Punktwolkenklasse in C++ . . . 24

5 Detektion von Zielzeichenkandidaten **27**
5.1 Circle-Hough-Transformation zur Kreissuche . . . 27
5.2 Kandidatensuche mit neuem Ansatz im Distanzbild . . . 32

6 Kandidaten-Validierung als Zielzeichen & Näherungswerte **39**
6.1 Validierung der Zielzeichen-Kandidaten im Bild . . . 39
6.2 Auswahl einer Subpunktwolke . . . 40
6.3 RANSAC - Näherungswerte der Kugelmittelpunkte . . . 41

7 Schätzung der Zielzeichenkoordinate **45**
7.1 Gauß Helmert Modell der Kugel mit freiem Radius . . . 45
7.2 Gauß Helmert Modell der Kugel mit fixiertem Radius . . . 46
7.3 Algorithmisches Vorgehen & Effizienzsteigerung . . . 47
7.4 Evaluierung der Methoden zur Schätzung der Kugel-Parameter . . . 48

8 Die Registrierung **49**
8.1 2.5 dimensionale Helmert-Transformation . . . 49
8.2 Identifizierung von globalen Passpunkten . . . 50
8.3 Rücktransformation weiterer Kandidaten und Registrierung . . . 52

9 Evaluierung der Zielzeichendetektion an ausgewählten Projekten **53**

10 Fazit & Ausblick **59**

A Zusammenfassung in englischer Sprache **61**

B Wissenschaftliches Poster **67**

Kapitel 1

Einleitung

In den letzten Jahren hat der Einsatz moderner und flächenhaft messender Instrumente, wie z.B. Laserscannern, deutlich zugenommen. Für verschiedenste Aufgaben in der Ingenieur- und Industrievermessung werden punkthafte Objekterfassungen mittels Tachymeter durch moderne Punktwolken erzeugende Laserscanner abgelöst [vgl. Kuhlmann und Holst 2017, 167f.]. Zur Abbildung größerer Objekte, welche nicht durch die Vermessung von einem Standpunkt in Gänze abgebildet werden können, müssen mehrere Scans miteinander verknüpft werden. Ziel ist es alle einzelnen Ansichten und Teilaufnahmen des Objektes in einem gemeinsamen Koordinatensystem bereitzustellen um aus der resultierenden Gesamtpunktwolke Schnitte, Ansichten und ganze 3D-Modelle generieren zu können [z.B. Jahn, Kersten und Kinzel 2004]. Im Bauwesen und in der Architektur erfreuen sich die Ergebnisse des Laserscanning in Form von 3D-Bestandsmodellen als Teil des BIM (Building Information Modeling) immer größer werdender Beliebtheit [vgl. Ehm und Hesse 2014]. Hierzu werden die Scans, welche alle in einem eigenen lokalen, kartesischen Koordinatensystem liegen, dessen Ursprung im jeweiligen Scanner liegt, miteinander verknüpft. Dieser Vorgang wird Registrierung genannt [vgl. Joeckel, Stober und Huep 2008, S. 319].
Ein mögliches Verfahren der Registrierung ist es alle in den jeweiligen lokalen Scans vorhandenen Zielmarken als signalisierende Passpunkte in einem übergeordneten zusammenhängenden Koordinatensystem vorzuhalten. So werden alle lokalen Systeme auf der Grundlage identischer Passpunkte jeweils in das übergeordnete Koordinatensystem transformiert. Überlappende Bereiche in den Punktwolken sowie doppelt aufgenommene Zielzeichen sind somit nicht nötig.
Notwendige Passpunkte können durch verschiedenste Arten von Zielzeichen materialisiert werden. Diese sind in der jeweiligen Scannerumgebung verteilt und bilden durch ein lokales 3D-Netz (Niemeier 2008, 331 ff.) die Grundlage der anschließenden zielzeichenbasierten Registrierung [vgl. Wujanz u. a. 2016].
Diese Zielzeichen dienen in den Punktwolken als gut detektierbare Referenzpunkte. Eine besondere Art häufig verwendeter Zielzeichen bildet das perspektiv invariante Kugel-Zielzeichen. Um die notwendigen Transformationsparameter zuverlässig und redundant bestimmen zu können, sind in jeder Scanszene häufig mehr als vier solcher Passpunkte platziert.
Eine Herausforderung stellt die Detektion der Referenzpunkte in der aufgenommenen Punktwolke mit häufig sehr großen Datenmengen dar. So werden die Zielzeichen in den 3D-Punktwolken bislang oftmals manuell bzw. semi-automatisch selektiert.
Ziel dieser Arbeit ist es Kugel-Zielzeichen effizient und automatisiert zu detektieren und mit geeigneten Algorithmen auf die zur Registrierung notwendigen Passpunktkoordinaten zu reduzieren.
In den nächsten Abschnitten soll neben einer Spezifizierung der Aufgabenstellung auf bereits geleistete Vorarbeiten und bestehende Ansätze eingegangen werden. Abschließend wird in diesem Kapitel die Gliederung der Arbeit dargestellt.

1.1 Vorarbeiten

Im Rahmen einer Seminararbeit, welche dieser Masterarbeit vorangegangen ist, konnte gezeigt werden, dass ein automatisiertes Erkennen von kugelförmigen Zielzeichen in 3D-Punktwolken möglich ist. Die Seminararbeit kann als Machbarkeitsanalyse verstanden werden und konzentrierte sich auf Methoden der Bildverarbeitung. Die Berechnung von sog. modifizierten Distanzbildern (vgl. Abschnitt 4.2), welche eine Abwicklung der Punktwolke unter Berücksichtigung der Intensitätswerte und Entfernungen zu den einzelnen 3D-Punkten, bzw. Pixeln im Bild darstellt, dient als Grundlage der Kandidatensuche in einem auf zwei Dimensionen reduzierten Bild, in welchem die gesuchten Kugeln als Kreise erscheinen. Die Detektion von Kugel-Zielzeichen reduziert sich damit auf die Detektion von Kreisen.

In Schreinemachers 2020a werden Kreise im generierten Bild der Punktwolke gesucht und das detektierte Mittelpunktpixel zurück in die 3D-Punktwolke transformiert. Ohne weiter auf die Abläufe und Details dieser Methode einzugehen (diese werden ausführlich in Kapitel 5 besprochen) sei hier erwähnt, dass häufig sog. false-positive Kugelkandidaten in dem Bild extrahiert werden, was eine vergleichsweise rechenaufwändige Validierung aller Kandidaten notwendig macht.
Neben der Kugelsuche im Bild konnten in einem weiteren Schritt mögliche nicht identifizierte Kugeln mittels Näherungswerten aus einem übergeordneten Koordinatensystem ermittelt werden.

Schlussendlich konnte wie eingangs erwähnt eine automatisierte Extraktion von Passpunktkoordinaten aus 3D-Punktwolken realisiert werden. Wesentliche Schwachstellen sind die mangelnde Effizienz einzelner Teilmethoden im gesamten Workflow sowie eine nicht immer gegebene Robustheit gegenüber fälschlicherweise erkannten Kugelkandidaten. Ebenso mussten viele Schwellwerte für verschiedene Funktionen und Validierungsansätze angenommen werden, die teilweise ausschließlich durch das Testen weniger Datensätze entstanden sind. Die implementierte Variante ist somit unter realen, vielseitigen Bedingungen schlecht einsetzbar. Die angewandten Methoden, Ansätze sowie die Ergebnisse dieser Vorarbeit können in [Schreinemachers 2020a] nachgelesen werden. Alle Implementierungen dort wurden in GNU Octave (GNU Octave 2020b), einer Matlab nahen Skriptsprache realisiert.

1.2 Bestehende Ansätze & Stand der Auswertetechnik

In der vermessungstechnischen Praxis werden in vielen Fällen die für die Registrierung nötigen Passpunkte halbautomatisch aus der 3D-Punktwolke extrahiert [vgl. Kersten u. a. 2008]. Der allgemeine Arbeitsablauf umfasst das Einlesen der Punktwolken in meist Instrumenten(-Hersteller) spezifischer Software. Hier werden anschließend Subpunktwolken mit Werkzeugen zur Segmentierung und Selektion gebildet, welche möglichst ausschließlich Punkte auf dem Kugel-Zielzeichen beinhalten. Die Segmentierung erfolgt rein visuell und ist offensichtlich operateurabhängig. Ggf. fälschlich selektierte Punkte wirken in der Analyse als Ausreißer und können je nach Ansatz und Robustheit der Parameterschätzung das Ergebnis signifikant verfälschen. Nach der Segmentierung werden dann Kugel-Objekte durch die ausgewählten Punkte geschätzt. Die Identifizierung identischer Punkte aus einem übergeordneten System kann teilweise direkt in der Software vorgenommen werden. Abschließend wird dann Standpunkt für Standpunkt registriert [z.B. Reidl und Ohlmann-Lauber 2014].
Es ist leicht nachvollziehbar, dass der Arbeitsaufwand mit der Anzahl eingesetzter Kugel-Zielzeichen pro Standpunkt und der Anzahl der Standpunkte im gesamten Messprojekt skaliert. Die Automatisierung der Kandidatensuche in 3D-Punktwolken führt somit unmittelbar zu einer signifikanten Reduzierung des Arbeitsaufwandes in der Nachbearbeitung des Laserscannings. Bestehende Arbeiten befassen sich oftmals mit verschiedenen Methoden der Kugelschätzung zur Bestimmung der Passpunktkoordinate. So werden beispielsweise Ansätze der Minimierung als Regression (*algebraic*

fitting) beschrieben sowie die geometrische Lösung als Minimierung der Verbesserungsquadrate zu vorläufigen Kugeloberflächen (*geometric* fitting) [vgl. Witzgall, Cheok und Kearsley 2006]. Eine möglichst optimale Auswahl der für die Kugelschätzung verwendbaren Punkte im Sinne der Ausreißerminimierung wurde mit verschiedensten Ansätzen bereits in [Rachakonda u. a. 2017] untersucht und Ergebnisse und Erkenntnisse dieser Arbeit sollen später auch in der eigenen Implementierung genutzt werden.
In [Franaszek, Cheok und Witzgall 2009] konnte gezeigt werden, dass eine automatisierte Registrierung aus abgeleiteten Distanzbildern zwischen zwei Standpunkten bei Aufnahme identischer Kugel-Zielzeichen möglich ist. Die im Rahmen der Untersuchung genutzten Punktwolken mit maximalen Datensatzgrößen von 6.4 Millionen 3D-Punkten im ersten Standpunkt und 3.4 Millionen Punkten im zweiten Standpunkt liegen jedoch deutlich unter den Größen üblicher Punktwolken aus Rundumscans in mittleren Auflösungsstufen. Die Datensätze, welche dieser Arbeit vorliegen, wurden im Rahmen mehrerer Bestandsdokumentationen mit einer einheitlichen Auflösung von 6.3 mm @10 m (P16) bzw. 6.0 mm @10 m (RTC 360) gescannt. In Tabelle 2.1 werden die zugehörigen maximalen Punktwolkengrößen von rund 40 Millionen 3D-Punkten aufgeführt. Die bestehenden Strategien aus [Franaszek, Cheok und Witzgall 2009] benötigen für die untersuchten Punktwolkengrößen von 6.4 und 3.4 Millionen 3D-Punkten weniger als 30 Sekunden, was ohne Informationen zu eingesetzten Rechenleistungen lediglich einen groben Vergleich zu später dargestellten Prozessierungzeiten bietet, jedoch zeigt, dass bereits kleinere Punktwolken eine gewisse Prozessierungszeit in Anspruch nehmen.
Im Rahmen weiterer Untersuchungen zur vollautomatischen Registrierung mit Kugel-Zielzeichen wurden in [Wang u. a. 2014] neue Ansätze zur Detektion von Kugelkandidaten in 3D-Punktwolken vorgestellt. So werden kreisförmige Objekte anhand von zuvor errechneten Randpixeln in Bildern berechnet. Mittels RANSAC werden anschließend Kugelkandidaten an den zuvor gefundenen Kandidatenorten detektiert. Auch hier wurden Angaben zur Geschwindigkeit der Prozessierung gemacht, welche sich auf zwei Punktwolken mit jeweils vier Kugel-Zielzeichen und mehr als 10 Millionen 3D-Punkten beziehen. Mit einer maximalen Abweichung von <2 mm zwischen identischen Zielzeichen konnte die Registrierung binnen rund 1.5 Minuten (Testsystem: 64-bit Windows 7 Professional PC mit Intel Core i7 2.4 GHz CPU und 16 GB RAM) berechnet werden. Auch hier ist zu erkennen, dass die eingesetzten Methoden rechenintensiv und auf großen Datensätzen demnach zeitintensiv sind. Ebenfalls wurden auch hier lediglich Ausschnitte einer Umgebung und somit ein Bruchteil der in dieser Arbeit anfallenden Punktwolkengrößen bearbeitet. Die idealisierten Laborbedingungen und einzelnen Projekte aus den beschriebenen Arbeiten decken die breite Anwendung nicht ab. Abschließend kann festgehalten werden, dass automatisierte Ansätze stark von der Umgebung des Laserscanners abhängen und bei sehr objektreichen Aufnahmesituationen stark in ihrer Laufzeit skalieren.

1.3 Spezifizierung der Aufgabenstellung und Ziel der Arbeit

Wie in Kapitel 1 beschrieben, befasst sich diese Arbeit mit der zielzeichenbasierten Registrierung von 3D-Punktwolken im Allgemeinen. Die speziellen Aufgaben sind das automatisierte Lokalisieren von Kugeln in der Punktwolke, die Ableitung von robusten Passpunktkoordinaten und die abschließende Registrierung als Transformation der gesamten 3D-Punktwolken mit den geschätzten Transformationsparametern. Diese Schritte sollen mit Blick auf die großen Datenmengen einzelner Punktwolken möglichst effizient implementiert werden und automatisiert einsetzbar sein. Hierzu wird in dieser Arbeit auf die im Rahmen der Anwendungsprogrammierung häufig eingesetzte kompilierte Programmiersprache C++ zurückgegriffen. Weiter bietet C++ neben der maschinennahen und somit effizienten Programmierung die Möglichkeit der Integration bereits implementierter Programmbibliotheken, wie OpenCV. In Abschnitt 5.1 wird unter anderem näher auf genutzte Teile der verwendeten freien Programmbibliothek OpenCV eingegangen. Eine wesentliche Verbesserung gegenüber bereits bestehenden Ansätzen kann die Prozessierungsgeschwindigkeit großer Punkt-

wolken sein. Wie in Abschnitt 1.2 gezeigt benötigen vollautomatische Ansätze oftmals bereits für kleinere Punktwolken mehrere Sekunden bis wenige Minuten. Unter Berücksichtigung der in der vermessungstechnischen Praxis anfallenden Anzahl an Standpunkten pro Objekt/Projekt stellt die Prozessierungszeit demnach ein wesentliches Kriterium der tatsächlichen Nutzbarkeit solcher vollautomatischen Registrierungen dar.

1.4 Gliederung der Arbeit

Zu Beginn werden in Kapitel 2 die messtechnischen Grundlagen erläutert. Hierzu wird das Messprinzip eines Panorama-Scanners, jener Baugruppe, welche in der Vermessung der vorliegenden Punktwolken eingesetzt wurde, dargestellt und beschrieben. Neben der Funktionsweise der Scanner sind physikalische Zusammenhänge zwischen ausgesandter Intensität, der Messkonfiguration, wie dem Einfallswinkel am Messobjekt und der empfangenen Intensität beschrieben. Diese Größen und Zusammenhänge dienen in späteren Kapiteln und Abschnitten zum Verständnis der angewandten Segmentierungen von Subpunktwolken zur anschließenden Schätzung von Passpunktkoordinaten. Weiter werden neben den Kugel-Zielzeichen die eingesetzten Instrumente mit ihren, für die Arbeit wesentlichen, Merkmalen vorgestellt und beschrieben. Abschließend sind Besonderheiten und Restriktionen an die Durchführung der Vermessung am Objekt aufgeführt, die als Annahme in die entwickelten Algorithmen eingehen, sowie ein Überblick über die Datengrundlage gegeben.
Ab Kapitel 3 gliedert sich die Arbeit nach dem implementierten Workflow. Hierzu ist neben einem kurzen textlichen Überblick ebenfalls ein Programmablaufplan dargestellt. Die Struktur der folgenden Kapitel wird somit später noch einmal ausführlich dargestellt.

Kapitel 2

Messtechnische Grundlagen

Wie in den Abschnitten 1.3 und 1.4 bereits dargestellt, soll der Fokus auf der Prozessierung der aufgenommenen 3D-Punktwolken liegen. Ohne das Beschreiben wesentlicher Grundlagen zum Messprinzip der eingesetzten Instrumente sowie zu den verwendeten Zielzeichen würden ggf. in späteren Kapiteln Fragen aufkommen oder Zusammenhänge nicht nachvollziehbar erscheinen. Aus diesen Gründen sollen die folgenden Abschnitte kurz die wesentlichen und für die Auswertung der Daten notwendigen Kenntnisse vermitteln.

2.1 Messprinzip & Zusammenhänge zwischen Einfallswinkel, Intensitäten und Genauigkeiten

Die in Abschnitt 2.2 aufgeführten Laserscanner, welche alle der Arbeit vorliegenden Punktwolken aufgenommen haben, sind sogenannte Panorama-Scanner. Diese nehmen anders als Kamera-Scanner oder Hybrid-Scanner ein größtmögliches Umfeld auf [vgl. Möser u. a. 2012, S. 286]. Lediglich ein kleiner Ausschnitt unter dem Scanner kann aus Gründen der Abschattung durch den Scanner und seinen Aufbau, wie in Abbildung 2.1a weiß gekennzeichnet, nicht abgedeckt werden. Realisiert wird die räumliche Vermessung durch das in Abbildung 2.1b dargestellte Strahlablenkungsprinzip. Hier rotiert der Scanner selbst um seine vertikale Drehachse (Stehachse) und ein Spiegel zur vertikalen Strahlablenkung auf das Messobjekt um die horizontale Drehachse. Neben dem Laser emittierenden Sender verfügt der Scanner über eine Empfangseinheit, welche den reflektierten Anteil des ausgesandten Laserstrahls detektiert [vgl. Joeckel, Stober und Huep 2008, S. 330].
Neben dem Messprinzip sind die Zusammenhänge zwischen der Intensität des reflektierten Laserstrahls, dem Einfallswinkel des Lasers am Messobjekt und der Genauigkeit eines 3D-Punktes wesentlich.
Die reflektierte Intensität P_r nimmt nach der Laser Radar Gleichung

$$P_r = \frac{P_t D_r^2 \rho}{4R^2} \eta_{\text{sys}} \eta_{\text{atm}} \cos\alpha \tag{2.1.1}$$

in der P_t die gesendete Leistung, D_r den Durchmesser des Empfängers, r die Distanz, η_{sys} einen systemabhängigen Parameter, η_{atm} den Atmosphärenparameter, ρ die Oberflächenreflektivität und α den Einfallswinkel beschreiben, mit dem Kosinus des Einfallswinkels ab [vgl. Höfle und Pfeifer 2007; Schmitz u. a. 2019; Jelalian 1992, 419 f.].
Frühere Untersuchungen zeigen den beschriebenen Verlauf in Abhängigkeit des Einfallswinkels und der Objektfarbe, welche zu einer Skalierung der absoluten Intensität führt. So ist in Abbildung 2.2a der in Gleichung (2.1.1) enthaltene Kosinus-Anteil des Einfallswinkels deutlich erkennbar. Gerade unter Berücksichtigung der eingesetzten sphärischen Kugel-Zielzeichen, welche in Abschnitt 2.2 näher beschrieben werden, ist die Abhängigkeit der Intensität vom Einfallwinkel am Messobjekt wesentlich. Punkte am Rand der sichtbaren Halbkugel weisen offensichtlich einen größeren

(a) (b)

Abbildung 2.1: (a) räumliche Abdeckung eines Panorama-Scanners, (b) Prinzip der Strahlablenkung eines Panorama-Scanners [angelehnt an Möser u. a. 2012, S. 286]

(a) (b)

Abbildung 2.2: (a) Abnahme der Intensität mit steigendem Einfallswinkel am Objekt für unterschiedliche Farben (weiß, hellgrau, grau, dunkelgrau, schwarz); (b) Intensitätsabhängige Entfernungsmessgenauigkeit aus Streckenresiduen zu einer Referenz-Ebene [vgl. Schreinemachers 2020b]

(a) (b)

Abbildung 2.3: Auftretender Kometenscheif am Kugel-Zielzeichen; (a) seitliche Ansicht der Aufnahmesituation; (b) Aufsicht mit deutlich erkennbaren Pixeln aus der beschriebenen Kometenschweif-Problematik

Einfallswinkel auf als jene Punkte in der Mitte. In zahlreichen, teilweise eigenen (vgl. Abbildung 2.2b), Untersuchungen konnte gezeigt werden, dass die Genauigkeit eines Punktes in Strahlrichtung von seiner Intensität abhängt [vgl. z.B. Wujanz u. a. 2017; Schmitz u. a. 2019]. Es kann demnach allein aus theoretischen Überlegungen festgehalten werden, dass Punkte auf den sphärischen Kugel-Zielzeichen unterschiedlich genau sind. Diese relativen Genauigkeiten können jedoch allein aus der bekannten Geometrie und den messbaren Intensitäten abgeleitet werden. Diese Überlegungen erscheinen wichtig, da Randpixel auf der Kugel für die spätere Schätzung der Kugelparameter geometrisch wichtig sind. Ein weiterer Effekt bei großen Einfallswinkeln am Objekt beschreibt der sogenannte Kometenschweif [vgl. Joeckel, Stober und Huep 2008, S 338; Möser u. a. 2012, S. 289]. An Randbereichen eines Objektes (vgl. Abbildung 2.3) bildet der Strahl durch seine entfernungsabhängige Divergenz und daraus resultierende Laserspotgröße Teile des Vordergrundes und Abschnitte im Hintergrund liegender Objekte ab. Am Rand einer Kugel führt das zu nicht eindeutigen Streckeninformationen entlang des Messstrahls (siehe Abbildung 2.3b).

2.2 Instrumente zur Punktwolkenerfassung

Im Rahmen der Arbeit wurde auf bereits durchgeführte Vermessungen zurückgegriffen. Alle Datensätze stammen aus realen Projekten eines Ingenieurbüros und bilden keineswegs Laborbedingungen ab. Demnach konnte die Auswahl der Instrumente zur Aufnahme, sowie die getroffenen Einstellungen an den Geräten, wie auch die Wahl und Platzierung der Zielzeichen nicht beeinflusst werden. Im Folgenden werden die eingesetzten Instrumente kurz vorgestellt. Sofern bekannt, sind die projektbezogenen Einstellungen aufgeführt. Die eingesetzten Laserscanner Leica P16 und Leica RTC360 unterscheiden sich wesentlich durch die im RTC360 eingesetzte Navigationssensorik, sowie die deutlich unterschiedlichen Abmessungen der Instrumente. So ist laut Hersteller eine relative Bewegungsverfolgung der Scannerposition zum jeweils vorherigen Standpunkt durch ein videogestütztes Intertialmesssystem (VIS) möglich. Die methodischen und baulichen Unterschiede der beiden Scanner haben jedoch keine Auswirkung auf die in dieser Arbeit durchgeführten Berechnungen und Auswertevorgänge. Auf weitere Besonderheiten der Scanner soll aus diesem Grund nicht eingegangen werden. Lediglich der größere effektive Messbereich (360° horizontal & 150° vertikal) des RTC360 gegenüber dem der P16 (360° horizontal & 135° vertikal) ist für die spätere Bildgenerierung zu berücksichtigen [vgl. Leica P16 Datenblatt 2015] [vgl. Leica RTC360 Datenblatt 2018]. Beide Laserscanner verfügen, wie für Instrumente dieser Bauart üblich, über Neigungssensoren, welche nach manueller Grobhorizontierung eine eigenständige Feinhorizontierung der Instrumente

(a) (b)

Abbildung 2.4: eingesetzte Instrumente zur Aufnahme der Punktwolken; (a) Leica ScanStation P16 [aus Leica P16 Benutzerhandbuch 2016]; (b) Leica RTC360 [aus Leica RTC360 Benutzerhandbuch 2018]

Auflösung [mm @10 m]		Winkel-inkrement [°]		Zeit für Rundum-Scan [mm:ss]		maximale Punktanzahl für Rundum-Scan	
P16	RTC360	P16	RTC360	P16	RTC360	P16	RTC360
12.5	12.0	0.072	0.069	[00:58]	[00:25]	9.375 x 10^6	10.4 x 10^6
6.3	**6.0**	**0.036**	**0.034**	**[01:49]**	**[00:50]**	**37.5 x** 10^6	**41.7 x** 10^6
3.1	3.0	0.018	0.017	[03:30]	[01:40]	150 x 10^6	166.7 x 10^6
1.6	-	0.009	-	[13:33]	-	600 x 10^6 *	666.7 x 10^6 *

Tabelle 2.1: Auflösungsstufen und benötigte Zeit für einen Rundum-Scan, *lediglich mit Auswahl eines Umgebungsausschnittes sinnvoll, da im Rundumscan sehr große Punktmengen generiert werden [vgl. Leica P16 Benutzerhandbuch 2016, 110 f.][vgl. Leica RTC360 Benutzerhandbuch 2018, S. 38]

rechnerisch ermöglichen. Die lokale Z-Achse des Koordinatensystems jedes Standpunktes richtet sich demnach am Lot aus und ermöglicht später eine Vereinfachung der nötigen Transformationen. Das Messprinzip der eingesetzten Laserscanner wurde bereits in Kapitel 2 beschrieben und in den Abbildungen 2.1a und 2.1b dargestellt. In Tabelle 2.1 ist aufgeführt, welche Auflösungen an den beiden Instrumenten einstellbar sind und welche Datenmengen jeweils erzeugt werden. Außerdem sind die zu erwartenden Messzeiten für alle Auflösungsstufen angegeben. Diese Größen spielen neben den messtechnischen Anforderungen eines Projektes eine wesentliche Rolle in der Planung und späteren Weiterverarbeitung der erzeugten 3D-Punktwolken.

2.3 Kugel-Zielzeichen als Messobjekt

Nachfolgend werden die eingesetzten Kugel-Zielzeichen vorgestellt. Die Kugeln haben einen Durchmesser von 145 mm und eine mittels M8-Gewinde adaptierbare Magnethalterung, welche das Anbringen an metallischen Oberflächen zulässt, wie in Abbildung 2.5a dargestellt. An nicht metallischen Oberflächen können diese ebenfalls montiert werden. Kompatible Schrauben ermöglichen

(a) (b) (c)

Abbildung 2.5: Kugelzielzeichen-Standorte; (a) Kugel-Zielzeichen mit Magnethalterung auf einem Fahrscheinentwerter im Bonner Hbf; (b) Kugel-Zielzeichen mittels Wandbolzen (M8-Innengewinde) an Schachtwand angebracht; (c) Klebemarke mit Punktnummer zur Punktsignalisierung auf metallischem Geländer

(a) (b)

Abbildung 2.6: eingesetzte Kugel-Zielzeichen zur Referenzierung der Punktwolken; (a) Kugel-Zielzeichen-Set mit sechs Kugeln; (b) einzelnes Zielzeichen mit Magnetfuß

beispielsweise das Anbringen an Tunnel-/Kanal- oder Schachtwänden, wie in Abbildung 2.5b beispielhaft dargestellt. In Abbildung 2.6b ist eine Kugel mit angeschraubtem Magnethalter abgebildet. Auffällig ist hier, dass der magnetische Fuß nach der Verschraubung unmittelbar an die Kugeloberfläche grenzt, was eine scharfe Segmentierung in der Punktwolke nötig macht. Eine Verbesserung bzw. Vereinfachung der Segmentierung könnte die Adaption einer Stange ähnlich einer Gewindestange zwischen Kugel und Magnetfuß darstellen. Für die Registrierung bedarf es mehrerer Kugeln in einer Scannerumgebung. In Abbildung 2.6a ist ein typisches Kugel-Zielzeichen-Set, wie es in allen hier analysierten Projekten genutzt wurde, abgebildet.
Alle Kugeln sind aus einem mattweiß beschichtetem Kunststoff gefertigt. Aus verschiedensten Untersuchungen ist bekannt, dass die Genauigkeit reflektorloser Distanzmessungen wesentlich von der Oberflächenstruktur und -rauigkeit, der Farbe des Objekts sowie geometrischen Besonderheiten, wie Ecken, Kanten, Stufen und Wölbungen abhängt [vgl. Möser u. a. 2012, S. 256, 257]. Die Farbe Weiß stellt hier eine möglichst große Menge an zurückgestrahlter Intensität sicher. Die unverspiegelte leicht raue Oberfläche der Kugeln realisiert ein wie in 2.7c dargestelltes diffus-spiegelnd reflektierendes Lasersignal mit hohem Intensitätsniveau. In den Abbildungen 2.8a bis 2.8f sind Ausschnitte aus Punktwolken dargestellt. Die in der Software CloudCompare nach ihren Intensitäten einge-

(a) (b) (c)

Abbildung 2.7: Arten möglicher Reflexionen an verschiedenen Oberflächen [angelehnt an Joeckel, Stober und Huep 2008, S. 335] und [angelehnt an Kern 2003, S. 40]; (a) Totalreflexion an der Objektoberfläche; (b) diffus-reflektierende Oberfläche (c) diffuse Reflexion mit Spiegelung an Objektoberfläche

(a) seitliche Ansicht (b) frontale Ansicht (c) nach Kugel-Fit

(d) seitliche Ansicht (e) frontale Ansicht (f) nach Kugel-Fit

Abbildung 2.8: Messpunkte auf der Kugelzielzeichenoberfläche; erste Zeile: Kugel in 0.65 m Entfernung und 110.948 Punkte; zweite Zeile: Kugel in 8.33 m Entfernung und 517 Punkte

Abbildung 2.9: Schweif am Kugel-Zielzeichen in naher Umgebung zum Laserscanner

Abbildung 2.10: Öffnungswinkel am Kugel-Zielzeichen

färbten Messpunkte auf den Kugeloberflächen vermitteln einen Eindruck der vom Einfallswinkel abhängigen Intensitäten (vgl. Gleichung 2.1.1) sowie den entfernungsabhängigen Punktdichten.

In Abbildung 2.9 ist deutlich zu erkennen, dass einige Datenpunkte als Schweif der Kugeloberfläche auftreten. Diese *Schein-Messungen* repräsentieren nicht vorhandene Objekte und müssen zur Realisierung einer bestmöglichen Schätzung per Segmentierung der Kugel ausgeschlossen werden [vgl. Rachakonda u. a. 2017, S. 5–6].

In weiteren Untersuchungen des Einfallswinkels und dessen Einfluss auf Genauigkeiten der Kugelparameter konnten Optimierungen bei der Selektion von Messpunkten erarbeitet werden [vgl. Rachakonda u. a. 2017, S. 17–21].

So kann gezeigt werden, dass der Ausschluss jener Messpunkte mit schlechten Intensitäten und Einfallswinkeln, sowie ggf. auftretenden Schweif-Effekten mittels definiertem Öffnungswinkel im Kugelzentrum (vgl. Abbildung 2.10) zu einer Verbesserung der geschätzten Kugelparameter führt.

In Abbildung 2.11 ist ein Anstieg des RMS (Root Mean Square) der Verbesserungen ab einem Öffnungswinkel von ca. 125 ° deutlich erkennbar. Wendet man die Segmentierung auf das Kugel-Zielzeichen aus Abbildung 2.8a an, werden jene Punkte verworfen, welche schlechte Intensitäten aufweisen (vgl. Abbildung 2.12). Tatsächlich sind jedoch eben diese Punkte für die Bestimmung der Geometrie der Kugel wichtig, da sie den Rand der Kugel abbilden und wesentlichen Einfluss auf die Schätzergebnisse der vertikalen und horizontalen Position des Zielzeichens aus Sicht des Laserscanners nehmen. Die Segmentierung stellt demnach einen Kompromiss zwischen den einerseits geometrisch wichtigen Randpunkten und den sich aufgrund der beschriebenen Zusammenhänge ergebenen Fehlern und Ungenauigkeiten dar. Dieser Ansatz der Segmentierung soll in später beschriebenen Implementierungen Verwendung finden.

Abbildung 2.11: Rauschen auf mehreren Kugeln unter verschiedenen Öffnungswinkeln; Ergebnisse mehrerer Laserscanner [vgl. Rachakonda u. a. 2017, S. 18].

(a) (b)

Abbildung 2.12: Segmentierung anhand definiertem Öffnungswinkel im Kugelzentrum zum Laserscanner

Abbildung 2.13: Intensitätsbild (invertiert) Scan Bahnübergang I

2.4 Restriktionen an die Vermessung & Datengrundlage

Im Rahmen der Spezifizierung der Aufgabenstellung dieser Arbeit und den in der vermessungstechnischen Praxis üblichen Rahmenbedingungen haben sich folgende Restriktionen an die Vermessung und daraus Annahmen für die hier entwickelten Algorithmen ergeben.
So ist die Wahl der Zielzeichen auf sphärische Kugelzielzeichen mit bekanntem Radius ähnlich der in 2.3 beschriebenen Kugeln begrenzt. Innerhalb eines Projektes weisen diese homogene und starke Reflexionseigenschaften auf. Ebenso haben alle verwendeten Kugel-Zielzeichen bekannte und feste Radien.
Neben dem Laserscanning, welches mit geeigneter Auflösung mit einem Panorama-Scanner als Rundum-Scan durchzuführen ist, sind die genutzten Passpunkte (Standorte der Kugel-Zielzeichen) z.B. tachymetrisch im Rahmen einer Netzmessung in einem übergeordneten Koordinatensystem zu vermessen. Die Signalisierung der Zielzeichenstandorte kann, wenn nicht durch bestehende Schraubgewinde statisch definiert, durch Klebemarken, wie in Abbildung 2.5c realisiert werden.
Der Laserscanner ist jeweils grob zu horizontieren und die Aufnahme der Punktwolke sollte aus in Abschnitt 2.2 erläuterten Gründen mit eingeschaltetem Kompensator erfolgen.
Möglichst viele, mindestens aber drei, der vorhandenen sechs Kugel-Zielzeichen sollten in einem Entfernungsbereich von 1 m bis maximal 25 m um den Scannerstandpunkt plaziert werden. Die Konfiguration der Zielzeichen sollte geodätischen Ansprüchen gleich denen einer Freien Stationierung in einem räumlichen Netz gerecht werden [vgl. Heunecke u. a. 2013, S. 223]. Eine auch teilweise Abschattung der Kugel-Zielzeichen durch im Vordergrund existierende Objekte sollte vermieden werden.
Als Datengrundlage dienen in dieser Arbeit mehrere Punktwolken aus verschiedenen Projekten. Die folgend aufgeführten Projekte stellen reale Aufgabenstellungen der Ingenieurvermessung dar mit teils sehr unterschiedlichen Umgebungsbedingungen, welche keineswegs laborähnlichen Aufnahmesituationen gleichen. Im ersten Projekt wurde ein Bahnübergang (BÜ) von mehreren Standpunkten gescannt. Der Arbeit dienen zwei dieser Punktwolken als Testdatensätze (s. Abb. 2.13). Im zweiten Projekt wurde der Bahnsteig 1 des Bonner Hauptbahnhofes (Hbf) während einer Sanierungsphase gescannt. Aus diesem Projekt liegt eine Punktwolke als Testdatensatz vor (s. Abb. 2.14). In einem weiteren Projekt wurden Durchlässe und Kanalabschnitte eines Kanalsystems zur Bestandsdokumentation vermessen. Hier liegen insgesamt 30 Punktwolken zur Auswertung vor (ähnlich Abb. 2.16). Da zur Bestandsdokumentation hochauflösende Detailaufnahmen nicht nötig sind, wurde in der Planung der Scans eine einheitliche Auflösung von 6.3 mm@10 m gewählt. Durch diese Voreinstellung wird die Messzeit pro Standpunkt gegenüber höheren Auflösungen, wie Tabelle 2.1

Scan Nr.	Projekt	Punktanzahl	Kugeln
1	A	20 176 029	6
2	A	18 003 764	6
3	B	34 948 134	5
4-33	C	ca. 3 850 000	6

Tabelle 2.2: Standpunktinformationen aller in dieser Arbeit genutzten Scans

Abbildung 2.14: Intensitätsbild (invertiert) Bahnsteig 1 Bonner Hauptbahnhof

entnommen werden kann, enorm reduziert.
Im Folgenden sollen die Besonderheiten der einzelnen Aufnahmesituationen und der resultierenden Punktwolken beschrieben werden. Außerdem sind in Tabelle 2.2 neben der Punktanzahl aller beschriebenen Punktwolken auch die jeweilige Anzahl installierter Kugeln aufgeführt. Die Scans aus Abb. 2.13 wurden im Freien aufgenommen. Gemessen wurden maximale Distanzen von 80 m. Auffällig in der Betrachtung der Intensitätsbilder (vgl. Abschnitt 4.2) ist außerdem die nahezu vollständige Himmelsfreiheit. Dies führt dazu, dass große Teile des generierten Intensitätsbildes leer (weiß) bleiben. Die in Abbildung 2.14 dargestellte Aufnahmesituation am Bonner Hbf unterscheidet sich hinsichtlich der nicht vorhandenen Himmelsfreiheit und hohen Objektdichte deutlich von den beiden zuvor beschrieben Punktwolken. Hierzu ist in Abbildung 2.15 ein Ausschnitt aus der Punktwolke dargestellt. Zu erkennen ist, dass die beiden markierten Kugel-Zielzeichen im Hintergrund eines Bauzaunes und wie beschrieben in eine sehr objektreiche Umgebung platziert sind. Folglich nimmt trotz gleicher Einstellung der Auflösung die Punktanzahl im Vergleich zu rund 20 Millionen und 18 Millionen der Scans des Bahnübergangs mit knapp 35 Millionen Punkten am Bahnhof deutlich zu. Weiter sind durch viele metallische und spiegelnde Oberflächen im nahen Umfeld des Scanners viele Ausschnitte mit hohen Intensitäten ähnlich denen auf den gesuchten Kugel-Zielzeichen vorhanden. In allen Scans des Kanalsystems gibt es, mit Ausnahme der Wasseroberflächen, keine Lücken in der Punktwolke, was zu den größten Punktwolken dieser Arbeit führt. In Abbildung 2.16 ist deutlich zu erkennen, dass bei großen Einfallswinkeln an den Randflächen der Kanalwände, wie in Abschnitt 2.1 beschrieben, nur noch wenig Intensität zum Instrument reflektiert wird. Die teilweise orthogonalen Auftreffwinkel an den Kugeln sowie deren in Abschnitt 2.3 aufgeführten Eigenschaften führen zu den guten Intensitätswerten, welche als dunkle Flächen in Abbildung 2.16 zu erkennen sind. Zu den Punktwolken in diesem Projekt liegen zusätzlich HDR-Bilder (High Dynamic Range Bilder oder Hochkontrastbilder), welche mit dem eingebauten Kamerasystem des

Abbildung 2.15: Punktwolkenausschnitt in CloudCompare, zwei Kugel-Zielzeichen an Stahlträgern hinter Bauzäunen

Abbildung 2.16: Intensitätsbild (invertiert) Durchlass Kanalsystem I

Abbildung 2.17: HDR-Bild eines Standpunktes im Kanalsystem, Spiegelung zweier Kugel-Zielzeichen an der Wasseroberfläche

RTC360 zu jedem Standpunkt aufgenommen wurden, vor. Auf diesen ist eine Besonderheit der Scans mit Wasseroberflächen erkennbar. In Abbildung 2.17 dargestellt spiegeln sich beide Kugel-Zielzeichen auf der Wasseroberfläche. Tatsächlich ergeben sich somit auch 3D-Punkte unterhalb der Oberfläche, was teilweise zur Detektion weiterer Kugelkandidaten führt (vgl. Abschnitt 42). In den dargestellten Abwicklungen der Punktwolken aus den Abbildungen 2.13 und 2.14 befinden sich jeweils fünf Kugel-Zielzeichen, in Abbildung 2.16 insgesamt sechs, was deutlich macht, dass ein einfaches visuelles Erkennen aller Zielzeichen nicht gelingt.

Kapitel 3

Überblick über den Programmablauf

In diesem Kapitel soll der entwickelte Workflow und das realisierende Programm in seiner Gesamtheit in einem Überblick dargestellt werden. Zur Übersicht der einzelnen Teilschritte dient der in Abbildung 3.1 dargestellte Programmablaufplan. Neben den Schritten der Verarbeitung werden in diesem Plan jeweils die Kapitel und Abschnitte aufgeführt, in denen die aufgeführten Teilschritte ausführlich beschrieben und diskutiert werden. Als Input dienen der implementierten Software lokale Punktwolken im binären `sbf`-Format (vgl. 4.1) sowie eine `ASCII`-Datei (American Standard Code for Information Interchange) mit den globalen Passpunkten des Projektes, auf welche die Registrierung durchgeführt werden soll. Die genutzten Speicherformate sowie das Vorprozessieren der Punktwolken sind in Kapitel 4 beschrieben. Aus der Vorprozessierung ergeben sich Bilder als Abwicklungen der Punktwolken. In diesen findet die Suche von potenziellen Kugel-Zielzeichen statt. Verschiedene Arten von Repräsentationen der Punktwolke und die notwendigen Berechnungen zur Generierung dieser dimensionsreduzierenden Darstellung sind in Kapitel 5 ausführlich dargestellt. Dem Programmablaufplan ist zu entnehmen, dass alternative Ansätze der Kandidatensuche implementiert wurden. Diese liefern Zielzeichenkandidaten im Bild, welche nach einer Validierung im Bild in die Punktwolke überführt werden. Das Ergebnis dieser Teilschritte sind Subpunktwolken, in denen alle 3D-Messpunkte, die auf den potenziellen Kugel-Zielzeichen liegen, enthalten sind. Wie in Abbildung 3.1 im zweiten Block dargestellt ist, erfolgt eine weitere Validierung dieser Kandidaten in der 3D-Subpunktwolke. Das anschließende Klassifizieren der zur Kugel gehörenden 3D-Messpunkte sowie das robuste Bestimmen von Näherungswerten des Kugelmittelpunktes wird in Kapitel 6 behandelt. Kapitel 7 stellt die notwendigen Methoden und implementierungsseitigen Auswerteschritte der endgültigen Schätzung der Zielzeichenkoordinaten dar. Die Ergebnisse dieser Schätzung bilden die Grundlage für die Bestimmung der Transformationsparameter zwischen den jeweiligen lokalen und dem globalen Koordinatensystemen. Im Bild nicht detektierte Zielzeichen werden durch Rücktransformation der globalen Passpunkte in den jeweiligen lokalen Standpunkt analog zu den zuvor beschriebenen Schritten in die endgültige Berechnung der Transformationsparameter einbezogen. Die notwendigen Schritte zur zielzeichen-basierten Registrierung bei identischen Passpunkten werden in Kapitel 8 beschrieben. Das finale Produkt der Software ist, wie in Abbildung 3.1 unten zu sehen, ein Set aus registrierten Punktwolken.
Die Reihenfolge der nächsten Kapitel wurde an den Ablauf der Auswertung des implementierten Programms gewählt. Alle Teilschritte der Software können im Programmablaufplan aus Abbildung 3.1 nachvollzogen werden.

Abbildung 3.1: Programmablaufplan

Kapitel 4

Datenfluss und Vorprozessierung

In diesem Kapitel wird die Repräsentation der Punktwolke in einem spezifischen Datenformat erläutert und das Einlesen der Daten in die Software dargestellt. Weiter werden die nötigen Schritte der Vorprozessierung der 3D-Punktwolken und deren Überführung in verschiedene Bilder ausführlich dargestellt. Ziel dieses Kapitel ist abschließend das Bereitstellen von Bildern für die effiziente Suche potenzieller Kugel-Zielzeichen.

4.1 Datenformat der Punktwolken

Für das Ein- und Auslesen der Punktwolken musste gleich zu Beginn der Programmierung ein geeignetes Datenformat aus einer Vielzahl möglicher Formate zur Speicherung von 3D-Punktwolken gewählt werden. Grundsätzlich sind die bestehenden Datenformate zur Speicherung von Punktwolken und ggf. deren Metadaten in zwei Kategorien und Unterklassen einzuordnen. Die erste äußerst einfache Variante besteht darin, die Punktwolke in einer `ASCII`-Datei in der Form $[x_i, y_i, z_i, int]_{[nx4]}$ vorzuhalten. Formate wie `pts` können zusätzlich RGB Werte zu jedem Punkt speichern. Ein weiteres Format ist das `ptx`-Format, welches zusätzlich Informationen zu bereits registrierten Punktwolken abbildet. Diese Formate sind einfach und menschenlesbar und damit leicht und flexibel in eigene Software zu integrieren. Sollen segmentierte Ausschnitte aus Punktwolken beispielsweise modelliert oder weiter analysiert werden, bietet sich das Speichern in einem dieser Formate an.
Neben den ASCII-Formaten sind gerade für größere Punktmengen oder wie in diesem Fall beim Speichern ganzer Punktwolken binäre Datenformate unverzichtbar. Das Ein- und Auslesen erfolgt signifikant schneller und der benötigte Speicherbedarf reduziert sich deutlich. Zum Vergleich benötigt eine Punktwolke im `pts`-Format rund 2.1GB bei einer Punktzahl von ca. 38.75 Millionen Punkten $P_i = [x_i, y_i, z_i, int_i]$. Die gleiche Punktwolke in binären Formaten wie E57, CloudCompare entities BIN oder dem CloudCompare `sbf` (Simple Binary File) Format benötigt mit lediglich 620 MB, rund 29% der ASCII Größe.
Das Einlesen einer Testpunktwolke mit den oben genannten rund 40 Mio. Punkten benötigt im `pts`-Format ca. 50 Sekunden, im `sbf`-Format lediglich mit rund 5.6 Sekunden nur 11%. Die Vorteile der binären Speicherung der Daten sind nach Betrachtung der Abbildungen 4.1a und 4.1b ausschlaggebend für die Wahl des für die Implementierung einzusetzenden Formates. Das in der Arbeit genutzte binäre Format ist das in der CloudCompare Softwareumgebung bekannte `sbf`-Format. Im folgenden soll kurz auf die Struktur der Daten eingegangen werden.

SBF-Format

Um möglichst einfach größere Punktwolken zu exportieren und in externen Programmen zu prozessieren gibt es neben den mächtigeren Datenformaten in CloudCompare, welche alle erstellten Objekte und Metainformationen von prozessierten Daten speichern, das explizit einfach gehaltene

Abbildung 4.1: Speicherbedarf (a) und Einlesegeschwindigkeiten der Punktwolke in CloudCompare (b) in Abhängigkeit des Datenformates

Datenformat SBF (Simple Binary File). Hier werden die Daten der Punktwolke auf zwei Dateien aufgeteilt:

*.**sbf** ASCII Textdatei mit Metadaten der Punktwolke, Vorteil: menschenlesbar

*.**data.sbf** die eigentliche Datei zur Speicherung von großen Punktwolken, Vorteil: Komplexität und Effizienz.

Für den Datenfluss ist bei Punktwolken der Struktur $P_i = [x_i, y_i, z_i, int_i]$ das Erzeugen einer Metadatei (*.sbf) nicht zwingend notwendig, für die nachhaltige Ablage der Daten aber sinnvoll. Hier reicht das Anlegen der binären Datei (*data.sbf) aus.
Bevor die 3D-Punktdaten und deren Intensitäten als Skalarfeld im Body der Datei geschrieben werden, folgt ein 64 Byte langer Header (siehe Tabelle 4.1).

Byte	Inhalt	Datentyp in C++
0-1	Header flag	`std::uint8_t [2]`
2-9	Punktanzahl	`std::uint64_t`
10-11	Anzahl der Skalarfelder	`std::uint16_t`
12-19	X-Koordinatenoffset	`double`
20-27	Y-Koordinatenoffset	`double`
28-35	Z-Koordinatenoffset	`double`
36-63	freier Platzhalter	`char [28]`

Tabelle 4.1: Byte-Muster des Headers einer binären *.data.sbf Datei

Nach einer identifizierenden flag folgen Informationen zur Menge der 3D-Punkte und die Anzahl deren zusätzlicher Datenfelder, wie z.B. mit abgespeicherten Intensitäten. Bei Punktwolken, meist bereits registrierten Datensätzen, folgen ggf. angebrachte Koordinaten-Offsets, wodurch nicht zu jedem Punkt große Koordinatenwerte gespeichert werden müssen. Weiterer Speicherplatz im Header sind für ggf. in Zukunft wichtige Informationen reserviert. In Tabelle 4.2 ist weiter die Struktur des Body-Teils der *data.sbf Datei dargestellt.
Die eigentlichen Punktdaten sind damit sequentiell einlesbar. Da bereits im Header definiert ist, wie viele Skalarfelder zu jedem Punkt gelesen werden müssen, kann die Ablagestruktur in eigenen Programmen vor dem tatsächlichen Einladen dynamisch angepasst werden. Die in dieser Arbeit genutzten Punktwolken sind ausschließlich mit einem Skalarfeld, der Intensität, gespeichert.

bytes	Inhalt	Datentyp in C++
0-3	X-Koordinate	`float`
4-7	Y-Koordinate	`float`
8-11	Z-Koordinate	`float`
12-15	Skalarwert 1	`float`
16-19	Skalarwert 2	`float`
...	...	

Tabelle 4.2: Body-Teil der binären *.data.sbf Datei

Bevor nun auf die Punktwolkenklasse und die implementierten Ansätze der automatisierten Detektion von Kugel-Zielzeichen eingegangen werden soll, sei an dieser Stelle abschließend auf folgende Besonderheit des SBF-Format in CloudCompare hingewiesen.
Die Byte-Reihenfolge (engl. endianness) ist für das binäre Lesen und Schreiben der Daten an das aktuelle System anzupassen. Die Byte-Reihenfolge beschreibt, unterschieden in big-endian und little-endian, die Position des höchstwertigen Byte im Speicher [vgl. Lee und Halligan 1998]. So ist das big-endian Format wie die Angabe einer Uhrzeit hh:mm:ss (höchstwertige Größe vorne) zu interpretieren, wohingegen das little-endian Format mit der Schreibweise des Datums dd.mm.yyyy (höchstwertige Größe hinten) vergleichbar ist. Bei der Implementierung der beiden Routinen zum Einlesen und Schreiben der Punktwolken mussten die Daten jeweils mit der in Listing 4.1 aufgeführten Funktion transformiert werden.

```
1        void swap_endianness(float & fvalue) { // float: x y z intensity
2                auto re_fvalue = reinterpret_cast<uint8_t *>(&fvalue);
3                std::reverse(re_fvalue, re_fvalue + sizeof(fvalue));
4        }
```

Listing 4.1: Vertauschung der Byte-weisen Speicherreihenfolge in C++ zur korrekten Interpretation der binären Daten (big-little endian conversion)

4.2 Dimensionsreduktion durch Bildabwicklung Punktwolke

Zur Reduktion der Dimension werden zu Beginn alle Punkte der 3D-Punktwolke für jeden Standpunkt mit

$$P_i = \begin{bmatrix} d_i \\ h_i \\ v_i \end{bmatrix} = \begin{bmatrix} \sqrt{x_i^2 + y_i^2 + z_i^2} \\ \text{atan2}(x_i, y_i) \\ \text{acos}(\frac{z_i}{d_i}) \end{bmatrix} \tag{4.2.1}$$

aus der kartesischen Form in ihre polaren Messelemente umgerechnet. Durch die konstante Winkelabtastung ergibt sich ein 2D-Gitter, welches als Bild dargestellt werden kann. In Gleichung (4.2.1) beschreiben h_i die Horizontalrichtung, v_i den jeweiligen Vertikalwinkel und d_i die Raumstrecke (Distanz) zum Messpunkt. Die polaren Messelemente lassen sich 2D in Bildern repräsentieren, welche abhängig von der bekannten Auflösungseinstellung am Panorama-Laserscanner in ihrer Dimension initialisiert werden. Kugeln erscheinen in diesen Bildern als Kreise, was die automatisierte Suche der Kugel-Zielzeichen stark vereinfacht. Für die in Tabelle 2.1 aufgeführte Auflösung von 6.3 mm @10 m und das vertikale Sichtfeld von 270° der P16 ergibt sich nach

$$\begin{aligned} \#\ \text{rows} &= \frac{\frac{2\pi}{360^\circ} \cdot 135^\circ}{6.3 \cdot 10^{-4}} = 4000 \\ \#\ \text{columns} &= \frac{2\pi}{6.3 \cdot 10^{-4}} = 10\,000 \end{aligned} \tag{4.2.2}$$

(a) (b) (c)

Abbildung 4.2: (a) Punktwolkenausschnitt in CloudCompare; (b) Ausschnitt des Intensitätsbildes; (c) Ausschnitt des Distanzbildes

(a) (b)

Abbildung 4.3: exemplarische Darstellung zweier Kugel-Zielzeichen im jeweiligen Intensitätsbild

eine Bildgröße von 4000 $\times$10 000 Pixel.
Die Bildkoordinaten in Pixeln ergeben sich aus der Horizontalrichtung h_i und dem Vertikalwinkel v_i der polaren Messelemente eines Punktes. Jedem Pixel wird im **Distanzbild** I_D die Entfernung d_i und im **Intensitätsbild** I_I der Intensitätswert des 3D-Punktes zugewiesen. Pixel, für deren Position keine Scanpunkte vorliegen, bleiben somit Null. Solche nicht gefüllten Pixel entstehen bei der Zuordnung der Pixelindizes über verrauschte Horizontalrichtungen und Vertikalwinkel sowie aus Messungen, zu denen keine Intensität zurückgestrahlt wurde. Die generierten Bilder stellen somit keinen Informationsverlust aus der 3D-Punktwolke dar und die abgebildeten Messpunkte sind später aus den Bildern zurück in die 3D-Darstellung der Punktwolke umrechenbar.
In Abbildung 4.2b und 4.2c sind Ausschnitte aus den jeweiligen Bildern des in Abbildung 4.2a dargestellten Punktwolkenausschnittes wiedergegeben. In den Abbildungen 4.3a und 4.3b sind exemplarisch zwei Intensitätsbildausschnitte um Kugel-Zielzeichen dargestellt. Um die Informationen (Distanz und Intensität) aus beiden generierten Bildern für die Suche nach Kreisen zu nutzen, wird ein weiteres Bild eingeführt, das **modifizierte Distanzbild**.
Hierzu werden einige Annahmen getroffen (vgl. Abschnitt 2.4) und Schwellwerte für zulässige Pixelwerte definiert. Diese ergeben sich bspw. aus dem Vorwissen, dass weiße Kugeln günstige Reflektionseigenschaften aufweisen (vgl. Abschnitt 2.3) oder, dass sich das Platzieren von Kugel-Zielzeichen in zu großer Entfernung negativ auf die Punktanzahl auswirkt und solche Zielzeichen keinen Mehrwert für die Registrierung darstellen.

Abbildung 4.4: Datenreduzierung durch Abbildung im modifizierten Distanzbild für alle Standpunkte der Testdatensätze

(a) (b)

Abbildung 4.5: exemplarische Darstellung zweier Kugel-Zielzeichen im jeweiligen modifizierten Distanzbild

Objektpunkte werden nur dann im modifizierten Distanzbild abgebildet, wenn sie zwischen 0.3 m und 25 m entfernt sind. Dies macht Sinn, da aus Gründen der Punktdichte auf dem Zielzeichen und der daraus resultierenden Güte des Schätzergebnisses schon im Außendienst explizit darauf geachtet wird, die Kugeln nicht in zu großer Entfernung aufzustellen. Die maximale Entfernung von 30 m bietet hier genügend Puffer, beachtet man, dass die Zielzeichen üblicherweise in Entfernungen bis max. 20 m platziert sind.
Neben diesen Restriktionen werden weiter nur jene Punkte mit ihrer tatsächlichen Entfernung im modifizierten Distanzbild abgebildet, deren gemessene Intensitäten zwischen 40 % und 100 % liegen. Alle Punkte mit Intensitäten kleiner 40 % werden mit der maximalen Distanz 30 m abgebildet. Dadurch lässt sich die Anzahl der zu untersuchenden Pixel auf <36 % bis 0.04 % reduzieren (vgl. Abbildung 4.4).
In den Abbildungen 4.5a und 4.5b sind die zu den Ausschnitten aus 4.3a und 4.3b generierten modifizierten Distanzbilder $I_{D'}$ dargestellt.
Einzelnen Pixeln, denen keine modifizierten Distanzen d_{mod} zugeordnet wurden, welche aber von gefüllten Pixeln umgeben sind, wird mittels Bildfaltung ein Nachbarwert zugeordnet. Implementiert

ist dazu ein normierter Kernel mit einem hohem Gewicht auf das aktuell gewählte Pixel. Mit Hilfe dieser Funktion werden mit Null initialisierte Pixel in solchen Umgebungen wie beschrieben neu berechnet. Hierzu erfolgt eine Interpolation ohne vormals leere Pixel aus den Werten der Umgebung.

4.3 Punktwolkenklasse in C++

Die zentrale Komponente des implementierten Programms bildet eine Klasse `PointCloud` (vgl. Listing 4.3).

```
class Pointcloud_sbf {// class for the Pointcloud stored in simple binary format (sbf)
 public:
  void find_spheres(std::vector<sphere_candidate_image> &vec_sphere_candidates, double
      sphere_radius);
  void get_PCoi(const double *ptr_bounds, std::vector<pointXYZ> &PCoi, int max_pts);
  static void save_PCoi(const std::string filename, std::vector<pointXYZ> &PCoi);
  void readBinaryData(const std::string fName);
  void writeBinaryData(const std::string fName);
  void registerPC(const trafo_param_set trafo_param);
  void generate_Image(const std::string image_mode);
  template <typename> static void zeroconv(cv::Mat &image);
  // scanner resolution results from multiples of PI, i.e. 6.3mm@10m (2*Pi)
  double instrument_resolution = 2 * M_PI * 0.001 / 10;
  float instrument_vertical_field_of_view = 146.0; // v. field of view [degree]

  // distance and intensity ranges within sphere targets
  float min_distance = 0.3;     // min distance [m]
  float max_distance = 25.0;    // max distance [m]
  float min_intensity = 40.0;   // min intensity [percent]
  float max_intensity = 100.0;  // max intensity [percent]

  // the images (distance, modified distance)
  cv::Mat distance_image;
  cv::Mat mod_distance_image;

 private:
  // data members of class pointcloud
  // declaration of the header file data
  std::uint8_t _flag[2];              // SBF header flag
  std::uint64_t _point_count;         // Point count (Np)
  std::uint16_t _scalar_field_count;  // Scalar field count (Ns)
  double _x_coord_shift;              // X coordinate shift
  double _y_coord_shift;              // Y coordinate shift
  double _z_coord_shift;              // Z coordinate shift
  char _reserved_space[28];           // reserved

  // declaration of the actually pointcloud data
  std::vector<float> _data_x_coords;
  std::vector<float> _data_y_coords;
  std::vector<float> _data_z_coords;
  std::vector<float> _data_intensity;
  std::vector<float> _calc_distance;
};
```

Listing 4.2: PointCloud.hpp

Neben dem Vorhalten der Punktwolkendaten mit ihrer Geometrie sowie den Intensitäten aller 3D-Punkte, werden in ihr auch alle vorhandenen Metadaten berücksichtigt.
Die Speicherung erfolgt analog zu den beiden Teilen des gewählten `sbf`-Formates aufgeteilt in Header und Body Informationen. Jedes Objekt der Klasse beinhaltet somit Angaben zu den in Tabelle 4.1 aufgeführten Metadaten und den eigentlichen 3D-Punkten aus Tabelle 4.2.

Neben der Klasse als Datenstruktur sind alle Funktionen, welche zur automatisierten Detektion von Kugel-Zielzeichen implementiert sind, an die Klasse angebunden. Zusätzlich zu der Berechnung der Intensitäts- und Distanzbilder wird analog zu den Beschreibungen aus Abschnitt 4.2 zu jeder Punktwolke ein modifiziertes Distanzbild generiert.
Eine effiziente Methode der Bildgenerierung wurde wie in Algorithmus 1 implementiert.

Algorithm 1: generiere Bilder aus Punktwolke

input : Punktwolke $[x, y, z, int]_n$ mit n Punkten, Instrumentenauflösung Δ
output: Intensitätsbild I_I, Distanzbild I_D

1 **for** $i \leftarrow 1$ **to** n **do**
2 $\quad d = \sqrt{x_i^2 + y_i^2 + z_i^2}$;
3 $\quad$ r $= \text{acos}(\frac{z_i}{d_i})$ / Δ;
4 $\quad$ c $= \text{atan2}(x_i, y_i)$ / Δ;
5 $\quad I_I$(r,c) $\leftarrow int_i$;
6 $\quad I_D$(r,c) $\leftarrow d_i$;
7 **end**

Die Repräsentation der benötigten Bilder sowie die beschriebenen Berechnungen in den Bildmatrizen erfolgen aufbauend auf der `cv::Mat`-Klasse der OpenCV-Bibliothek für C++ [vgl. Open Source Computer Vision, OpenCV 2020]. In der Klasse `cv::Mat` können n-dimensionale Einkanal- oder Mehrkanal-Arrays abgebildet werden, welche Zeile für Zeile (row-major order) und bei mehreren Kanälen Ebene für Ebene gespeichert sind [vgl. OpenCV 2020a]. Im Sinne der effizienten Implementierung ist auf die Speicherreihenfolge zu achten. Die Zugriffe in den Funktionen zur Kandidatensuche und Interpolation erfolgen somit stets in gleicher Form.
Um ggf. auftretende Datenlücken im Bild zu vermeiden, ist eine weitere Funktion zur Füllung der Null-Pixel unmittelbar an die Erstellung der Bilder geknüpft. An alle Positionen im Bild, deren Pixelwert Null ist, wird, sobald in der direkten Umgebung nicht leere Pixel vorhanden sind, das Mittel jener nicht leeren Pixel als neuer Wert geschrieben. Außer in tatsächlich freien Ausschnitten der Punktwolke (Standpunkte mit Himmelsfreiheit) ist somit sichergestellt, dass eine vollständige Bildmatrix zur Kandidatensuche vorliegt. Die Methode kann als Filter mit gleichen Gewichten verstanden werden, in der leere Pixel jedoch nicht berücksichtigt sind, was eine Interpolation der fehlenden Daten zulässt.

Kapitel 5

Detektion von Zielzeichenkandidaten

In den zuvor generierten Bildern sollen potenzielle Zielzeichen, somit hier die geometrische Form der Kugel, detektiert werden. Zwei Ansätze deren theoretische Betrachtung sowie implementierungsseitige Umsetzung werden in den folgenden Abschnitten dargestellt. Ergebnis dieser Ansätze sind Zentrumskoordinaten oder Regionen im Bild, an denen ein Kugel-Zielzeichen vermutet wird. Diese Kandidaten werden dann mit den in Kapitel 6 beschriebenen Methoden validiert.

5.1 Circle-Hough-Transformation zur Kreissuche

Betrachtet man z.B. Abbildung 4.5a, so ist unmittelbar ersichtlich, dass sich die Kugeln im modifizierten Distanzbild eindeutig als Kreise von ihren lokalen Umgebungen abheben. Somit erfolgt der erste logische Ansatz als die Suche von Kreisen in modifizierten Distanzbildern. Als Standardverfahren der Kreissuche in der Computer Vision findet die Circle Hough Transformation in unterschiedlichsten Disziplinen Anwendung. So werden mit dieser Methode beispielsweise in mikroskopisch-medizinischen Bildern einzelne Zellkerne, welche als Kreise erscheinen, detektiert [Smereka und Dulęba 2008]. In einer anderen Arbeit konnte ein Ballerkennungssystem zur Unterstützung von Fußballschiedsrichtern bei der Entscheidungsfindung in zweifelhaften Situationen realisiert werden. Fußbälle erscheinen ebenso als Kreise in Bildern und können somit mit der Circle Hough Transformation relativ zur kalibrierten Torlinie erkannt werden [vgl. D'Orazio u. a. 2004]. Andere Arbeiten zeigen das Potenzial der Circle Hough Transformation bei der automatisierten Erkennung von Personen in Bildern. Teilweise werden Iris der menschlichen Augen als zu detektierende Objekte genutzt (vgl. Cherabit, Chelali und Djeradi 2012), eine andere Arbeit beschäftigt sich mit der unmittelbaren Erkennung von Köpfen aus der Vogelperspektive einer Aufzugkamera [Liu, Qian und Lin 2010].

Im folgenden Teilabschnitt werden schematisch die Grundlagen der zur Kreissuche verwendeten Circle-Hough-Transformation dargestellt und weiter erläutert. Die Circle Hough Transformation (CHT) ist eine spezielle Variante der allgemeineren Hough Transformation, deren Idee P.V.C. Hough 1962 in einer Patentschrift beschrieb [vgl. Bässmann und Kreyss 2013]. Im Folgenden soll ausschließlich auf die speziellere CHT eingegangen werden [vgl. Davies 2004, S. 284–313].
Zu Beginn der Kreissuche im Bild´ werden per Filterung in zwei Dimensionen alle vorhandenen Kanten extrahiert. In den meisten Implementierungen findet der Canny-Edge-Detector zur zweidimensionalen Kantenextraktion Anwendung [vgl. Canny 1986, S. 690–697]. Nach zweidimensionaler Filterung des Eingangsbildes bspw. mit den Ableitungsfilterkernen

$$k_{G_1/x} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & -4 & 1 \\ 0 & 0 & 1 \end{bmatrix}; \quad k_{G_1/y} = \begin{bmatrix} 0 & 1 & 1 \\ 0 & -4 & 0 \\ 1 & 1 & 0 \end{bmatrix} \tag{5.1.1}$$

Abbildung 5.1: binäres Filterergebnis nach Anwendung von (5.1.1) und (5.1.2) auf Bild 4.5a

Abbildung 5.2: schematische Darstellung der Circle Hough Transformation in drei Schritten

ergeben sich die Gradientenbilder G_1 und G_2. Diese werden mit

$$G(x,y) = \sqrt{G_1^2(x,y) + G_2^2(x,y)} \tag{5.1.2}$$

zu einem gemeinsamen Gradientenbild zusammengeführt, welches die Norm darstellt [vgl. Mitra, Chandra und Halder 2013]. Für ein Eingangsbild aus Abb. 4.5a ergibt sich dann das in Abb. 5.1 dargestellte Filterergebnis G. Hier ist das Kugel-Zielzeichen deutlich als Kreis erkennbar.
Der beschriebene Ansatz kann als Kantenextraktionsfilterung interpretiert werden. Kanten sind jene Bereiche im Bild, in denen sich benachbarte Pixelwerte stark von einander unterscheiden. Die Norm des Gradienten an solchen Stellen ist im Vergleich zu homogenen Bereichen groß.

Detektion von Kreisen im Kantenbild

Alle Kantenpixel werden im Ablauf der Circle Hough Transformation (CHT) als Mittelpunkt eines möglichen Kreises mit festem Suchradius R_{fix} interpretiert. Die in den Abbildungen 5.2a bis 5.2c schwarzen Punkte auf dem gesuchten roten Kreis liegenden Kantenpixel bilden die Mittelpunkte der gestrichelten Kreise. In einer Akkumulationsmatrix, die in der Größe des Bildes initialisiert ist, wird ihr Wert an der Stelle des Pixels, in dem sich Kreise schneiden, um 1 erhöht. Wird dieser Vorgang für möglichst viele Kantenpixel nacheinander durchgeführt, ergibt sich, wie in Abbildung 5.2c zu sehen, ein Pixel, welches in der Akkumulationsmatrix dominiert. Dieses dominante Pixel

repräsentiert dann den Mittelpunkt des erfolgreich gefundenen Kreises im Bild.
Ist der Kreisradius unbekannt oder sind mehrere Radien möglich, so ergibt sich der im Pseudocode 2 dargestellte Ablauf.
Ein weiterer Implementierungsansatz beschreibt das Vorgehen anders. Dort werden mehrere Kreise um die in Abb. 5.2 dargestellten schwarzen Punkte gebildet. Die Kantenpixel liegen auf den Rändern der neuen Kreise. Akkumuliert werden dann die Mittelpunkte der generierten Kreise, was wiederum zu einer identischen Akkumulationsmatrix führt. Auf eine Darstellung der beschriebenen zweiten Variante soll hier verzichtet werden.

Algorithm 2: Circle Hough Transformation - Voting mit Akkumulator-Matrix

input : Kantenbild **I**, mögliche Radien **R**, Akkumulator-Matrix **A**
output: Akkumulator-Matrix **A**

```
1 foreach Pixel = I(x,y) do
2     foreach r in R do
3         foreach t = 0 to 2π do
4             row = x - r · cos t;
5             col = y - r · sin t;
6             A(row,col,r) ++;
7         end
8     end
9 end
```

Die CHT ist sowohl in der Funktion `imfindcircles()` im image package für GNU Octave (GNU Octave 2020a) implementiert, als auch in der Funktion `HoughCircles()` aus der Open Computer Vision Library (OpenCV) für C++ (OpenCV 2020b). Der Radius R_{fix} kann den Funktionen als solcher übergeben werden, alternativ kann ein definierter Bereich von minimalem und maximalem Kreisradius $R_{\text{region}} = [R_{\text{min}}, R_{\text{max}}]$ übergeben werden. Für diesen Fall wird die bereits beschriebene Suche für diskrete Werte zwischen R_{min} und R_{max} wiederholt. Die Definition einer solchen Region ist für die aktuelle Problemstellung zwingend nötig, da Kugeln in unterschiedlicher Entfernung zum Scanner zu verschieden großen Kreisen im aus dem modifizierten Distanzbild $I_{D'}$ gerechneten Kantenbild führen.
Als Ergebnis der CHT ergeben sich die Mittelpunkte der gefundenen Kreise als Bildkoordinaten. Diese Bildkoordinaten lassen sich in einem nächsten Schritt in 3D-Koordinaten der Punktwolke umrechnen. Um diese Zentrumskoordinaten werden jeweils Subpunktwolken erzeugt, in denen mit dem in Abschnitt 6.3 dargestellten Verfahren robuste Näherungswerte der Kugelmittelpunkte bestimmt werden. In [Schreinemachers 2020a] konnte bereits für gutmütige Konfigurationen der Kugel-Zielzeichen sowie deren möglichst freie nähere Umgebung gezeigt werden, dass die CHT eine mögliche Methode zur Kandidatensuche in modifizierten Distanzbildern darstellt.

5.1.1 Algorithmisches Vorgehen

Wie in Abschnitt 5.1 beschrieben, kann die Methode der Circle-Hough-Transformation zur Kandidatensuche in modifizierten Bildern genutzt werden. Um möglichst kleine Regionen valider Suchradien zu realisieren, erfolgte die Berechnung in verschiedenen Distanzklassen. Dazu werden mehrere Bilder aus der Punktwolke abgeleitet, die jeweils unterschiedliche Distanzbereiche abbilden (vgl. Abbildung 5.3). Die Anzahl der Bilder erhöht sich somit von einem einzigen auf mehrere Teilbilder, diese sind jedoch dünn besetzt und die Laufzeit der CHT kann pro Bild stark reduziert werden. Die Suchradien werden anhand des bekannten Radius der Kugel-Zielzeichen und den zugelassenen Entfernungen pro Bild jeweils spezifisch berechnet, somit muss in jedem Distanzklassenbild lediglich ein deutlich kleineres Intervall valider Radien durchlaufen werden.

Abbildung 5.3: Intensitätsbilder verschiedener Distanzklassen (invertiert)

Abbildung 5.4: Intensitätsbild (invertiert) generiert für Distanzen von 0.3 m bis 1.0 m, Suchradien von 770 Pixel bis 230 Pixel

Die Suche der Kreise in den Bildern kann zusätzlich parallelisiert erfolgen, was die Effizienz erneut steigert. Realisiert wurde die parallele Verarbeitung der Bilder mittels OpenMP (Open Multi-Processing), eine Programmierschnittstelle (API) für Shared-Memory-Programmierung in C++ und weiteren Programmiersprachen [vgl. Dagum und Menon 1998]. Die Parallelisierung erfolgt auf der Ebene von unabhängigen Programmabschnitten in Schleifen. So werden die distanzabhängigen Bilder nacheinander aus der Punktwolke erzeugt und die Kandidatensuche auf den einzelnen Bildern parallel ausgeführt. Zu beachten ist hier, dass die Abbildungen der Punktwolken in Teilbilder mit Überlappungsbereichen in der Größenordnung eines Kugeldurchmessers erfolgen müssen. Kugelkandidaten können somit ggf. mehrfach detektiert werden. In den Abbildungen 5.4 bis 5.6 sind exemplarisch zwei Intensitätsbilder dargestellt, welche mit vorgegebenen Distanzklassen erzeugt wurden. Weiße Bereiche im Bild erzeugen in der Kantenextraktion keine Pixel, welche mittels der CHT berücksichtigt würden, und steigern somit neben den beschriebenen verkleinerten Regionen der Suchradien weiter die Effizienz. Die verschiedenen Größen, in denen die Kugel-Zielzeichen (rot markiert) dieses Standpunktes erscheinen, verdeutlichen das Optimierungspotenzial. Tatsächlich liefern die parallelisierten Suchen in den einzelnen Bildern in gutmütigen Datensätzen ähnlich den

Abbildung 5.5: Intensitätsbild (invertiert) generiert für Distanzen von 1.0 m bis 2.0 m, Suchradien von 230 Pixel bis 115 Pixel

Abbildung 5.6: Intensitätsbild (invertiert) generiert für Distanzen von 7.5 m bis 10.0 m, Suchradien von 31 Pixel bis 23 Pixel

Abbildung 5.7: Pixelwerte in der Umgebung eines Kugelkandidaten

dargestellten Punktwolken gute Ergebnisse. Punktwolken mit wenig validen Messpunkten im Sinne der Restriktionen an die Abbildung im modifizierten Distanzbild (valide Intensitäten: 40 % bis 100 % und den verschiedenen Distanzbereichen) konnten mit der beschriebenen Methode verarbeitet werden.
Das Verfahren hat jedoch signifikante Schwächen, welche im Laufe der Arbeit aufgedeckt wurden und zur Entwicklung des in Abschnitt 5.2 beschriebenen Ansatzes führten. Die wesentlichen Schwachstellen sind das Detektieren vieler falscher Kreiskandidaten in den abgeleiteten Kantenbildern und deren Aufdeckung. Diese sogenannten false-positiv Kandidaten müssen in weiteren Schritten im Bild validiert und können teilweise erst in der dreidimensionalen Punktwolke als false-positiv aufgedeckt werden.

5.2 Kandidatensuche mit neuem Ansatz im Distanzbild

Anders als im bisher diskutierten Verfahren der CHT-gestützten Kandidatensuche in modifizierten Distanzbildern, findet die Suche im neuen Ansatz unmittelbar im Distanzbild der Punktwolke statt. Ohne das Extrahieren von Kanten werden hier die erwarteten geometrischen Eigenschaften der Kugel-Zielzeichen direkt im Distanzbild zur Suche genutzt. Durch das Einbinden der Distanzen in die Suche der Kugeln wird, motiviert durch (Franaszek, Cheok und Witzgall 2009), eine weitere Dimension genutzt.
Das vorbereitende Generieren von Teilbildern verschiedener Distanzklassen kann somit vermieden werden. Hier müssen keine Kreise mit verschiedenen Suchradien in einem Kantenbild detektiert werden, sondern geometrische Eigenschaften der bekannten Kugeln. Nachfolgend ist der Ablauf dieses Verfahrens erläutert.
Für jedes Pixel im Distanzbild werden abhängig von seinem aktuellen Wert (Pixelwert im Distanzbild $\widehat{=}$ Distanz zum Messpunkt) Soll-Eigenschaften einer potenziellen Kugel im Bild abgeleitet. Da der Radius aller Kugel-Zielzeichen (R = 72.5 mm) bekannt ist, kann mit der ebenfalls bekannten Distanz des aktuellen Pixels berechnet werden, wie eine Kugel an dieser Position im Bild abgebildet ist, würde sich das aktuelle Pixel aus der Sicht des Laserscanner mittig auf der Oberfläche befinden. Um diese Eigenschaften zu prüfen werden Randpixel (siehe Abbildung 5.7, blau) und Umgebungspixel (siehe Abbildung 5.7, rot) selektiert und getestet. Randpixel befinden sich in einem Abstand von 60 mm am Rand der Kugel, jedoch immer auf der Kugeloberfläche. Für jedes aktuell betrachtete Pixel (Abbildung 5.7 grün) wird der Abstand zu diesen Randpixeln in Pixeln berech-

Abbildung 5.8: Detektierte Kugelkandidaten Standpunkt 2 (Bahnübergang), abgebildet im Intensitätsbild (invertiert)

net. Abhängig von der gewählten Auflösung und der Distanz (Wert des grünen Pixels) ist dieser Abstand eindeutig bestimmbar. Für diese Pixel ist dann ebenfalls unmittelbar bekannt, wie viel sie bezogen auf das Zentrumspixel weiter entfernt sein müssen, sollte das aktuelle Pixel tatsächlich ein Zentrumspixel eines Kugel-Zielzeichens darstellen. Neben den Randpixeln werden analog die Abstände für die rot gekennzeichneten Umgebungspixel sowie ebenfalls valide Pixelwerte für diese berechnet. So wird für alle Umgebungspixel getestet, ob diese mindestens um den Durchmesser der Kugel weiter entfernt sind als das aktuelle Zentrumspixel.

Für die Distanzbilder der vorliegenden Datensätze konnten die Tests für alle Pixel in 1.2 Sekunden erfolgen. Bei den gewählten Auflösungen von 6.3 mm @10 m und 6.0 mm @10 m ergeben sich, wie in Tabelle 2.1 aufgeführt, ca. 40 Megapixel (4000 ×10 000 Pixel) große Bilder. In den folgenden Abbildungen 5.8 bis 5.11 sind exemplarisch für vier verschiedene Aufnahmesituationen alle detektierten Kugel-Kandidaten dargestellt. In den gewählten Bildern, welche die verschiedenen Projekte mit ihren jeweiligen Besonderheiten (vgl. Abschnitt 2.4) aus den Datengrundlagen abbilden, werden teilweise alle sechs Kugeln mindestens jedoch vier der platzierten Kugel-Zielzeichen detektiert, sodass eine inertiale Schätzung der Transformationsparameter möglich wird. In den Distanzbildern der Punktwolken im Kanalsystem werden alle Kugel-Zielzeichen erkannt. Die deutlich verschieden weit entfernten Kugeln aus Abbildung 5.10 werden ebenso zuverlässig detektiert wie jene sehr klein erscheinenden Kugeln in Abbildung 5.8. Im Distanzbild der objektreichen Punktwolke des Bonner Hbf wird lediglich die teilweise durch einen Bauzaun verdeckte Kugel nicht erkannt. Im Vergleich zur CHT-Methode kann jedoch für alle vier vorgestellten Punktwolken festgehalten werden, dass keine false-positiv Kandidaten in den Bildern erzeugt wurden. Da die benötigte Rechenzeit wie zuvor bereits beschrieben im Wesentlichen von der Anzahl nötiger, ggf. sehr aufwändiger, Validierungen abhängt, konnte mit der neu implementierten Methode eine äußerst effiziente und robuste Kandidatensuche umgesetzt werden.
Für jene 3D-Punktwolken, welche mit feineren Auflösungen gescannt wurden, etwa mit 3.1 mm @10 m oder 1.6 mm @10 m ergeben sich, wie in Tabelle 2.1 aufgeführt, deutlich größere Bilder. Eine Skalierung der Rechenzeiten bei der Kandidatensuche ergibt sich jedoch für höhere Auflösungen nicht, da die gesuchten Soll-Eigenschaften der Kugel-Zielzeichen konstant bleiben. Anders als in den Testdaten wird nicht jedes Pixel in Zeilen- und Spalten-Richtung abgefragt, sondern lediglich jedes zweite bzw. vierte Pixel pro Richtung. So ergäben sich aus höheren Auflösungen lediglich in den Funktionen des Ein- und Auslesens der Punktwolkendaten, sowie beim Generieren der Bilder höhere Laufzeiten. Die Laufzeiten der eigentlichen Kandidatensuche in Bildern bleibt jedoch identisch.

Abbildung 5.9: Detektierte Kugelkandidaten Standpunkt 10 (Kanalsystem), abgebildet im Intensitätsbild (invertiert)

Abbildung 5.10: Detektierte Kugelkandidaten Standpunkt 21 (Kanalsystem), abgebildet im Intensitätsbild (invertiert)

Abbildung 5.11: Detektierte Kugelkandidaten Standpunkt 1 (Bonner Hauptbahnhof), abgebildet im Intensitätsbild (invertiert)

Abbildung 5.12: Scanner Foto

5.2.1 Algorithmisches Vorgehen

Doppelte Kandidaten durch Spiegelung

Die Auswertung weiterer Testdatensätze ergab bis auf wenige Ausnahmen fehlerfrei detektierte Kugel-Zielzeichen im Bild. In Abbildung 5.12 ist ein HDR-Bild des Leica RTC360 für einen Standpunkt im Kanalsystem dargestellt. Die sich an der Wasseroberfläche spiegelnden Kugel-Zielzeichen sind in Abbildung 5.12 erkennbar. Tatsächlich ergeben sich in der Punktwolke scheinbar unter der Wasseroberfläche ebenfalls Kugeln. Diese resultieren aus den durch die Spiegelung abgelenkten Messstrahlen zu den eigentlich oben angebrachten Zielzeichen. In Abbildung 5.13a ist der Ausschnitt dieser Punktwolke seitlich dargestellt. In rot sind jene Messpunkte eingefärbt, welche durch die Strahlablenkung eine Kugel scheinbar unter der Wasseroberfläche repräsentieren.
Da, wie in Abschnitt 5.2 erläutert, geometrische Soll-Eigenschaften der Kugel-Zielzeichen in den Distanzbildern gesucht werden, liefert der Algorithmus auch für die gespiegelten Kugeln Ergebnisse. Abbildung 5.13b stellt die Situation als Ausschnitt des Distanzbildes mit den insgesamt vier detektierten Kugeln dar. Die rot markierten Kreise liegen exakt unter den eigentlichen Zielzeichen im Bild. Bei genauerer Betrachtung der Messpunkte unterhalb der Wasseroberfläche kann

Algorithm 3: geometrischer Ansatz der Kandidatensuche im Distanzbild

input : Distanzbild $\mathbf{I_D}$, modifiziertes Distanzbild $\mathbf{I}_{\text{mod}}$, Kugel-Zielzeichenradus $\mathbf{r}$, Instrumentenauflösung Δ

output: Kugelkandidaten $\mathbf{K}$

```
1  foreach Pixel = I_D(x,y) do
2      if I_mod(x,y)== 255 then
3          continue;                    // überspringe nicht valide Pixel aus I_mod
4      end
5      dist = I_D(x,y);
6      /* berechne Abstände in Pixeln                                          */
7      RPS = Δ· dist · 0.06;                          // Randpixel Schrittweite
8      UPS = Δ· dist · 0.09;                      // Umgebungspixel Schrittweite
9      /* berechne valide Distanzwerte für Rand- und Umgebungspixel            */
10     RP_min = dist + 0.015;                     // valide min. Dist. Randpixel
11     RP_max = dist + 0.040;                     // valide max. Dist. Randpixel
12     UP_min = dist + 0.145;                // valide max. Dist. Umgebungspixel
13     /* Distanzwerte an Pixelpositionen (top, right, bottom, left, ...)      */
14     RP_t  = I_D(x−RPS,y      );  RP_t_val = RP_t > RP_min && RP_t < RP_max;
15     RP_r  = I_D(x     ,y+RPS);  RP_r_val = RP_r > RP_min && RP_r < RP_max;
16     RP_b  = I_D(x+RPS,y      );  RP_b_val = RP_b > RP_min && RP_b < RP_max;
17     RP_l  = I_D(x     ,y−RPS);  RP_l_val = RP_l > RP_min && RP_l < RP_max;
18     RP_tl = I_D(x−RPS,y−RPS);  RP_tl_val = RP_tl > RP_min && RP_tl < RP_max;
19     RP_tr = I_D(x−RPS,y+RPS);  RP_tr_val = RP_tr > RP_min && RP_tr < RP_max;
20     RP_bl = I_D(x+RPS,y−RPS);  RP_bl_val = RP_bl > RP_min && RP_bl < RP_max;
21     RP_br = I_D(x+RPS,y+RPS);  RP_br_val = RP_br > RP_min && RP_br < RP_max;
22     UP_tl = I_D(x−UPS,y−UPS);  UP_tl_val = RP_tl > UP_min;
23     UP_tr = I_D(x−UPS,y+UPS);  UP_tr_val = RP_tr > UP_min;
24     UP_bl = I_D(x+UPS,y−UPS);  UP_bl_val = RP_bl > UP_min;
25     UP_br = I_D(x+UPS,y+UPS);  UP_br_val = RP_br > UP_min;
26     /* teste Rand- und Umgebungspixel                                       */
27     R_valid = RP_t_val && RP_r_val && ... && RP_bl_val && RP_br_val;
28     U_valid = UP_tl_val + UP_tr_val + UP_bl_val + UP_br_val ≥ 3;
29     if R_valid && U_valid then
30         K.append (x, y, dist + r);
31     end
32 end
33 /* lösche doppelte Kugelkandidaten                                          */
34 sortiere K nach x;
35 foreach K_i in K do
36     lösche K_i wenn | x_i − x_{i+1} | < 50 && | y_i − y_{i+1} | < 50;
37 end
38 sortiere K nach y;
39 foreach K_i in K do
40     lösche K_i wenn | x_i − x_{i+1} | < 50 && | y_i − y_{i+1} | < 50;
41 end
42 return K;
```

(a) (b)

Abbildung 5.13: (a) seitliche Ansicht des Punktwolkenausschnittes; (b) Visualisierung der Kugelkandidaten ohne Validierung mit modifiziertem Distanzbild und Algorithmus aus Abschnitt 5.2

(a) (b)

Abbildung 5.14: (a) modifiziertes Distanzbild des Ausschnittes; (b) Visualisierung der Kugelkandidaten mit Validierung mit modifiziertem Distanzbild und Algorithmus aus Abschnitt 5.2

festgestellt werden, dass die gemessenen Intensitäten deutlich kleiner ausfallen, als jene auf den eigentlichen Kugel-Zielzeichen. Da in den modifizierten Distanzbildern durch die in Abschnitt 4.2 beschriebenen Restriktionen nur gut reflektierende Messpunkte dargestellt werden, befinden sich im Ausschnitt des modifizierten Distanzbildes an den zuvor rot markierten Stellen keine validen Pixelwerte (vgl. Abbildung 5.14a). Der Abgleich aller im Distanzbild detektierten Kugel-Kandidaten mit den Zentrumspixeln im jeweiligen modifizierten Distanzbild liegt als Validierung nahe. Nach der beschriebenen Validierung werden in der neu implementierten Kandidatensuche im Bild sowohl die geometrischen Soll-Eigenschaften der Kugel-Zielzeichen, als auch die jeweiligen Intensitäten der Messpunkte genutzt. Dieser Ansatz nutzt demnach alle in den Daten der Punktwolken vorhandenen Informationen und liefert äußerst robust detektierte Kandidaten.

Kapitel 6

Kandidaten-Validierung als Zielzeichen & Näherungswerte

Die zuvor mit beiden vorgestellten Ansätzen in verschiedenen Bildern detektierten Kandidaten müssen im nächsten Schritt als tatsächliche Zielzeichen validiert werden. Das geschieht in einer ersten Analyse bereits direkt in den Bildern. Die bildbasierte Validierung ist im nächsten Abschnitt beschrieben. Das endgültige und deutlich zuverlässigere Testen der Kandidaten erfolgt in der 3D-Punktwolke. Die nötigen Prozessierungsschritte sowie das methodische Vorgehen werden ab Abschnitt 6.2 dargestellt. Abschließend werden Punkte auf den validierten Kugel-Zielzeichen klassifiziert und zur finalen Schätzung des Zielzeichen-Referenzpunktes übergeben.

6.1 Validierung der Zielzeichen-Kandidaten im Bild

Im Bild funktioniert die Validierung bspw. durch Testen der Streuung aller Pixelwerte. Dies kann im Distanzbild sowie im Intensitätsbild erfolgen. In Abbildung 6.1 sind sowohl eine tatsächliche Kugel (vgl. Abbildung 6.1a) als auch ein falsch detektierter Kandidat, hier aus der Kreissuche der CHT (vgl. Abbildung 6.1b), dargestellt. So kann der in Abbildung 6.1b gefundene Kreis als Kugel-Kandidat ausgeschlossen werden, da die berechnete Varianz innerhalb des Kreises zu groß ist. In anderen Fällen gelingt eine zuverlässige Überprüfung der Kreise als Kugel-Kandidaten jedoch nicht. In solchen Szenarien muss die Validierung, wie in Abschnitt 6.3 beschrieben, direkt in der 3D-Punktwolke erfolgen. Das hat erhöhte Rechenzeiten zur Folge, welche diese Methode als weniger geeignet erscheinen lassen. In einigen Fällen (bei vielen false-positive Kandidaten aus der

(a) (b)

Abbildung 6.1: rot: mittels CHT gefundener Kreis in modifiziertem Distanzbild, (a) positives Bsp. mit homogenen Pixelwerten, (b) negatives Bsp. (false-positiv)

Abbildung 6.2: schematische Darstellung einer selektierten Subpunktwolke mit grünen Punkten auf der Oberfläche des Kugel-Zielzeichens und roten Umgebungspunkten

CHT) müssen pro Punktwolke (Standpunkt in einem Projekt) wenige Minuten ausschließlich für die Validierung der fälschlicherweise im Bild gefundenen Kandidaten aufgewendet werden.

6.2 Auswahl einer Subpunktwolke

Nachdem die Kugel-Zielzeichen-Kandidaten, welche mit dem vorgestellten Algorithmus aus Abschnitt 5.2 detektiert wurden und auch nach der bildbasierten Validierung als Kandidaten verbleiben, in ihre kartesischen Mittelpunktkoordinaten zurückgerechnet sind, folgt die Auswahl einer Subpunktwolke. Diese soll alle 3D-Punkte enthalten, die auf einem Kugel-Zielzeichen liegen. Aus den Zeilen und Spalten im Bild werden die Horizontalrichtung und der Vertikalwinkel eines jeden Kandidaten berechnet. Die Distanz als dritte Größe der polaren Koordinaten ergibt sich aus dem Zentrums-Pixelwert des Kandidaten selbst. Mit

$$P_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \begin{bmatrix} -d_i \cdot \sin v_i \sin h_i \\ -d_i \cdot \sin v_i \cos h_i \\ d_i \cdot \cos v_i \end{bmatrix} \tag{6.2.1}$$

kann der kartesische Mittelpunkt des Kugel-Zielzeichens als Näherungswert für das Zentrum der Subpunktwolke berechnet werden. Um diesen Punkt wird für jeden Kandidaten ein Würfel, dessen Kanten an den Koordinatenachsen ausgerichtet sind, generiert (siehe Abbildung 6.2). In jede der drei Raumrichtungen beträgt die Kantenlänge den 1.25-fachen Kugelradius. Es ergeben sich somit Würfel, in denen sich alle Punkte der Kandidaten zuzüglich ggf. vorhandener naher Umgebungspunkte befinden. In den Abbildungen 6.5a und 6.5b sind solche Würfel mit größeren Kantenlängen dargestellt.
Die Wahl eines, an den Koordinatenachsen ausgerichteten, Würfels stellt die rechen-effizienteste Methode zur Extraktion einer Subpunktwolke dar. Hier werden alle Punkte der Punktwolke gegen insgesamt sechs Koordinatenkomponenten getestet ($x_{\min}$, $x_{\max}$, $y_{\min}$, $y_{\max}$, $z_{\min}$, $z_{\max}$). Diese Methode ist deutlich effizienter als das Berechnen der Raumstrecken jedes Punktes zum genäherten Kugel-Zielzeichen-Zentrum und liefert nur unwesentlich mehr Umgebungspunkte. Alle segmentierten Punktwolken werden im nächsten Schritt zur Bestimmung der Näherungsparameter einer möglicherweise enthaltenden Kugel an die implementierte RANSAC-Funktion zur robusten Schätzung übergeben.

6.3 RANSAC - Näherungswerte der Kugelmittelpunkte

Die für die Bestimmung der Näherungswerte des jeweiligen Kugelmittelpunktes verwendeten Ansätze und Gleichungen sollen in diesem Abschnitt eingeführt und erläutert werden. Um die Näherungswerte möglichst recheneffizient und robust gegenüber ggf. vorhandenen Ausreißern bzw. Messpunkten der nahen Umgebung zur Kugel zu bestimmen oder den Extremfall keiner enthaltender Kugel aufzudecken, wird im Folgenden eine eindeutige Bestimmung der gesuchten Parameter durch vier zuvor zufällig ausgewählte Punkte beschrieben. Dies stellt die Grundidee der Robustheit von RANSAC dar [vgl. Förstner und Wrobel 2016, 153 ff.].
Mit der allgemeinen Kugelgleichung

$$(x - x_m)^2 + (y - y_m)^2 + (z - z_m)^2 = r^2 \tag{6.3.1}$$

ist die Kugel durch den Kugelmittelpunkt $\left[x_m, y_m, z_m\right]$ und ihren Radius r beschrieben [vgl. Zwillinger 2002, S. 238]. Ausmultipliziert erhält man

$$x^2 + y^2 + z^2 + ax + by + cz + d = 0 \tag{6.3.2}$$

mit vier Hilfsparametern

$$a = -2x_m; \quad b = -2y_m; \quad c = -2z_m; \quad d = x_m^2 + y_m^2 + z_m^2 - r^2. \tag{6.3.3}$$

Mit Hilfe der neu eingeführten Parameter $\left[a, b, c, d\right]$ lässt sich Gleichung 6.3.2 in die Form

$$ax + by + cz + d = -(x^2 + y^2 + z^2) \tag{6.3.4}$$

bringen. Nach Einsetzen aller vier Punkte auf der Kugeloberfläche ergeben sich folgende nun in den Hilfsparametern lineare Gleichungen

$$\begin{aligned} ax_1 + by_1 + cz_1 + d &= -(x_1^2 + y_1^2 + z_1^2) \\ ax_2 + by_2 + cz_2 + d &= -(x_2^2 + y_2^2 + z_2^2) \\ ax_3 + by_3 + cz_3 + d &= -(x_3^2 + y_3^2 + z_3^2) \\ ax_4 + by_4 + cz_4 + d &= -(x_4^2 + y_4^2 + z_4^2). \end{aligned} \tag{6.3.5}$$

Ein lineares Gleichungssystem mit vier Parametern und vier unabhängigen Gleichungen ist, bis auf besondere Szenarien der Konfiguration (vier Punkte auf einem Kreis führen zu Rangdefekt), eindeutig lösbar. Aus ((6.3.5)) ergeben sich in Matrix-Vektor-Schreibweise

$$\mathbf{A} = \begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{bmatrix}; \quad \mathbf{b} = \begin{bmatrix} -(x_1^2 + y_1^2 + z_1^2) \\ -(x_2^2 + y_2^2 + z_2^2) \\ -(x_3^2 + y_3^2 + z_3^2) \\ -(x_4^2 + y_4^2 + z_4^2) \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}. \tag{6.3.6}$$

Nach der Lösung des linearen Gleichungssystems $\mathbf{Ax} = \mathbf{b}$ lassen sich die Kugelparameter mit

$$\mathbf{K} = \begin{bmatrix} x_m \\ y_m \\ z_m \\ r \end{bmatrix} = \begin{bmatrix} -a/2 \\ -b/2 \\ -c/2 \\ \sqrt{x_m^2 + y_m^2 + z_m^2 - d} \end{bmatrix} \tag{6.3.7}$$

berechnen.
Um nun mit dem beschriebenen Modell der Kugelberechnung mittels RANSAC optimale Näherungswerte bestimmen zu können sind folgend der allgemeine Ansatz des RANSAC sowie eine modifizierte Variante zu Kugelschätzung erläutert.

Abbildung 6.3: Prinzip des RANSAC am Beispiel einer Geraden durch zwei Punkte, optimale Lösung mit größtem Support (maximale Anzahl von Inlier Punkten in Epsilon-Umgebung)

In Abbildung 6.3 soll aus den gegebenen verrauschten Punkten eine Gerade geschätzt werden. Hierzu sind die Geradengleichung, ein Schwellwert als minimale Anzahl der Punkte, welche in einer definierten Toleranzumgebung liegen und die Anzahl der maximal durchzuführenden Iterationen bekannt. Analog zum Algorithmus im Pseudocode 4 werden jeweils zwei Punkte aus der Menge aller Punkte zufällig gewählt. Durch diese wird die Gerade eindeutig bestimmt. Nach Berechnung der Abstände aller verbleibenden Punkte zur Gerade, werden alle Punkte innerhalb der definierten Toleranz als Inlier und alle Punkte außerhalb dieser Umgebung als Outlier klassifiziert. Ist der zuvor definierte Schwellwert mit der Inlier-Menge der aktuellen Iteration erreicht, wird abschließend eine Schätzung mit allen Inlier-Punkten zu Bestimmung der endgültigen Geradenparameter durchgeführt. Andernfalls werden zwei neue Punkte ausgewählt und alle beschriebenen Schritte wiederholt.

Algorithm 4: RANSAC Algorithmus [vgl. Derpanis 2010]

input : Punktset $\mathbf{l}$, Modell $\mathbf{M}$, Toleranz ϵ, Schwellwert t, maximale Anzahl Iterationen N

output: Parameter $\mathbf{p}$

```
Wähle zufällig genau so viele Punkte aus l, damit das Modell M eindeutig lösbar ist.
Berechne die eindeutigen Modellparameter p.
n ← Zähle alle Punkte l_in, welche innerhalb der definierten Umgebung ϵ zum Modell
  liegen.
Teste, ob Schwellwert t erreicht ist.
if n > t then
    Schätze die Modellparameter mit allen Inliern l_in endgültig.
else
    wiederhole 1-4 jedoch maximal N mal.
end
```

Im Rahmen der Bestimmung von Näherungswerten der Kugelmittelpunkte sowie der Klassifikation der Punkte im Ausschnitt als Kugel-Punkte oder nicht Kugel-Punkte für die abschließende Parameterschätzung im Gauß-Helmert-Modell (vgl. Abschnitt 7.1 und 7.2) hat sich ein modifiziertes Kriterium zur Konsistenzprüfung bewährt. Generell sind verschiedenste Kriterien zur Konsistenzprüfung der verbleibenden Punkte mit dem vorgegebenen Modell denkbar [vgl. Förstner und Wrobel 2016, S. 153]. Das angewandte Kriterium besteht aus drei Teilen. Im ersten Schritt wird überprüft, ob der ermittelte Radius in der aktuellen Iteration dem vorgegebenen Zielzeichenradius in einem

(a) (b)

Abbildung 6.4: Visualisierung der Outlier in unmittelbarer Umgebung zum Kugelkandidaten, (a) rot schraffierte Fläche lediglich im Bereich des Magnetfußes des Zielzeichens, (b) schematische Darstellung eines falsch vermuteten Kugelkandidaten an einem zylindrischen Objekt (z.B. Laternenmast) mit großen Teilflächen im Umfeld des Kandidaten

definierten Spielraum entspricht. Ist das der Fall, so werden die orthogonalen Abstände aller Punkte aus dem Punktset zur Kugeloberfläche der aktuellen Iteration gebildet. Hierzu wird erneut der bekannte Radius gewählt. Die Anzahl der ermittelten Inlier bilden zusammen mit dem dritten Teilkriterium die Grundlage der Validierung, ob im gegebenen Punktset eine Kugel existiert oder nicht. Wie in Abbildung 6.4 dargestellt ist, wird im letzten Schritt die Anzahl der im unmittelbaren Umfeld zur vermuteten Kugel liegenden Outlier bestimmt. Ergibt sich ein zu großes Verhältnis von Inliern und Umfeldpunkten wird die Lösung der jeweiligen Iteration verworfen. Überschreitet das ermittelte Verhältnis den Schwellwert nicht, ist der aktuelle Kandidat mit der Anzahl seiner Inlier als Gütekriterium so lange der beste Kandidat, bis es ein Parameterset mit größerer Inlier-Anzahl ablöst.

Die in Abschnitt 6.3 dargestellte Methode zur Validierung und robusten Bestimmung der Näherungsparameter wurde unter Berücksichtigung der in Abschnitt 2.3 beschriebenen und in den Abbildungen 2.10 sowie 2.11 dargestellten Besonderheiten der Selektion per Öffnungswinkel implementiert. In Abbildung 6.5 sind exemplarisch zwei Subpunktwolken nach der Bestimmung der Inlier-Punkte durch die beschriebene Methode dargestellt. Die selektierten Inlier-Punkte der Kugeln sowie deren Näherungsparamter bilden die Grundlage der Schätzung im Gauß-Helmert-Modell, die eine möglichst genaue und zuverlässige Schätzung des Mittelpunktes liefern soll.

Algorithm 5: modifizierter RANSAC zur robusten Kugelsuche-/schätzung

input : Punktset $\mathbf{l}$, Kugel-Modell $\mathbf{M}$, Zielzeichenradius R, Toleranz ϵ, Schwellwert t, maximale Anzahl Iterationen N

output: Kugel ja/nein isSphere, (opt.) Kugelparameter $\mathbf{p}$

```
Wähle zufällig 4 Punkte aus l.
Berechne die eindeutigen Kugelparameter p = [x_m, y_m, z_m, r].
Teste, ob aktueller Radius r zum gegebenen Zielzeichenradius R passt.
if |r − R| < ϵ then
    n ← Zähle alle Punkte l_in, welche innerhalb der definierten Umgebung ϵ zur
      Kugeloberfläche mit Radius R liegen.
    n_near ← Zähle alle Punkte l_near, welche innerhalb einer 10 mm Umgebung zur
      Kugeloberfläche liegen.
    if n_near/n < t then
        if n > n_opt then
            Ersetze optimale Parameter durch aktuelle Parameter.
            isSphere ← true
        end
end
Wiederhole alle Schritte N mal.
if isSphere != true then
    isSphere ← false // Es konnte nach N Iterationen keine Kugel gefunden werden.
end
```

(a) (b)

Abbildung 6.5: Inlier Selektion nach Anwendung des RANSAC auf zwei Subpunktwolken

Kapitel 7

Schätzung der Zielzeichenkoordinate

Die zu den endgültig validierten Kugel-Kandidaten gehörenden Inlier-Punkte der einzelnen Subpunktwolken dienen in diesem Kapitel als Grundlage der bestmöglichen Schätzung der Passpunktkoordinaten, die durch den Kugelmittelpunkt realisiert sind. In den folgenden Abschnitten sind die Methoden der Ausgleichung sowie deren Implementierung und Laufzeit-Optimierung dargestellt. Ziel dieser Prozessierung sind die endgültigen Passpunktkoordinaten der zuvor detektierten Kugel-Kandidaten.

7.1 Gauß Helmert Modell der Kugel mit freiem Radius

Eine Kugel lässt sich wie in (6.3.1) bereits gezeigt durch ihren Mittelpunkt und ihren Radius beschreiben. Für den Ausgleich im Gauß-Helmert-Modell ergibt sich daraus folgender funktionaler Zusammenhang

$$F = (x - x_m)^2 + (y - y_m)^2 + (z - z_m)^2 - r^2 \tag{7.1.1}$$

mit dessen Hilfe sich die Bedingungsmatrix B und die Designmatrix A aufstellen lassen [vgl. Niemeier 2008, S. 173–178]. In $\mathbf{B}$ wird der funktionale Zusammenhang für jede Beobachtung $l_i = [x_i, y_i, z_i]$ nach den drei Beobachtungsgrößen x, y, z abgeleitet. Die Bedingungsmatrix setzt sich als Blockmatrix aus den drei Diagonalmatrizen $\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z}$ zusammen

$$\mathbf{B} = \begin{bmatrix} \frac{\partial F}{\partial x_1} & 0 & \dots & 0 & \frac{\partial F}{\partial y_1} & 0 & \dots & 0 & \frac{\partial F}{\partial z_1} & 0 & \dots & 0 \\ 0 & \frac{\partial F}{\partial x_2} & & \vdots & 0 & \frac{\partial F}{\partial y_2} & & \vdots & 0 & \frac{\partial F}{\partial z_2} & & \vdots \\ \vdots & & \ddots & 0 & \vdots & & \ddots & 0 & \vdots & & \ddots & 0 \\ 0 & \dots & 0 & \frac{\partial F}{\partial x_n} & 0 & \dots & 0 & \frac{\partial F}{\partial y_n} & 0 & \dots & 0 & \frac{\partial F}{\partial z_n} \end{bmatrix}_{n \times 3n} . \tag{7.1.2}$$

Analog wird der funktionale Zusammenhang zur Aufstellung der Designmatrix $\mathbf{A}$ nach den Parametern $\mathbf{x} = [x_m, y_m, z_m, r]$ abgeleitet

$$\mathbf{A} = \begin{bmatrix} \frac{\partial F_1}{\partial x_m} & \frac{\partial F_1}{\partial y_m} & \frac{\partial F_1}{\partial z_m} & \frac{\partial F_1}{\partial r} \\ \frac{\partial F_2}{\partial x_m} & \frac{\partial F_2}{\partial y_m} & \frac{\partial F_2}{\partial z_m} & \frac{\partial F_2}{\partial r} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial F_n}{\partial x_m} & \frac{\partial F_n}{\partial y_m} & \frac{\partial F_n}{\partial z_m} & \frac{\partial F_n}{\partial r} \end{bmatrix}_{n \times 4}$$

$$= \begin{bmatrix} \frac{x_1 + v_{x_1} - x_m^0}{r_1^0} & \frac{y_1 + v_{y_1} - x_m^0}{r_1^0} & \frac{z_1 + v_{z_1} - x_m^0}{r_1^0} & 1 \\ \frac{x_2 + v_{x_2} - x_m^0}{r_2^0} & \frac{y_2 + v_{y_2} - x_m^0}{r_2^0} & \frac{z_2 + v_{z_2} - x_m^0}{r_2^0} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ \frac{x_n + v_{x_n} - x_m^0}{r_n^0} & \frac{y_n + v_{y_n} - x_m^0}{r_n^0} & \frac{z_n + v_{z_n} - x_m^0}{r_n^0} & 1 \end{bmatrix}_{n \times 4} . \tag{7.1.3}$$

Es folgt die Berechnung des Widerspruchsvektors $\mathbf{w}$

$$\mathbf{w} = -\mathbf{B} \cdot \mathbf{v} + \mathbf{F}(\mathbf{l} + \hat{\mathbf{v}}, \hat{\mathbf{x_0}}), \tag{7.1.4}$$

wobei $\mathbf{v}$ die Verbesserungen der Beobachtungen sind (in erster Iteration $\mathbf{v} = \mathbf{0}$). Die Kovarianzmatrix der Beobachtungen ist eine Diagonalmatrix mit den Punktgenauigkeiten $\sigma_x = \sigma_y = \sigma_z = \sigma_{xyz}$

$$\boldsymbol{\Sigma_{ll}} = \begin{bmatrix} \sigma_{xyz}^2 & & & \\ & \sigma_{xyz}^2 & & \\ & & \ddots & \\ & & & \sigma_{xyz}^2 \end{bmatrix}_{n \times n} . \tag{7.1.5}$$

Es entsteht die Blockmatrixform der Normalgleichungsmatrix, sowie deren rechte Seite

$$\mathbf{N} = \begin{bmatrix} \mathbf{B \Sigma B^T} & \mathbf{A} \\ \mathbf{A^T} & \mathbf{0} \end{bmatrix} \qquad \mathbf{n} = \begin{bmatrix} -\mathbf{w} \\ \mathbf{0} \end{bmatrix}. \tag{7.1.6}$$

Aus der Lösung des Gleichungssystems

$$\begin{bmatrix} \tilde{\mathbf{k}} \\ \tilde{\mathbf{x}} \end{bmatrix} = \mathbf{N}^{-1}\mathbf{n} \tag{7.1.7}$$

ergeben sich die Parameterzuschläge der jeweiligen Iteration. Die ausgeglichenen Parameter ergeben sich aus

$$\tilde{\mathbf{x}} = \hat{\mathbf{x}} + \tilde{\mathbf{x}}, \tag{7.1.8}$$

wobei $\hat{\mathbf{x}}$ und $\tilde{\mathbf{x}}$ aus der vorherigen Iteration beziehungsweise in erster Iteration als Näherungsparameter bekannt sind. Mit $\tilde{\mathbf{k}}$ können nun die Verbesserungen $\mathbf{v}$ der Beobachtungen berechnet werden

$$\mathbf{v} = \boldsymbol{\Sigma} \mathbf{B^T} \tilde{\mathbf{k}}. \tag{7.1.9}$$

Die für die erste Iteration benötigten Näherungswerte $\mathbf{x^0} = [x_m^0, y_m^0, z_m^0, r^0]$ für den Kugelmittelpunkt sowie deren Radius werden aus den Ergebnissen des RANSAC aus Abschnitt 6.3 übernommen und stellen vergleichsweise gute, vor allem aber robuste Näherungswerte dar.

7.2 Gauß Helmert Modell der Kugel mit fixiertem Radius

Soll der Kugelradius nicht mitgeschätzt werden, sondern als bekannt und fixiert in die Ausgleichung eingeführt werden, so ergeben sich folgende Veränderungen im Vergleich der Gleichungen aus Abschnitt 7.1. Ein bekannter Radius liegt insbesondere bei den eingesetzten Kugel-Zielzeichen vor und sollte nach den Überlegungen aus Abschnitt 2.3 zwingend festgehalten werden.
Die Bedingungsmatrix lässt sich unverändert wie in (7.1.2) dargestellt aufstellen. Der Parametervektor ergibt sich ohne den zu schätzenden Radius zu $\mathbf{x} = [x_m, y_m, z_m]$. Daraus folgt die um eine Spalte verkürzte Designmatrix

$$\mathbf{A} = \begin{bmatrix} \frac{\partial F_1}{\partial x_m} & \frac{\partial F_1}{\partial y_m} & \frac{\partial F_1}{\partial z_m} \\ \frac{\partial F_2}{\partial x_m} & \frac{\partial F_2}{\partial y_m} & \frac{\partial F_2}{\partial z_m} \\ \vdots & \vdots & \vdots \\ \frac{\partial F_n}{\partial x_m} & \frac{\partial F_n}{\partial y_m} & \frac{\partial F_n}{\partial z_m} \end{bmatrix} = \begin{bmatrix} \frac{x_1 + v_{x_1} - x_m^0}{r_1^0} & \frac{y_1 + v_{y_1} - x_m^0}{r_1^0} & \frac{z_1 + v_{z_1} - x_m^0}{r_1^0} \\ \frac{x_2 + v_{x_2} - x_m^0}{r_2^0} & \frac{y_2 + v_{y_2} - x_m^0}{r_2^0} & \frac{z_2 + v_{z_2} - x_m^0}{r_2^0} \\ \vdots & \vdots & \vdots \\ \frac{x_n + v_{x_n} - x_m^0}{r_n^0} & \frac{y_n + v_{y_n} - x_m^0}{r_n^0} & \frac{z_n + v_{z_n} - x_m^0}{r_n^0} \end{bmatrix}_{n \times 3} . \tag{7.2.1}$$

Der fixierte Radius R geht mit

$$F = (x - x_m)^2 + (y - y_m)^2 + (z - z_m)^2 - R^2 \tag{7.2.2}$$

in den Widerspruchsvektor $\mathbf{w}$

$$\mathbf{w} = -\mathbf{B} \cdot \mathbf{v} + \mathbf{F}(\mathbf{l} + \hat{\mathbf{v}}, \hat{\mathbf{x_0}}), \tag{7.2.3}$$

ein. Analog zu Abschnitt 7.1 folgt der iterative Ausgleich nun mit fixiertem Radius der Kugel.

Abbildung 7.1: Abweichungen bei zufälliger Auswahl von verschieden vielen Punkten zur Schätzung der Passpunktkoordinaten

7.3 Algorithmisches Vorgehen & Effizienzsteigerung

Wie in mehreren Tests gezeigt werden konnte, stellt eine zufällige Ausdünnung der Messpunkte auf der Kugeloberfläche eine wesentliche Laufzeitoptimierung der Ausgleichung im GHM dar. Hierzu wurden nach Anwendung der Inlier-Selektion per RANSAC (vgl. Abschnitt 6.3), sowie der in Abschnitt 2.3 beschriebenen Selektion per Öffnungswinkel jeweils 500 Messpunkte zur endgültigen Schätzung der Parameter zufällig ausgewählt. In Abbildung 7.1 sind die Abweichungen der Schätzergebnisse in Abhängigkeit von der Anzahl der zufällig gewählten Punkte dargestellt. Als Referenz dient das jeweilige Schätzergebnis mit der nicht ausgedünnten (aber durch den RANSAC ausreißerbereinigten) Subpunktwolke. Es ist deutlich erkennbar, dass ab ca. 500 Punkten seltenst eine Abweichung größer 0.2 mm auftritt. Eine Größe, welche mit Blick auf die Messgenauigkeit der Panorama-Laserscanner, sowie den geforderten Genauigkeiten als nicht signifikant einzustufen ist. Gerade im äußersten Nahbereich stellt die Variante großes Optimierungspotenzial der benötigten Rechenzeiten im iterativen Ausgleich dar. So kann bei einer Kugelentfernung von ca. 1.3 m die Punktanzahl von über 35 000 auf unter 2 % (500 Punkte) verringert werden.
Weiter wurde die Kovarianzmatrix der Beobachtungen als Einheitsmatrix angenommen. Diese Vereinfachung der stochastischen Annahmen zu den gemessenen Punkten auf der Oberfläche der Kugeln führt ebenfalls zu einer Optimierung der Laufzeiten im Ausgleich. Durch die Segmentierung per Öffnungswinkel werden Punkte mit großem Einfallswinkel gänzlich aus der Schätzung ausgeschlossen, was eine aufwändigere Gewichtung nach den Intensitäten bzw. Einfallswinkeln auf der Kugel überflüssig macht.
Implementiert sind, anders als im Methoden-Teil der Arbeit dargestellt (vgl. Gleichung 7.1.7), keine Invertierungen der Normalgleichungsmatrix. Durch effiziente Rechenverfahren wie das Ausnutzen bekannter Symmetrie-Eigenschaften, die Cholesky-Zerlegung sowie das Lösen durch Rückwärts- und Vorwärtseinsetzen konnten die ursprünglich aufwändigen Berechnungen weiter optimiert werden. Hierzu wurden Subroutinen der LAPACK Bibliothek mit OpenBLAS in eine Matrix-Klassen-Implementation integriert (vgl. Tabelle 7.1). Somit werden die parallelen Mehrkern-Prozessierungen in OpenBLAS genutzt.

LAPACK Name	Funktion aus Matrix-Klasse
`dsyrk`	`Matrix::isSymmetricProductOf`
`dtrmm`	`Matrix::dtrmm`
`dgemm`	`Matrix::isProductOf`
`dpotrf`	`Matrix::choli`
`dpotrs`	`Matrix::solveWithCholReduced`
`dtrsm`	`Matrix::dtrsm`

Tabelle 7.1: LAPACK Subroutinen

Abbildung 7.2: Differenzen der im Ausgleich geschätzten Radien zum bekannten Radius der Kugel-Zielzeichen

7.4 Evaluierung der Methoden zur Schätzung der Kugel-Parameter

Eine Evaluierung der in den Abschnitten 7.1 und 7.2 beschriebenen Methoden zur bestmöglichen Schätzung der Kugel-Mittelpunkte ist hier anhand der Abweichungen zum bekannten Radius der Kugel-Zielzeichen realisiert (vgl. Abbildung 7.2). Bei der in Abschnitt 7.1 beschriebenen Methode zur Schätzung der Parameter inklusive des Radius fällt auf, dass sich in einigen Szenarien bis zu 2 mm große Differenzen zum bekannten Radius der Zielzeichen ergeben. Das war zu erwarten, da ggf. auftretende Effekte an der Kugeloberfläche, welche zu Stauchungen bzw. Aufweitungen der Kugel führen, durch den bekannten Radius in der Ausgleichung unterdrückt werden und durch das gezielte Segmentieren und Auslassen der Randpixel weiter minimiert wurden. Die teilweise bspw. durch den Kometenschweif systematisch verfälschten Messpunkte am Rand der Kugel-Zielzeichen (vgl. Abbildungen 2.3, 2.9) und die teilweise deutlich ungenaueren Kugel-Randpunkte mit geringen Intensitäten werden jedoch wichtig, sobald der Radius mitgeschätzt werden soll. Eben diese Punkte stützen die Kugelgeometrie wesentlich und können hier nur aufgrund des bekannten Radius der Zielzeichen vernachlässigt werden.

Kapitel 8

Die Registrierung

Nachdem die Kugelmittelpunkt-Koordinaten abschließend im Gauß-Helmert-Modell geschätzt wurden, folgt die Transformation der lokalen Passpunkte auf identische Punkte im übergeordneten Koordinatensystem. Hierzu ist die Zuordnung der detektierten Zielzeichenkoordinaten zu den Passpunkten in den übergeordneten Systemen notwendig. Die Umsetzung der Identifizierung gleicher Punkte sowie die Transformationen ggf. zuvor nicht detektierter Zielzeichen im Bild sind in den nächsten Abschnitten beschrieben.

8.1 2.5 dimensionale Helmert-Transformation

Sollten nicht alle N in einer Punktwolke vorhandenen Kugel-Zielzeichen durch die Ansätze aus Kapitel 5 detektiert worden sein, werden die globalen Passpunktkoordinaten in den lokalen Standpunkt transformiert und auf Kugel-Zielzeichen geprüft. Für die Rücktransformation müssen die Transformationsparameter des übergeordneten Systems in das jeweilige lokale Koordinatensystem des Scanners berechnet werden. Die allgemeine 3D-Transformation kann durch bekannte Eigenschaften der beiden Koordinatensysteme auf eine 2.5 dimensionale Helmert-Transformation vereinfacht werden. Die Ausrichtung der z-Achse ist in beiden Systemen gleich, da der Laserscanner vor Beginn der Messung horizontiert wurde. Die zu bestimmenden Transformationsparamter beschränken sich somit auf $[t_x, t_y, t_z, \alpha, m]$, wobei t_x, t_y und t_z die Translationen in der Lage und Höhe darstellen. Der Höhenoffset t_z wird in einem eigenen vermittelnden Ausgleich unabhängig von der 2D-Transformation der Lage bestimmt. Der Winkel α beschreibt den Drehwinkel zwischen den beiden Koordinatensystemen um die z-Achse. Gerade unter Berücksichtigung der oftmals nicht verzerrungsfreien übergeordneten Koordinatensysteme wird ein Maßstab m für jeden Standpunkt mitgeschätzt.
Die Berechnung erfolgt in zwei voneinander unabhängigen Schritten.

2D-Helmert-Transformation

Die Transformationsparameter $[t_x, t_y, \alpha, m]$ der ebenen Transformation lassen sich mit den Beobachtungsgleichungen

$$\begin{aligned} X_i + v_{X_i} &= t_x + x_i \cdot a - y_i \cdot b \\ Y_i + v_{Y_i} &= t_y + y_i \cdot a + x_i \cdot b \end{aligned} \tag{8.1.1}$$

in welchen die Hilfsparameter $a = m \cdot \cos\alpha$ und $b = m \cdot \sin\alpha$ die jeweilige Auswirkung des Maßstabes auf die Koordinatenachsen beschreiben und somit den Drehwinkel α und den Maßstab m enthalten und folgendem Ansatz im Gauß-Markov-Modell bestimmen [vgl. Caspary und Beineke 2003, S. 242]. Die Designmatrix ergibt sich nach den partiellen Ableitungen der Beobachtungsgleichungen (8.1.1)

nach den Parametern $\mathbf{x} = [t_x, t_y, a, b]$ zu

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & x_1 & -y_1 \\ 0 & 1 & y_1 & x_1 \\ 1 & 0 & x_2 & -y_2 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & y_n & z_n \end{bmatrix}_{n\times 4} . \tag{8.1.2}$$

Der Beobachtungsvektor $\mathbf{l}$ sowie die zugehörigen Verbesserungen v_i lassen sich mit

$$\begin{aligned} \mathbf{l} &= \left[X_1, Y_1, X_2, \dots, Y_n\right]^T \\ \mathbf{v} &= \left[v_{X_1}, v_{Y_1}, v_{X_2}, \dots, v_{Y_n}\right]^T \end{aligned} \tag{8.1.3}$$

aufstellen. Nach Lösung des bekannten Gleichungssystems $\mathbf{l} + \mathbf{v} = \mathbf{A}\mathbf{x}$ ergeben sich die Transformationsparameter t_x und t_y unmittelbar, sowie die verbleibenden Parameter m und α über die Hilfsparameter a und b jeweils aus

$$\begin{aligned} m &= \sqrt{a^2 + b^2} \\ \alpha &= \arctan\left(\frac{b}{a}\right). \end{aligned} \tag{8.1.4}$$

1D-Höhentransformation

Analog zu Abschnitt 8.1 lässt sich auch der Translationsparameter t_z als Höhenoffset bestimmen. Nach Reduzierung der beiden Punktsätze um ihre mittleren Höhen $\bar{z}$ im lokalen System und $\bar{Z}$ im globalen System

$$\bar{z} = \frac{1}{n}\sum_{i=1}^{n} z_i, \qquad \bar{Z} = \frac{1}{n}\sum_{i=1}^{n} Z_i \tag{8.1.5}$$

ergeben sich der Beobachtungsvektor $\mathbf{l}$ sowie die Designmatrix $\mathbf{A}$ wie folgt

$$\begin{aligned} \mathbf{l} &= \left[z_1 - \bar{z}, z_2 - \bar{z}, \dots, z_n - \bar{z}\right]^T \\ \mathbf{A} &= \left[Z_1 - \bar{Z}, Z_2 - \bar{Z}, \dots, Z_n - \bar{Z}\right]^T. \end{aligned} \tag{8.1.6}$$

Der Translationsparameter t_z ergibt sich nach Lösung des Gleichungssystems $\mathbf{l} + \mathbf{v} = \mathbf{A}\mathbf{x}$ und der Addition der zuvor subtrahierten mittleren Höhe.

8.2 Identifizierung von globalen Passpunkten

Zu Beginn der eigentlichen Registrierung müssen die Punktnummern (IDs) der lokalen Passpunkte im übergeordneten System ermittelt werden. Hierzu werden jeweils alle möglichen Kombinationen aus den lokalen sowie globalen Passpunkten gebildet und deren räumliche Entfernung zueinander berechnet. Gleiche Punktpaare weisen in beiden Systemen gleiche Entfernungen auf. Nach dem Abgleich mit allen weiteren Punktpaaren ergeben sich eindeutige Zuordnungen zwischen lokalen und globalen IDs. Ein Punktset bestehend aus den lokalen Kugel-Zielzeichen-Mittelpunkten sowie den übergeordneten Netzkoordinaten liegt damit vor und bildet die Grundlage für die in Abschnitt 8.1 beschriebene 2.5D-Helmert-Transformation. Abschließend kann die Zuordnung der jeweiligen Passpunkte durch die Interpretation der Transformationsparameter sowie durch die berechneten Verbesserungen (vgl. (8.1.1)) validiert werden. Die Auswertungen der Testdaten in dieser Arbeit lieferte in keinem Fall falsch zugeordnete Punkte. Theoretisch ist dies jedoch möglich, speziell in Szenarien, in welchen sich lokale Geometrien im globalen Passpunktfeld wiederholen. Ein Beispiel könnte hier ein schlauchförmiges Netz in Tunnelröhren oder entlang von Bahntrassen sein.

(a) (b)

Abbildung 8.1: schematische Darstellung der Passpunktsituation mit paarweiser Distanzbildung; (a) lokale Passpunkte; (b) globale Passpunktsituation und identische Distanzen zur Punktzuordnung

id	x	y	z	ID	X	Y	Z
1	-8.2938	1.1650	-0.2043	8122	2577567.1541	5622382.6200	59.0855
2	-3.3461	5.1970	-0.2279	8125	2577561.3773	5622385.3360	59.0609
3	0.6908	3.6856	-0.5028	8126	2577560.9947	5622389.6280	58.7866
4	4.1680	-3.1584	-0.9635	8123	2577565.6613	5622395.7235	58.3247
5	-7.6885	-5.5364	-1.7073	8119	2577572.9328	5622386.0642	57.5785

Tabelle 8.1: Beispiel eines Punktsets zur Transformation der lokalen und globalen Passpunkte

Abbildung 8.2: Detektierte Kugel-Zielzeichen im Bild und Anzahl platzierter Kugel-Zielzeichen pro Datensatz

Abbildung 8.3: endgültig gefundene Kugel-Zielzeichen und Anzahl platzierter Kugel-Zielzeichen pro Datensatz

8.3 Rücktransformation weiterer Kandidaten und Registrierung

Werden in einem Datensatz weniger als die bekannte Anzahl an Kugel-Zielzeichen mit der in Abschnitt 5.2 dargestellten Methode detektiert, folgt eine Rücktransformation der globalen Passpunkte in das lokale Koordinatensystem der aktuellen Punktwolke. In Abbildung 8.2 ist erkennbar, dass in allen 33 Testpunktwolken mindestens vier der maximal sechs platzierten Zielzeichen erkannt wurden. Analog zu dem in Abschnitt 6.2 beschriebenen Verfahren werden für alle Passpunkte, welche nach der Transformation in das lokale System des Scanners nicht weiter als 25 m entfernt sind, neue Subpunktwolken generiert. Befinden sich tatsächlich Messpunkte in den neu erzeugten Würfeln (vgl. Abbildung 6.2), so wird mittels der RANSAC geprüft, ob ein Kugel-Zielzeichen vorliegt. Alle weiteren Schritte werden analog zu den bisher detektierten Zielzeichen ausgeführt. Nachdem alle transformierten Passpunkte im lokalen System auf möglicherweise vorhandene Zielzeichen geprüft wurden, ergab sich eine Quote gefundener Kugeln von 100 % (vgl. Abbildung 8.3).
Nachdem die Transformationsparameter endgültig mit allen vorhandenen Kugel-Zielzeichen bestimmt wurden, kann die gesamte Punktwolke transformiert bzw. im geodätischen Sinne registriert werden.

Kapitel 9

Evaluierung der Zielzeichendetektion an ausgewählten Projekten

In diesem Kapitel wird der vorgeschlagene Workflow und die Implementierung aus Abbildung 3.1 anhand drei ausgewählter Punktwolken evaluiert. Die in Abschnitt 2.4 aufgeführten Datensätze stammen aus drei unterschiedlichen Projekten. Die Evaluierung der Software mit Punktwolken aus allen drei Projekten ist sinnvoll, da sich die Umgebungsbedingungen an einzelnen Standpunkten zwischen den Projekten stark unterscheiden (vgl. Tabelle 9.1) und typische Szenarien abbilden. In den folgenden drei Abbildungen werden die gewählten Punktwolken durch ihre Intensitätsbilder repräsentiert. Lediglich im Intensitätsbild der Punktwolke im Kanalsystem gelingt es durch visuelle Betrachtung die sechs platzierten Kugel-Zielzeichen zu finden. Hier wird besonders deutlich, dass die verschieden weit entfernten Zielzeichen unterschiedlich groß erscheinen. Wie bereits in Abschnitt 2.4 beschrieben wurde, unterscheiden sich die Punktwolken aus den Abbildungen 9.1, 9.2 und 9.3 wesentlich durch den Grad der Himmelsfreiheit, die Objektdichte in naher Umgebung sowie in den unterschiedlichen Distanzen zu den platzierten Kugel-Zielzeichen. In grün sind alle Kugel-Zielzeichen markiert, die bereits mittels Suche im Bild detektiert wurden. Das gelingt für 15 der insgesamt 17 platzierten Kugel-Zielzeichen. In gelb markiert ist in Abbildung 9.1 ein Kugel-Zielzeichen, welches nicht als Kugel-Kandidat aus der Suche im Bild resultierte, jedoch nach Rücktransformation korrekt gefunden wurde. In dieser Punktwolke konnten somit abschließend alle 5 platzierten Zielzeichen detektiert werden. Lediglich in einem Datensatz der insgesamt 34 wurde ein platziertes Kugel-Zielzeichen endgültig nicht erkannt.
Das in rot markierte Zielzeichen aus Abbildung 9.3 befindet sich in 21.45 m Entfernung zum Laserscanner. Es wurde mittels Magnethalterung in der oberen rechten Ecke eines stark reflektierenden Andreas-Kreuzes des Bahnübergangs angebracht (vgl. Abbildung 9.4a). Neben der großen Entfernung zu diesem Zielzeichen stellt auch die nahe Oberfläche des Andreas-Kreuzes, welche so stark reflektiert, dass leere Bereiche in der Punktwolke entstehen ein Problem dar. Die ca. 70 Punkte, welche auf der Kugel-Oberfläche liegen, weisen stark streuende Intensitäten sowie viele durch den Kometenschweif systematisch falsche Messpunkte auf (vgl. Abbildung 9.4b). Nach Anwendung des Kugel-Fit Tools in CloudCompare ergibt sich für die segmentierten Punkte (ohne Kometenschweif-Pixel) ein geschätzter Radius der Kugel von 63.5 mm, welcher um 9 mm vom bekannten Radius der Zielzeichen abweicht. Diese durch die ungünstigen Umgebungsbedingungen des Zielzeichens sowie der großen Distanz resultierende Verkleinerung der Kugel führt in der Validierung per RANSAC zum Ausschluss des Kandidaten. Tatsächlich stellt dieses Zielzeichen mit den wenigen, teils stark verfälschten Messpunkten im Sinne einer Anschlusspunkt-Qualität ohnehin keinen für die Transformation nutzbaren Passpunkt dar. Durch den Ausschluss mittels abschließender Validierung in der Punktwolke werden somit schlecht repräsentierte Zielzeichen ignoriert, was zur Verbesserung der geschätzten Transformationsparameter und somit zur Verbesserung der Registrierung führt.
In Abbildung 9.1 ist in gelb das am Fuß einer Werbetafel angebrachte Kugel-Zielzeichen markiert, welches durch die Algorithmen der bildbasierten Suche (vgl. Kapitel 5) nicht detektiert wurde.

Abbildung 9.1: Intensitätsbild der Punktwolke am Bonner Hbf, grün: im Bild gefundene ZZ, gelb: nach Rücktransformation gefundenes ZZ

Abbildung 9.2: Intensitätsbild einer Punktwolke im Kanalsystem, grün: im Bild gefundene ZZ

Abbildung 9.3: Intensitätsbild einer Punktwolke am Bahnübergang, grün: im Bild gefundene ZZ, rot: endgültig nicht erkanntes Kugel-Zielzeichen

Punktwolke	Objektdichte	Himmelsfreiheit	Kugel-Zielzeichen [#, Distanzen]	Datensatzgröße [# Punkte, MB]
Hbf	sehr hoch	sehr gering	5, 3.78 m - 9.63 m	34 948 134, 559.2
Kanalsystem	gering	nicht vorhanden	6, 0.61 m - 8.80 m	38 323 027, 613.2
Bahnübergang	normal	sehr groß	6, 9.09 m - 21.45 m	20 176 029, 322.8

Tabelle 9.1: Eigenschaften der zur Evaluierung genutzten Punktwolken

(a) Umgebung des ZZ in 21.45 m Entfernung, Datenlücken auf der Oberfläche des Andreas-Kreuzes

(b) Ausschnitt der nahen Umgebung des ZZ mit deutlich erkennbarem Kometenschweif

(c) Kugel-Fit in CloudCompare mit 68 Punkten auf der Kugel, Radius aus Fit: 63.5 mm

Abbildung 9.4: nicht gefundenes Kugel-Zielzeichen

Durch die Abschattung eines im Vordergrund der Kugel aufgestellten Bauzauns (vgl. Abbildung 9.5a) und die sich daraus ergebenen Datenlücken (vgl. Abbildung 9.5b) auf der Kugeloberfläche erfüllen nicht alle Rand- und Umgebungspixel aus dem vorgestellten Algorithmus (vgl. Abschnitt 5.2 und Abbildung 5.7) die berechneten Kriterien. So sind Pixel in der nahem Umgebung des gesuchten Zentrums-Pixel nicht vorhanden (vgl. Abbildung 9.6). Auch die Validierung der Randpixel gelingt für einige der abgetasteten Bereiche nicht. Nach der Rücktransformation der globalen Passpunktkoordinaten und der erneuten Validierung der Subpunktwolke per RANSAC wird das platzierte Kugel-Zielzeichen jedoch zuverlässig detektiert. Die 316 Messpunkte auf der Kugel-Oberfläche repräsentieren nach einem Kugel-Fit in CloudCompare einen Kugel-Radius von 72.4 mm, welcher den bekannten Radius der Kugel-Zielzeichen bis auf einen Zehntelmillimeter widerspiegelt (vgl. Abbildung 9.5c).
Eine Evaluierung der geschätzten Kugel-Parameter (vgl. Abschnitt 7.1 und 7.2) erfolgte durch die Betrachtung der Restklaffen nach Transformation auf vorhandene Passpunkte im übergeordneten System. In den Abbildungen 9.7 und 9.8 sind exemplarisch die Restklaffen für beide Varianten des Ausgleiches dargestellt. Für beide Standpunkte ist festzuhalten, dass die Verbesserungen grundsätzlich im Rahmen der geforderten Genauigkeiten und kleiner als zu erwarten ausfallen. In wenigen Fällen lagen die Verbesserungen einzelner Koordinatenkomponenten über 2.0 mm. Da sich sowohl im tachymetrischen Netz der globalen Passpunkte, als auch in den lokalen Passpunktkoordinaten der Laserscans die jeweiligen messtechnischen Unsicherheiten der Instrumente widerspiegeln, war mit Abweichungen bis zu 5.0 mm zu rechnen. Die Genauigkeiten der Transformationen lagen in allen bearbeiteten Punktwolken unter der abgeschätzten maximalen Abweichung beider Systeme. Weiter ist für alle berechneten Transformationen festzuhalten und ebenfalls in den Abbildungen 9.7 und 9.8 zu erkennen, dass die Ausgleichung mit fixiertem Radius der Kugel-Zielzeichen stets bessere Ergebnisse liefert. Dies war aufgrund der in Abschnitt 7.4 beschriebenen Effekte und der Evaluierung der Methoden zu erwarten.
Das Prozessieren der aufgeführten Datensätze benötigt ohne Berücksichtigung der Zeiten zum Ein-

(a) Umgebung des ZZ hinter einem Bauzaun, Datenlücken durch Abschattung

(b) Ausschnitt der nahen Umgebung des ZZ mit gitterförmiger Abschattung

(c) Kugel-Fit in CloudCompare mit 316 Punkten auf der Kugel, Radius aus Fit: 72.4 mm

Abbildung 9.5: per Rücktransformation gefundenes Kugel-Zielzeichen

Abbildung 9.6: Punkte auf der Oberfläche des teilweise verdeckten Kugel-Zielzeichens und schematische Darstellung der geometrischen Suche im Bild

Abbildung 9.7: Verbesserungen nach 2D Helmert-Transformation eines ersten Standpunktes mit (links) Kugelschätzung mit Radius; (rechts) Kugelschätzung mit fixiertem Radius

Abbildung 9.8: Verbesserungen nach 2D Helmert-Transformation eines zweiten Standpunktes mit (links) Kugelschätzung mit Radius; (rechts) Kugelschätzung mit fixiertem Radius

Punktwolke	Einlesen	Bilder generieren	Kandidatensuche im Bild	RANSAC & GHM	Registrierung	Speichern
Hbf	2678 ms	1784 ms	200 ms	536 ms	101 ms	3339 ms
Kanalsystem	3030 ms	2150 ms	31 ms	1123 ms	112 ms	3663 ms
Bahnübergang	1589 ms	1023 ms	106 ms	48 ms	60 ms	2076 ms

Tabelle 9.2: benötigte Zeiten zur Prozessierung der Teilschritte aller zur Evaluierung genutzten Punktwolken

lesen und Schreiben der Daten von 1.24 Sekunden bis 3.42 Sekunden (Testsystem: 64-bit Ubuntu Linux 20.04 LTS mit AMD Ryzen 5 2600 6(12)x 3.40GHz CPU, 16GB (2x8192MB) DDR4-3200 DIMM CL16 RAM und 250GB Samsung 970 Evo Plus M.2 2280 PCIe 3.0 x4 NVMe). Bei der parallelen Verarbeitung aller Testdatensätze dieser Arbeit ergab sich eine Gesamtlaufzeit von 114 s, was bezogen auf eine einzelne Punktwolke 3.45 Sekunden ergibt. Wie Tabelle 9.2 entnommen werden kann, bilden das Einlesen und Speichern der binären Daten den Flaschenhals der effizienten und automatisierten Registrierung. Die hierzu benötigen Zeiten können jedoch durch eigene Implementierungen nicht wesentlich beeinflusst werden.

Kapitel 10

Fazit & Ausblick

Ziel der Arbeit ist es Kugel-Zielzeichen in 3D-Scan-Punktwolken effizient und automatisiert zu detektieren. Entwickelt wurden, neben den eigentlichen Algorithmen zur Detektion der Zielzeichen, zusätzlich alle nötigen Teilschritte um den gesamten Workflow der zielzeichenbasierten Registrierung abzubilden. So umfasst die in C++ performant implementierte und modular aufgebaute Software den Datenfluss großer Punktwolken mit binärer Speicherstruktur zum Einlesen und Speichern der Daten. Effiziente Algorithmen zur Abbildung in verschiedene Bilder zu Dimensionsreduzierung sowie das effiziente und zuverlässige Suchen von Kandidaten in diesen Bildern wurde in Teilen parallelisiert implementiert. In verschiedensten Umgebungen der einzelnen Testdatensätze gelang die Detektion der Kugeln durch Abgleich mit geometrischen Soll-Eigenschaften zuverlässig. Eigenschaften wie Objektdichte, Verteilung der Intensitäten in der Punktwolke, sowie die teilweise sehr unterschiedlichen Distanzen zu den platzierten Kugel-Zielzeichen, welche ggf. zu einer erschwerten manuellen Suche der Zielzeichen in Standardsoftware führen, beeinflussen die implementierten Algorithmen in ihrer Zuverlässigkeit nicht. Das Validieren, Klassifizieren der Messpunkte auf den Kugel-Zielzeichen und robuste Schätzen von Näherungslösungen konnte ebenfalls äußerst effizient umgesetzt werden und spiegelt im gesamten Workflow einen Bruchteil der Prozessierungszeit wider. Das abschließende bestmögliche Schätzen der Kugelmittelpunkte sowie das Registrieren der gesamten Punktwolke auf Basis zuvor als identische Punkte im lokalen wie globalen Koordinatensystem aufgedeckte Passpunkte stellt den letzten Teil des Workflows dar. Auch hier profitiert der Nutzer von der parallelen Verarbeitung mehrerer Punktwolken eines Projektes in der Software.
Es konnte gezeigt werden, dass das automatische Detektieren von Kugel-Zielzeichen in 3D-Scan-Punktwolken auf Basis der optimierten Objektsuche in abgeleiteten Bildern äußerst effizient ist und im Rahmen einer performanten Software vollautomatisch zu Registrierung lokaler Punktwolken genutzt werden kann.
In weiteren Arbeiten zu diesem Thema sollte ein Fokus auf die Integration weiterer Datenformate der Punktwolken gelegt werden. Das bisher genutzte Datenformat stellt zwar binäre und effizient abgelegte Daten bereit, es bedarf jedoch der vorbereitenden Konvertierung der instrumentenspezifischen Exportformate.
Weiter könnte die Stochastik der Punktwolke in bestehende Auswertungen stärker einfließen. Aufgrund der geforderten Genauigkeiten und dem sich ergebenen großen Optimierungspotenzial bei der Vernachlässigung der stochastischen Größen wurde auf die Modellierung der Stochastik der einzelnen Messpunkte auf der Kugeloberfläche verzichtet. Untersuchungen zu diesem Thema und die Integration der Genauigkeitsinformationen in die gewichtete Auswertung könnten zu höheren Genauigkeiten der abschließenden Registrierung führen.

Anhang A

Zusammenfassung in englischer Sprache

Efficient and automated detection of sphere targets in 3D scan point clouds

Matthias Schreinemachers

Institute of Geodesy and Geoinformation
Department of Theoretical Geodesy
University of Bonn

October 15, 2020

Abstract

In this master thesis an efficient and performant implementation for the target-based registration of 3D scan pointclouds with sphere targets is developed and implemented in the compiled C++ programming language.
By processing the pointcloud and representing it as intensity and distance images, the associated dimension is reduced, with all information remaining recoverable. This images form the basis for the efficient algorithms for the target detection developed in this thesis. Since the spheres in those images appear as circles, the detection of sphere target candidates is simplified in a first approach to the detection of cirlces wich could be solved efficiently. The generation of the images and the detection for sphere candidates in images are done in parallel processing. Two different approaches for the detection of the target candidates were developed.
The analysis of several pointclouds of a project is also implemented in parallel and represents an efficient approach of the entire registration for larger surveying projects. The developed software represents a performant and holistic implementation of a fully automatic registration by means of geometrical sphere targets.
Key words: terrestrial laser scanning, pointclouds, automated target-based registration, sphere targets

1 Introduction and an Overview of the Implemented Software

1.1 Introduction and Context

In geodetic-engineering practice larger objects can often not be captured by the laserscanner from a single survey point. By measuring from several positions distributed around the object, the complete recording of the object can be achieved. The local 3D pointclouds of the individual positions must then be transformed into a common reference system within the scope of the so called registration process. One approach is the target-based registration of the individual pointclouds. For this purpose, for example sphere targets are mounted at control points in the surroundings of each survey point. These serve as identical reference points to determine the transformation between the local and global system, assuming that the coordinats in the global system are known.

The sphere targets used in several real projects - which serve as test data sets in this thesis - have been processed semi-automatically in standard software packages so far. The registration requires the manual detection and segmentation of all target marks, the manual segmentation of points on the sphere targets by visual observation of an operator as well as the final estimation of the sphere centre points.
In this master thesis an efficient and performant implementation for a fully automated target-based registration of 3D scan pointclouds with sphere targets is developed and implemented in C++. For an overview of the implemented workflow realized by the software package Figure 1 shows the program flow chart with all individual processing steps. In this chapter an overview of all necessary processes of the evaluation is provided. The two sub-steps of the processing highlighted in red are presented in more detail in the following. This summary of the master thesis ends with some conclusions.

1.2 Overview of the developed workflow

The implemented software package uses local pointclouds in binary `sbf` format as main input. Additionally an `ASCII` file with the coordinates of the control points in the global system of the project on which the registration should be processed is required.

In a first step the pointcloud is transformed into specific images. The measured intensities of the measuring points form the pixel values in the intensity image. The calculated distances are assigned to the pixels of the so called distance image. Additionally modified distance images are generated, which only show certain pixel values based on threshold values for intensity and distance in the image to directly filter out pixels not relevant for target detection.

In these now more sparse images the detection of potential sphere targets takes place. The program flow chart shows that two alternative approaches of the candidate search have been developed and implemented. These identify potential sphere targets in the two dimensional image, based on different geometric properties. Intensity variations in the neighborhood are used to validate the candidates already in the image. One of these approaches will be described in more detail later. The result of these partial steps are positions of potential sphere targets in the image.

After back transformation to the 3D positions of these candidates, the next step generates subpointclouds in which all 3D measuring points lying on the potential sphere target are generated. As shown in figure 1 in the second block, a further validation of these candidates is made in the 3D subpointcloud. Robust 3D sphere fitting is used to validate the candidate and to derive approximate center coordinates in case of successful validation. Details are discussed in section 3. The generated subpointclouds with the classified points on the sphere target form the basis of the final estimate of the target coordinates. This estimated coordinates as well as the given reference coordinates are used to derive the transformation parameters between the respective local and global reference systems.

Remaining targets not detected in the image are included in the final calculation of the transformation parameters by re-transforming the global control points into the respective local position analogous to the steps described above. The final product of the software is a set of registered pointclouds, as shown in the bottom row of Figure 1.

Figure 1: overview of the developed workflow and the implementation

2 Candidate Detection in the Distance Image

A previously prepared distance image of the pointcloud serves as the basis for the detection of potential target candidates. In this approach, the expected geometric properties of the spherical targets are used directly in the distance image for the detection. By including the distances in the detection of the spheres (motivated by [1]), a further dimension is used [1]. For each pixel in the distance image, target properties of a potential sphere in the image are derived depending on its current value (pixel value in the distance image $\widehat{=}$ distance to the measuring point). Since the radius of all spherical targets (R = 72.5 mm) is known, the distance of the current pixel, which is also known, can be used to calculate how a hypothetical sphere is represented at this position in the image, in case the current pixel was located centrally on the surface.

Figure 2: Pixel values in the neighbourhood of a sphere candidate in the distance image.

To check these properties, border pixels (see Figure 2, blue) and surrounding pixels (see figure 2, red) are selected and tested for every pixel sequentially. Margin pixels are located at a distance of 60 mm on the edge of the sphere, but always on the sphere surface. For each currently tested pixel (Figure 2 green) the distance to these edge pixels is calculated in pixels. Depending on the chosen resolution and the distance (value of the green pixel) this distance can be determined unambiguously. For these pixels the sphere geometry defines the expected range difference in case the current pixel actually represents a centre pixel of a spherical target. Comparing the expected range in addition to the border pixels, the distances for the surrounding pixels marked in red as well as valid pixel values for these are calculated analogously. It is tested for all surrounding pixels whether they are at least by the diameter of the sphere further away than the current centre pixel.

[1] compared to the alternative approach wich utilizes the fact that the spheres show up as circles in the image representation

Figure 3: detected sphere candidates shown in the intensity image (inverted), position at a railway crossing

Figure 4: detected sphere candidates shown in the intensity image (inverted), position in a canal system

For the distance images of the available data sets, the tests could be performed for all pixels in 1.2 seconds. With the selected instrument resolutions of 6.3 mm @10 m and 6.0 mm @10 m this results in approx. 40 Megapixel (4000 × 10 000 pixels) images. In Figure 3 and 4 all detected sphere candidates are shown exemplarily for one exposure situation. In the distance images of the pointclouds in a canal system, all sphere targets are detected successfully. The spheres from Figure 4, which are clearly at different distances, are detected just as reliably as the very small appearing spheres shown by Figure 3.

Compared to an alternative approach using the circle hough transformation (CHT [2], [3, S. 284-313]), it can be stated for all pointclouds that no false-positive candidate was detected. Since the required computing time is dominated by the validation of the detected candidates, avoiding false positives results in a robust and efficient implementation.

For 3D pointclouds, scanned with higher resolutions, the resulting images are significantly larger.

However, a scaling of the computing times for the candidate detection can be avoid for higher resolutions. The amount of checks per tested pixel remain constant. In contrast to the tests done here, not every pixel needs to be checked in line and column direction, but only every second or fourth pixel per direction. Thus, higher resolutions would only result in longer runtimes in the functions of reading the pointcloud data. However, the runtimes of the actual candidate search in images remain identical.

3 Subpointclouds and Classification

Even after a validation in the image, the detected candidates remain as candidates until they are validated as a sphere in the 3D pointcloud. For that purpose, the center coordinates are transformed back to their cartesian coordinates. A selection of a subpointcloud follows. This should contain all 3D points which are located on a spherical target. The horizontal direction and the vertical angle of each candidate are calculated from the lines and columns in the image. The distance as the third quantity of the polar coordinates results from the centre pixel value of the candidate itself. With

$$P_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \begin{bmatrix} -d_i \cdot \sin v_i \sin h_i \\ -d_i \cdot \sin v_i \cos h_i \\ d_i \cdot \cos v_i \end{bmatrix} \tag{1}$$

the cartesian centre of the sphere target can be calculated as an approximation of the centre of the subpointcloud. Around this point a cube is generated for each candidate whose edges are aligned with the coordinate axes. In each of the three spa-

Figure 5: inlier-selection after applying RANSAC to a subpointcloud

tial directions the edge length is 1.25 times the radius of the sphere. This results in cubes, in which all points of the candidates are located plus any close surrounding points. In the figure 5 a cube with larger edge lengths is shown.
The choice of a cube aligned with the coordinate axes represents the most computationally efficient method for extracting a subpointcloud. Here all points of the pointcloud are tested against a total of six coordinate components ($x_{\min}$, $x_{\max}$, $y_{\min}$, $y_{\max}$, $z_{\min}$, $z_{\max}$). This method is much more efficient than calculating the distances of each point to the approximated sphere target only a few more surrounding points enter the subpointcloud. All segmented pointclouds are passed to a RANSAC ([4, pp. 153-157]) implementation for a robust sphere fit. It determines the four approximation parameters (i.e. 3D center coordinates and the radius) of a possibly contained sphere and classifies all points as in- or outliers of a sphere. The selected inlier-points as well as their approximative parameters form the basis of the final reference point estimation in the Gauss-Helmert model (center coordinates and optionally the radius). This provides the most accurate and reliable estimation of the centre-coordinates.

4 Conclusion

In this master thesis it could be shown that the automatic detection of sphere targets in 3D scan pointclouds based on optimized object detection in generated images is extremely efficient. It can be used to register local laser scanning pointclouds automatically.

References

[1] Marek Franaszek, Geraldine S Cheok, and Christoph Witzgall. Fast automatic registration of range images from 3d imaging systems using sphere targets. *Automation in Construction*, 18(3):265–274, 2009.

[2] Henning Bässmann and Jutta Kreyss. *Bildverarbeitung ad oculos*. Springer-Verlag, 2013.

[3] E Roy Davies. *Machine vision: theory, algorithms, practicalities*. Elsevier, 2004.

[4] Wolfgang Förstner and Bernhard P Wrobel. *Photogrammetric computer vision*. Springer, 2016.

Anhang B

Wissenschaftliches Poster

Efficient and automated detection of sphere targets in 3D scan point clouds

Matthias Schreinemachers

Motivation

Problem definition:

Larger objects can often not be captured by the laserscanner from a single survey point. → **registration** requierd

This registration requires the **manual detection** and segmentation of all targets, the segmentation of points on the sphere targets by **visual observation** of an operator as well as the estimation of the sphere center coordinates.

The goal of this thesis:

In this master thesis an **efficient** and performant implementation for the **fully automated target-based registration** of 3D scan pointclouds with sphere targets is developed and implemented in C++.

Developed method for efficient sphere detection

- **reduce dimension** by transform the poitclouds to **images**
- detect **known geom.-properties** of the spheres in the distance image
  - check these properties at each center-pixel for:
    - 8 border pixels:
    - 4 surrounding pixels:
  - The sphere geometry defines the expected range difference in case the current pixel actually represents a target centre pixel.
- **validate** the detected **candidates** in:
  - images directly
  - generatet subpointclouds

Classification and robust estimation

- **RANSAC** implementation for a robust sphere fit
  - **classifies** all points as in- or outlier of a sphere
  - determines the **approximation parameters** of a possibly contained sphere
- inlier-points & approximative sphere parameters form the basis of the **final reference point** estimation in the **GHM**

Conclusion

In this master thesis it could be shown that the **automatic detection** of **sphere targets** in 3D scan pointclouds based on **optimized** object detection in generated **images** is **extremely efficient**. It can be used automatically to register local laser scanning pointclouds.

References

[1] Marek Franaszek, Geraldine S Cheok, and Christoph Witzgall. Fast automatic registration of range images from 3d imaging systems using sphere targets. Automation in Construction, 18(3):265–274, 2009.

[2] Rachakonda, Prem u. a. (2017). Methods and considerations to determine sphere center from terrestrial laser scanner point cloud data. Measurement Science and Technology 28.10.

[3] E Roy Davies. Machine vision: theory, algorithms, practicalities. Elsevier, 2004.

Overview of the developed workflow

Institute of Geodesy and Geoinformation
Department of Theoretical Geodesy
http://www.tg.uni-bonn.de/

Nußallee 17
53115 Bonn
phone.: +49 228 73-2626

Contact persons:
Univ. Prof. Dipl.-Ing. Dr.techn. Wolf-Dieter Schuh
schuh@uni-bonn.de

Abbildungsverzeichnis

2.1 (a) räumliche Abdeckung eines Panorama-Scanners, (b) Prinzip der Strahlablenkung eines Panorama-Scanners [angelehnt an Möser u. a. 2012, S. 286] 6
2.2 (a) Abnahme der Intensität mit steigendem Einfallswinkel am Objekt für unterschiedliche Farben (weiß, hellgrau, grau, dunkelgrau, schwarz); (b) Intensitätsabhängige Entfernungsmessgenauigkeit aus Streckenresiduen zu einer Referenz-Ebene [vgl. Schreinemachers 2020b] 6
2.3 Auftretender Kometenscheif am Kugel-Zielzeichen; (a) seitliche Ansicht der Aufnahmesituation; (b) Aufsicht mit deutlich erkennbaren Pixeln aus der beschriebenen Kometenschweif-Problematik 7
2.4 eingesetzte Instrumente zur Aufnahme der Punktwolken; (a) Leica ScanStation P16 [aus Leica P16 Benutzerhandbuch 2016]; (b) Leica RTC360 [aus Leica RTC360 Benutzerhandbuch 2018] 8
2.5 Kugelzielzeichen-Standorte; (a) Kugel-Zielzeichen mit Magnethalterung auf einem Fahrscheinentwerter im Bonner Hbf; (b) Kugel-Zielzeichen mittels Wandbolzen (M8-Innengewinde) an Schachtwand angebracht; (c) Klebemarke mit Punktnummer zur Punktsignalisierung auf metallischem Geländer 9
2.6 eingesetzte Kugel-Zielzeichen zur Referenzierung der Punktwolken; (a) Kugel-Zielzeichen-Set mit sechs Kugeln; (b) einzelnes Zielzeichen mit Magnetfuß 9
2.7 Arten möglicher Reflexionen an verschiedenen Oberflächen [angelehnt an Joeckel, Stober und Huep 2008, S. 335] und [angelehnt an Kern 2003, S. 40]; (a) Totalreflexion an der Objektoberfläche; (b) diffus-reflektierende Oberfläche (c) diffuse Reflexion mit Spiegelung an Objektoberfläche 10
2.8 Messpunkte auf der Kugelzielzeichenoberfläche; erste Zeile: Kugel in 0.65 m Entfernung und 110.948 Punkte; zweite Zeile: Kugel in 8.33 m Entfernung und 517 Punkte 10
2.9 Schweif am Kugel-Zielzeichen in naher Umgebung zum Laserscanner 11
2.10 Öffnungswinkel am Kugel-Zielzeichen 11
2.11 Rauschen auf mehreren Kugeln unter verschiedenen Öffnungswinkeln; Ergebnisse mehrerer Laserscanner [vgl. Rachakonda u. a. 2017, S. 18]. 12
2.12 Segmentierung anhand definiertem Öffnungswinkel im Kugelzentrum zum Laserscanner 12
2.13 Intensitätsbild (invertiert) Scan Bahnübergang I 13
2.14 Intensitätsbild (invertiert) Bahnsteig 1 Bonner Hauptbahnhof 14
2.15 Punktwolkenausschnitt in CloudCompare, zwei Kugel-Zielzeichen an Stahlträgern hinter Bauzäunen 15
2.16 Intensitätsbild (invertiert) Durchlass Kanalsystem I 15
2.17 HDR-Bild eines Standpunktes im Kanalsystem, Spiegelung zweier Kugel-Zielzeichen an der Wasseroberfläche 16

3.1 Programmablaufplan 18

4.1 Speicherbedarf (a) und Einlesegeschwindigkeiten der Punktwolke in CloudCompare (b) in Abhängigkeit des Datenformates 20

4.2 (a) Punktwolkenausschnitt in CloudCompare; (b) Ausschnitt des Intensitätsbildes; (c) Ausschnitt des Distanzbildes . . . 22
4.3 exemplarische Darstellung zweier Kugel-Zielzeichen im jeweiligen Intensitätsbild . . . 22
4.4 Datenreduzierung durch Abbildung im modifizierten Distanzbild für alle Standpunkte der Testdatensätze . . . 23
4.5 exemplarische Darstellung zweier Kugel-Zielzeichen im jeweiligen modifizierten Distanzbild . . . 23

5.1 binäres Filterergebnis nach Anwendung von (5.1.1) und (5.1.2) auf Bild 4.5a . . . 28
5.2 schematische Darstellung der Circle Hough Transformation in drei Schritten . . . 28
5.3 Intensitätsbilder verschiedener Distanzklassen (invertiert) . . . 30
5.4 Intensitätsbild (invertiert) generiert für Distanzen von 0.3 m bis 1.0 m, Suchradien von 770 Pixel bis 230 Pixel . . . 30
5.5 Intensitätsbild (invertiert) generiert für Distanzen von 1.0 m bis 2.0 m, Suchradien von 230 Pixel bis 115 Pixel . . . 31
5.6 Intensitätsbild (invertiert) generiert für Distanzen von 7.5 m bis 10.0 m, Suchradien von 31 Pixel bis 23 Pixel . . . 31
5.7 Pixelwerte in der Umgebung eines Kugelkandidaten . . . 32
5.8 Detektierte Kugelkandidaten Standpunkt 2 (Bahnübergang), abgebildet im Intensitätsbild (invertiert) . . . 33
5.9 Detektierte Kugelkandidaten Standpunkt 10 (Kanalsystem), abgebildet im Intensitätsbild (invertiert) . . . 34
5.10 Detektierte Kugelkandidaten Standpunkt 21 (Kanalsystem), abgebildet im Intensitätsbild (invertiert) . . . 34
5.11 Detektierte Kugelkandidaten Standpunkt 1 (Bonner Hauptbahnhof), abgebildet im Intensitätsbild (invertiert) . . . 35
5.12 Scanner Foto . . . 35
5.13 (a) seitliche Ansicht des Punktwolkenausschnittes; (b) Visualisierung der Kugelkandidaten ohne Validierung mit modifiziertem Distanzbild und Algorithmus aus Abschnitt 5.2 . . . 37
5.14 (a) modifiziertes Distanzbild des Ausschnittes; (b) Visualisierung der Kugelkandidaten mit Validierung mit modifiziertem Distanzbild und Algorithmus aus Abschnitt 5.2 . . . 37

6.1 rot: mittels CHT gefundener Kreis in modifiziertem Distanzbild, (a) positives Bsp. mit homogenen Pixelwerten, (b) negatives Bsp. (false-positiv) . . . 39
6.2 schematische Darstellung einer selektierten Subpunktwolke mit grünen Punkten auf der Oberfläche des Kugel-Zielzeichens und roten Umgebungspunkten . . . 40
6.3 Prinzip des RANSAC am Beispiel einer Geraden durch zwei Punkte, optimale Lösung mit größtem Support (maximale Anzahl von Inlier Punkten in Epsilon-Umgebung) . 42
6.4 Visualisierung der Outlier in unmittelbarer Umgebung zum Kugelkandidaten, (a) rot schraffierte Fläche lediglich im Bereich des Magnetfußes des Zielzeichens, (b) schematische Darstellung eines falsch vermuteten Kugelkandidaten an einem zylindrischen Objekt (z.B. Laternenmast) mit großen Teilflächen im Umfeld des Kandidaten . . . 43
6.5 Inlier Selektion nach Anwendung des RANSAC auf zwei Subpunktwolken . . . 44

7.1 Abweichungen bei zufälliger Auswahl von verschieden vielen Punkten zur Schätzung der Passpunktkoordinaten . . . 47
7.2 Differenzen der im Ausgleich geschätzten Radien zum bekannten Radius der Kugel-Zielzeichen . . . 48

8.1 schematische Darstellung der Passpunktsituation mit paarweiser Distanzbildung; (a) lokale Passpunkte; (b) globale Passpunktsituation und identische Distanzen zur Punktzuordnung . . . 51
8.2 Detektierte Kugel-Zielzeichen im Bild und Anzahl platzierter Kugel-Zielzeichen pro Datensatz . . . 52
8.3 endgültig gefundene Kugel-Zielzeichen und Anzahl platzierter Kugel-Zielzeichen pro Datensatz . . . 52

9.1 Intensitätsbild der Punktwolke am Bonner Hbf, grün: im Bild gefundene ZZ, gelb: nach Rücktransformation gefundenes ZZ . . . 54
9.2 Intensitätsbild einer Punktwolke im Kanalsystem, grün: im Bild gefundene ZZ . . . 54
9.3 Intensitätsbild einer Punktwolke am Bahnübergang, grün: im Bild gefundene ZZ, rot: endgültig nicht erkanntes Kugel-Zielzeichen . . . 54
9.4 nicht gefundenes Kugel-Zielzeichen . . . 55
9.5 per Rücktransformation gefundenes Kugel-Zielzeichen . . . 56
9.6 Punkte auf der Oberfläche des teilweise verdeckten Kugel-Zielzeichens und schematische Darstellung der geometrischen Suche im Bild . . . 56
9.7 Verbesserungen nach 2D Helmert-Transformation eines ersten Standpunktes mit (links) Kugelschätzung mit Radius; (rechts) Kugelschätzung mit fixiertem Radius . . 56
9.8 Verbesserungen nach 2D Helmert-Transformation eines zweiten Standpunktes mit (links) Kugelschätzung mit Radius; (rechts) Kugelschätzung mit fixiertem Radius . . 57

Tabellenverzeichnis

2.1 Auflösungsstufen und benötigte Zeit für einen Rundum-Scan, *lediglich mit Auswahl eines Umgebungsausschnittes sinnvoll, da im Rundumscan sehr große Punktmengen generiert werden [vgl. Leica P16 Benutzerhandbuch 2016, 110 f.][vgl. Leica RTC360 Benutzerhandbuch 2018, S. 38] 8
2.2 Standpunktinformationen aller in dieser Arbeit genutzten Scans 14

4.1 Byte-Muster des Headers einer binären *.data.sbf Datei 20
4.2 Body-Teil der binären *.data.sbf Datei 21

7.1 LAPACK Subroutinen 48

8.1 Beispiel eines Punktsets zur Transformation der lokalen und globalen Passpunkte . . 51

9.1 Eigenschaften der zur Evaluierung genutzten Punktwolken 55
9.2 benötigte Zeiten zur Prozessierung der Teilschritte aller zur Evaluierung genutzten Punktwolken 57

Literatur

Bässmann, Henning und Jutta Kreyss (2013). *Bildverarbeitung ad oculos*. Springer-Verlag.

Canny, John (1986). „A computational approach to edge detection“. In: *IEEE Transactions on pattern analysis and machine intelligence* 6, S. 679–698.

Caspary, Wilhelm und D Beineke (2003). „Robuste Helmert-Transformation“. In: *AVN* 7, S. 242–247.

Cherabit, Noureddine, Fatma Zohra Chelali und Amar Djeradi (2012). „Circular hough transform for iris localization“. In: *Science and Technology* 2.5, S. 114–121.

Dagum, Leonardo und Ramesh Menon (1998). „OpenMP: an industry standard API for shared-memory programming“. In: *IEEE computational science and engineering* 5.1, S. 46–55.

Davies, E Roy (2004). *Machine vision: theory, algorithms, practicalities*. Elsevier.

Derpanis, Konstantinos G (2010). „Overview of the RANSAC Algorithm“. In: *Image Rochester NY* 4.1, S. 2–3.

D'Orazio, Tiziana u. a. (2004). „A new algorithm for ball recognition using circle Hough transform and neural classifier“. In: *Pattern recognition* 37.3, S. 393–408.

Ehm, Markus und Christian Hesse (2014). „3D-Laserscanning zur Erfassung von Gebäuden–Building Information Modeling (BIM)“. In: *Bautechnik* 91.4, S. 243–250.

Förstner, Wolfgang und Bernhard P Wrobel (2016). *Photogrammetric computer vision*. Springer.

Franaszek, Marek, Geraldine S Cheok und Christoph Witzgall (2009). „Fast automatic registration of range images from 3D imaging systems using sphere targets“. In: *Automation in Construction* 18.3, S. 265–274.

GNU Octave (2020a). *Dokumentation imfindcircles(), GNU Octave, image package (version 2.12.0)*. [Online; abgerufen am 09.10.2020]. URL: `https://octave.sourceforge.io/image/function/imfindcircles.html`.

— (2020b). *GNU Octave (version 5.2.0)*. [Online; abgerufen am 04.10.2020]. URL: `https://www.gnu.org/software/octave/`.

Heunecke, O u. a. (2013). *Auswertung geodätischer Überwachungsmessungen, 2. Aufl. Handbuch Ingenieurgeodäsie*. Wichmann Verlag, Berlin.

Höfle, Bernhard und Norbert Pfeifer (2007). „Correction of laser scanning intensity data: Data and model-driven approaches“. In: *ISPRS journal of photogrammetry and remote sensing* 62.6, S. 415–433.

Jahn, Ingo, Thomas Kersten und Roland Kinzel (2004). „Erfahrungen mit einem 3D-Laserscanning-System bei der Erfassung einer Industrieanlage und des Lübecker Holstentores“. In: *Photogrammetrie-Laserscanning-Optische 3D-Messtechnik*, S. 222–229.

Jelalian, Albert V (1992). *Laser radar systems*. Artech House.

Joeckel, Rainer, Manfred Stober und Wolfgang Huep (2008). *Elektronische Entfernungs-und Richtungsmessung und ihre Integration in aktuelle Positionierungsverfahren*. 5. Wichmann Verlag, Berlin.

Kern, Fredie (2003). *Automatisierte Modellierung von Bauwerksgeometrien aus 3D-Laserscanner-Daten*. Technische Universität Carolo-Wilhelmina zu Braunschweig.

Kersten, Thomas u. a. (2008). „Datenfluss im terrestrischen Laserscanning-Von der Datenerfassung bis zur Visualisierung“. In: *Terrestrisches Laserscanning (TLS 2008), Schriftenreihe des DVW* 54, S. 31–56.

Kuhlmann, Heiner und Christoph Holst (2017). „Flächenhafte Abtastung mit Laserscanning“. In: *Ingenieurgeodäsie*. Springer, S. 167–212.

Lee, Sherman und JoAnne K Halligan (1998). *Method for compiling a software program and executing on a system which converts data between different endian formats*. US Patent 5,828,884.

Leica P16 Benutzerhandbuch (2016). *Leica ScanStation P16 User Manual, 827810-4.0.0en*. Leica Geosystems AG, Heerbrugg, Schweiz.

Leica P16 Datenblatt (2015). *Leica ScanStation P16 Datenblatt, 835468de–03.15–INT*. Leica Geosystems AG, Heerbrugg, Schweiz.

Leica RTC360 Benutzerhandbuch (2018). *Leica RTC360 User Manual, 870891-1.0.0en*. Leica Geosystems AG, Heerbrugg, Schweiz.

Leica RTC360 Datenblatt (2018). *Leica RTC360 Datenblatt, 2018.872752de-06.18*. Leica Geosystems AG, Heerbrugg, Schweiz.

Liu, Hong, Yueliang Qian und Shouxun Lin (2010). „Detecting Persons using Hough Circle Transform in Surveillance Video.“ In: *VISAPP (2)*, S. 267–270.

Mitra, Jubin, Abhijit Chandra und Tanmay Halder (2013). „Peak trekking of hierarchy mountain for the detection of cerebral aneurysm using modified hough circle transform“. In: *ELCVIA Electronic Letters on Computer Vision and Image Analysis* 12.1, S. 57–84.

Möser, Michael u. a. (2012). *Handbuch Ingenieurgeodaesie: Grundlagen*. 4. Wichmann Verlag, Berlin.

Niemeier, Wolfgang (2008). *Ausgleichungsrechnung: Statistische Auswertemethoden*. Walter de Gruyter.

Open Source Computer Vision, OpenCV (2020). *cv::Mat Class Reference*. [Online; abgerufen am 04.10.2020]. URL: `https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html`.

OpenCV (2020a). *cv::Mat Class Reference*. [Online; abgerufen am 10.10.2020]. URL: `https://docs.opencv.org/3.4/d3/d63/classcv_1_1Mat.html#details`.

— (2020b). *Dokumentation HoughCircles(), OpenCV für C++*. [Online; abgerufen am 09.10.2020]. URL: `https://docs.opencv.org/3.4/dd/d1a/group__imgproc__feature.html#ga47849c3be0d0406ad3ca45db65a25d2d`.

Rachakonda, Prem u. a. (2017). „Methods and considerations to determine sphere center from terrestrial laser scanner point cloud data“. In: *Measurement Science and Technology* 28.10.

Reidl, Fabian und J Ohlmann-Lauber (2014). „Reverse Engineering eines hydraulischen Versuchsstandes-Laserscanning und 3D-Modellierung zur Erstellung eines CAD-Plans“. In:

Schmitz, Berit u. a. (2019). „How to Efficiently Determine the Range Precision of 3D Terrestrial Laser Scanners“. In: *Sensors* 19.6, S. 1466.

Schreinemachers, Matthias (2020a). „Automatisierte Erkennung von Kugel-Zielzeichen in 3D-Punktwolken und deren Registrierung bei vorhandenen Näherungskoordinaten (unveröffentlicht), Seminararbeit im Rahmen des Moduls Ausgewählte Kapitel der Numerik, Geowissenschaftliches Rechnen, 3/2020“.

— (2020b). „Stochastik der Punktwolke Untersuchungen zu intensitäts- und geometrieabhängigen Entfernungsmessgenauigkeit sowie deren stochastische Abhängigkeiten (unveröffentlicht), Abschlussbericht zur Projektarbeit im Rahmen des Moduls Deformationsanalyse mit Punktwolken, 3/2020“.

Smereka, Marcin und Ignacy Dulęba (2008). „Circular object detection using a modified Hough transform“. In: *International Journal of Applied Mathematics and Computer Science* 18.1, S. 85–91.

Wang, Yanmin u. a. (2014). „Automatic registration of laser point cloud using precisely located sphere targets“. In: *Journal of applied remote sensing* 8.1, S. 433–453.

Witzgall, Christoph, Geraldine S Cheok und Anthony J Kearsley (2006). „Recovering circles and spheres from point data“. In: *Perspectives In Operations Research*. Springer, S. 393–413.

Wujanz, Daniel u. a. (2016). „Standpunktplanung beim terrestrischen Laserscanning“. In: *Schriftenreihe des DVW Band 85, Terrestrisches Laserscanning*.

Wujanz, Daniel u. a. (2017). „An intensity-based stochastic model for terrestrial laser scanners“. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 125, S. 146–155.

Zwillinger, Daniel (2002). *CRC standard mathematical tables and formulae*. CRC press.