Hochschule für Technik und Wirtschaft Dresden

Fakultät Geoinformation

Diplomstudiengang Vermessungswesen

FV33 - Diplomarbeit

Automatisierte Bestimmung von Schüttgutvolumen aus Punktwolken

Eingereicht von

Mario Rosenbohm

Seminargruppe: 15/061/21

Matrikelnummer: 39581

1. Gutachter: Prof. Dr. rer. nat. Stephan Joachim Kopf

2. Gutachter: Prof. Dr. - Ing. Christian Clemen

Eingereicht am: 15.05.2020

Kurzzusammenfassung / abstract

Ziel dieser Diplomarbeit ist es, eine Software zu entwickeln, welche den Benutzer bei der Volumenberechnung aus Punktwolken unterstützt.
Dazu werden verschiedene Methoden diskutiert, mit deren Hilfe eine Trennung der Punktwolke in zwei Klassen (»gehört zur Volumenoberfläche« und »gehört nicht zur Volumenoberfläche«) möglich ist. Eine Methode stellt sich, unter bestimmten Bedingungen, als geeignet dar. Mit dieser Methode lässt sich die geforderte Klassifizierung effizient durchführen.
Weiterhin gilt es, für die eigentliche Volumenberechnung, Methoden zu analysieren und auf deren Tauglichkeit für die Anwendung in der Punktwolke zu prüfen. Aus diesen Ergebnissen wird ein Verfahren zur Klassifizierung der Scanpunkte und direkt anschließender Volumenmodellerstellung erarbeitet.
Die Darstellung des Volumens ist an vorgegebene Rahmenbedingungen, das Verwenden einer bestimmten CAD-Software, gebunden. Mit Hilfe dieser CAD-Software ist eine Darstellungsvariante zu wählen, welche ein zügiges Arbeiten mit dem Volumenmodell ermöglicht.
Alle Verfahren und Methoden, einschließlich der dabei auftretenden Probleme werden in Programmcode umgesetzt, damit am Ende eine funktionsfähige, unterstützende Software entsteht.

abstact

The intention of this thesis is to develop a software which supports the user in calculating volumes from point clouds.
For this purpose, different methods are discussed that allow a separation of the point cloud into two classes (i.e. »belongs to the volume surface« and »does not belong to the volume surface«). One method is shown to be suitable under certain conditions. With this method, the required classification can be carried out efficiently.
Furthermore, methods for the volume calculation itself are analyzed and tested to ensure that they are suitable for use in the point cloud. Based on these results, a method for the classification of the scan points and the ensuing generation of the volume model are developed.
The representation of the volume is bound to given framework conditions, including the use of a specific CAD software. With the help of this CAD software, a representation variant is to be selected which enables a user to easily work with the volume model.
All procedures and methods, including the problems that arise in the process, are converted into program code, so that in the end a functional, helpful software is created.

Inhaltsverzeichnis

Abkürzungsverzeichnis

2D Zweidimensional, eben, Ausdehnung nur in X- und Y-Achse

3D Dreidimensional, räumlich, Ausdehnung in X-, Y- und Z-Achse

CAD Computer Aided Design, computerunterstütztes Konstruieren

C# Objektorientierte, Java ähnliche, C++ Notations basierende Programmiersprache

DGM Digitales Gelände Modell, ein Dreiecksmaschenmodell, welches (Gelände)Oberflächen beschreibt

DLL Dynamic Link Library, eine Datei, welche Funktionalitäten über eine öffentliche Schnittstelle zur Verfügung stellt

GB Maßeinheit in der Informationsverarbeitung 1 GB = 10^9 Bytes = $8 \cdot 10^9$ Bits

MVVM Model-Viewer-ViewModel, ein Entwicklungsmuster für WPF-XAML-Oberflächen zur Trennung von Daten, Geschäftslogik und Darstellung

PC Personal Computer

RAM Random Access Memory, der Arbeitsspeicher eines PC

SSD Solid State Disk, Festplatte ohne rotierende Scheiben, Daten werden auf Speicherchips abgelegt, ähnlich einer SD-Karte, wesentlich schneller als herkömmlich Festplatten

STL STereoLithographie, ein Standarddateiformat um Körper als Außendreiecksflächen auszutauschen

WPF Windows Presentation Foundation, ein von Microsoft entwickeltes Framework um Benutzeroberflächen auflösungsunabhängig, vektorbasierend darzustellen

XAML eXtensible Application Markup Language, Microsoft eigene XML-Erweiterung für die Beschreibung einer visuellen Benutzerschnittstelle

XML eXtensible Markup Language, eine textbasierte Beschreibungssprache

1 Einleitung

In der K+S Minerals and Agriculture GmbH werden vorwiegend mineralische Dünger mit den Komponenten Kalium, Magnesium, Natrium und Schwefel produziert. All diese Dünger sind wasserlöslich und werden daher in großen Hallen, den sogenannten Produktspeichern, gelagert. Diese Lager, am Standort, haben ein Fassungsvermögen von jeweils etwa 15000 t bis 45000 t Düngemittel. Zu Revisionszwecken werden in regelmäßigen Abständen die Lagerbestände an Produkt überprüft. Dies geschieht bisher durch ein tachymetrisches Aufmaß der Oberfläche der Produkthaufen, der Modellierung eines digitalen Geländemodells (DGM) und anschließender Berechnung des Volumens aus diesem.
Das Aufmaß der Produktoberfläche soll zukünftig durch Messung mit einem Laserscanner erfolgen. Die Berechnung des Volumens muss dann aus der Punktwolke abgeleitet werden. Zielsetzung des Projektes ist es, zum einen eine der bisher drei benötigten Arbeitskräfte einzusparen und zum anderen die Genauigkeit des Aufmaßes zu steigern. Eine zwingende Bedingung ist, das praktische Aufmaß und auch die vollständige Auswertung der Messungen innerhalb eines Arbeitstages, in maximal acht Zeitstunden, zu vollziehen. Die Realisierbarkeit des praktischen Aufmaßes mit einem terrestrischen Laserscanner wurde bereits im Rahmen eines vorherigen Studienprojektes nachgewiesen.
Im Rahmen dieser Diplomarbeit soll, ein Arbeitsablauf entwickelt werden, um die Auswertung der registrierten Punktwolke in einem Arbeitsgang zu ermöglichen. Mit diesem Ziel wird eine Software entwickelt, die den Anwender von der Schaffung notwendiger Grundlagen bis zur Ausgabe von Protokolldaten unterstützt.

1.1 Thesen / Fragen

Aus der Aufgabenstellung, das Volumen eines Produkthaufen aus einer Punktwolke abzuleiten, ergeben sich folgende Fragestellungen und aus diesen resultieren Thesen:

Faktum 1: Die Punktwolke eines Laserscan beschreibt alle sichtbaren Oberflächen, also nicht nur den Produkthaufen sondern auch alle Wand-, Dach- und andere -Oberflächen.
Frage 1.1: Ist es möglich, aus den empfangenen Daten (Intensität, Farbe) auf den Typ der Oberfläche (Produkt / Nichtprodukt) zu schließen?
Frage 1.2: Welche anderen Möglichkeiten der Klassifizierung gibt es und sind diese praktisch umsetzbar?
These 1: *Die Punkte können automatisch in zwei Gruppen unterteilt, genauer klassifiziert, werden, in Produkt und Nichtprodukt.*

Faktum 2: Die Punkte auf der Oberfläche des Produkthaufens beschreiben nur die sichtbare Oberfläche, nicht das Volumen des Produktes.
Frage 2.1: Welche Volumenberechnungsverfahren eignen sich für die großen Punktmengen?
Frage 2.2: Wie lässt sich ein fehlertolerantes Berechnungsverfahren beschreiben?
These 2: *Aus den gemessenen Punkten wird das Volumen effektiv und fehlertolerant berechnet.*

Faktum 3: Durch das Scanverfahren entstehen hinter hohen Kanten Sichtschatten, in denen sich keine, die Oberfläche beschreibenden, Punkte befinden.
Frage 3.1: Wie kann man Rückschlüsse auf den, der Wirklichkeit am nächstliegenden, Oberflächenverlauf ziehen?
Frage 3.2: Lassen sich diese »Löcher« automatisch schließen?
These 3: *Die entstehenden Oberflächenlöcher lassen sich möglichst wahrheitsgetreu schließen.*

In den folgenden Kapiteln werden die Fragen und Thesen immer wieder aufgegriffen, analysiert und es wird geklärt, in wie weit die Thesen zutreffen.

1.2 Literatur

Mit Hilfe der Literaturrecherche soll ein Überblick über die bestehenden Möglichkeiten zur Beantwortung der im vorherigen Kapitel aufgekommenen Fragen gegeben werden.

Wie in [Alb09][S.12] erläutert wird, steht jeder Körper in Wechselwirkung mit seiner Umwelt. Dies geschieht durch die elektromagnetische Strahlung. Trifft diese auf den Körper, wird je ein Teil absorbiert, reflektiert und/ oder durchdringt ihn. Die Verteilung zwischen den drei Arten der Reaktion hängen sowohl von der Wellenlänge der elektromagnetischen Strahlung, als auch von dem Material des Körpers ab. Anhand der reflektierten Energiemenge einer bestimmten Wellenlänge, dem Reflektionsgrad, kann man Rückschlüsse auf die Stoffeigenschaften des Reflektors ziehen. Um jedoch verlässliche Aussagen über den reflektierenden Stoff zu erhalten reicht es nach [Alb09][S.18 ff] nicht aus, den Reflektionsgrad nur einer Wellenlänge zu bestimmen. In der Fernerkundung sind mindestens vier Wellenlängenbereiche üblich. Damit wird der Bereich der Infrarot- bis zur Microwellenstrahlung abgedeckt. Erst mit solch umfassenden Daten lassen sich sicher Rückschlüsse auf die Materialien der reflektierenden Oberflächen ziehen. Der verwendete Laserscanner sendet nur einen Laserstrahl in einer Frequenz aus und registriert bei dessen Empfang die empfangene Energiemenge. Durch den Transport der Produkte über Förderbänder entsteht Produktabrieb und daraus Staub. Beim Einspeichern der Produkte

in die Speicher lagert sich dieser Staub auf nahezu allen Oberflächen ab. Dadurch ist es auch kaum möglich eine genaue Aussage über das Material zu treffen, auf dessen Oberfläche der Laserscanpunkt gemessen wurde.

Ein sehr interessanter Ansatz erschien im Januar 2020 in einem Artikel des internationalen Magazin »THE PHOTOGRAMMETRIC RECORD«. In [Liu+20] beschrieben acht chinesische Wissenschaftler, wie aus, mit Hilfe mobiler Laserscanner erzeugter, Punktwolken geometrische Objekte zu erkennen sind und damit die Punktwolke gruppiert werden kann. In dem beschriebenen Fall wurde versucht aus dieser Punktwolke Bäume, Laternen, Ampeln und Verkehrsschilder zu selektieren. Dies geschah in einem mehrstufigen Verfahren. Im ersten Schritt fand das RANSAC-Verfahren[1], ein sehr robuster Schätzer für Ausgleichungsberechnungen, Anwendung, um Bögen in den Punktwolken zu erkennen. In nächsten Schritt wurden Säulen ermittelt und in einem dritten Schritt, anhand der oberen Punkteverteilung, grob auf Baumkrone, Ampel und Pfahl unterschieden. Die erreichten Ergebnisse sind sehr beeindruckend.

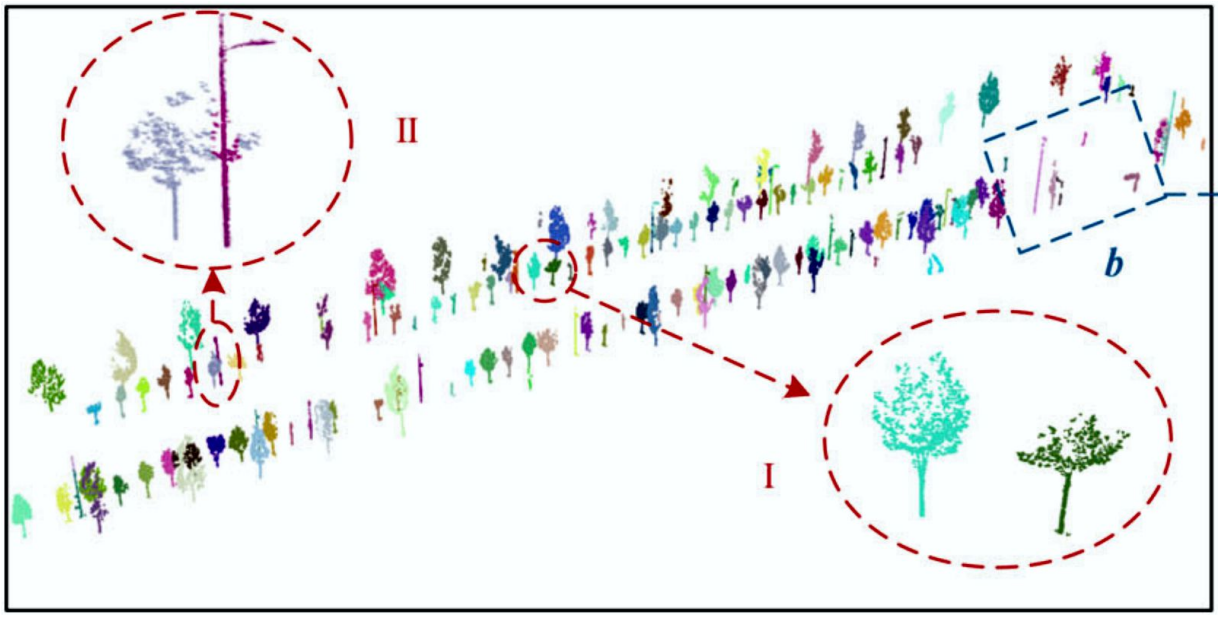


Abbildung 1.1: Ausschnitt aus der Abbildung [Liu+20][S.19, Fig12 (b1)], sie zeigt die erkannten Objekte aus der Punktwolke

Ein solcher Ansatz könnte auch bei den bestehenden Daten Anwendung finden, wenn dabei nicht das Zeitproblem im Wege steht. Denn eine solche Auswertung hat, nach Angaben des Artikels, zwischen 43 bis 134 Minuten benötigt.

[1] Erläuterung des RANSAC-Verfahrens: `https://de.mathworks.com/discovery/ransac.html`, 25.03.2020

In dem Magazin »Computers & Geosciences« des niederländischen Verlages Elsevier[2] wurde im Jahr 2013 ein Artikel von vier Wissenschaftlern der »Technological Univerity Dublin« und dem »University College Dublin« veröffentlicht. In diesem Artikel [Sch+13] wurde die Indexbildung über einer Punktwolke mit Hilfe eines Octrees in einer Oracle-Spatial-Datenbank diskutiert.
Ein Octree bildet eine Würfelstruktur, wobei der erste Würfel die gesamte Ausdehnung der Punktwolke abbildet. Die Kanten des Raumwürfels werden an den Achsen des Koordinatensystems ausgerichtet. Im nächsten Schritt wird dieser Würfel in 8 kleinere Würfel unterteilt, welche dann die halbe Kantenlänge aufweisen. Die Teilwürfel, welche nach der Teilung mehr als eine vorher definierte Anzahl von Punkten enthalten, werden wiederum geteilt. Dieser Schritt geschieht so oft, bis alle entstandenen Würfel nicht mehr als die erlaubte Anzahl von Punkten enthalten.
In [Sch+13] wurde die Aufteilung exemplarisch mit 2.9 Mio. Punkte und 66 Mio. Punkte berechnet. Der Rechner benötigte bei 2.9 Mio. Punkten nur 9.4 Sekunden, bei 66 Mio. Punkten jedoch schon 5:21 Minuten. Dieser Ansatz bietet den Vorteil, dass durch die erfolgte Indizierung alle Punkte gruppiert werden.

Derzeit steht im programmiertechnischen Umfeld ein Thema stark im Focus, »Voxelsysteme«. Auch im Bereich der K+S werden mit solchen Systemen Programieraufgaben umgesetzt. Ein sehr prominentes Beispiel von Voxelsystemen ist das Computerspiel Minecraft[3]. Um zu klären, ob auch diese Systeme einen Lösungsansatz bieten, wurde die Literaturrecherche erweitert.
In dem Magazin »Graphical Models and Image Processing« des niederländischen Verlages Elsevier wurde bereits im Jahr 1995 ein Artikel zu Grundlagen der Oberflächen Voxelerzeugung veröffentlicht. In diesem Artikel ([CK95]) werden von Cohen-Or und Kaufmann die Grundlagen erläutert, mit deren Hilfe unregelmäßige Oberflächen in Voxelsysteme überführt werden. Dabei wird zur Vereinfachung die Zuordnung der zur Oberfläche gehörenden Voxel anhand von 2D-Kurven erläutert. Bei der Voxelerzeugung aus dreidimensionalen Oberflächen wird eine schnittweise Voxelgenerierung getestet. Hierbei wird festgestellt, dass durch die Oberflächen im Inneren Hohlräume entstehen. Diese Hohlräume können laut [CK95] nur über topologische Nachbarschaftsbeziehungen (siehe Abildung 1.2, S. 5) geschlossen werden.

[2] `https://www.elsevier.com/de-de`, 25.03.2020
[3] `https://www.minecraft.net/de-de/`

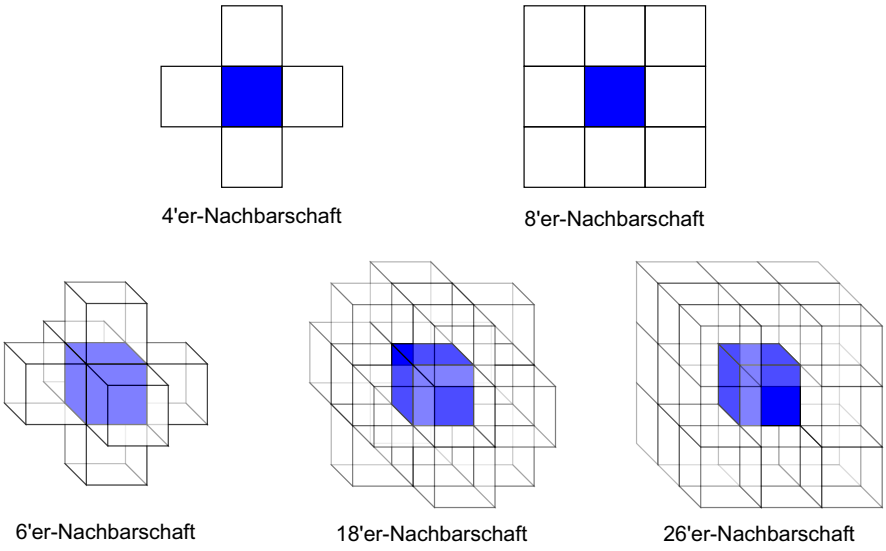


Abbildung 1.2: Darstellung der Nachbarschaftsbeziehungen (obere Reihe 2D, untere Reihe 3D)

Auf einem Symposium der IEEE[4] im Oktober 1998 in Durham, Nord Carolina (USA), zum Thema Volumendarstellung wurde ein Vortrag von vier Wissenschaftlern der Ohio State Univertity mit dem Thema »Eine genaue Methode zum Voxelerzeugen aus Polygonnetzen« gehalten ([Hua+98]). In diesem Vortrag wurden die Probleme, welche bei der Voxelerzeugung aus Polygonnetzten entstehen, analysiert. Dabei stellt man fest, dass die Voxelerzeugung eine Kombination aus Berechnung und Topologie ist. Gerade im 3D-Raum bedarf es der Betrachtung der topologischen Nachbarn. In der 2D-Ebene wurde gezeigt, dass es nur mit den topologischen Nachbarn möglich ist, ein Polygon in eine geschlossene Voxelkurve zu überführen. Im 3D-Raum konnte dargestellt werden, dass eine Ebene immer nur Teile von Voxel schneiden kann ([Hua+98][S.5]).

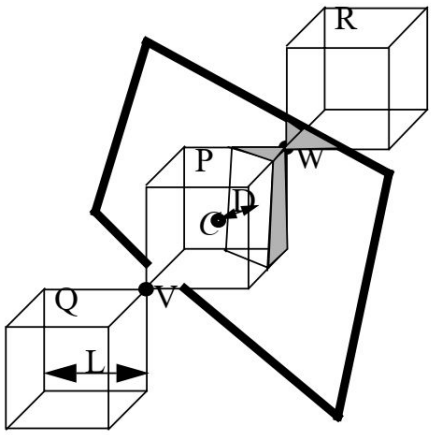


Abbildung 1.3: Abbildung [Hua+98][S.5, Fig11], sie zeigt den Schnitt einer Randfläche durch die Voxel

[4] https://www.ieee.org/

Ansatz der beschriebenen Entscheidungsmethode ist, den Abstand zwischen der Voxelmitte und der Polygonnetzfläche zu betrachten und diesen mit dem Wert der Nachbarschaftsvoxel zu berechnen. Liegt diese Fläche innerhalb der »26'er Nachbarschaft« wird der Voxel einbezogen.

Auch ein solches Voxelsystem stellt einen interessanten Ansatz dar. Welche der Filteransätze effektiv arbeiten und gut anwendbar sind, wird zu einem späteren Zeitpunkt diskutiert.

Ein weiteres Diskussionsfeld ist die Volumenberechnung. Zu diesem Thema werden in [Sch+15] [S.227 ff], im Kapitel »Erdmengenberechnung« ausführlich jegliche Möglichkeiten erläutert. Da es sich bei dem Schüttgut im weiteren Sinne auch um »Erdmengen« handeln könnte, lassen sich hier alle diese Möglichkeiten anwenden.
Da als Datengrundlage jedoch nur Punktdaten vorliegen, lassen sich daraus ausschließlich flächenhafte Objekte ableiten. Es ist nicht möglich Volumen- oder Profilkörper abzuleiten, da keine Informationen über die unteren, verdeckten Bereiche vorliegen. Für die Volumenberechnung flächenhafter Objekte stellt [Sch+15] drei Möglichkeiten vor. Zum einen die Berechnung aus Vierseitprismen. Nach [Sch+15] bedarf es dazu eines gleichmäßigen, quadratischen Punktrasters. Dieses gleichmäßige Raster kann in den Punktwolken nicht immer gewährleistet werden. Als zweite Variante wird die Berechnung mittels Dreiseitprismen vorgestellt. Dazu wird aus den vorhandenen Punkten ein gleichmäßiges Dreiecksnetz gebildet. Aus den entstandenen Dreiecken lässt sich, über deren Grundfläche und der mittleren Höhe der drei Eckpunkthöhen, das Volumen berechnen. Als dritte und letzte Möglichkeit wird die Erdmengenberechnung aus Höhenlinien erläutert. Bei dieser Methode werden die für die Berechnung benötigten Höhen von Eckpunkten oder Trassen aus Höhenschichtlinienmodellen gewonnen. Da alle Punktwolken-Punkte bereits Höhen besitzen entfällt diese Variante.

Für die Anwendung im vorliegenden Fall verbleiben beide Prismenvarianten, entweder über ein gleichmäßiges, quadratisches oder rechteckiges Punktraster oder über ein Dreiecksraster. Letzteres hat den Vorteil, dass es sich aus jeder vorhandenen Punktverteilung erzeugen lässt.

Auch die Darstellung, Visualisierung und Dokumentation der Volumendaten ist ein wichtiger Bestandteil des mit dieser Arbeit abzubildenden Prozesses. In dem Magazin »Informatik Spektrum«[5] der »Gesellschaft für Informatik e.V«[6] erschien, in der 22. Auflage des Magazins im Jahr 1999, ein Beitrag über »Visualisierungen und ihre Rolle in Multimedia-Anwendungen« [RS99]. Herausgeber des Magazins ist der Springer Verlag. In diesem Artikel stellen Reichenberger und Steinmetz die zentrale Bedeutung der Visualisierung, speziell die Erzeugung von Bilder für

[5] `https://www.springer.com/journal/287`, 26.03.2020
[6] `https://gi.de/`, 26.03.2020

die Vermittlung von Informationen in multimedialen Anwendungen, dar. Es werden verschiedene Ansätze der Visualisierungen besprochen, wie die gegenständliche - und die abstrakte Visualisierung sowie ihre jeweils optimalen Anwendungsfelder. Weiterhin wird auf die »Ausdrucksmächtigkeit« der Visualisierungen eingegangen und wie am übersichtlichsten Kontinua und Diskreta dargestellt werden. Auch mögliche Fehler in der Darstellung von Informationen werden erörtert ([RS99][S.11(92)]). Abschließend wird auf die Themen Typografie, Layout und Ästhetik von Grafiken eingegangen.

Dieser Artikel ist, gesehen vom Veröffentlichungsdatum her, nicht sehr aktuell. Die behandelten Themen, die herausgearbeiteten Schlussfolgerungen und Anleitungen hingegen gelten bis heute und sind so allgemeingültig gehalten, dass sie als zeitlos gültig angesehen werden können. In diesem Artikel sind die Techniken und ihre Wirkung auf den Betrachter (ergo den Softwarenutzer) umfassend beschrieben.

Eine Software, genauer die Benutzerschnittstelle, soll nach der ISO 9241-110:2006[7] auch selbstbeschreibend und lernförderlich sein. Dies lässt sich auch mit »intuitiv bedienbar« umschreiben, genau dabei kann die Visualisierung von Informationen ihre volle Ausdrucksmächtigkeit entfalten.

Ebenfalls bedarf eine Frage des Dateihandlings einer Klärung:
Welchen Geschwindigkeitsvorteil bringt Multithread Lesen und Schreiben von Dateien?
Hierzu wurden zwei aussagekräftige Internetquellen gefunden:
Im Jahr 2009 wurde eine Untersuchung zu »Multithread File I/O« [Ste09] von Stefan Wörthmüller durchgeführt. Ein Thread beschreibt laut [FH11][S. 903] den ablaufenden Programmteil, welcher einem Prozessor zugeordnet ist. Die ablaufenden Programmsequenzen können durch aus parallel, also Multithread ablaufen. Hier wird für dieses parallele Lesen und Schreiben jedoch der Begriff Multithread verwendet um die Trennung zum sequenziellen (hintereinander) Lesen und Schreiben zu verdeutlichen. Dabei wurde die Geschwindigkeit bei Multithread Lesen aus einer oder mehrerer Dateien und das Multithread Schreiben einer oder mehrerer Dateien getestet. Von Interesse für diese Arbeit ist das Multithread sequenzielle (schrittweise nach einander) Lesen und Schreiben einer einzelnen Datei. Diese Variante des Dateizugriffes ist nur unter einer speziellen Konfiguration schneller als der Zugriff mit nur einem Prozess. Die hierfür benötigte Konfiguration sind mehrere Festplatten im RAID-5-Verbund und genau vier gleichzeitige Zugriffsprozesse.

Auch ein weiterer Test von David Lozinski im Jahr 2013 [Dav13], befasste sich mit einer pas-

[7] `https://www.iso.org/standard/38009.html`, 26.03.2020

senden Fragestellung: »Was ist der schnellste Weg den Inhalt einer Textdatei mir C# zu lesen und zu verarbeiten?«. Diese Untersuchung kam zu dem Schluss, dass es am schnellsten ist, den kompletten Dateiinhalt in ein Array zu lesen und im Anschluss mit einer parallel laufenden Schleife (*Parallel.For*) zu verarbeiten.

Die vollständige Beantwortung aller Fragen ist allein durch die Literatur nicht möglich. Daher muss in den folgenden Kapiteln eine intensivere Auseinandersetzung mit den Fragestellungen und Thesen erfolgen.

1.3 Abgrenzung

In dieser Arbeit wird gezeigt, wie eine Software entwickelt wird, mit deren Hilfe der Nutzer in die Lage versetzt wird, aus einer registrierten und vorgefilterten Punktwolke in wiederholten Messungen Volumen abzuleiten. Dieses Volumen wird visualisiert und es werden Werkzeuge entwickelt, mit deren Hilfe der Nutzer dieses Volumen manipulieren kann.
Dies geschieht, in dem die aufgestellten Thesen durch Beantwortung der zugehörigen Fragen überprüft werden.
Diese Arbeit befasst sich nicht mit dem Erstellen des Quelltextes, sondern legt den Focus auf die Begleitung des Erarbeitens der Softwareabläufe. Es werden Algorithmen entwickelt und besprochen. Muss in diesem Zusammenhang Quellcode dargestellt werden, um bestimmte Abläufe zu erläutern, dann wird nur Pseudocode verwendet. Im Allgemeinen werden die einzelnen Schritte verbal beschrieben und mit Diagrammen optisch visualisiert. Die Ausgaben der Software werden als Bildschirmfotos abgebildet.

Das entstandene AutoCAD-Plugin wird in der beiliegenden CD sowohl als textbasiertes Quellcodeprojekt, sowie als lauffähige Version in einem Windows Setup-Paket enthalten sein. Diese Dateien unterliegen einem Sperrvermerk, da der entstandene Quellcode nicht unter einer quelloffenen Lizenz entwickelt wurde und somit Eigentum des Entwicklers bzw. dessen Arbeitgebers bleibt.

2 Grundlagen

Im Folgenden werden die Ausgangsbedingungen für die anstehende Problemlösung erörtert. Hierzu muss der Rahmen geklärt werden, in dem die zu entwickelnde Software arbeiten muss und in welcher Umgebung die Programmierung erfolgt. Auch die Organisation, Struktur des Programmierprojekt und die Datengrundlage, genauer die Ausgangsdaten, werden besprochen.

2.1 Rahmenbedingungen

In der K+S Minerals and Agriculture GmbH wird als Unternehmens-CAD-Software AutoCAD und Civil 3D der Firma Autodesk in der Version 2017 eingesetzt. In der zuständigen Vermessungsabteilung werden seit vielen Jahren mit Hilfe dieser Software alle CAD-Arbeiten durchgeführt. Die CAD-Software AutoCAD verfügt über eine Script-Sprache, das AutoLISP. Mit dieser Script-Sprache lassen sich vielfältige Problemstellungen lösen. Ebenfalls verfügt AutoCAD über eine gut dokumentierte Erweiterungsschnittstelle. Diese Schnittstelle lässt sich durch verschiedene Programmiersprachen ansprechen. Es ist eine Programmierung über C++ und auch über die Microsoft-.NET-Technologie möglich. Für die .NET-Technologie stehen verschiedene Programmiersprachen zur Verfügung, unter anderem C#, C++, JavaScript und weitere. Mit Hilfe der AutoCAD-API-Schnittstelle ist es möglich Erweiterungen, Plugins, zu programmieren. Diese Plugins erweitern AutoCAD mit zusätzliche Funktionalitäten oder nehmen dem Benutzer monotone Tätigkeiten ab. Empfohlen wird für die Programmierung der AutoCAD-Plugins die Entwicklungsumgebung Visual-Studio der Firma Microsoft. Auch diese Entwicklungsumgebung ist in der K+S vielfach im Einsatz. Für die derzeitige AutoCAD Version 2017 wird von Autodesk[8] die .NET-Framework-Version 4.6 und Visual-Studio in der Version 2015 empfohlen.
Die Auswertung der nativen Laserscannerdaten erfolgt in der Vermessungsabteilung mit der Software Z+F LaserControl® Office Premium, des Scannerherstellers Zoller + Fröhlich. Mit Hilfe dieser Software erfolgt die Vorverarbeitung der Laserscannerdaten, die Registrierung und der Export der Punktwolke.
Als Hardware stehen Workstations der Firma Lenovo, hier die ThinkStation P500, zur Verfügung. Diese bieten 32 GB RAM, eine 512 GB SSD-Festplatte, als Grafikkarte ist eine NVIDIA Quadro M4000 im Einsatz und als CPU ein Intel Xenon E5-1620. Diese Workstation verfügt über genügend Leistung, um mit größeren Datenmengen umzugehen.
Die in dieser Arbeit aufgeführten Tests wurden auf dem Entwicklungsrechner durchgeführt. Dies ist ein Lenovo Thinkpad P50 mit 64 GB RAM, einer 512 GB SSD-Festplatte. Grafikkarte ist eine NVIDIA Quadro M2000M und als CPU ist ein Intel I7 6820HK im Einsatz. Dieser Laptop

[8] https://knowledge.autodesk.com/de/search-result/caas/CloudHelp/cloudhelp/2017/DEU/AutoCAD-NET/files/GUID-450FD531-B6F6-4BAE-9A8C-8230AAC48CB4-htm.html, 21.03.2020

ist in der Hardwarekonfiguration mit den verwendeten Workstations durchaus vergleichbar.
Die Umsetzung der Programmieraufgaben erfolgt streng objektorientiert in der Programmiersprache C#. Alle Oberflächenprogrammierungen, sprich Benutzerschnittstellen, werden im WPF-XAML im MVVM-Pattern realisiert. Für das MVVM-Pattern exestiert eine eigene Umsetzung inklusive Grundklassen und Mediatoren. Die Organisation der Klassen findet in einer Projekt-Struktur statt. Hierzu werden Klassen mit ähnlichem Sachbezug in einem DLL-Projekt zusammengefasst. Durch diese Strukturierung erzeugt man eine hohe Wiederverwendbarkeit des erzeugten Quellcodes. Neue Programmieraufgaben nutzen diese DLL-Projekte und führen den Quellcode fort. Dabei ist eine hohe Disziplin in der Führung der API-Dokumentation der Klassen notwendig.

2.2 Struktur des Programmierprojektes

Wie bereits erläutert, erfolgt die Organisation der Klassen, des erzeugten Quellcodes, in einer Projekt-Struktur. Im Visual-Studio werden die einzelnen DLL-Projekte in einer Projektmappe zusammengefasst. In dieser Projektmappe wird ein Projekt als Startprojekt festgelegt. Dieses Projekt stellt die aufrufbaren Programmfunktionen, im Fall von AutoCAD-Plugins als Befehlsaufruf, zur Verfügung. Die einzelnen DLL-Projekte sind so gegliedert, dass zum einen innerhalb des Projektes ein sinnvoller Sachbezug herrscht. Zum anderen müssen auch Abhängigkeiten Beachtung finden. Bestimmte Funktionalitäten werden von externen Bibliotheken zur Verfügung gestellt, zum Beispiel wird die AutoCAD-API über Autodesk AutoCAD-DLLs bereitgestellt. Projekte, welche diese AutoCAD-API nutzen wollen, müssen diese DLLs einbinden. Da die DLL-Projekte in vielen, auch in »nicht AutoCAD-Plugin-Projekten«, Anwendung finden, müssen alle Funktionalitäten, welche die AutoCAD-API nutzen, in einem AutoCAD-DLL-Projekt gebündelt werden. Die folgende Tabelle liefert eine Übersicht über die vorhandenen DLL-Projekte:

Projektname	**Beschreibung**
AutoCAD - Plugin Projekt = »APP_*«	
KSRO_ScanVolume	stellt die aufrufbaren AutoCAD-Befehle zur Verfügung
Kerprojekte = »Core_*«	
CSharpTools	vielfältige grundlegende Funktionalitäten, Datentypen, Pattern, Converter
VermessungsTools	klassische 2D- und 3D-Berechnungen, Transformationen, STL-, Voxel-, Texel-Klassen
WPF / MVVM - Projekte = »WPF_*«	
CSharpWPFTools	Datentypen, Converter, Filter, eigene Controls für die WPF-Umsetzung

WPFImagesRes16	Bereitstellung gemeinsam genutzter 16-Pixel-Icons für WPF-Anwendung
WPFImagesRes48	Bereitstellung gemeinsam genutzter 48-Pixel-Icons für WPF-Anwendung
MVVM_Tools	beinhaltet die eigene Umsetzung des MVVM-Patterns incl. Wizzard's, Themen, Commands etc.
AutoCAD - Projekte = »Ext_*«	
AutoCADRequester	kapselt die Aufrufe AutoCAD eigener Dialoge, stellt häufig genutzte WPF-MVVM-Auswahldialoge zur Verfügung
AutoCADTools	Klassen und Funktionen für den Zugriff auf die AutoCAD-API incl. eigenem, globalem Transactionshandling u.v.m.

Tabelle 2.1: Liste der vorhandenen Projekte

Um die einzelnen Projekte in dem Visual Studio Projektmappen-Explorer sinnvoll anzuordnen, wurden den Projektnamen Präfixe hinzugefügt. Diese sind in der obigen Tabelle in der Zwischenüberschriftszeile ersichtlich. Durch die alphabetische Anordnung der Projekte erfolgt eine Gruppierung. Die Änderung des Projektnamens hat keinerlei Auswirkung auf die Bezeichnung der Projekte (Applikationsname, Namespace etc.).
Eine Auflistung der, für diese Software, erstellten Klassen ist in den Tabellen A.1 und A.2 in den Anlagen auf den Seiten XVII und XVIII ersichtlich.

2.2.1 Speicherung von Einstellungen / Parametern

Jede Applikation, jedes Programm muss Werte von Variablen persistent speichern, damit diese Werte bei dem nächsten Programmstart wieder zur Verfügung stehen. Um diese Funktionalität zu gewährleisten wurde eine Klasse *ClassAppSettings* entwickelt. Die Ableitung dieser Klasse wird in das Plugin eingebunden. Sie stellt Funktionen zur Verfügung, um die in der Ableitung enthaltenen Properties persistent zu speichern. Die Speicherung kann in der Windows-Registry erfolgen oder als Datei im Filesystem. Als Dateiformate stehen eine XML-Formatierung sowie das Initialisierungdatei-Format (Schlüssel=Wert) zur Auswahl. Die Konvertierung der einzelnen Datentypen in eine Textform und zurück, wird durch eine eigene Klasse »*ClassConverters*« vollständig unabhängig von getätigten Ländereinstellungen realisiert. Die Serialisierung komplexerer Datentypen wird über die Klasse »*ClassSerializeBeanXML*« abgebildet. Mit ihrer Hilfe kann der Entwickler durch Vergabe von Attributen an den Properties Eingriff auf die Datenstruktur nehmen. Einstellungen können als »global«, dass heißt, für alle Benutzer gleich, oder als »benutzerspezifisch« deklariert werden. Für letztere erfolgt die Speicherung, für jeden Win-

dowsbenutzer, in einem persönlichen Verzeichnis.

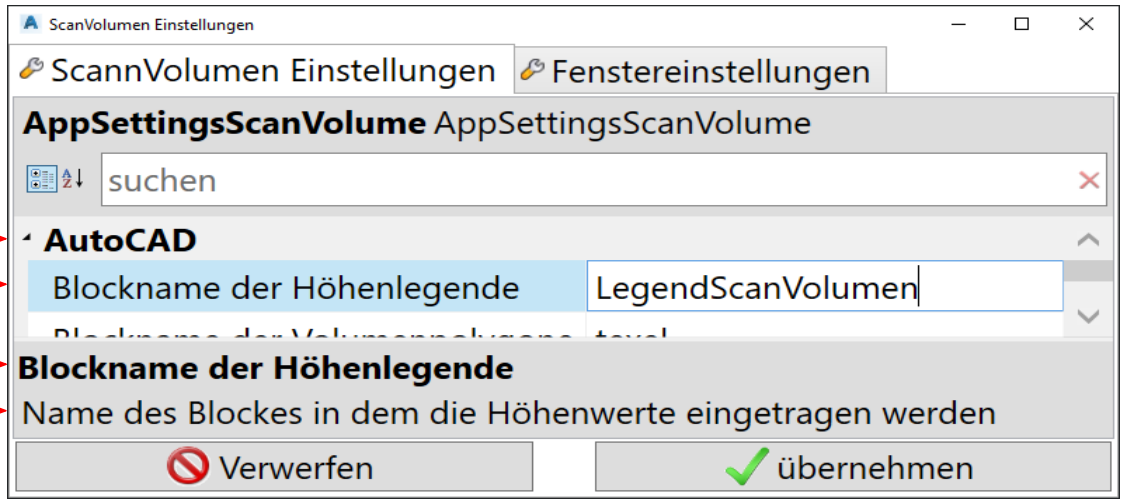
In AutoCAD-Plugins wird die persistente Speicherung der Einstellungen und Parameter als XML formatierte Datei in den dafür vorgesehenen Ordner »*C:\ProgramData\...*«, in einer eigenen Ordnerstruktur, abgelegt.

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <configuration>
3    <applicationSettings>
4      <BlocknameLegendHeight>LegendScanVolumen</BlocknameLegendHeight>
5      ...
6    </applicationSettings>
7  </configuration>
```

Ausschnitt aus der Einstellungsdatei "C:\ProgramData\.....\ScanVolume\Plugin.xml"

```
116   /// <summary>Blockname der Höhenlegende</summary>
117   private string blocknameLegendHeight;
118   /// <summary>Blockname der Höhenlegende</summary>
119   [CategoryAttribute("AutoCAD"),
120   DisplayName("Blockname der Höhenlegende"),
121   DescriptionAttribute("Name des Blockes in dem die Höhenwerte
        eingetragen werden"),
122   DefaultValueAttribute("LegendScanVolumen"),
123   ConfigurationAssign(PropertyAssign.AssignToApplication)]
124   public string BlocknameLegendHeight
125   {
126       get { return blocknameLegendHeight; }
127       set { blocknameLegendHeight = value; }
128   }
```

Ausschnitt aus der Klassendatei AppSettingsScanVolume.cs



Benutzeroberfläche zum Ändern der Einstellungen

Abbildung 2.1: Verknüpfung zwischen persistenter XML-Datei, ClassAppSettings und GUI

Die Speicherung der Einstellungen in Dateiform bietet den Vorteil, dass eine zentrale Änderung von Einstellungen aller Benutzer über z.B. ein Administratoren-Skript wesentlich einfacher

möglich ist, als bei einer Ablage der Einstellungen in der Registry. Ein weiterer Vorteil zeigt sich bei der Deinstallation. Nach dem Löschen der Daten von der Festplatte sind alle Spuren der Applikation entfernt. Bei der Ablage in der Registry verbleiben immer Geistereinträge in den Registry-Zweigen der anderen Windows-Benutzer.
Das Ändern der Einstellungen in der Benutzeroberfläche erfolgt über ein entsprechendes XAML Control, ein PropertyGrid[9]. Dieses Control liest die Gruppierungen, Anzeigenamen, default Werte und Beschreibungen aus den Attributen der C#-Property.

2.3 Datengrundlage

Die zur Volumenbestimmung benutzten Punktwolken entstehen durch Bearbeitung der nativen Laserscannerdaten mit der, vom Scannerhersteller, mitgelieferten Software. In dieser Software erfolgt die Filterung von groben Fehlpunkten. Die Daten werden registriert, sie werden an einem übergeordneten Koordinatensystem ausgerichtet. Im Falle der Produktenspeicher erfolgt eine Orientierung an einem lokalen Koordinatensystem mit Hilfe von Zieltafeln. Dadurch wird gewährleistet, dass Wiederholungsmessungen direkt miteinander vergleichbar sind.
Als Ergebnis dieser Auswertung entsteht eine Punktwolke mit 11.1 Mio. Punkten oder 44.4 Mio. Punkten, je nach verwendeter Scanauflösung. Die Punktdichte wird in der Scannersoftware nicht reduziert, da sich dabei die Rechenzeit nur in die Scannersoftware verlagern würde. Die Algorithmen zur Ausdünnung der Punktwolke sind nicht trivial und würden voraussichtlich mehr Zeit in Anspruch nehmen, als das Einlesen der kleineren Punktwolke sparen würde.
Der verwendete Laserscanner liefert zu den Koordinaten der Punkte noch die Intensität, mit welcher der Laserstrahl reflektiert wurde, die so genannte Reflektivität. Werden bei dem Scan zusätzlich mit der integrierten Kamera Bilder aufgenommen, lässt sich zu jedem Scanpunkt sein Farbwert hinterlegen. Dieser wird als RGB-Wert gespeichert. In ihm wird der Anteil des roten, grünen und blauen Lichtes in der Skala von 0 bis 255 gespeichert.
Die erzeugte Punktwolke muss aus der Scannersoftware in die Auswertesoftware übertragen werden. Dies kann über die Windows-Zwischenablage erfolgen, bei der Datenmenge ist die Zwischenablage jedoch kein geeignetes Übertragungsmittel. Auch unterstützt die Scannersoftware diese Möglichkeit nicht. Ein weiteres Problem stellt die Flüchtigkeit, dieser Variante, der Datenübertragung dar. Die Punktwolke sollte mindestens bis zum nächsten Messzeitpunkt für Kontrollzwecke verfügbar sein.
Aus diesem Grund muss die Datenübergabe von der Scannersoftware zur Auswertesoftware über eine Austauschdatei erfolgen. Maßgebend für die Auswahl des Speicherformates der Austauschdatei sind zu gleichen Teilen: eine hohe Geschwindigkeit beim Lesen der Daten, ein standardisiertes Format, welches dauerhaft lesbar ist und eines, das eine möglichst geringe

[9] https://github.com/xceedsoftware/wpftoolkit/wiki/PropertyGrid

Dateigröße ermöglicht.

2.3.1 Dateiformate der Punktwolkendateien

Die Speicherformate für Punktwolkendaten unterteilen sich in textbasierende und binäre Formate.
Textbasierende Dateiformate sind natürlicherweise lesbar, da die Zahlen direkt als Zahlzeichen gespeichert werden. Diese Formate zeichnen sich durch eine hohe Langlebigkeit aus, da sie auf Dauer lesbar sind. In ihnen werden die Punktdaten meist zeilenweise gespeichert. In einer Zeile sind die Daten zu einem Punkt als direkt lesbare Zahlen abgelegt. Diese werden durch ein festgelegtes Trennzeichen unterteilt. Um sie zu lesen muss die Zeichenkodierung der Datei bekannt sein. Da es sich bei den Punktwolkendaten hauptsächlich um Zahlenwerte handelt, wird eine falsche Zeichenkodierung bei dem Auslesen der Daten kaum zu Problemen führen. Unbestreitbarer Nachteil dieser Dateiformate ist die entstehende Dateigröße. Diese steht zu binären Formaten durchaus im Verhältnis 2:1.
Der Aufbau der binären Formate ist im wesentlichen in drei Bereiche aufgeteilt:
In der ersten Zeile befindet sich die Dateitypkennung, diese wird meist durch eine Raute »#« eingeleitet. Anschließend wird der Typ angegeben, die Version und oft eine längere Bezeichnung.
Ab der zweiten Zeile beginnt der Datenheader, der beschreibende Datenteil. In ihm werden Angaben zum Datenformat, der Datenstruktur und der Datenmenge gemacht. Anhand dieser Informationen ist das Programm in der Lage, bereits vor dem Auslesen der eigentlichen Punktdaten den benötigten Speicherplatz zu reservieren. Die dabei verwendeten Schlüsselwörter werden von den Entwickler der Formate in Eigenregie festgelegt.
Der dritte Block umfasst dann die Daten zu den Scanpunkten. Diese sind in der Regel zeilenweise organisiert und werden mit einem, dem jeweiligen Format bekannten, Schlüsselwort eingeleitet.
Bekannte Vertreter der textbasierten Punktwolkenformate sind:

- Leica Cyclone[10] PTS / PTX mit der Dateiendung »(*.pts / *.ptx)«. Beide Formate wurden durch die Firma Leica Geosystems entwickelt. Sie beinhalten keinerlei Schlüsselwörter, haben jedoch einen festgelegten zeilenweisen Aufbau. Der Unterschied zwischen einer PTS- und PTX- Datei besteht darin, dass in der PTS-Datei nur die reinen Punkte enthalten sind und in der PTX-Datei die Registrierungsinformation der Scannerstandpunkte integriert sind.

[10] Leider sind die Informationen zu Dateiformaten auf den Seiten von Leica/Hexagon nicht mehr auffindbar., 24.03.2020

- Point Cloud Data[11] mit der Dateiendung »(*.pcd)«. Dieses Dateiformat wurde durch das Open-Source-Project Point Cloud Library (PCL) [12] entwickelt und wird von vielen Scannersoftwares unterstützt. Die aktuelle Version ist 0.7.

- Wavefront OBJ[13] mit der Dateiendung »(*.obj)«. Dieses Dateiformat wurde bereits 1989 von der damaligen Firma Wavefront Technologies als Speicherformat für 3D-Modelle entwickelt. In ihm werden die Punkte als Eckpunkte abgelegt.

In binären Dateiformaten werden die Information, die Zahlen, in ihrem bitweisen Speicherabbild abgelegt. Dadurch kann eine Gleitkommazahl, welche 15 signifikante Zahlstellen enthält, in 8 Byte gespeichert werden. In Textform wären dafür 16 Byte notwendig. Diese Ersparnis an Speicherplatz wird jedoch auf Kosten der Lesbarkeit erkauft. Solche binären Punktwolken sind nur unter Kenntnis der Struktur und mit entsprechender Software oder zumindest passender Programmierbibliotheken lesbar. Daher ist es bei Verwendung von binären Dateiformaten wichtig ein Format zu wählen, welches sich bereits als Quasi-Standard etabliert hat.
Derzeit die etabliertesten binären Punktwolkenformate sind:

- ASTM 57[14] mit der Dateiendung »(*.e57)«. Dieses Dateiformat wurde bereits 2006 durch die amerikanisch/ internationale Standardisierungsvereinigung ASTM für den Austausch verschiedener von LIDAR-Daten und Daten optischer Kameras erschaffen. In dem E57-Format können sowohl Punkt- als auch Bild-Daten abgelegt werden. Dieses binäre Punktwolkenformat wird zur Zeit von beinahe jeder Scannersoftware unterstützt.

- ASPRS LAS(er)[15] mit der Dateiendung »(*.las)«. Ein im November 2011 durch die »Amerikanische Gesellschaft für Photogrammetrie und Fernerkundung« entwickelter Datenaustauschstandard für 3D-Punktdaten. Er wurde 2018 vom Open Geospatial Consortium (OGC) als Standard übernommen. Derzeit gültige Version ist 1.4. Die Punkte werden nicht direkt als native 3D-Koordinaten gespeichert, sondern mit 3D-Offset verschoben und mit einem in X,Y,Z unterschiedlichem Faktor skaliert. Dadurch werden die Punktkoordinatenwerte nur noch als 4 Byte Integer gespeichert. Die Translations- und Skalierungsparameter werden im Dateikopf gespeichert.

- Leica Cyclone[16]-PTG mit der Dateiendung »(*.ptg)«. Auch dieses Format wurde durch die Firma Leica Geosystems entwickelt. Es stellt eine binäre Variante der oben erläuterten PTS / PTX-Formate dar. Auch hier werden die Punkte ohne Translation und Skalierung gespeichert.

[11] http://pointclouds.org/documentation/tutorials/pcd_file_format.php#pcd-file-format, 24.03.2020
[12] http://pointclouds.org/about/, 24.03.2020
[13] https://de.wikipedia.org/wiki/Wavefront_OBJ, 24.03.2020
[14] https://www.astm.org/COMMITTEE/E57.htm, 24.03.2020
[15] https://www.asprs.org/divisions-committees/lidar-division/laser-las-file-format-exchange-activities, 24.03.2020
[16] Leider sind die Informationen zu Dateiformaten auf den Seiten von Leica/Hexagon nicht mehr auffindbar. 24.03.2020

Zur Auswahl stehen die Dateiformate, welche die Scannersoftware für den Export anbietet. Diese sind im wesentlichen alle hier aufgelisteten Dateiformate. Zusätzlich sei noch ein unbekanntes Dateiformat genannt:

- das »Oldenburger Scan Format«[17] mit der Dateiendung »(*.osf)«. Es ist vollständig quelloffen und erzeugt erstaunlich kleine Dateien.

Es wurde für die beiden möglichen Scanauflösungen (11.1 Mio. Punkte und 44.4 Mio. Punkte) getestet, welche Zeit der Datenexport benötigt und wie groß die entstehenden Dateien sind:

Exportformat	11.1 Mio. Punkte		44.4 Mio. Punkte	
	Größe MB	Zeit mm:ss	Größe MB	Zeit mm:ss
textbasierende Formate				
OBJ	329	0:17	1367	1:08
ASC	339	0:49	1408	3:08
PTS	404	0:57	1683	4:02
PTX	450	1:03	1836	4:21
binäre Formate				
PTG	193	0:05	802	0:19
OSF	220	0:05	876	0:20
E57	248	0:08	995	0:28
LAS	262	0:10	1090	0:40
RCP	193	0:44	528	2:26

Tabelle 2.2: Vergleich der implementierten Exportformate

Die Ergebnisse in der Tabelle 2.2 (S.16) wurden nach der benötigten Zeit für den Export der großen Auflösung sortiert. Im Wesentlichen lässt sich eine Linearität zwischen den Auflösungen, den entstehenden Dateigrößen und den benötigten Zeiten erkennen. Alle Werte steigen bei 4-facher Auflösung auch etwa um das 4-fache. Der Vorteil der Binärformate liegt eindeutig in der Dateigröße. Diese benötigen etwa nur 60% der Dateigröße von textbasierenden Dateien. Zum Exportieren in ein binäres Format wird nur etwa 17% der Zeit benötigt, die für einen textbasierenden Export aufgewendet werden muss. Dieser hohe Zeitvorsprung resultiert zum Teil aus der geringeren Dateigröße und damit aus der weniger benötigten Zeit, um die Information auf die Festplatte zu schreiben. Es lässt sich jedoch nicht der gesamte Zeitvorsprung

[17] `www.xdesy.de`, 24.03.2020

damit erklären. Ein weiterer erklärender Ansatz ist, dass für den textbasierenden Export jeder Zahlenwert formatiert ausgegeben werden muss, bei der binären Ausgabe hingegen nur die Bitwerte der Variablen geschrieben werden.
In den Test wurde zusätzlich zur obigen Liste ein weiteres binäres Format aufgenommen, das Autodesk ReCAP[18] - Punktwolken(Projekt) - Format, »(*.rcp)«. Darin werden die einzelnen Scans in einen separaten Unterordner als »(*.rcs)«-Dateien binär gespeichert. Dieses Format lässt sich sehr performant in Autodesk AutoCAD einlesen und man kann auch mit sehr großen Punktwolken noch flüssig arbeiten. Leider ist die Struktur des Dateiformates nicht offengelegt.

Für die Erstimplementierung wurde als Austauschformat das Waveform OBJ-Format gewählt, da es ein standardisiertes Format ist. Besser wäre es, ein binäres Format zu wählen, jedoch gibt es für das E57-Format noch keine nutzbare .NET-/ C#-Implementierung. Die Umstellung auf das Austauschformat E57 wird eine zukünftige Aufgabe werden.

[18] `https://www.autodesk.de/products/recap/features`, 25.03.2020

3 Filterung der Punktwolke

Dieses Kapitel befasst sich ausschließlich mit der **These 1:** »***Die Punkte können automatisch in zwei Gruppen unterteilt, genauer klassifiziert, werden, in Produkt und Nichtprodukt.***«
In der Literaturrecherche (Kapitel 1.2, S.2) wurde die *Frage 1.1: »Ist es möglich, aus den empfangenen Daten (Intensität, Farbe) auf den Typ der Oberfläche (Produkt / Nichtprodukt) zu schließen?«* verneinend beantwortet. Da sich auf allen, im Produktenspeicher befindlichen, Oberflächen Staub aus Abrieb vom Produkt absetzt, kann nicht mit abschließender Sicherheit aus den Reflektionswerten eines Scanpunktes auf dessen Zugehörigkeit geschlossen werden. Auch sendet der Laserscanner nur eine Frequenz aus, damit ist keine multispektrale Auswertung möglich.
Mit der *Frage 1.2: »Welche anderen Möglichkeiten der Klassifizierung gibt es und sind diese praktisch umsetzbar?«* befasste sich im Ansatz die Analyse der Artikel [Liu+20] und [Sch+13]. Diese Art der Filterung ist für die derzeitige Rechentechnik zu aufwendig, im Grundsatz jedoch durchaus als Lösungsansatz geeignet.
Aus der Betrachtung von Möglichkeiten der Auswahl von Elementen in einer CAD-Software heraus formte sich der Gedanke, »*Scanpunkte auszuwählen, welche sich in einem Positivkörper befinden*«, analog den Booleschen Volumenkörper Operationen. Da die jeweiligen Scanepochen immer an den gleichen Passpunkten ausgerichtet werden, tragen sie auch immer das gleiche geodätische Datum, den gleichen Koordinatenbezug. Damit befinden sich die nicht benötigten Elemente, Wände, Dächer etc. immer an den gleichen Koordinaten, jedoch mit einer gewissen Messunsicherheit behaftet. In dem vorhergehenden Studienprojekt wurde eine 3D-Lagesicherheit von <20 mm nachgewiesen. Also sollte es doch möglich sein, einen Volumenkörper zu konstruieren, der den maximalen Raum beschreibt, in dem sich das Produkt befinden kann. Dieser Körper dient dann als Auswahlkriterium für die Klassifizierung in Produkt und Nichtprodukt, er wird in Folge als *Sammelkörper* bezeichnet.
Für die Konstruktion eines solchen Sammelkörpers wird die Erstmessung eines Produktspeichers genutzt. Die Punktwolke wird in das Autodesk ReCAP-Format exportiert. Dieses Format kann dann in Autodesk AutoCAD eingelesen werden. In einer rein manuellen Tätigkeit wird dann ein solider Volumenkörper erzeugt. Dies erfolgt mit Hilfe von im AutoCAD enthaltenen Volumenkörperwerkzeuge. In diesen Sammelkörper muss die beschriebene Messunsicherheit einfließen. Auch ragen bei einigen Wandkonstruktionen aus Stahlblech die Schraubenköpfe etwa 15 mm in den Innenraum. Um dies darzustellen, halten die Außenflächen des Sammelkörpers 40 mm Abstand zu den in der Punktwolke erkennbaren Außenwänden, Dachbalken und anderen Störobjekten.
Die Sammelkörper müssen nach der reinen Konstruktion einem Feinschliff unterzogen werden, da in den einzelnen Produktspeichern an unterschiedlichen Stellen Störkörper in den Raum ra-

gen, z.B. Seile oder Kabel, die von der Dachkonstruktion herunter hängen oder Abdeckgummis für Fensteröffnungen. Diese Störkörper müssen so herausgearbeitet werden, dass sie nicht in den Sammelkörper hineinragen. Zwei Beispieldarstellungen sind in den Anlagen, Abbildungen A.6 und A.7 auf der Seite XIX aufgeführt.
Der zeitliche Aufwand ist stark von der Form und Struktur des jeweiligen Produktspeichers abhängig, eine einfache Form (etwa eine Klötzchenhausform) bedarf etwa 10-15 Minuten, eine komplizierte Form auch bis zu 60-240 Minuten Arbeitszeit. Da dieser Sammelkörper für jede weitere Messepoche Verwendung findet, ist dessen Konstruktion eine einmalige Tätigkeit..

Das Hauptproblem bei dieser Sammelkörperauswahl besteht darin, dass es derzeit, bei der zur Verfügung stehenden Software, keine Auswahlmöglichkeit »Scanpunkt in Volumenkörper« gibt. Diese muss in die Software implementiert werden. Hierzu bedarf es eines effektiven, schnellen Algorithmus, da diese Entscheidung in der großen Punktwolke etwa 44 Mio. mal getroffen werden muss.
Um einen Test »*Punkt im Volumenkörper*« durchzuführen, kann man testen, ob sich der Punkt auf den Innenseiten der Außenflächen befindet ([Lui16]). Dieser Test funktioniert nur bei regelmäßigen Volumenkörpern zuverlässig. Eine weitere Möglichkeit wäre ein abgewandelter »Punkt-im-Polygon-Test«. Dazu müsste der Schnittpunkt eines Vektors vom Punkt aus (z.B. nach +Z) mit den Außenflächen gerechnet werden. In einer zweiten Berechnung kann dann in der Außenflächenebene ein Test »Punkt-im-Polygon« durchgeführt werden um festzustellen, ob der Schnittpunkt tatsächlich in der Außenfläche liegt. Es ist bei einem solchen Prüfmechanismus unerheblich, welche der beiden vorgestellten Varianten benutzt werden. Es wird immer ein sehr hoher Rechenaufwand entstehen.
Über eine freie Internetrecherche wurde die Idee des Nutzen eines Voxelsystems geboren. Mit Hilfe solcher Voxelsysteme werden Volumenkörper vereinfacht dargestellt. Ein Voxel wird nach [IW94][S. 296] als Äquivalent eines Pixels im dreidimensionalen Raum angesehen. Definiert wird es als unteilbares Volumenelement in der Form eines Würfels. Die Seiten eines Voxels verlaufen parallel zu den Koordinatenachsen. Durch dieses Anschmiegen an die Koordinatenachsen besteht ein direkter Zusammenhang zwischen den Koordinaten eines Scanpunktes und dem zugehörigen Voxel. Damit sollte es sehr schnell möglich sein zu tes-

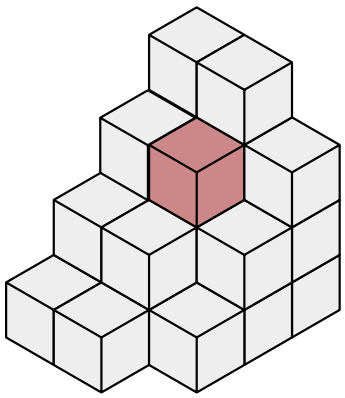


Abbildung 3.1: Visualisierung einer Voxelstruktur
Quelle:
https://commons.wikimedia.org/wiki/File:Voxels.svg

ten, ob ein Scanpunkt in dem Sammelkörper liegt.

3.1 Das Voxelsystem

Nun gilt es, den Sammelkörper in ein Voxelsystem umzuwandeln. Hierzu sind ein paar Grundfragen zu klären:

- Dürfen Eckpunkte der Voxel über den Sammelkörper hinausragen?
  Antwort: Nein, auf keinem Fall, da die Außenseiten des Sammelkörper bereits maximal 40 mm Sicherheitsabstand zu Störobjekten besitzen.

- Mit welcher Genauigkeit und daraus resultierend mit welcher Voxelgröße soll der Sammelkörper gefüllt werden?
  Antwort: Es sollen an den Rändern und Ecken keine Hohlräume entstehen, die größer als 50 mm sind. Dadurch wird gewährleistet, dass an jeder Außenseite der letzte verwendete Scanpunkt maximal 90 mm entfernt liegt. Daher wird die Voxelgröße auf 50 mm festgelegt.

Das Voxelsystem beschreibt immer ein dreidimensionales boolesches Array, mit der Zellengröße des Voxels, in dem nur die verwendeten Voxel den Zellwert »true« besitzen. Die Länge, Breite und Höhe ergibt sich aus der Größe des in Voxel zu rechnenden Objektes. Das Voxelarray umfasst immer ein Quader mit den jeweiligen Maximalwerten des Objektes.
Da die Ausrichtung der Voxel an dem Koordinatengitter erfolgt, muss bei der Festlegung des Koordinatenbezuges der Produktspeicher darauf geachtet werden, dass bei rechteckigen Objekten die Außenseiten möglichst parallel zu den Koordinatenachsen verlaufen. Diese Entscheidung optimiert die Genauigkeit der Voxelerzeugung.
Die Außenmaße der Produktspeicher liegen in den Größenordnungen (L x B x H) von 56 m x 56 m x 19 m bis 87 m x 36 m x 10 m. Bei der geforderten 50 mm Voxelgröße resultieren daraus Voxelarray mit einer Dimension von 1120 x 1120 x 380 Zellen (476.7 Mio. Zellen) bzw. 1740 x 720 x 200 Zellen (240.6 Mio. Zellen).
Während der Literaturrecherche ist bereits aufgefallen, dass die beschriebenen Verfahren zur Voxelerzeugung kaum die geforderte Genauigkeit »Keine Ecke außerhalb des Sammelkörpers« erreichten.
Innerhalb der oben erwähnten Internetrecherche wurde die Software »binvox« von Patrick Min ([Pat20] gefunden. Mit Hilfe dieser Software wird ein 3D-Modell aus einer STL-Datei[19] in ein Voxelsystem umgesetzt. Die Speicherung erfolgt als lineares binäres Array. Bei eingehenden Tests ist aufgefallen, dass die Voxelsysteme nicht gefüllt werden und das trotz Parameter »-e« (*exact voxelization*) die Voxel nicht immer vollständig innerhalb der Flächen enden. Des weiteren ist es nicht möglich, eine gewünschte Voxelgröße anzugeben.

[19] siehe Abkürzungsverzeichnis S.V

Der Sammelkörper wird als primitiver Volumenkörper in Autodesk AutoCAD erzeugt. Um die Voxelerzeugung in einer externen Software durchzuführen, muss der Sammelkörper exportiert werden, dies geschieht mit einem in Autodesk AutoCAD bereits implementiertem STL-Exportfilter. In diesem STL-Format werden Volumenkörper durch Ihre Außenflächen inklusive der Flächennormalen dargestellt. Das resultierende Dateiformat ist ein binäres Format. Bei der Konstruktion des Sammelkörpers muss darauf geachtet werden, dass keine Zylinder als Volumenkörper Verwendung finden, denn für diese Rundungen ist die AutoCAD-API nicht in der Lage, Ersatzflächen zu berechnen, sie werden bei dem Flächen-Rendern konsequent ignoriert. Abhilfe schafft hierbei, die gewünschten Zylinder durch einen segmentierten Kreis, aus welchem per Extrusion[20] ein zylinderähnlicher Volumenköper erstellt wird, zu ersetzten.
Um das Resultat optisch zu validieren und Tests durchzuführen, wurde eine Funktion zum Importieren von STL-Dateien in das Plugin realisiert. Im Projekt wurde eine Klasse »*STLReader*« implementiert, diese Klasse dient als Datenhalter der Dreiecksflächen.

3.1.1 Programmierung des Voxelsystems

Grundlegend wurde der Lagebezug eines einzelnen Voxel auf die dem Koordinatenursprung an nächsten gelegene Ecke festgelegt. Dadurch hat das erste Voxel den Koordinatenursprung als gültige dreidimensionale Koordinate.
Die Grundlage für die Voxelberechnung ist der »Punkt-im-Polygon-Test nach Jordan«. Dieser prüft, ob sich ein Punkt in einem ebenen Polygon befindet. Dazu wird von dem Punkt ein Vektor in beliebiger Richtung, in diesem Fall wurde die Vektorrichtung parallel zur Y-Achse in Richtung positiv-Y, gewählt. Von dem Punkt aus werden in der Vektorrichtung alle Linien geschnitten und die Schnittpunkte gezählt. Als Kriterium für die Entscheidung, ob der Punkt innen oder außen liegt, wird die Anzahl der Schnittpunkte benutzt. Ist sie eine ungerade Zahl, dann ist der Punkt innerhalb, bei gerader Anzahl ist der Punkt außerhalb.

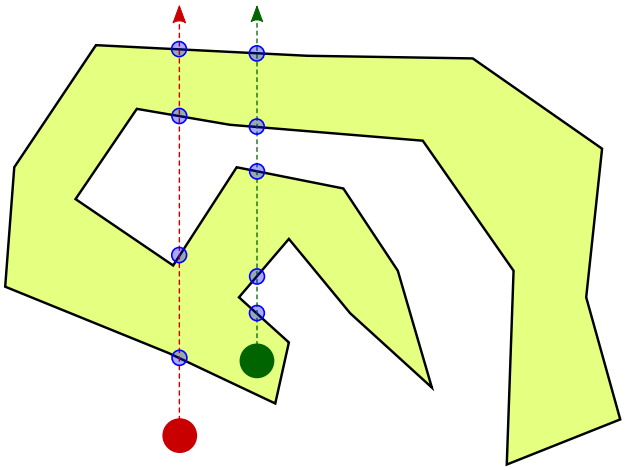


Abbildung 3.2: Visualisierung des Punkt-im-Polygon-Tests nach Jordan

[20] siehe Glossar, S.VII

Der Test verfügt über die mathematische Genauigkeit, korrekt über »innen und außen« zu entscheiden, jedoch hat auch dieser Test zwei Logik-Probleme mit besonderen Linienanordnungen. Bei dem Antreffen genau dieser Besonderheiten lässt sich nicht eindeutig anhand der Anzahl der Schnittpunkte festlegen, ob sich der Testpunkt innerhalb oder außerhalb der Polygonfläche befindet.

Diese Besonderheiten sind:

- Die Schnittachse schneidet einen Polygon-Eckpunkt oder
- In der Schnittachse liegt exakt ein Polygonteilstück, eine Linie.

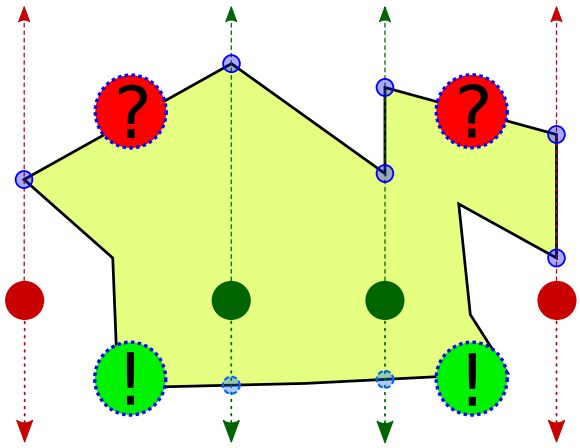


Abbildung 3.3: Darstellung logischer Probleme im Punkt-im-Polygon-Test nach Jordan

Die Ursache der beiden benannten Probleme ist die gleiche: Liegt der Schnittpunkt auf einem Polygoneckpunkt, existieren zwei Linien, bei denen die Schnittberechnung den selben Schnittpunkt liefert. Werden demnach zwei Schnittpunkte erzeugt, dann wird einem tatsächlich innen liegender Testpunkt, welcher in der Schnittachse ein Polygoneckpunkt hat, zwei Schnittpunkte zugewiesen und wird als außen liegend erkannt. Wird jedoch nur ein Schnittpunkt erzeugt, dann wird der tatsächlich außen liegende Testpunkt als innen liegend erkannt.
In der zweiten beschriebenen Besonderheit kann, lediglich mit dem genau auf der Schnittachse liegenden Polygonteilstück, kein Schnitt berechnet werden. Bei der Schnittberechnung entstehen ebenfalls bei innen und außen liegenden Testpunkten die gleiche Anzahl von Schnittpunkten. Anhand der folgenden Grafik lässt sich das Paradoxon sehr gut visuell erfassen.

Ein Lösungsansatz ist, die Schnittpunkte in beide Richtungen zu analysieren. Wird in der anderen Richtung ebenfalls ein Polygoneckpunkt als Schnittpunkt identifiziert, muss die Schnittachse etwas gedreht werden.
Für die Nutzung des Punkt-im-Polygon-Test im Voxelsystem sind einige Vorbereitungen zu treffen. Alle Eckpunkte der zu erzeugenden Voxel liegen in Ebenen mit dem parallelen Abstand

einer »Voxelgröße«. Es müssen Schnittpolygone in jeder dieser Ebenen erzeugt werden, in denen dann der Punkt-im-Polygon-Test erfolgen kann. Als Grundlage für die Schnittberechnung eignen sich die für den Export erzeugten STL-Daten, daher wurde die STL-Klasse um entsprechende Funktionen erweitert und zusätzliche Datenklassen erzeugt.
Die aus dem Schnitt der STL-Fläche mit der Schnittebene entstehende Gerade wird nicht als Ebenenschnitt gerechnet, denn nicht parallele Ebenen schneiden sich immer. Es muss der Schnitt der Außenlinien der STL-Fläche und der Schnittebene berechnet werden. Damit ist eine Kontrolle der Schnittpunkte auf »Schnittpunkt liegt innerhalb einer Außenseite des STL-Dreieckes« möglich. Die entstehenden Schnittlinien werden anschließend zu geschlossenen Polygonen geordnet.

Es wurde festgelegt, dass die Lage auf dem Grenzpolygon als »innen liegend« zu werten ist.

Um die Punkt-im-Polygon-Funktion zu optimieren, wird die Schnittrichtung in die Y-Achse des Koordinatensystems gelegt. Zur schnelleren Suche der Schnittlinien, werden die Verbindungslinien nach ihrem X-Wert ihrer Eckpunkte sortiert und als gerichtet (X-Anfang ist kleiner als X-Ende) gespeichert. Es werden nur Schnittpunkte mit den Linien berechnet, bei denen sich der Eckpunkt-X-Wert zwischen Linien X-Anfang und X-Ende befindet. Wird die erste Linie gefunden, bei der ihr X-Anfang größer ist als der Eckpunkt-X-Wert, so wird die Suche abgebrochen.

Der Berechnung des Schnittpunktes sind einige Logik-Prüfungen vor- und nachgeschaltet um die Besonderheiten abzubilden.
Als erstes wird geprüft, ob der Testpunkt einer der beiden Linieneckpunkte ist. Danach wird geprüft, ob der Delta-X-Wert der Linie genau 0.0 ist. Sollte dies der Fall sein, tritt das oben beschriebene Problem auf. Es wird trotzdem geprüft, ob sich der Testpunkt innerhalb der Linie befindet, damit würde er auf der Polygonlinie liegen. Ist der Delta-X-Wert ungleich 0.0, wird die Schnittberechnung durchgeführt und es wird geprüft, auf welcher Seite sich der Schnittpunkt befindet oder ob er gar auf der Polygonlinie (Abstand = 0.0) liegt. Damit eine gewisse mathematische Toleranz in den erforderlichen Berechnungen und Variablentypen dargestellt werden kann, erfolgt jeder Test mit einer erlaubten Abweichung von $1 \cdot 10^{-3}$ mm.
Zur Minimierung des oben beschriebene Punkt-im-Polygon-Problems, werden die Schnittpunkte nach Y-Wert größer oder kleiner Y-Testpunkt getrennt gespeichert. Zusätzlich werden zwei Marker geführt, die das Auffinden eines Polygoneckpunktes in der jeweiligen Schnittrichtung anzeigen. Sind beide Marker gesetzt wird die Suche abgebrochen. Dann wird in einer neuen Suche die Schnittrichtung um 1 Gon gedreht. Dies geschieht iterativ so oft, wie auch in der gedrehten Richtung in beiden Schnittrichtungen Polygoneckpunkte angetroffen werden. Dadurch funktioniert jedoch der X-Filter für den vorzeitigen Abbruch nicht mehr und es müssen

alle Linien geschnitten werden.
Jedes Voxel besitzt 8 Eckpunkte. Jeder dieser Eckpunkte wird, außer an den Außenseiten, wiederum in 8 Nachbar-Voxel verwendet. Daher wird für die bereits berechneten Voxeleckpunkte ein Puffer bereitgestellt, der den Wahrheitswert speichert und die Information, ob er bereits einmal berechnet wurde. Für die exakte Berechnung wird beim ersten außerhalb liegenden Punkt des Voxels abgebrochen.
Das Voxelarray wird als Datentyp *BitArray* (eindimensionales Array aus Bits) realisiert. Das Voxelsystem bildet ein dreidimensionales Array ab. Dieses Array wird mit ganzzahligen Indizes in den drei Koordinatenachsen abgebildet. Da der erste Indexwert »0« sein muss, werden die realen Koordinaten um den Minimalwert der Boundingbox des Sammelkörpers reduziert. Dies wird in den Werten $verschiebung^X$, $verschiebung^Y$ und $verschiebung^Z$ abgebildet. Um die Koordinaten in Achsindizes umzurechnen werden folgende drei kleine Formeln benutzt:

$$Index^X = ((X - verschiebung^X)/voxelgroesse) \tag{3.1}$$

$$Index^Y = ((Y - verschiebung^Y)/voxelgroesse) \tag{3.2}$$

$$Index^Z = ((Z - verschiebung^Z)/voxelgroesse) \tag{3.3}$$

In der programmiertechnischen Umsetzung erfolgt vor der Zuweisung ein Casten auf den Datentyp *INT32*. Dadurch werden die Nachkommastellen abgeschnitten und es wird gewährleistet, dass mit dem Erreichen der ersten Voxelgröße genau der nächste Indexwert folgt.
Aus den drei Achsindizes wird danach für das *BitArray* der lineare Index wie folgt berechnet:

$$\begin{aligned} linearIndex^{Voxel} = Index^Y + (Index^Z * arrayLaenge^Y) + \\ (Index^X * arrayLaenge^Y * arrayHoehe^Z) \end{aligned} \tag{3.4}$$

Im Quellcode wird vor der Berechnung eine Prüfung der angegebenen Achsindizes durchgeführt, denn sie dürfen sich nur in der Größe der Boundingbox bewegen.
Damit die Berechnung der Voxeleckpunkte mit der maximal zur Verfügung stehenden Rechnerleistung vollzogen werden kann, wurde im Quelltext eine Parallelisierung über eine Partitionierung der Anzahl Schnittebenen (Z-Achse) durchgeführt. Dadurch ist auch gewährleistet, dass es zu keiner Blockade bei dem Zugriff auf den gemeinsamen Schnittpunktpuffer geben kann. Partitionierer (*System.Collections.Concurrent.Partitioner*) stellen im .NET-Framework Funktionen zur Verfügung, welche den Inhalt von Arrays auf die Anzahl der im Computersystem zur Verfügung stehenden »gleichzeitig bearbeitbaren Abläufe« (engl. Threads) aufteilt. Diese Pakete werden anschließend in einer »Parallel.ForEach«-Schleife von allen Threads gleichzeitig abgearbeitet. Die Speicherung der Schnittpunkte muss in einer threadsicheren Liste erfolgen, da es bei Tests zu Laufzeitfehlern kam.

Für die Speicherung wurde der Datentyp *System.Collections.Concurrent.BlockingCollection* gewählt. Dieser Datentyp ersetzt die vorher verwendete *System.Collections.Generic.List*. In der *BlockingCollection* sind Funktionen wie clear (leeren) und contains (enthält?) nicht implementiert. Daher wurde eine Klasse von dem *BlockingCollection* abgeleitet und die benötigten Funktionen eingefügt.
Durch den hohen Rechenaufwand schläft der Computer während dieser Aktion auch anscheinend ein. Daher wird der Benutzer vor Beginn der Aktion darauf hingewiesen.

Die Optimierungen wurden exemplarisch an zwei Produktspeichern getestet, in der nicht optimierten Version ist bereits die Kontrolle auf den ersten außenliegenden Eckpunkt implementiert. Daher ist die Anzahl der maximal zu prüfenden Eckpunkte nur theoretisch richtig.

Folgende Beschreibung lässt sich den beiden Produktspeichern zuordnen:

- Rechteckiger Speicher, leichte Dachform, Linienanzahl der Schnittpolygone zwischen 10 und 20 Linien, Voxeldimensionen 1779 x 729 x 251 Zellen = 320 030 271 Voxel zu berechnen, theoretisch 2 560 242 168 Eckpunkte.
- »Runder« Speicher (16-eckig), der Umring wurde manuell digitalisiert, »Kegel«-Dachform auch (16-Eckig), Linienanzahl der Schnittpolygone zwischen 32 und 200 Linien, Voxeldimensionen 1121 x 1114 x 383 = 478 288 102 Voxel zu berechnen, theoretisch 3 826 304 816 Eckpunkte.

Es wurden drei Szenarien in den beiden Speichern untersucht:

- Testszenario 1 : serielle Abarbeitung des Voxelarray, ohne Eckpunktpufferung
- Testszenario 2 : serielle Abarbeitung des Voxelarray, mit Eckpunktpufferung
- Testszenario 3 : parallele Abarbeitung des Voxelarray, mit Eckpunktpufferung

Testszenario	Rechteckspeicher		Rundspeicher	
	Anzahl berechneter Schnittpunkte	Zeit hh:mm:ss	Anzahl berechneter Schnittpunkte	Zeit hh:mm:ss
1. seriell ohne	2 190 238 572	00:15:12	1 606 607 691	01:36:16
2. seriell mit	551 327 333	00:07:02	485 412 013	00:18:06
3. parallel mit	551 327 333	00:03:35	485 412 013	00:04:45

Tabelle 3.1: Vergleich der Ergebnisse des Leistungstests der Voxelberechnung

Die extrem lange Zeit des Rundspeichers im Testszenario 1 ist schlicht der Anzahl der Schnittlinien geschuldet, denn in dem digitalisierten unteren Bereich des Speichers müssen sehr viele Linien getestet werden. Die erreichte Geschwindigkeit des Voxelalgorithmus hält die benötigte Zeit in einem erträglichen Rahmen. Die getestete »binvox.exe«[Pat20] benötigt bei gleichen Voxelarraygrößen und im »exact«-Modus ebenfalls etwa 5 Minuten. Zu dem muss dieser Arbeitsschritt in jedem Produktspeicher nur einmalig ausgeführt werden. Die Voxeldaten werden gespeichert und bei der nächsten Volumenberechnung nur noch eingelesen.
Die grobe Struktur der Funktion »*Voxelsystem erzeugen*« ist in dem Ablaufdiagramm A.2, im Anhang, auf der Seite XIII dargestellt. In dem Ablaufdiagramm A.3 auf der Seite XIV ist der Programmschritt »*Voxel erzeugen*« detaillierter dargestellt.

3.1.2 Speicherung der Voxeldaten

Um die Voxeldaten zu speichern wurde ein binäres Format gewählt, da sich Bitarrays in binärer Form platzsparender ablegen lassen. Wie auch in anderen binären Dateiformaten üblich, wird am Dateianfang ein textbasierter Kopf geschrieben, dem dann die eigentlichen Binärdaten folgen. In dem Kopf sind Angaben zum Dateityp, der Voxelgröße, der Array-Dimensionen und zur Verschiebung des Nullpunktes lesbar gespeichert. Der binäre Datenteil wird von der Zeile »*binarydata:*« eingeleitet.

```
1 #exactbinvox3d 1
2 voxelsize 0.05
3 dim 1121 1114 383
4 translate 4.99158143612067 2.81214143764996 2.80
5 binarydata:
6 ...
```

Listing 3.1: Dateikopf einer *.binvox3d-Datei

Um das *BitArray* zu speichern wird eine Byte-Paar-weise Speicherung verwendet. Das erste Byte setzt den booleschen Wert (1 oder 0) und das zweite Byte die Anzahl der mit dem booleschen Wert behafteten Bits, maximal jedoch 255. Spätestes bei jedem Wechsel des booleschen Wertes wird ein neues Byte-Paar geschrieben. Ist das Voxelsystem sehr »zerklüftet« (viele kurze Wechsel), dann dürfte diese Art der Speicherung keine sehr effektive Speichervariante sein. Da die zu erwartenden Voxelsysteme sehr homogen sein werden, solle sie doch recht effektiv sein. Die oben im Leistungstest verwendeten Beispiele liefern Dateigrößen von:

- Rechteckiger Speicher, 320 030 271 Voxel = 3.4 MB
- Runder Speicher , 478 288 102 Voxel = 4.2 MB

Die erzeugten Dateien erhalten die Dateiendung »binvox3d« um eine Zuordnung zu einem

3D-Voxelsystem zu ermöglichen. Da das Speichern der Daten sequenziell erfolgt, werden die Daten nicht parallel geschrieben.

3.2 Filterung mit Hilfe des Voxelsystem

Um die Filterung der Voxel durchzuführen, wird das Voxelsystem aus der »*.binvox3d«-Datei geladen. Im Anschluss erfolgt die Auswahl der Punktwolkendatei. Diese Waveform-OBJ-Datei wird als Textdatei zeilenweise gelesen, der gelesene Datensatz wird sofort verarbeitet. Die ausgelesene Textzeile wird mit Hilfe von einem Trennzeichen zerlegt, dieses ist bei Waveform-OBJ ein Leerzeichen. Die drei gewonnenen Textteile werden in Double-Werte umgewandelt und stellen den X-, Y- und Z-Wert des Scanpunkt dar.
Mit Hilfe der Formeln 3.1-3.3 (S. 24) werden daraus die Achsindizes gerechnet. Die Entscheidung, ob der Punkt zur Produktoberfläche gehört, wird durch Abfrage des *BitArray* der Voxelklasse getroffen. Durch die Formel 3.4 (S.24) wird dessen linearer Index berechnet. Ist das Bit 1 bzw. *true* gesetzt, ist es ein Oberflächenpunkt und kann für die Volumenberechnung genutzt werden.

Abschließend lässt sich feststellen, die *Frage 1.2: »Welche anderen Möglichkeiten der Klassifizierung gibt es und sind diese praktisch umsetzbar?«* kann wie folgt beantwortet werden:

Die Klassifizierung von Scanpunkten mit Hilfe von voxelbasierenden Sammelkörper-Filter bietet eine effektive und schnelle Möglichkeit, Punktwolken in die zwei gewünschten Gruppen »Produkt und Nichtprodukt« einzuteilen, mit der Einschränkung, dass zu Beginn erst einmal ein Sammelkörper erzeugt, genauer konstruiert werden muss. Da diese Aufgabe auch recht zeitintensiv werden kann, bleibt diese Version eher den regelmäßig wiederholten Messungen vorbehalten.

Somit lässt sich auch die **These 1: »*Die Punkte können automatisch in zwei Gruppen unterteilt, genauer klassifiziert, werden, in Produkt und Nichtprodukt.*«** nur als »zum Teil bewiesen« betrachten. Es ist zwar eine automatische Klassifizierung, die jedoch nur unter bestimmten Bedingungen (Erzeugung des Sammelkörpers) funktioniert.

4 Volumenberechnungen

Die Volumenberechnung erfolgte bisher auf Grund der tachymetrischen Einzelpunktaufnahme. Dies bedeutet, dass nur Punkte aufgenommen wurden welche für die Oberflächendarstellung geeignet und wichtig waren. Grundsätzlich musste der äußere Umring des Produkthaufens immer bestimmt werden. Die dafür notwendigen Punkte wurden während der Tachymetermessung erfasst. Anschließend musste der Umring zum Teil manuell korrigiert werden. Das Volumen wurde dann nur bis zu dem Umring bestimmt, jedoch auch bis zu einer definierten Null-Höhe. In einer Punktwolke ist man, auf Grund der hohen Punktanzahl, nicht mehr in der Lage diesen Umring zu bestimmen. Daher wird hier das Volumen immer bis zur äußeren Begrenzung des Voxelsystems berechnet. Es wird auch immer bis zur lokalen Höhe 0.0 m berechnet. Dadurch ist eine so genannte Nullmessung, die Bestimmung des Restvolumens eines »leeren« Produktspeichers, unerlässlich. Das reine Produktvolumen ergibt sich dann aus der Differenz des aktuellen Aufmaßes und der Nullmessung.

Im Kapitel 1.1 wurde die **These 2:** »***Aus den gemessenen Punkten wird das Volumen effektiv und fehlertolerant berechnet.***« aufgestellt. Diese gilt es nun zu verifizieren oder auch zu falsifizieren. Dazu wurde im Kapitel 1.2 (Literatur) bereits herausgearbeitet, dass es für Berechnung von Volumen aus großen Punktmengen im wesentliche zwei Verfahren existieren:

- Volumenberechnung aus Vierseitprismen bei gleichmäßigem Punktraster oder
- Volumenberechnung aus Dreiseitprismen.

Die Berechnung aus Dreiseitprismen ist in der Vermessung ein Standardverfahren. Die daraus entstehenden Dreiecksnetze werden vielfältig eingesetzt, unter anderem zur Darstellung von Oberflächenmodellen, den digitalen Geländemodellen (DGM). Solche DGM werden auch in der Software Autodesk Civil 3D zur Volumenberechnung eingesetzt und sind auch in der bisherigen Technologie zum Produktspeicheraufmass die Methode zur Volumenberechnung.
Die Bildung des Dreiecksnetzes wird in der Civil 3D-Software nach der Methode der Delaunay-Triangulation vollzogen. In [RM04][S.49-52] wird die Konstruktion des Dreiecksnetzes ausführlich erörtert. In Kürze lässt sich das Verfahren wie folgt beschreiben: Aus den gegebenen Punkten wird ein Voronoi-Diagramm gebildet. Am ehesten lässt sich das Ergebnis des Algorithmus mit einer Seifenblasenstruktur beschreiben (siehe Abbildung 4.1). Die Dreiecke werden dann nur zwischen den Punkten gebildet, welche eine gemeinsame Grenze besitzen.

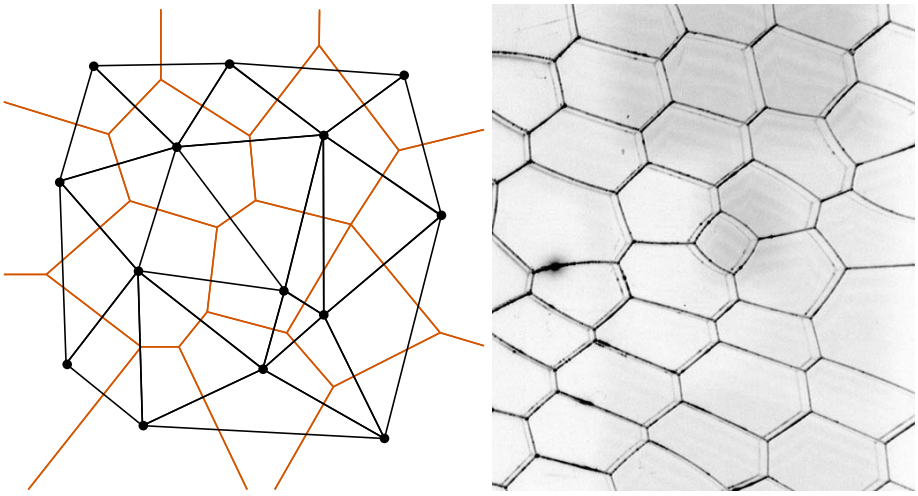


Abbildung 4.1: Vergleich zwischen Delaunay-Triangulation mit hinterlegtem Voronoi-Diagramm (links) und einer natürlichen Seifenblasenstruktur (rechts)

Quelle: links=https://de.wikipedia.org/wiki/Datei:Voronoi-Delaunay.svg

rechts=https://commons.wikimedia.org/wiki/File:2-dimensional_foam_(colors_inverted).jpg

Eine andere Möglichkeit, solche Punkt-Triple (eine Dreiergruppe) zu bilden, ist die Auswahl der nächstgelegen 2 Punkte. Dabei wird darauf geachtet, dass in dem Umkreis, welcher sich aus diesen drei Punkten bildet, kein weiterer Punkt hineinragt. Ist das doch der Fall, wird eine andere Kombination gewählt.

Aus einer Dreiecksmasche wird dann mit deren Flächeninhalt und der mittleren Höher ihrer drei Eckpunkte das Volumen berechnet. Solche Prismen-Volumenberechnungen erfolgen im Allgemeinen immer bis zur Höhe 0.0. Soll eine andere Höhe die Bezugsebene sein, wird sie von allen Punkthöhen abgezogen.

Die digitalen Geländemodelle haben ein entscheidendes Problem, es ist nicht möglich einen Überhang (vergleichbar mit einem Vordach am Hauseingang) darzustellen. Durch die Vermaschung entstehen völlig falsche Oberflächen. In solchen Fällen behilft man sich durch wissentliche Manipulation der Punktdaten. Der aus der Vogelperspektive verdeckte Unterkantenpunkt und auch der sichtbare Oberkantenpunkt werden so gegeneinander verschoben, dass eine natürliche, von oben sichtbare schräge Fläche entsteht. Diese Verfahrensweise ist aber bei einer Punktanzahl von mehreren Millionen Punkten nicht mehr handhabbar.

Auch ein zweiter Fakt spricht gegen die Verwendung der Dreiecksnetze; eine Forderung der These II ist »*effektiv zu berechnen*«. Bei diesen Dreiecksnetzen entstehen etwa doppelt so viele Dreiecke wie vorhandene Punkte, das wären in der kleinen Variante etwa 20 Millionen Dreiecke [RM04][S.49]. Diese Datenmenge zu handhaben, und darzustellen ist nicht effektiv und dürfte die Grenzen der Leistungsfähigkeit der zur Verfügung stehenden Hardware überschreiten.

Um die Volumenberechnung effektiver zu gestalten, muss die sehr große Punktanzahl generalisiert werden. Dies kann durch eine Abwandlung des in [RM04][S.53ff] beschriebenen Interpolationsverfahrens erreicht werden. In diesem Verfahren wird ein gleichmäßiges Raster über

das Messpunktfeld gelegt. Die Höhen der Eckpunkte des Raster werden aus den umliegenden Messpunkten interpoliert. Dadurch besitzen alle Eckpunkte des Rasters eine an die wirkliche Oberfläche angeschmiegte Höhe. Eine optische Beschreibung des Verfahrens wäre: Aus den Messpunkten wird ein DGM erzeugt und die Rasterpunkte werden auf dieses DGM fallen gelassen. Als Ergebnis steht ein gerastertes DGM. Im vorliegendem Fall haben wir jedoch kein »grobes« Messpunktnetz, sondern ein sehr feingliedriges Messnetz.

Der Ansatz mit dem gleichmäßigen Raster wird aufgegriffen und weiterentwickelt. Aus der Erfahrung über die Oberflächenstruktur der Produkthaufen heraus, wird eine Rastergröße von 20 Zentimeter vorgegeben. Mit dieser Rastergröße sind Kanten und Absätze bis 50 cm Ausdehnung gut erkennbar. An dem Beispiel des bereits beschriebenen Rundspeichers soll diese Möglichkeit erarbeitet werden. Der Rundspeicher besitzt einen Radius von etwa 28 m, dies ergibt eine Grundfläche von 2463 m^2. Mit der Rastergröße von 20 cm ergeben sich für die vollständige Abdeckung der Fläche ca. 61575 Rasterflächen. Diese Anzahl von Zellen ist eine handhabbare Größenordnung.

4.1 Erläuterung der verwendeten Säulenprismen

Die Umsetzung des Rasters erfolgt nicht wie in [RM04] beschrieben, mit unterschiedlichen Höhen an jedem Eckpunkt, sondern als Säulen mit ebenem Abschluss.

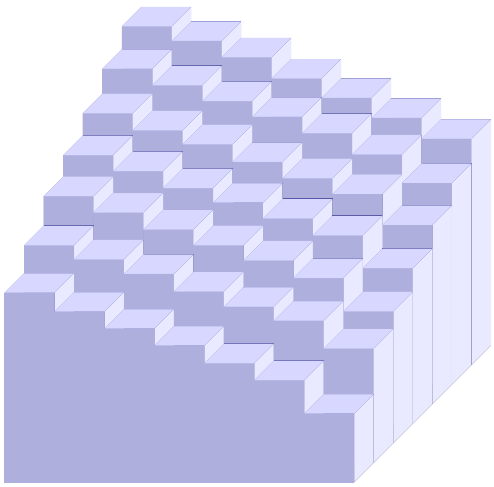


Abbildung 4.2: Beispieldarstellung eines Volumenmodelles aus Vierseitprismen

Um die Höhen der Säulen festzulegen, wird die mittlere Höhe aller in ihr befindlichen Scanpunkte bestimmt. Das Volumen wird dann einfach ausgezählt, dazu werden die Höhen aller Säulen aufaddiert und mit der Grundfläche multipliziert.
Auch in einem Säulenprismenmodell können keine Überhänge dargestellt werden, es ist jedoch möglich, senkrechte Absätze zu modellieren.

Um dieses Volumenmodell aus Vierseitprismen zu realisieren muss die Grundfläche, auf dem sich der Produkthaufen befindet, einer Rasterung unterzogen werden.

4.1.1 Rasterung der Grundfläche

Die Grundfläche der Säulenraster entspricht dem Raster eines Pixelbildes. Ein Voxel wird als ein Volumenpixel definiert, daher ist das 2D-Äquivalent eines Voxels ein Pixel. Das Pixel ist als quadratische Fläche definiert. Da der Begriff »Pixel« bereits durch die Bildverarbeitung sehr stark vereinnahmt ist, wurde nach einer anderen passenderen Begrifflichkeit gesucht. Gefunden wurde eine Bezeichnung, die dem sinnverwandtem Voxel nahekommt, das Texel. Dieser Begriff ist der Computergrafik entliehen, in [FH11][S. 900] beschreibt der Texel einen Farb- oder Bildpunkt, der auf einen Körper gerenderten wird. In dieser Arbeit wird das Texel als quadratischer Teil der gerasterten Grundfläche angesehen und im Folgendem auch als solcher benutzt.

Die Texel sollen die Grundfläche der Produktenspeicher vollständig ausfüllen und dabei nicht über den Rand hinaus ragen, da dadurch die Grundfläche als zu groß dargestellt wird. Ist die Produkthöhe an den Außenrändern mit 0.0 m gegeben, ist der Flächengewinn oder auch ein Flächenverlust für das Volumen als unschädlich zu betrachten. In den Rundspeichern wird oft an den Außenrändern bis zu einer Höhe von ca. 3 m eingespeichert. Legt man den Radius von etwa 28 m zu Grunde, entsteht, mit 1 cm umlaufender Schichtdicke, eine Volumendifferenz von 5.3 m^3. Um die Ungenauigkeit des berechneten Volumens, durch den beschriebenen Einfluss, so gering wie möglich zu halten, sollte versucht werden, die Flächendifferenz zu minimieren.
Als Außenring der zu rasternden Fläche wird das, in der Sammelkörperkonstruktion erzeugte Außenpolygon verwendet. Auf den, in der Konstruktion des Sammelkörpers verwendeten, Sicherheitsabstand von 40 mm wird verzichtet. Dadurch wird die, für die Volumenberechnung genutzte, Grundfläche auf die korrekte im Produktspeicher nutzbare Fläche erhöht.

Das entstehende Texelsystem nutzt den gleichen Berechnungsalgorithmus, Punkt-im-Polygon, wie das Voxelsystem. Es wird nur eine leichte Aufweichung der Klassifizierungsregel »*alle Eckpunkte innen*« implementiert, aus *»alle innen«* wird *»3 Eckpunkte von 4 innen«*. Dadurch werden zumindest die Flächen mit integriert, welche überwiegend in der Grundfläche liegen. Dies führt zu einer präziseren Gesamtfläche.

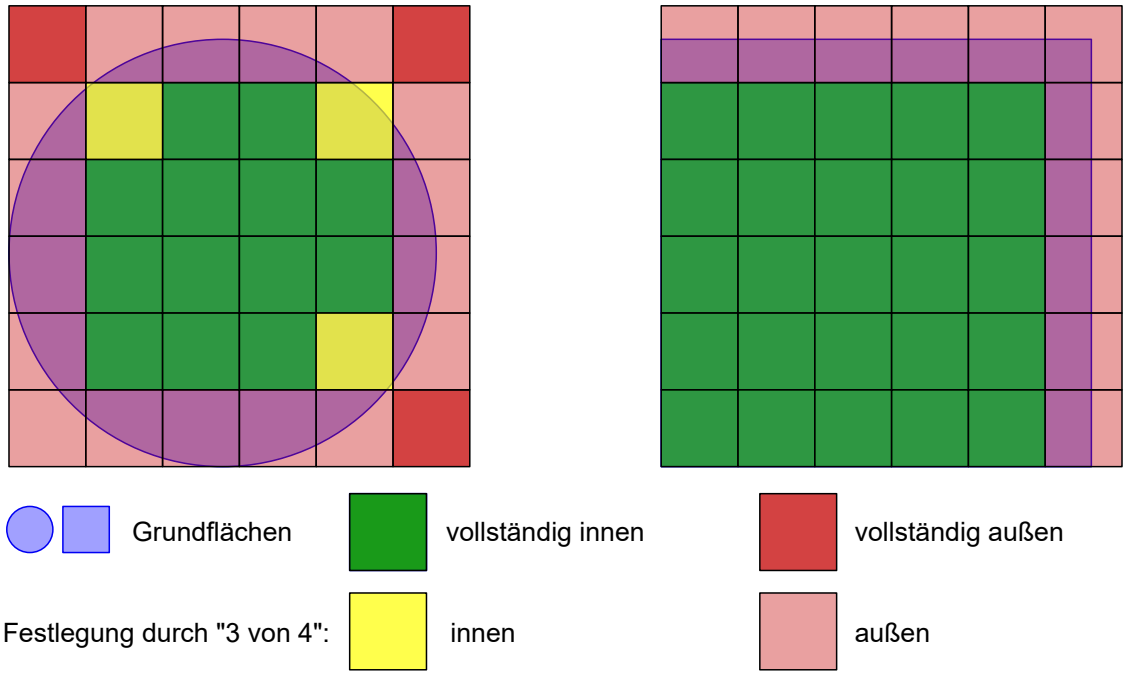


Abbildung 4.3: Exemplarische Darstellung des Texelsystems

In der Abbildung 4.3 ist erkennbar, dass der Flächenverlust durch das gewählte Raster doch recht erheblich ist. In der Realität ist der entstehende Flächenverlust anteilig an der Gesamtfläche nicht so hoch wie in diesem exemplarischen Beispiel, jedoch immer noch signifikant. Wenn man die Klassifizierungsregel auf *»2 von 4 innen«* oder gar *»1 von 4 innen«* herabsetzten würde, dann käme es zu einem ebenfalls nicht unerheblichen Flächengewinn. Beide Flächendifferenzen verfälschen das Volumen im oben beschriebenen Fall, bei einer Randhöhe größer 0.0 m, zu stark.

Für die beiden bereits oben verwendeten Musterbeispiele, dem Rechteck- und dem Rundspeicher, wurden die besprochenen Varianten berechnet:

Beispiel	reale Fläche m^2	Fehler bei »4 von 4« m^2	Fehler bei »3 von 4« m^2	Fehler bei »2 von 4« m^2	Fehler bei »1 von 4« m^2
Rechteckspeicher	3161,0	-0,8	-0,7	+9,6	+9,6
Rundspeicher mit	2491,2	-21,0	-7,9	+10,6	+23,7

Tabelle 4.1: Vergleich der Auswirkung der verschiedenen Auswahltests

Erkennbar ist hierbei, dass sich bei runden Objekten der Anteil von Flächenfehlern deutlich erhöht. In dem Rechteckspeicher ist der Fehleranteil nur so gering, weil dort die Grundfläche in einer durch 0.2 teilbaren Außengröße konstruiert wurde. Eine Aussparung ist auch hier enthalten, was die Unterschiede in den einzelnen Stufen erklärt.

In dem besprochenen Fall des Rundspeichers, mit einer Randhöhe von 3 m, entstehen Volumendifferenzen von minimal -23 m^3 bis maximal +71 m^3.
Um die Genauigkeit des Flächenrasters zu steigern, kann hier das System eines Quadtree angewendet werden. Ein Quadtree lässt sich nach [Ern+16][S.119 ff] als »... Baumstruktur von rechteckigen Bereichen in abnehmender Größe ...« beschreiben. In diesem Fall ist die Mutterfläche ein Quadrat, dessen Seitenlänge halbiert wird. So entstehen aus dem Mutterquadrat vier gleichgroße Tochterquadrate, welche in Summe die gleiche Fläche abbilden. Dieser Prozess lässt sich stufenweise, theoretisch beliebig oft wiederholen. Bei einer Ausgangs-Seitenlänge von 20 cm ist man in der 5. Stufe bereits bei 6.25 mm Seitenlänge.
Für dieses Texelsystem wird ein 2-stufiger Quadtree verwendet, dadurch wird die kleinste verwendete Seitenlänge 5 cm lang.

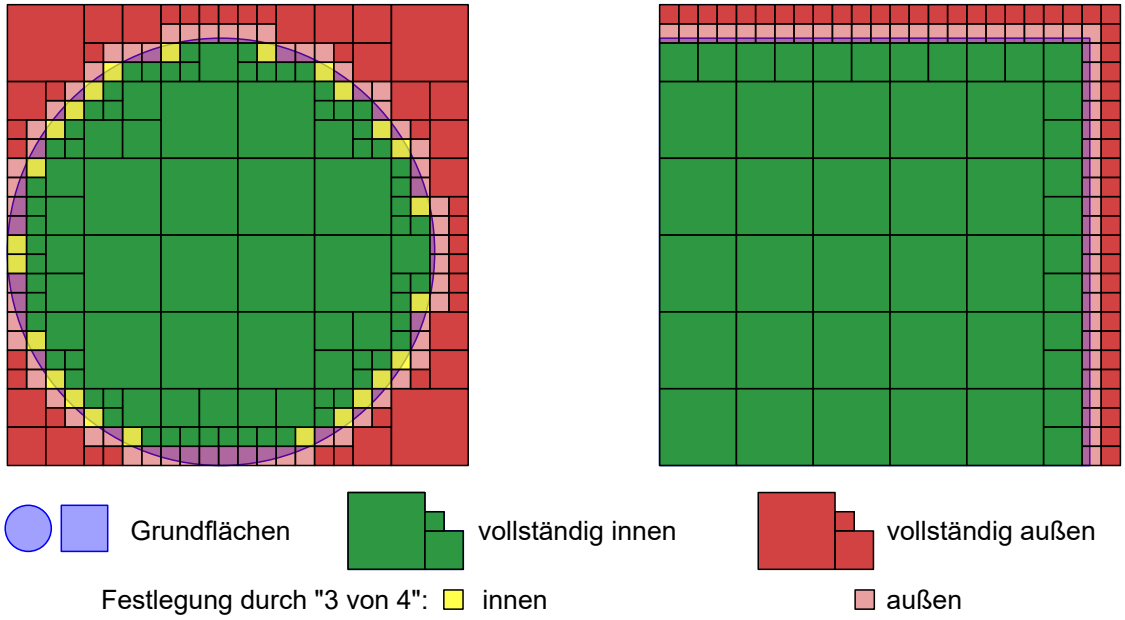


Abbildung 4.4: Exemplarische Darstellung des Texelsystems mit Nutzung des Quadtrees

Im der Abbildung 4.4 wird die Auswirkung des Quadtree auf das vereinfachte Beispiel verdeutlicht.
Wie sich direkt optisch erkennen lässt, fällt durch die Nutzung des zweistufigen Quadtree der Flächenverlust deutlich kleiner aus. Auch hier wird für die Klassifizierung eines Viertels in »innen oder außen« die *»3 von 4 Eckpunkten müssen in der Grundfläche liegen«*-Regel verwendet. Diese Regel wird nur für die Quadrate der untersten Schachtelstufe angewendet.

Wiederum wurden mit den beiden Beispiel-Produktspeichern Berechnungen durchgeführt und es zeigt sich folgendes Ergebnis:

Beispiel	reale Fläche m^2	Fehler bei »3 von 4« m^2	Fehler Quadtree »3 von 4« m^2
Rechteckspeicher	3161,0	-0,7	-0,4
Rundspeicher mit	2491,2	-7,9	-2,6

Tabelle 4.2: Vergleich zwischen normalem Texel und der Quadtree-Variante

Mit diesem bestehenden Flächenverlust ist im größtem anzunehmenden Problemfall (Rundspeicher mit 3.0 m Außenhöhe) mit einem maximalem Volumenverlust von -7,8 m^3 zu rechnen. Bei einem Gesamtvolumen in diesem Fall von zirka 23000 m^3 entspricht dies einem Anteil von -0,03%.

4.1.2 Umsetzung des Texelsystems als Quadtree

Die programmiertechnische Umsetzung des Texelsystem erfolgte in Anlehnung an das Voxelsystem. Die Elemente, welche in einem Texelsystem gerechnet werden können, sind derzeit Kreise und Polygone. In der Praxis werden ausschließlich Polygone in das Texelsystem gerechnet. Das Quellpolygon muss ebenfalls den gleichen Koordinatenbezug besitzen wie auch Punktwolke und Sammelkörper. Genau wie im Voxelsystem werden für die Arrayberechnung die Koordinaten um das Minimum der Boundingbox des Polygons verschoben. Die Berechnung der Indizes für das Array funktionieren analog Formeln 3.1-3.2 (S.24):

$$Index^X = ((X - verschiebung^X)/texelgroesse) \tag{4.1}$$

$$Index^Y = ((Y - verschiebung^Y)/texelgroesse) \tag{4.2}$$

Die Speicherung der vollständigen Texel erfolgt in einem »*BitArray*«. Daher wird hier ebenfalls ein linearer Index benötigt. Dessen Berechnung muss, gegenüber dem linearen Index des Voxelsytems (Formel 3.4, S. 24), um den Anteil der Z-Koodinate reduziert werden:

$$linearIndex^{Texel} = Index^Y + (Index^X * arrayLaenge^Y) \tag{4.3}$$

Es wurde eine Basisklasse *Texelizer2D* erzeugt. In dieser werden alle notwendigen Daten gehalten und die benötigten Funktionen, Methoden bereitgestellt. Für die Berechnung der Eckpunkte der Texel wird die gleiche Klasse »STLCutPolygon«, wie bei der Voxelerzeugung, genutzt.

Um die Daten der Texel, inklusive des eventuell zugehörigen Quadtree auch außerhalb der *Texelizer2D*-Klasse zur Verfügung zu stellen, wurde eine Klasse *TexelCell2D* erzeugt. Sie trägt alle Informationen zu der Texelzelle. Die Klasse *Texelizer2D* verfügt über die Funktion, um eine einzelne Zelle auszugeben. Auch die Ausgabe aller vorhandenen Zellen als Liste von *TexelCell2D* ist möglich. Erzeugt wird als *TexelCell2D* jeweils das kleinste belegte Viertel (Quater).
Die Viertel werden bei eins beginnend in dem Texel, welches dem Koordinatenursprung am nächsten gelegen ist, entgegen des Uhrzeigersinns gezählt. Um eine Indizierung der einzelnen Sub-Texel zu ermöglichen, wird als Basis der Index des Muttertexels benutzt. Für jede Schachteltiefe wird eine Dezimalstelle hinzugefügt und die Viertelnummer (Quaternummer) als diese Dezimalstelle geschrieben. Ein Beispiel: Der Index 0.11111 würde in unserem Fall das 6.25 mm - Texel ganz links unten bezeichnen.
Zur Darstellung des Quadtree, um die Verwaltung der Daten zu gewährleisten, wurden zwei Klassen eingeführt. Eine Basisklasse *LinearQuadTree*, in ihr wird das einzelne Texel mit seinen vier Tochtertexeln gehalten. Es werden die Texelgröße, Schachteltiefe, die Quaternummer sowie die Koordinaten X und Y des Basispunktes gespeichert. Zusätzlich noch notwendige Verwaltungsdaten, wie der Vorfahre und das erste Muttertexel. Weiterhin wurden Methoden zur Datenhaltung (getter, setter) und Informationsfunktionen (gibt es an X,Y einen gültigen Texel) implementiert.
Um die Besonderheiten des Muttertexel (Randtexel) abzubilden, wurde von der Basisklasse *LinearQuadTree* die Klasse *BoundaryTexel* abgeleitet. In dieser Klasse wurden zusätzlich der Texelarray-Index (IndexX und IndexY), die maximale Quadtreetiefe und ein *BitArray* zur Speicherung der Daten abgebildet. Des weiteren wurden Funktionen zum Datenhandling implementiert, wie die Erzeugung des Quadtree aus einem *BitArray* oder die Rückgabe aller gültigen Tochtertexel als Liste.
Zur Speicherung des einzelen Quadtree wird die Baumstruktur in ein *BitArray* abgebildet. Gespeichert werden die kleinsten Einheiten des Quadtree, daher wird das *BitArray* in der Größe $4^{\text{Schachteltiefe}}$, hier 4^2 = 16 Bit, erzeugt. Ist ein Tochtertexel vollständig innen liegend, werden dessen zugehörige Bits alle gefüllt. Dieses Füllen erfolgt in einer rekursiven Funktion »*setBitArray*«, welche durch den *BoundaryTexel* initiiert wird. Diese Funktion berechnet den linearen Index des gerade aktuellen Tochtertexel und ruft sich für dessen vier Tochtertexel wieder selber auf. Ist die maximale Schachteltiefe erreicht, dann wird der, zu dem aktuellen Tochtertexel gehörende, lineare Index benutzt um das zugehörige Bit im Array auf dessen Status (gültig oder ungültig) zu setzten.

Im folgenden Pseudocode wird der Ablauf erläutert:

```
1  // Vorbereitung
2  setze qTTiefe = 2 // QuadTreeDeep
3
4  // Aufruf in dem BoundaryTexel => ich
5  aufruf setBitArray(ich, 1, 0)
6
7  // Funktionsdefinition
8  begin funktion setBitArray (LinearQuadTree elternTexel, int
       aktTiefe, int preIndex)
9    fuer quaterNr = 0 bis quaterNr == 3
10     setze cuIndex = preIndex + (quaterNr * 4 ^ (qTTiefe -
           aktTiefe))
11     ist aktTiefe == qTTiefe dann
12       setze BitArrayZelle[cuIndex] = "Bin_ich_innen?"
13     sonst
14       // rekursiver Aufruf
15       aufruf setBitArray(elternTexel.Quater[quaterNr],
             aktTiefe+1, cuIndex)
16     ende ist
17   wiederhole quaterNr
18 ende funktion
```

Listing 4.1: Ablauf der Funktion setBitArray

Die Speicherung der Texeldaten erfolgt analog der Speicherung der Voxeldaten. Es wird ein anderer Dateityp angegeben, als Kennung wird »#exactbintex2d 1« verwendet. Die Muttertexel werden in einem *BitArray*, entsprechend des Linearen Index in dem »binarydata:«, abgelegt. Die Speicherung des »BitArray« erfolgt analog der Speicherung der Voxeldaten.
Die *BounderyTexel* müssen in einem eigenem Format gespeichert werden. Eingeleitet wird der Bereich von der Schlüsselzeile »binaryboundarydata:«. Im Anschluss folgen jeweils der lineare Index des BoundaryTexel als 32 Bit Integer und das *BitArray*, welches die Struktur des Quadtree binär speichert.
Als Dateiendung wurde »bintex2d« als Äquivalent zu der Voxel-Dateiendung »binvox3d« gewählt.

In der Abbildung 4.5 ist ein Ausschnitt des Texelsystem mit Texelgröße 20 cm, als 2 stufiger Quadtree mit der Forderung »3 von 4 Eckpunkten innerhalb« zu sehen. Die blaue Linie stellt das umgrenzende Polygon dar.

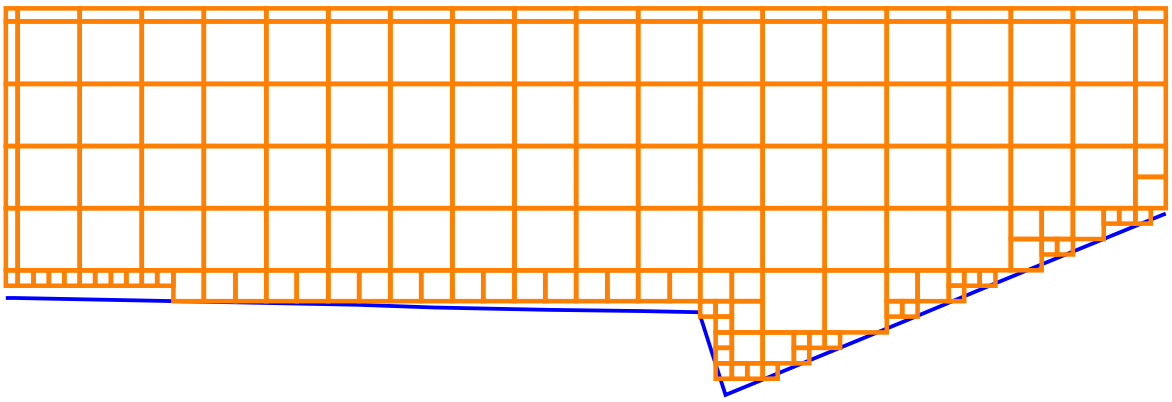


Abbildung 4.5: Ausschnitt aus dem Rundspeicher mit Texelsystem, 2 stufiger Quadtree und »3 von 4«

Das Erzeugen und Speichern des Texelsystems ist, genau wie das Erzeugen und Speichern des Voxelsystems in jedem Produktspeicher eine einmalige Tätigkeit. Bei Folgemessungen kann, wie in 3 und 3.1.1 beschrieben, die erzeugte Datei genutzt werden.

4.2 Einsatz des Texelsystem in der Volumenberechnung

Mit Hilfe des Texelsystems lässt sich noch kein Volumen berechnen, hierzu fehlen ihm zusätzliche Eigenschaften und Methoden. Da das Texelsystem nicht ausschließlich für die Volumenberechnung geeignet ist, wird für die Volumenberechnung eine neue Klassenstruktur erarbeitet, welche das Texelsystem nutzt.
Es wird eine Klasse zur Datenhaltung einer Volumensäule benötigt, dies geschieht in der Klasse *VolumeGridCell*. sie enthält die Information Ihrer Texelzelle und alle für die Volumenbestimmung notwendigen Informationen. Die einzelnen Volumensäulen werden in der Klasse *VolumeGrid* zusammengefasst. Diese Klasse trägt das gesamte Volumenmodell, sie hält das Voxelsystem und Texelsystem. Implementiert sind Methoden zur Volumenerzeugung, Neuberechnung, Speicherung und zum Einladen einer gespeicherten Volumenberechnung.

Das Erzeugen eines Volumens läuft wie folgt ab. Die Namen des zugehörigen Voxel- und Texelsystems werden aus dem Namen der aktuellen AutoCAD Zeichnung gewonnen. Existieren die Dateien nicht, können sie manuell, per Datei-öffnen-Dialog, ausgewählt werden. Im Anschluss sind eine oder mehrere Punktwolkendateien zu selektieren.
Die einzelnen Scanpunkte werden einzeln, zeilenweise eingelesen und sofort verarbeitet. Nach erfolgter Klassifizierung eines Scanpunktes als »Produktoberfläche«, mit Hilfe des Voxelsystems, wird mit Hilfe seiner X-Y-Koordinaten die zugehörige Rasterfläche gesucht. Ist sie in dem

Raster vorhanden, erfolgt eine Zuweisung seiner Z-Koordinate zu der Rasterflächen-Liste von Höhenwerten. Ist die Klassifizierung aller Scanpunkte abgeschlossen, wird zu jeder Rasterfläche der Mittelwert aller in ihr gespeicherten Höhenwerten berechnet. Das Volumen und weitere statistische Informationen, wie Anzahl der genutzten Texelzellen und Scanpunkte, werden in der Klasse *VolumeGrid* gespeichert.

Im Laufe einiger Tests wurde festgestellt, dass sich in den Scandaten auch Ausreißer befinden. Ausreißer sind Scanpunkte, welche sich in der Realität nicht an der Koordinate befinden können, da sich an diesem Ort nichts befindet, was den Laserstrahl hätte reflektieren können. Welche Ursache diesen Ausreißern zu Grunde liegt, lässt sich nicht abschließend klären. Um diese Höhen nicht für die Mittelbildung zu verwenden, wurde die Ermittlung des Höhenwertes vom normalen mathematischen Summenmittel auf die Bestimmung des Medianwertes umgestellt. Der Median beschreibt den Wert der mittleren Position in einer Liste von sortierten Werten. Dadurch wird zwar der Einfluss der Bandbreite der korrekten Werte verfälscht, jedoch ist der Median sehr beständig gegen Ausreißer.

Um den Zeitbedarf der Volumenberechnung festzustellen, wurden zwei Produktspeicher exemplarisch berechnet. Der Rundspeicher wurde zusätzlich aus zwei verschiedenen Auflösungen gerechnet. Die Ergebnisse sind in der folgenden Tabelle 4.3 ersichtlich.

Beispiel	Anzahl Punktwolken	Anzahl Punkte gesamt/ verwendet	Anzahl Texel	Anzahl leerer Texel	Zeit für Volumen erzeugen mm:ss	Zeit für Darstellung mm:ss
Rechteckspeicher 44.4 Mio. Punkte	2	78 645 721 30 587 690	79 158	0	1:42	1:43
Rundspeicher 44.4 Mio. Punkte	1	42 932 968 14 621 492	65 076	63	1:06	1:28
Rundspeicher 11.1 Mio. Punkte	1	10 337 894 3 073 653	64 968	171	0:18	1:32

Tabelle 4.3: Übersicht der Volumenberechnungen

Der Rechteckspeicher wurde von zwei Standpunkten aus aufgenommen, durch Überlappung beider Scanns wurden etwa 10 Mio. Punkte entfernt. Die zu der Scanauflösung fehlenden Punkte wurden durch die Vorfilter der Scannersoftware entfernt. Die verwendeten Punkte lagen sowohl in dem Sammelkörper, als auch über einem verwendeten Texel. Aus der Gesamtmenge der Scanpunkte befanden sich zirka 35 % auf der potentiellen Produktoberfläche. In den leeren Texelzellen waren keine Scanpunkte. Diese Zellen müssen in einem weiteren Arbeitsschritt

gefüllt werden. Die Zeiten des Einlesens und Verarbeitens der Scanpunkte sind durchaus akzeptabel, es werden durchschnittlich zirka 709 000 Punkte pro Sekunde verarbeitet. Die Darstellung in der CAD-Software AutoCAD benötigt mehr Zeit, dort werden etwa 737 Texelzellen je Sekunde gezeichnet.

Eine der im Kapitel 1.1 aufgestellten Fragen, die ***Frage 2.1*** befasst sich mit der Eignung der Volumenberechnungsverfahren für große Punktmengen. Dieses hier beschriebene Rasterverfahren ist für solche großen Punktmengen als sehr geeignet zu betrachten. Es kann die Vermutung aufgestellt werden, das dieses Verfahren quasi unabhängig von der Anzahl der Punkte pro Zelle ist. Eine Volumensäule benötigt mindestens einen Punkt, jeder zusätzliche Punkt erhöht die Sicherheit der ermittelten Säulenhöhe. Im Gegensatz zum Dreiseitprismen-Modell jedoch nicht das Datenvolumen der Darstellung. In den verwendeten Punktwolken die Punktdichte so hoch ist, dass in den Texelzellen mit 20 cm Seitenlänge von 2 Punkte bis zu 76 300 Punkte liegen. In einer Randzelle mit 5 cm Seitenlänge befinden sich auch noch bis zu 6 Scanpunkte. Ein Problem für das Säulenprismenmodell ist das Fehlen von Scanpunkten in Zellen. Dieses Problem wird im Kapitel 5.1 aufgegriffen und besprochen.

Mit der Umstellung des Berechnungsalgorithmus auf den Median ist die ***Frage 2.2*** abschließend geklärt: *Ein fehlertolerantes Berechnungsverfahren muss mit größtmöglicher Sicherheit ein der Realität entsprechendes Berechnungsergebnis liefern, auch wenn Parameter fehlerhaft sind.* Sollte jedoch in einer Rasterfläche nur ein Scanpunkt enthalten sein und dieser ist ein Ausreißer, kann auch mit diesem Verfahren kein sicheres Berechnungsergebnis erfolgen. Dieser Fall wird später, im Kapitel 5.2, analysiert und bleibt hier erst einmal ungeklärt.

Trotz der aufgezeigten Probleme kann die **These 2: »*Aus den gemessenen Punkten wird das Volumen effektiv und fehlertolerant berechnet.*«** als bestätigt betrachtet werden.

4.2.1 Speicherung der Volumendaten

Für die Speicherung der Volumendaten wurde ein textbasiertes Dateiformat erstellt. Der Dateikopf beinhaltet, neben der üblichen den Dateityp beschreibenden ersten Zeile, statistische Angaben über das Volumen. Es werden auch der Dateibezug zur Voxel- und Texel-Datei sowie das Berechnungsdatum abgelegt. Die Texelzellen werden zeilenweise gespeichert. In der Zeile stehen der lineare Index als Fließkomma-Zahl und ein Marker der anzeigt, ob die Zelle manipuliert ist. Weiterhin wird die Flächengröße, Punktanzahl, die Summe aller Punkthöhen und die Median-Höhe der Zelle gespeichert.
Als Dateiendung wurde »result« festgelegt. Die Dateien werden zentral im Netzwerk abgelegt,

erhalten als Dateinamen das Messdatum in der Reihenfolge (JahrMonatTag = YYYYMMDD) und den Namen der AutoCAD-DWG-Datei.

```
#texelvolume 1
volume 19055.792
groundarea 2488.597
volumeid 17962
usingcellcount 65139
usingpointcount 14621492
groundheight 0.000
texelcount 65139
texel2dbase Rundspeicher.bintex2d
voxel3dbase Rundspeicher.binvox3d
calculatedate 20200413
celldata:
949. 0 0.040000 64 193.108153
17380.13 0 0.002500 5 15.049774
17380.4 0 0.010000 19 57.152623
...
```

Listing 4.2: Ausschnitt einer *.result-Datei

Die AutoCAD-DWG und das gespeicherte Volumen werden in der AutoCAD Zeichnung miteinander verbunden. In der Zeichnung werden der Dateiname und die Volumen-Kennzahl (siehe obiges Listen »*volumeid*«) gespeichert. Mit diesen Informationen ist die Applikation in der Lage, nach dem erneuten Öffnen der Zeichnung, die logische Verbindung zu der zugehörigen »*.result«-datei herzustellen.

4.3 Darstellung der Volumendaten

Die Darstellung der Volumendaten erfolgt in der CAD-Software Autodesk AutoCAD. Diese bietet vielfältige Möglichkeiten, eine Darstellung zu realisieren. Erste Versuche zielten auf eine flächenhafte Darstellung der Texel inklusive bestimmter Kennzahlen, wie Punktanzahl und Spannweite der Höhenwerte. Diese Darstellungsart wird in AutoCAD über Blöcke realisiert. Blöcke sind symbolartige Zeichnungen, welche sichtbare und unsichtbare Attribute besitzen. Sie werden selbst auch als »*.dwg«-AutoCAD-Zeichnung abgelegt. Diese Attribute können während des Einfügens eines Blockes an der gewünschten Position ausgefüllt werden. Sichtbare Attribute werden als Text, in relative Lage zum Block-Referenzpunkt, angezeigt. Dadurch können wichtige Informationen, wie zum Beispiel Namen o.ä., direkt abgelesen werden. Mit der Auswahl eines Blockes können die Attribute in dem Eigenschaftsfenster angezeigt werden.

Leider stellte sich das Verwenden von sichtbaren Attributen als extrem zeitaufwendig heraus. Die Regeneration, das reine Darstellen der Zeichnung im AutoCAD, benötigte so viel Zeit, dass es nicht möglich war, flüssig in der Zeichnung mit Zoom und Pan zu arbeiten. Das AutoCAD interne Handling der sichtbaren Attribute von 65139 Blöcken verbraucht zu viel Zeit. Daher wurden alle Attribute auf »nicht sichtbar« umgestellt.
Bei Ansichten aus der Profilsicht (von links, rechts, vorn und hinten) entstand ein sehr löchriger, zerklüfteter optischen Eindruck. Es ließ sich kein Volumenkörper erkennen. Damit in diesen Ansichten die Zwischenräume geschlossen sind, wurden zwei weitere Farbflächen in den Block integriert. Eine in X-Achsrichtung und eine in Y-Achsrichtung. Dadurch ist ein Betrachten aus dem Profil als "geschlossene Fläche"möglich.
Die Blöcke werden mit ihrer korrekten Lage und Höhe gezeichnet. Um in der Draufsicht einen Eindruck der Höhe zu erhalten, wird eine Farbzuordnung der Höhe verwendet. Dem niedrigsten Höhenwert wird die Farbe Blau (RGB 0 0 255) zugewiesen, dem größten Höhenwert die Farbe Rot (RGB 255 0 0). Die Mitte zwischen dem Minimal- und dem Maximalwert erhält die Farbe Grün (RGB 0 255 0). Texel ohne zugewiesene Höhen (ohne Scanpunkte) erhalten die Farbe Schwarz oder Weiß, je nach Hintergrund der Zeichnung. Durch diese Falschfarbendarstellung erhält der Betrachter einen sehr plastischen Eindruck der Höhen (Siehe Abbildung A.10, Anlagen S.XXI))

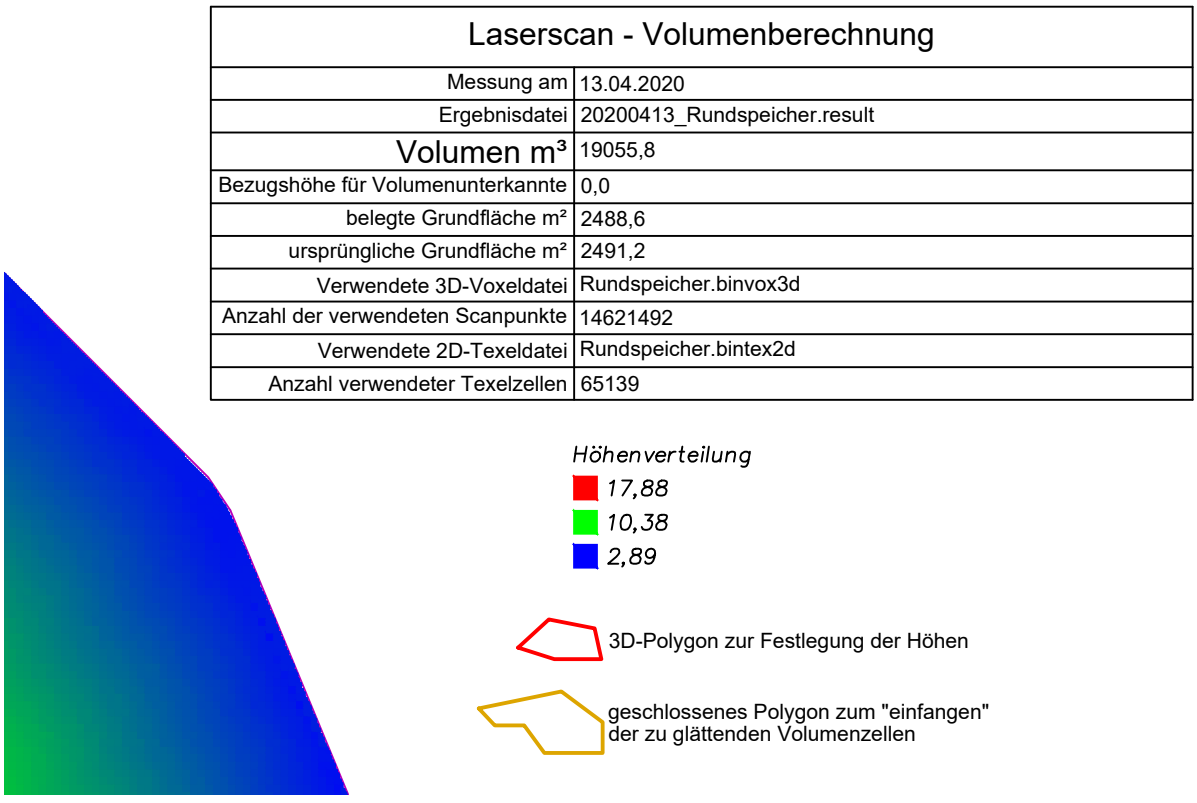


Abbildung 4.6: Ausschnitt der PDF-Ausgabe, mit Tabelle der Volumendaten

Zur Darstellung der Volumendaten gehört auch die Ausgabe von geforderten Protokolldaten. Diese erfolgt zum einen als Ablage der Volumen »*.result«-Datei, zum anderen wird über den Layoutbereich der Software AutoCAD eine PDF-Ausgabe der Grundrissdarstellung erzeugt. Diese PDF-Ausgabe enthält, neben der Draufsicht und üblichen Angaben in einem Kartenstempel wie Firma, Bearbeiter, Dateiablageort, Erstellungsdatum noch ein Informationsfeld. In diesem finden sich alle notwendigen Angaben zu dem Volumenmodell, zum Beispiel: Volumen, Bezugshöhe, Anzahl der verwendeten Scanpunkte und andere. Die AutoCAD-Zeichnung, die PDF-Ausgabe und die »*.result«-Datei werden gemeinsam in einem gepackten Archiv abgelegt. Dadurch sind Nachweise auch zu einem späteren Zeitpunkt möglich. Die Dauer der Aufbewahrungsfrist solcher Archive muss entweder der Arbeitgeber festlegen oder es gelten gesetzliche Regelungen.

5 Bearbeitungen der Volumendaten

Laserscanner sind optisch arbeitende Vermessungsinstrumente, sie können nur direkt sichtbare Punkte anmessen. Alles was hinter hervorstehenden Konturen liegt, bleibt verborgen, es bilden sich Scanschatten. Diese Bereiche bilden in der Draufsicht regelrechte »Löcher«. Diese, nicht gemessenen, Bereiche tragen ebenfalls Informationen. Dabei stellen sich die Fragen. Wie ist der Oberflächenverlauf in diesem Bereich? Lässt sich der wahre Oberflächenverauf nachträglich rekonstruieren? Diese Fragestellungen münden in der **Frage 3.1:** *Wie kann man Rückschlüsse auf den, der Wirklichkeit am nächstliegenden, Oberflächenverlauf ziehen*?
Das eingelagerte Produkt ist ein Schüttgut, daher wird sich immer eine Art Kegelstruktur einstellen. In den Rundspeichern entsteht ein gestauchter Kegel. Dies wird durch die kinetische Energie des fallenden Schüttgutes im Auftreffen auf den bereits liegenden Teil bewirkt. Daher ist die Außenfläche des Kegels nicht gerade, sondern etwas bauchig gewölbt.
Da das Produkt auch sehr hygroskopisch, es zieht sehr stark Feuchtigkeit an, ist, neigt es auch zum Verklumpen. Diese Klumpen sind wiederum von sehr lockerer Struktur und zerfallen bei leichtem Krafteintrag. Durch diese Eigenschaft können im Ausspeicherprozess nahezu senkrechte Kanten, sogenannte Wechten, entstehen. Befinden sich diese Wechten am Rande des Produkthaufens, bewirken sie automatisch einen Scanschatten.
Eine weitere Ursache für Scanschatten können herabhängende Störkörper sein, zum Beispiel Kabel oder Ketten. In einem Speicher ist beispielsweise noch ein längeres Einspeicherrohr montiert. Wird der Scanner nicht tief genug herunter gelassen, bildet sich ein recht großer Scanschatten, der sich vom Dach aus über die Außenwände, bis hin zum Boden ausbreitet (Siehe Abbildung 5.4, Anlagen S.47).

Um die oben gestellte *Frage 3.1* zu beantworten, muss man eingestehen, dass Rückschlüsse auf den tatsächlichen Verlauf der Oberfläche in verdeckten Bereichen nur aus der Erfahrung mit dem Produkt und Kenntnis der Örtlichkeit zu treffen sind. Eine realistische Einschätzung allein aus der Punktwolke heraus ist selten möglich.
Alternativ muss eine Begehung der Örtlichkeit nach oder auch vor der Messung erfolgen. Ob dies grundsätzlich immer möglich ist bleibt eher fraglich, da alle Messungen und auch Auswertungen in einem engen zeitlichen Rahmen abgeschlossen werden müssen.

All diese verschiedenen Probleme lassen »Löcher« in der Oberfläche entstehen, nun gilt es diese zu schließen. Die Lösung des Sachverhaltes wird im folgenden Kapitel (5.1) diskutiert.
Erste Tests deckten ein bisher nicht beachtetes Problem auf. In den Produktspeichern stehen, wenn sie nicht bis zum Außenrand gefüllt sind, Werkzeuge (Besen und Schippe), gelegentlich auch größere Objekte. In dem Falle einer Nullmessung stand ein Fahrzeug in dem Produkt-

speicher (Siehe Abbildung A.9, Anlagen S.XX). Von dem Laserscannerstandort aus ist der Speicher nicht einsehbar, daher sind die störenden Objekte erst in der Punktwolke erkennbar. Solche Objekte verfälschen das entstehende Volumen. Die Werkzeuge werden das Volumen nicht im signifikanten Bereich verfälschen, ein Besen zum Beispiel erzeugte im Volumenmodell, im Maximalfall, einen Volumenfehler von 0.25 m^3. Es zeigt sich dadurch die Notwendigkeit den entstandenen Volumenkörper nachträglich manipulieren zu können.
In dem bisherigen tachymetrischen Messverfahren wurden die Messpunkte durch den Mitarbeiter subjektiv, einzeln ausgewählt. Solche, nicht zum Produkt gehörende, Objekte wurden ignoriert. Der Laserscanner hingegen unterscheidet nicht nach »sinnvoll« oder »nicht sinnvoll«. Mit diesem Thema befasst sich das Kapitel 5.2, ab Seite 46.

5.1 Füllen von Löchern in der Volumenfläche

Die Volumensäulen in dem Modell, welche keine Scanpunkte erhalten, werden schwarz oder weiß (je nach Zeichnungshintergrund) dargestellt. Zusätzlich liegen diese Texel-Blöcke auf einem separatem Layer. Die Layer lassen sich im AutoCAD separat einblenden und auch sperren, dadurch wird die Darstellung verblasst dargestellt. Damit ist es möglich, die betroffenen Texel gut zu lokalisieren.
Um die manuelle Arbeit in dem Volumenmodell zu minimieren, wird direkt nach der Volumenmodellerstellung eine Funktion *VolumeGrid.fillBorder()* aufgerufen. Diese versucht, die leeren Randtexel (schwarze Quadrate) mit Höhen aus ihren 8'er-Nachbartexel (siehe Abbildung 1.2, S. 5) zu füllen.

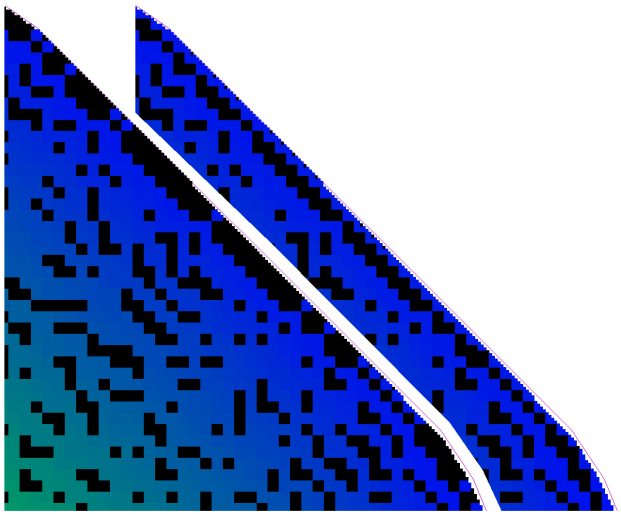


Abbildung 5.1: Wirkung der Funktion *fillBorder()*, links vor der Funktion, rechter Bereich danach

Die gefüllten, manipulierten Texel werden markiert, sie erhalten die Punktanzahl »-1« und werden für weitere Mittelbildungen nicht verwendet. Wird bei dieser Suche kein Nachbar gefunden, kommt eine weitere Suchfunktion zur Anwendung, die von dem betroffenen Texel aus in Rich-

tung des Flächenschwerpunktes des Texelsystems nach dem nächsten Texel mit gültiger, nicht manipulierter Höhe sucht. Diese Höhe wird dann übernommen und der Indikator für »manipuliert« gesetzt. Um größere Verfälschungen zu vermeiden, wird hier nach drei Texel (drei mal 20 cm) die Suche abgebrochen. In den bisherigen Tests hat diese Abbruchregel zu keiner größeren Beeinträchtigung geführt. Der Abbruch der Suche hat die großen »Löcher«, welche auf Grund größerer Hindernisse entstanden sind, nicht geschlossen und damit nicht verfälscht.
Der AutoCAD-Befehl *FILLVOLUME* muss vom Benutzer aufgerufen werden. Dieser Befehl sucht die Texel ohne Scanpunkte. Als Grundlage der Funktion wird in einem ersten Schritt aus allen Texelmittelpunkten, mit einer gültigen Höhe, ein Dreiecksnetz berechnet. Dieses Netz wird mit Hilfe der Delaunay-Triangulation, nur einmalig, erstellt. Für jedes gefundene Texel ohne Scanpunkte wird aus diesem Netz die Höhe des Texelmittelpunktes interpoliert und gesetzt. Der Texel erhält als Markierung die Punktanzahl »-1«.
In einem zweiten Durchgang wird noch einmal kontrolliert, ob trotz der Netzglättung nicht interpolierte Texel ohne Scanpunkte vorhanden sind. Ist dies der Fall, wird dem Benutzer, unter Angabe der Anzahl der betroffenen Texel, die Frage gestellt, ob die Höhe dieser Texel aus dem Mittelwert der umliegenden Texel gerechnet werden soll. Der Mittelwert wird dann als mathematisches Mittel aus allen 8er-Nachbarschafts-Texel berechnet, gleichgültig ob diese interpoliert sind oder nicht.
Alternativ kann, bei diesem Befehl, auch ein vorher konstruiertes, geschlossenes 3D-Polygon ausgewählt werden. In diesem Fall werden nur die Texel, bzw.die Volumenblöcke ohne Scanpunkte innerhalb dieses Polygones betrachtet. Aus dem gewählten 3D-Polygon wird ein Dreiecksnetz erstellt und die Höhe der betroffenen Texel aus diesem Netz gesetzt.

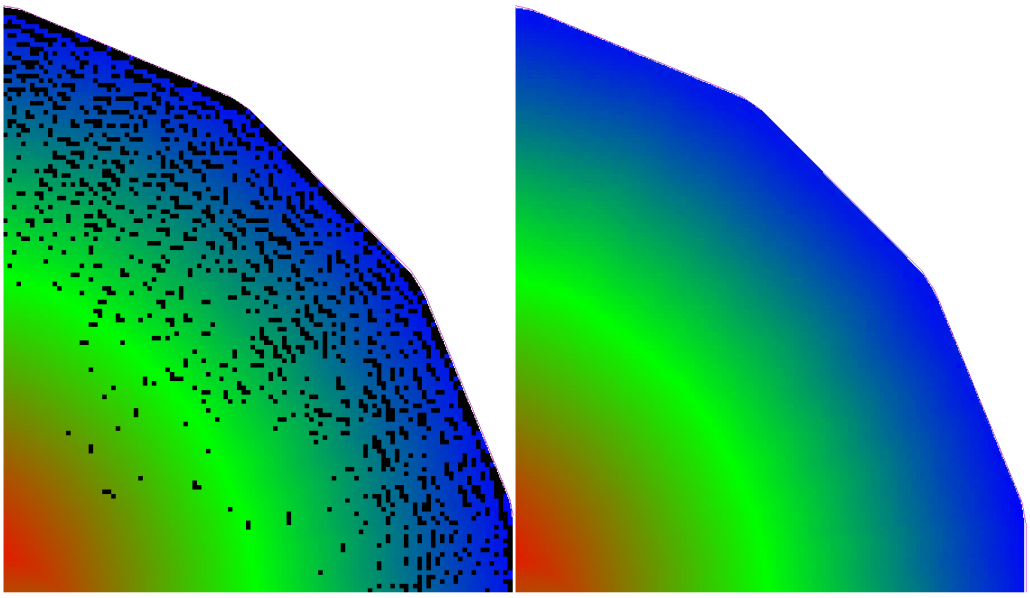


Abbildung 5.2: Auswirkung des Befehls FILLVOLUME, links vor Anwendung, rechts nach dem Befehl

In dem Kapitel 1.1 wurde die **Frage 3.2:** »*Lassen sich diese ›Löcher‹ automatisch schließen?*« gestellt. Im Abschluss der Analyse lässt sich feststellen, dass es nur bis zu einem gewissen Teil möglich ist, die entstandenen Scanschattenlöcher automatisch schließen zu lassen. Im Randbereich ist das recht gut möglich. Auch größere Löcher lassen sich, mit Hilfe des beschriebenen Befehl »FILLVOLUME«, automatisch füllen. Da die Programmierung keine Rückschlüsse auf die Ursache der Löcher ziehen kann, wurde diese Funktion in einem extra aufzurufenden Befehl ausgelagert.
Damit obliegt es dem Benutzter, im ersten Schritt die Scanschattenlöcher zu bewerten und gegebenenfalls bestimmte Bereiche mit Hilfe eines 3D-Polygons und des FILLVOLUME extra zu füllen. Im zweiten Schritt können dann die restlichen Löcher gefüllt werden.

5.2 Glätten von Bereichen

Da, wie bereits beschrieben, auch Störkörper auf dem Boden vorhanden sein können, muss eine Möglichkeit geschaffen werden diese aus dem Volumenmodell zu entfernen. Dies geschieht durch das »Glätten« der betroffenen Bereiche.

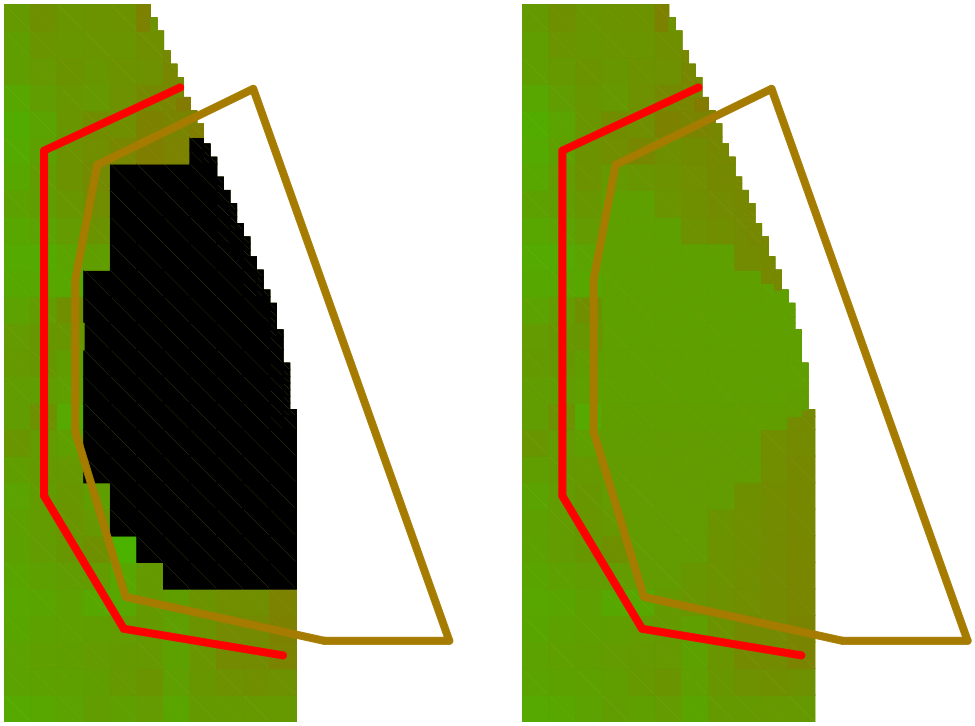


Abbildung 5.3: Auswirkung des Befehles PADVOLUME, links mit Texel-Loch durch Scanschatten, rechts mit PADVOLUME gefüllt. Rotes Polygon = 3D-Polygon, grünes Polygon »Einfangpolygon« zum Einsammeln der Texel

Es werden einzelne oder mehrere Volumensäulen, bzw. Texel auf eine bestimmte Höhe gesetzt. Hierzu wurde eine Funktion »PADVOLUME« implementiert, welche Höhen von einem nicht geschlossenen Polygon abgreift. Dabei wird, in der Draufsicht vom Mittelpunkt des Texels auf das Polygon ein Lot gefällt und an dem Lotfußpunkt am 3D-Polygon die Höhe interpoliert. Auch

das Nutzen der Höhe bestehender Texel und die manuelle Eingabe einer Höhe stehen zur der Auswahl. Die Auswahl der zu manipulierenden Texel erfolgt entweder mit AutoCAD-Bordmitteln oder durch den PADVOLUME-internen Befehl »POLY«. Dadurch wird ein geschlossenes Polygon ausgewählt. Die in dem Polygon gelegenen Texel werden ausgewählt.

Eine weitere Möglichkeit ist »PADVOLUME« mit einem geschlossenen 3D-Polygon zu nutzen. In dieser Verwendung wird aus dem geschlossenen 3D-Polygon ein Dreiecksnetz gebildet und es werden alle, sich in dem Polygon befindliche, Texel auf die Netzhöhe gesetzt.

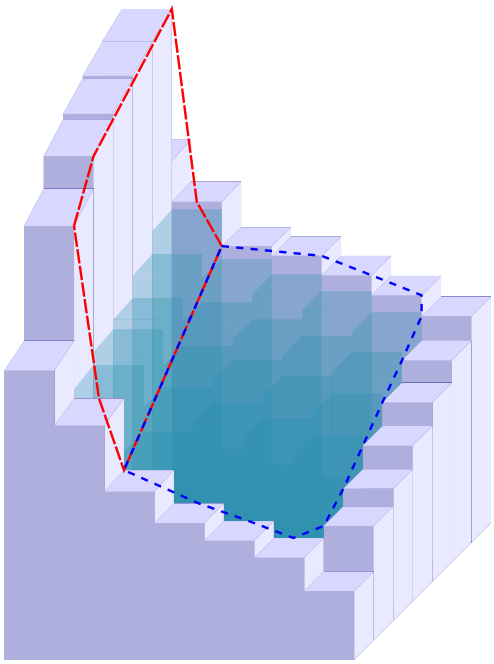


Abbildung 5.4: Schematische Darstellung eines Scanschattens, verursacht durch eine Wechte, mit beiden 3D-Polygonen. Oberer Teil rot, unterer Teil blau

Mit diesem Verfahren wird der Benutzer in die Lage versetzt, das Verhalten der Oberfläche im Schatten einer Wechte abzubilden. Es werden dafür zwei 3D-Polygone erzeugt. Eines verläuft um die Oberkante des Oberflächenloches (Abbildung 5.4, rotes Polygon). In der Höhe, bei der sich, in der Realität, die Wechtenunterkante befindet, kreuzt das Polygon das Oberflächenloch und schließt wieder nach oben. Das zweite 3D-Polygon verläuft über die Unterkante (blaues Polygon) und kreuzt an der gleichen Stelle das Oberflächenloch. So ist es möglich, auch eine solche Oberfläche annähernd realistisch nachzubilden.

Einen besonderen Stellenwert erlangt das Glätten des Volumen während der Auswertung der notwendigen Nullmessung. Eine solche Nullmessung kann nur in einem sehr kleinen Zeitfenster durchgeführt werden, da ein Produktspeicher nie längere Zeit leer steht. Bei der Auswertung einer Nullmessung störte ein defektes Ladefahrzeug, welches zum Messzeitpunkt nicht herausgefahren werden konnte. Ein solches Objekt muss in dem zu erstellenden Volumenmo-

dell entfernt werden. Eine Wiederholung der Messung in einem leeren Produktspeicher ist, in absehbarer Zeit, nicht möglich.

Zum Abschluss dieser Problematik lässt sich nach Beantwortung der Fragen 3.1 und 3.2 die **These 3:** ***»Die entstehenden Oberflächenlöcher lassen sich möglichst wahrheitsgetreu schließen.«*** durchaus bestätigen, wenn auch mit kleinen Einschränkungen. Der auswertende Benutzer muss zwingend Kenntnisse über die Örtlichkeit und auch über das Verhalten des zu modellierenden Produktes besitzen. Erst durch diese Kenntnisse ist er in der Lage, die entstandenen Oberflächenlöcher *möglichst nahe der Wahrheit* zu schließen.

6 Die Software in Gänze betrachtet

Im Verlauf dieser Arbeit erfolgte eine Betrachtung verschiedener Bereiche, des in der Einleitung genannten »Arbeitsablaufes zur Auswertung einer Punktwolke in einem Arbeitsschritt«. Zu den im Entwicklungsprozess aufgetretenen Problemen wurden Lösungsansätze diskutiert und umgesetzt. Die in der Einleitung gestellten Fragen konnten beantwortet werden. Die Verifizierung bzw. Falsifizierung der aufgestellten Thesen war möglich, wenn auch zum Teil nur mit Einschränkungen.

Es ist nicht gelungen den kompletten Ablauf in einem einzigen Arbeitsschritt abzubilden. Dazu ist die Tätigkeit zu komplex, es müssen zwischen den Einzelschritten Entscheidungen durch den Benutzer getroffen werden. Diese wiederum sind zum Teil nur *»auf Grund seiner Erfahrung«* möglich. Trägt eine im Programmablauf zu treffende Entscheidung ein solches Merkmal, so ist diese in der Informatik sehr schwer umsetzbar. In zukünftigen Softwaresystemen könnte bei solchen Entscheidungen der Bereich der »Künstlichen Intelligenz« Anwendung finden.

Die wesentlichste Einschränkung der entstandenen Software ist, dass es nur unter Vorbehalt möglich ist, ein Volumen aus einer einzigen Messung zu ermitteln. Denn dies ist nur unter der Bedingung, dass der Boden unter dem Produkthaufen eben ist, möglich. Dann kann sowohl der Sammelkörper, als auch das Volumen direkt aus dieser Messung erzeugt werden.

Sobald die Produktmenge in regelmäßigen Abständen bestimmt werden muss, ist diese Software ein effektives Werkzeug zur Bestimmung des Volumens.

Die wesentlichen Arbeitsschritte einer Nullmessung und einer Wiederholungsmessung sind in der nachfolgenden Abbildung (6.1) aufgeführt. Im Detail ist der Ablauf einer Wiederholungsmessung in der Abbildung A.5 auf der Seite XVI im Anhang ersichtlich.

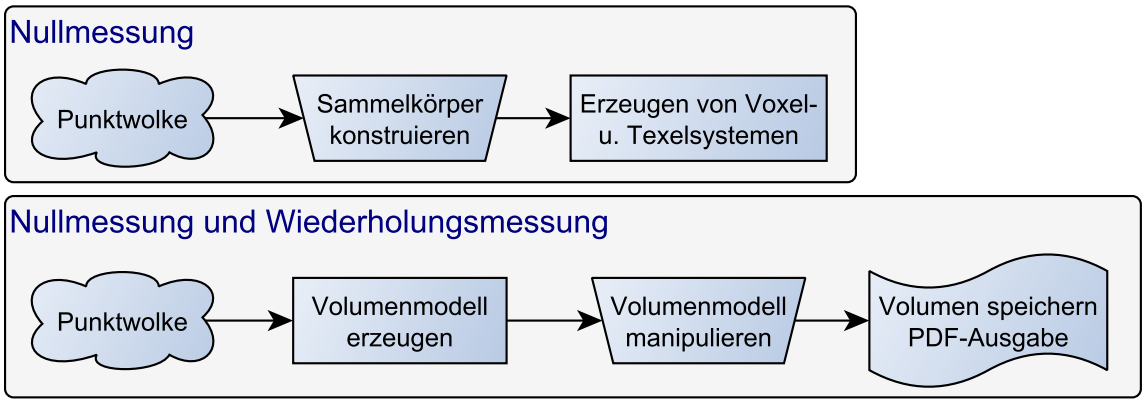


Abbildung 6.1: Darstellung der wesentlichen Arbeitsschritte beider Messungstypen

Am Ende dieses Prozesses steht eine Software, mit deren Hilfe die geforderte Auswertung erfolgen kann. Sie wurde vollständig als AutoCAD-Plugin umgesetzt und stellt derzeit 16 AutoCAD-Befehle zur Verfügung. Mit deren Hilfe kann der Anwender Volumenkörper in ein Voxelsystem umsetzten und aus einem 2D-Polygon ein Texelsystem rechnen lassen. Es sind Befehle für die Analyse der Berechnungsvorgänge implementiert, zum Beispiel kann man ein Texelsystem oder einen Bereich eines Voxelsystems zeichnen lassen. Auch ist es möglich, zwei Voxelsysteme zu vergleichen, die Unterschiede werden visualisiert. Um die vorrangige Arbeit, die Volumenberechnung aus Punktwolken zu ermöglichen, wurden Befehle zum Laden und Speichern, zur Volumenmodellerzeugung und dessen Manipulation implementiert.
Diese Befehle lassen sich im AutoCAD per Tastatur eingeben oder auch aus einem Ribbonpanel per Mausklick aufrufen.

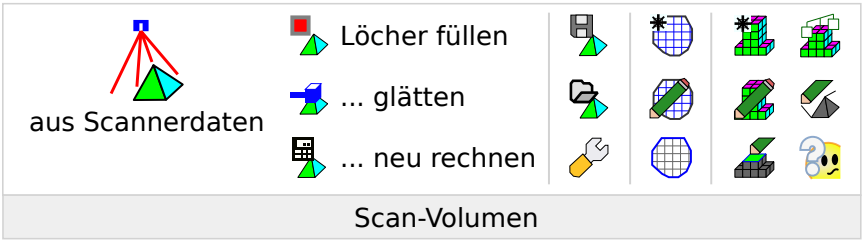


Abbildung 6.2: Ribbontoolbar des AutoCAD-Plugins

Es wurde darauf geachtet, möglichst selbsterklärende Symbole für die Befehle/ Arbeitsschritte zu erstellen. Zusätzlich wird in AutoCAD bei der Mausbewegung über das Symbol der Befehlsname und eine Beschreibung angezeigt. Mit dem Nutzungsbeginn der Software ist eine Online-Hilfe verfügbar, die jeden Arbeitsschritt und Befehl erläutert.

6.1 Analyse der Effektivität

Um die Volumenberechnung in ihrer Gesamtheit zu betrachten, wurden drei verschiedene Punktwolken aus einem Rundspeicher an zwei verschiedenen Rechnern ausgewertet. Zum einen an der im Kapitel Rahmenbedingung aufgeführten Workstation und zum anderen an dem Entwicklungsrechner.

Folgende Punktwolken wurden verwendet:

- **44.4 Mio. mit Produkt** = Testscan im Rundspeicher mit Produkthaufen
- **44.4 Mio. leer** = Nullscan im Rundspeicher, ohne Produkt
- **11.1 Mio. leer** = Nullscan im Rundspeicher mit niederer Auflösung, ohne Produkt

Die bewerteten Arbeitsschritte/ Programmschritte wurden in folgende Bereiche aufgeteilt:

- **Initialisierung** = Einlesen der zugehörigen Voxeldatei und Texeldatei, erzeugen des Texelsystems und Volumensystems

- **Punkte einlesen** = Einlesen und Verarbeiten der Scanpunkte in das Volumensystem
- **Volumen zeichnen** = Die Blöcke in der AutoCAD-Zeichnung erzeugen, zeichnen
- **Volumen aktualisieren** = Löcher füllen, das Volumen neu berechnen und die Blöche in der AutoCAD-Zeichnung aktualisieren
- **Gesamtzeit** = Summe der oberen Programmschritte, der Zeitbedarf für Aktionen des Benutzers bleibt unbeachtet!
- **Volumen** = Summe des berechneten Volumen, dies sollte bei Berechnung durch verschiedenen PCs immer gleich bleiben

Die Zeiten für die Analyse wurden der Protokolldatei entnommen, um dies zu ermöglichen sind im Quellcode entsprechende Einträge enthalten. Zusätzlich muss das Plugin mit dem AutoCAD-Befehl SETDEBUG in den erweiterten Protokollmodus gesetzt werden.

Programm-schritt	Workstation			Entwicklungs PC		
	44.4 Mio. gefüllt	44.4 Mio. leer	11.1 Mio. leer	44.4 Mio. gefüllt	44.4 Mio. leer	11.1 Mio. leer
Initialisierung (mm:ss)	00:20	00:21	00:19	00:22	00:24	00:18
Punkte einlesen (mm:ss)	01:03	01:02	00:13	01:07	01:06	00:13
Volumen zeichnen (mm:ss)	01:21	01:22	01:20	01:34	01:35	01:38
Volumen aktualisieren (mm:ss)	02:21	01:22	01:26	02:28	01:28	01:31
Gesamtzeit (mm:ss)	05:05	04:07	03:18	05:31	04:33	03:40
Volumen (m^3)	19 055.80	7 422.05	7 431.60	19 055.80	7 422.05	7 431.60

Tabelle 6.1: Übersicht der Volumenberechnungen

Die *Initialisierung* benötigt bei allen Varianten etwa 21 Sekunden, den Großteil der Zeit benötigt das Lesen der Voxeldatei und Texeldatei aus dem Netzlaufwerk. Bei Tests mit Lesen von der lokalen Festplatte reduzierte sich die Zeit auf 10 Sekunden. Die zu lesenden Dateien verbleiben auf dem Netzlaufwerk um die Daten jedem zugriffsberechtigten Nutzer im Netz zur Verfügung zu stellen. Ein weiterer Grund für die zentrale Datenhaltung ist, der redundanter Datenhaltung und den damit aufkommenden Problemen vorzubeugen.
Das *Einlesen der Punkte* benötigt bei beiden Rechnern in etwa die gleiche Zeit. Da beide Rechner eine vergleichbare Festplatte und auch CPU besitzen, sind die Ergebnisse plausibel. Das

Einlesen und Verarbeiten der kleineren Punktwolke benötigt erwartungsgemäß auch weniger Zeit.
Bei dem Zeitverbrauch für das *Zeichnen der Volumenblöcke* in der AutoCAD-DWG zeigt sich ein Geschwindigkeitsvorteil der Workstation, bedingt durch ihre leistungsfähigere Grafikkarte.
Das *Volumen Aktualisieren* benötigt in dem Speicher mit Produkt etwa eine Minute mehr Zeit als in den anderen Punktwolken. Begründet ist dies durch die größere Anzahl von Volumensäulen ohne Scanpunkte. In den Zeiten ist zu erkennen, dass das Aktualisieren meist schneller ist, als das Neuzeichnen der Volumenblöcke.
Bei den Punktwolken ohne Produkt wurde für den Zeittest keine Bearbeitung der Volumenmodelle vorgenommen.

Wird zu Grunde gelegt, dass in einem Speicher mit Produkt eine Minute für das »Füllen der Löcher« benötigt wird, kann man in der 44.4 Mio. Scanpunkte Auflösung von einer Bearbeitungszeit von 5:30 Minuten ausgehen. Mit der 11.1 Mio. Scanpunkte Auflösung wird sich diese auf etwa 4:30 Minuten reduzieren.
Keine Beachtung findet bei dieser Schätzung, ob in dem Volumenmodell weitere Bearbeitungen/ Glättungen notwendig sind. Aus der Tabelle ist zu erkennen, dass zu der benötigten Zeit für das reine Manipulieren noch einmal 1:30 Minuten für die Aktualisierung hinzukommen. In diesem Fall kann man von einer kompletten Bearbeitungszeit von zirka neun bis zehn Minuten ausgehen.

Um einen direkten Vergleich der beiden Punktwolken aus der Nullmessung (44.4 Mio. Punkte und 11.1 Mio. Punkte) zu ermöglichen, wurden beide Volumen geglättet und berechnet. Wie es sich in der Tabelle sichtbar andeutet, ist das Volumen der kleineren Punktwolke (7 426.6 m^3) größer, als das der Punktwolke mit höherer Auflösung (7 417.1 m^3). Auf die Volumengrundfläche von 2 488.6 m^2 berechnet beschreibt die Differenz von 9.5 m^3 einen Höhenfehler von 4 mm. In der Abbildung A.11 in den Anlagen auf der Seite XXII ist das Volumenmodell des Differenzvolumen (»11.1 Mio. Punkte« minus »44.4 Mio.Punkte«) zu sehen. In diesem Differenzvolumen sind Höhenunterschiede 0.0 m grün dargestellt. Positive Höhenunterschiede sind rot und negative blau gezeichnet. Die größten Höhendifferenzen sind in der Mitte zu sehen, dort befindet sich in der Realität ein Gitterrost. Dadurch entstehen in den unterschiedlichen Auflösungen diese Differenzen. Die fächerförmige Riffelung des Bildes zum Rand des Speichers beschreibt einen Höhenunterschied von maximal vier Millimeter. Die Ursache der (eventuell im Druck nicht erkennbaren) Riffelung liegt in dem »11.1 Mio. Punkte«-Scan, dieser trägt bereits diese Riffelung.

Für die Nullmessungen der Produktspeicher wird die höhere Auflösung empfohlen.
Die Wiederholungsmessungen können in der niedrigen Auflösung gescannt werden. Die entstehende Volumendifferenz ist akzeptabel. Der Zeitgewinn durch die niedrige Auflösung wirkt bereits im Scanvorgang selbst, dort ergibt sich eine Einsparung von 2:18 Minuten. In der Bearbeitung der Punktwolke in der Sannersoftware ergeben sich, durch die geringere Punktanzahl, weitere Zeitgewinne. Der Filtervorgang benötigt statt 2:12 Minuten nur 31 Sekunden. Auch bei dem Export der Punktwolke aus der Scannersoftware sind in dem derzeit verwendeten Dateiformat Wave-OBJ 51 Sekunden einzusparen. Die Volumenauswertung ist, wie oben beschrieben etwa eine Minute schneller. Damit ergibt sich durch die geringere Auflösung eine Zeitersparnis von 5:50 Minuten. Ein weiterer angenehmer Effekt ist die Verringerung der Dateigröße der Exportdatei auf ein Viertel im Vergleich zur »normalen Auflösung«.

6.2 Ausblick und Ausbau

Die Entwicklung dieser Software ist nicht abgeschlossen und wird auch nie vollständig abgeschlossen sein. Dafür gibt es bereits zu viele Ansätze, um die Arbeitsabläufe und auch die Software internen Routinen zu optimieren. Eine Softwareentwicklung sollte auch nie als »abgeschlossen« betrachtet werden, denn die äußeren Bedingungen, unter denen die Software arbeiten muss, unterliegen ständigen Änderungen. Bereits eine Versionsänderung in der eingesetzten CAD-Software AutoCAD bedarf einer Anpassung in der Programmierung, denn auch die API-Schnittstelle von AutoCAD erfährt eine Weiterentwicklung. Auch ist die Entwicklung neuer Laserscanner und Punktwolkendateiformate nicht abgeschlossen. Daher wird dauerhaft ein Bedarf an Anpassung und Optimierung der Software bestehen.

Bereits angedacht sind folgende Punkte:

- In der Voxelsystemerzeugung erfolgt derzeit die Schnittberechnung im Punkt-im-Polygon Algorithmus von einem Testpunkt aus in »+Y«-Richtung und in »-Y«-Richtung, dies geschieht für jeden Testpunkt auf dieser Schnittachse. In Überlegung ist, nur noch die Schnitte in der »X=0«-Ebene zu berechnen und die Schnittpunkte zu puffern. Dann können, ausgehend vom Schnittpunkt mit dem größten Y-Wert, die Bereiche festgelegt werden, welche »innen« liegen. Daraus lässt sich dann die Lage aller Voxelecken (Testpunkte) ableiten. Problematisch wird es nur, wenn ein Polygoneckpunkt auf der Schnittachse liegt. Betrachtet man den Fakt, dass nur ein kleiner Teil (Rundspeicher= 551 Mio. Schnittberechnungen 15 525 Rotationen notwendig) der Schnittachsen diese Problematik aufweist, ist der zu erwartende Effekt doch recht hoch.
- Eine zentrale Steuerdatei für jeden Produktspeicher in der Software implementieren. In dieser wird dann die Bezeichnung des Speichers, die AutoCAD-DWG-Vorlagendatei, die zu-

gehörige Voxel- und Texel-Datei und das Nullmessungsvolumen gespeichert. In einer Folgemessung wird dann nur die Steuerdatei gewählt und die verknüpften Dateien werden automatisch nachgeladen. Als Ausgabe der Volumenbestimmung wird dann das Differenzvolumen, das reine Produktvolumen, ausgewiesen.

- Die Mittelbildung der Volumensäulen soll so umgestellt werden, dass die wirklichen Ausreißer erkennbar sind und diese ausgesondert werden können. Damit kann eine Aussage über die Spannweite der Höhendaten getroffen werden. Auch das dann entstehende mathematische Mittel sollte eine bessere Genauigkeit aufweisen als der Medianwert. Weiterhin ist eine Speicherung der Ausreißerpunkte denkbar, um über mehrere Messungen hinweg diese zu beobachten. Treten diese Punkte an gleicher Stelle häufiger auf, muss dort ein Störkörper vorhanden sein. Diesen kann man dadurch erkennen und in den Sammelkörper einarbeiten.
- Es ist angedacht, die in [Dav13] empfohlene Verfahrensweise, das vollständige Einlesen der Daten in eine Textliste und anschließende parallele Verarbeiten der Textzeilen in dem Einlesen der Punktwolke und dem Einlesen der Volumendaten testweise umzusetzen. Danach muss eine Analyse des Zeitbedarfs erfolgen, erst dadurch kann festgestellt werden ob diese Methode effektiver arbeitet.

An diesen bereits vorhandenen vier Ansätzen erkennt man, dass eine Software immer weiter entwickelt werden kann und auch sollte.

Eine Software, welche mit und für den Anwender entwickelt wird und diesen bei seiner Arbeit unterstützt, wird eine sehr hohe Akzeptanz erfahren. Sie wird wie selbstverständlich eingesetzt und erleichtert die notwendigen Arbeitsabläufe.
In unserer komplexen Arbeitswelt sollten die Computer zur Unterstützung und Entlastung der Arbeitskräfte eingesetzt werden. Dies verlangt nicht nur eine »effektive Software«, sondern vor allem Mitarbeiter, welche in der Lage versetzt werden mit moderner Rechentechnik umzugehen. Dazu ist es unbedingt notwendig, den Wissensstand eines jeden Mitarbeiters zu analysieren und gegebenenfalls durch passende Weiterbildungsangebote auf das nötige Level anzuheben. Nur dann kann auch die »effektive Software« ihr Leistungspotential ausspielen und zur Verbesserung des Arbeitsablaufes beitragen. Die viel und oft beschworene Digitalisierung bedarf nicht nur der Investition in Hardware und Software, sondern mindestens gleichermaßen in die Weiterbildung der Mitarbeiter.

Glossar

boolesche Werte

Sind in der Informatik Wahrheitswerte, wahr oder falsch, »true« oder »false«, 1 oder 0 VI

Double

Ist in der Programiersprache C# (auch in vielen anderen) ein Schlüsselwort, es beschreibt eine Fließkomma-Zahl mit doppelter Genauigkeit. Sie benötigt 64 Bit Speicherplatz und kann Zahlenwerte mit 16 signifikante Stellen speichern. VI

Epoche

Bezeichnet in der Gedäsie den Zeitpunkt einer Messung, oft bei sich wiederholenden Messungen. VI

Extrusion

Bezeichnet in der Volumenkörpermodellierung das »Aufziehen« von Flächen entlang einer Mittelachse zu Körpern. VI

Integer

Ist in der Programiersprache C# (auch in vielen anderen) ein Schlüsselwort, es beschreibt eine natürliche Zahl. Sie wird in unterschiedlicher Größe gespeichert, in 16 Bit, 32 Bit oder 64 Bit. Je nach Größe ist der mögliche Zahlendarstellungsbereich unterschiedlich. VI

Plugin

Beschreibt eine Funktionserweiterung eines bestehenden Programmes. Das Plugin bedient sich der vorhandenen Schnittstelle des Programmes und fügt neue im Programm nutzbare aufrufbare Funktionen zur Verfügung. Das Plugin ist häufig eine DLL-Datei und wird während des Programmstartes automatisch geladen. VI

Property (Properties)

Eigenschaft(en) einer Klasse, die durch öffentliche Getter- und Setterfunktionen erreichbar ist(sind). Die Verwendung solcher Eigenschaften gegenüber einfacher public-Eigenschaften hat den Vorteil, dass sie durch Funktionen des .NET-Framworks aufgelistet und ausgelesen werden können. VI

Pseudocode

Programmiersprachenähnliche Codedarstellung, jedoch für den Leser in lesbarer verständlicher Textform. VI

RGB-Wert

Hierbei wird die Farbe eines Bildpunktes als Mischkombination aus rotem, grünem und blauem Licht abgelegt. Diese drei Zahlwerte, einem Triple, werden in verschiedenen Varianten gespeichert, häufig als Bereich zwischen 0 und 255. VI

Tachymeter (tachymetrisch bestimmt)

Ein Tachymeter ist ein Vermessungsinstrument, mit dem man Richtungen am Horrizontalkreis (= »Winkelmesser« in der Ebene), Vertikalwinkel (0 GON = im Zenit) und die Distanz (schräge Strecke vom Instrument zum Zielpunkt) erfassen kann. Bei modernen Tachymetern geschieht dies elektronisch. Aus diesen Werten können mit Hilfe der Steuersoftware Koordinaten gerechnet werden. Diese werden gespeichert und im Büro ausgelesen. VI

Texel

Begriff aus dem Rendering. Dort bezeichnet der Texel die kleinste Einheit einer Textur, ein Texturpixel. In dieser Arbeit bezeichnet ein Texel eine quadratische Fläche, sprich ein Pixel. VI

Voxel

Ein Raumwürfel mit festgelegter Kantenlänge. Mit diesen Raumwürfeln können komplexe Volumenkörper vereinfacht dargestellt werden. VI

Zoom und Pan

Zoom und Pan beschreiben in der CAD-Software das Bewegen in der Zeichnung. Zoom ist, analog zum Fotoapparate-Zoom, das Vergrößern und Verkleinern des sichtbaren Bereiches. Die dargestellten Objekte werden kleiner (heraus zoomen) oder größer (heran zoomen). Pan beschreibt das Verschieben des sichtbaren Bereiches der Zeichung. VI

Literaturverzeichnis

Monographien

[Alb09] Jörg Albertz. *Einführung in die Fernerkundung Grundlagen der Interpretation von Luft- und Satellitenbildern. Grundlagen der Interpretation von Luft- und Satellitenbildern*. ger. 4., aktualisierte Aufl. Darmstadt: WBG (Wiss. Buchges.), 2009. 254 S. isbn: 978-3-534-23150-8.

[Ern+16] Hartmut Ernst, Jochen Schmidt und Gerd Hinrich Beneken. *Grundkurs Informatik Grundlagen und Konzepte für die erfolgreiche IT-Praxis - Eine umfassende, praxisorientierte Einführung. Grundlagen und Konzepte für die erfolgreiche IT-Praxis - Eine umfassende, praxisorientierte Einführung*. ger. 6. Auflage 2016. Ernst, Hartmut (VerfasserIn) Schmidt, Jochen (VerfasserIn) Beneken, Gerd Hinrich (VerfasserIn). Wiesbaden: Springer Vieweg, 2016. 809 S. isbn: 978-3-658-14633-7. doi: 10.1007/978-3-658-14634-4.

[FH11] Peter Fischer und Peter Hofer. *Lexikon der Informatik*. ger. 15., überarb. Aufl. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2011. isbn: 9783642151255. doi: 10.1007/978-3-642-15126-2.

[IW94] Alfred Iwainsky und Wolfgang Wilhelmi. *Lexikon der Computergrafik und Bildverarbeitung Mit über 1000 Eintragungen, zahlreichen Querverweisen. Mit über 1000 Eintragungen, zahlreichen Querverweisen*. Braunschweig: Vieweg, 1994. 351 S. isbn: 3528053429.

[RM04] Wolfgang Reinhardt und Michael Möser. *Raumbezogene Informationssysteme*. ger. Bd. / Michael Möser ... (Hrsg.) Handbuch Ingenieurgeodäsie. Heidelberg: Wichmann, 2004. 226 S. isbn: 3879072949.

[Sch+15] Bettina Schütze, Andreas Engler und Harald Weber. *Lehrbuch Vermessung - Fachwissen*. ger. 2., vollständig überarbeitete Auflage. Schütze, Bettina (VerfasserIn) Engler, Andreas (VerfasserIn) Weber, Harald (VerfasserIn). Dresden: Schütze Engler Weber Verlags GbR, 2015. 481 S. isbn: 978-3-936203-27-1.

Zeitschriftenartikel

[CK95] Daniel Cohen-Or und Arie Kaufman. »Fundamentals of Surface Voxelization«. In: *Graphical Models and Image Processing* 57.6 (1995). PII: S1077316985710398, S. 453–461. issn: 10773169. doi: 10.1006/gmip.1995.1039.

[Liu+20] Rufei Liu u. a. »Hierarchical Classification of Pole–like Objects in Mobile Laser Scanning Point Clouds«. In: *The Photogrammetric Record* 1.5 (2020), S. 76. issn: 0031-868X. doi: 10.1111/phor.12307.

[RS99] Klaus Reichenberger und Ralf Steinmetz. »Visualisierungen und ihre Rolle in Multimedia-Anwendungen«. In: *Informatik-Spektrum* 22.2 (1999). PII: TWD6YFM58PDEHM5F, S. 88–98. issn: 0170-6012. doi: 10.1007/s002870050128.

[Sch+13] Bianca Schön u. a. »Octree-based indexing for 3D pointclouds within an Oracle Spatial DBMS«. In: *Computers & Geosciences* 51 (2013). PII: S009830041200307X, S. 430–438. issn: 00983004. doi: 10.1016/j.cageo.2012.08.021.

Konferenzschriften

[Hua+98] Jian Huang u. a. »An accurate method for voxelizing polygon meshes«. In: IEEE Symposium on Volume Visualization (Cat. No.989EX300). 1998. doi: 10.1109/SVV.1998.729593.

Beiträge im Internet/ Webseiten

[Dav13] David Lozinski. *The Fastest Way to Read and Process Text Files using C# .Net*. 2013. url: https://cc.davelozinski.com/c-sharp/the-fastest-way-to-read-and-process-text-files (besucht am 29. 03. 2020).

[Lui16] Luis Colorado. *Point inside a 3d closed Volume*. Hrsg. von stackoverflow.com. 2016. url: https://stackoverflow.com/questions/38214038/point-inside-a-3d-closed-volume (besucht am 27. 03. 2020).

[Pat20] Patrick Min. *Homepage der binvox-Software,* 2004-2020. url: http://www.patrickmin.com/binvox (besucht am 28. 03. 2020).

[Ste09] Stefan Wörthmüller. *Multithreaded File I/O*. 2009. url: https://www.drdobbs.com/parallel/multithreaded-file-io/220300055 (besucht am 29. 03. 2020).

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Anlagen

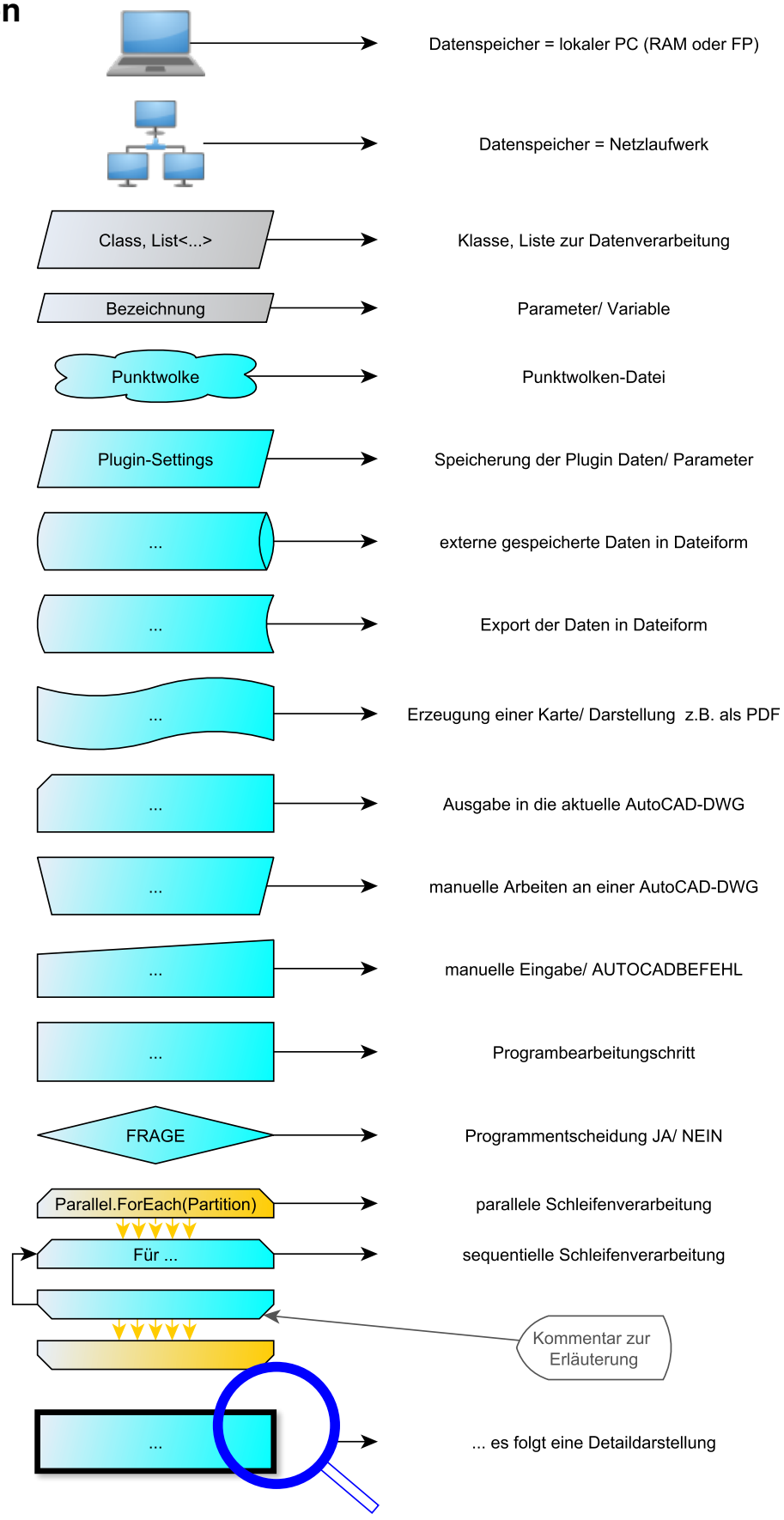


Abbildung A.1: Legende der verwendeten Diagrammsymbole

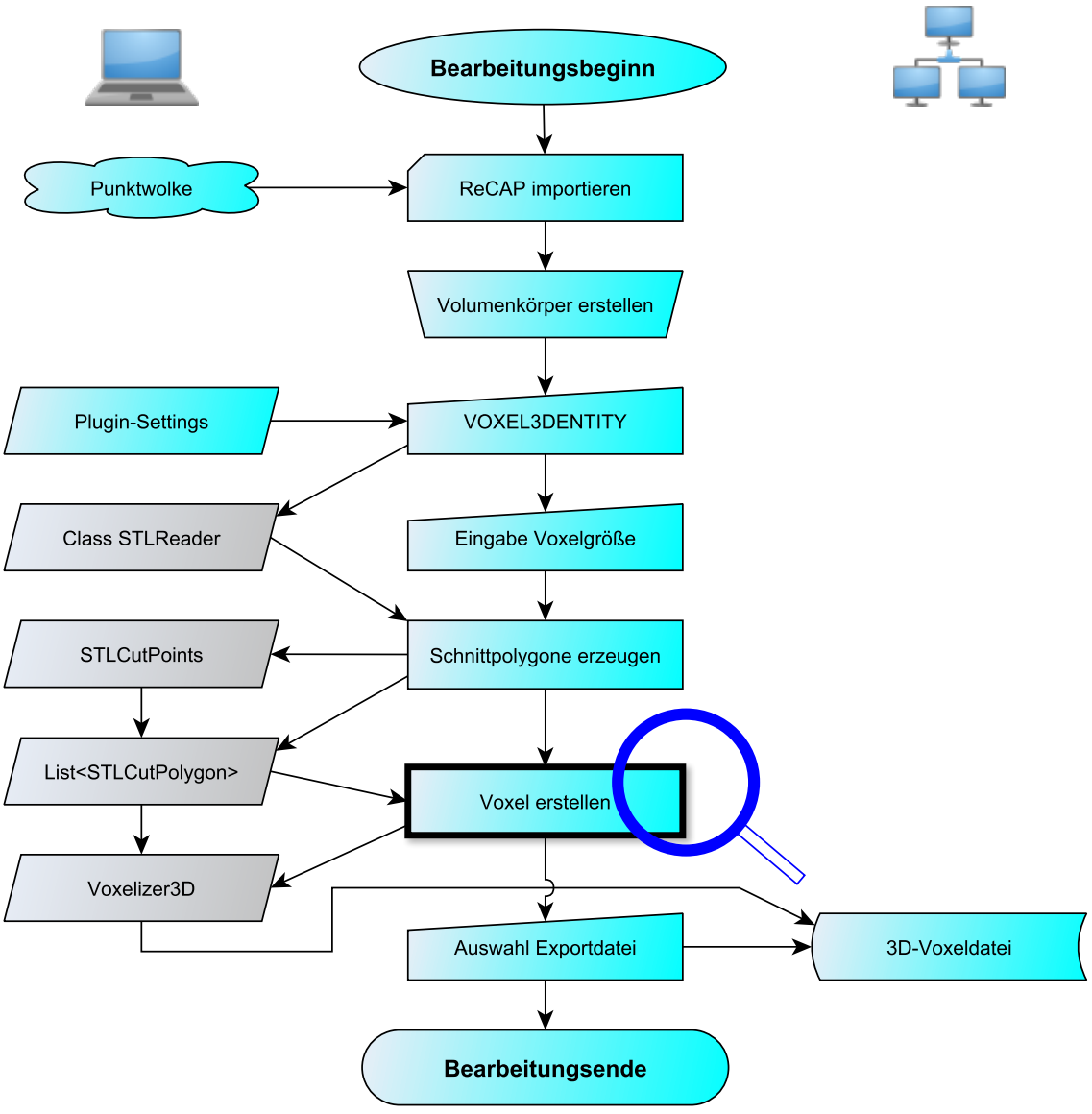


Abbildung A.2: Ablaufdiagramm der Voxelerzeugung

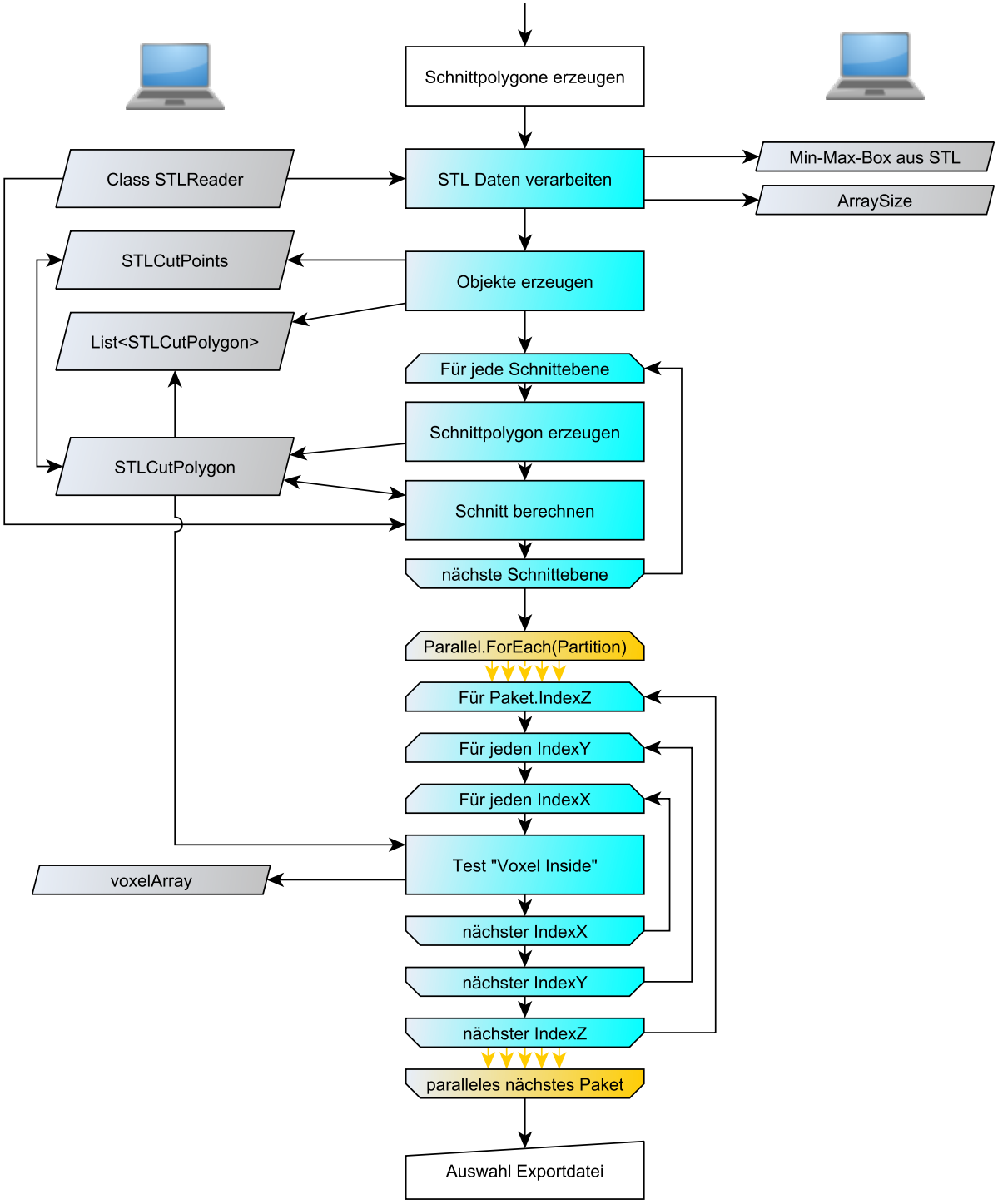


Abbildung A.3: Detaildiagramm des Programmschrittes »Voxel erstellen«

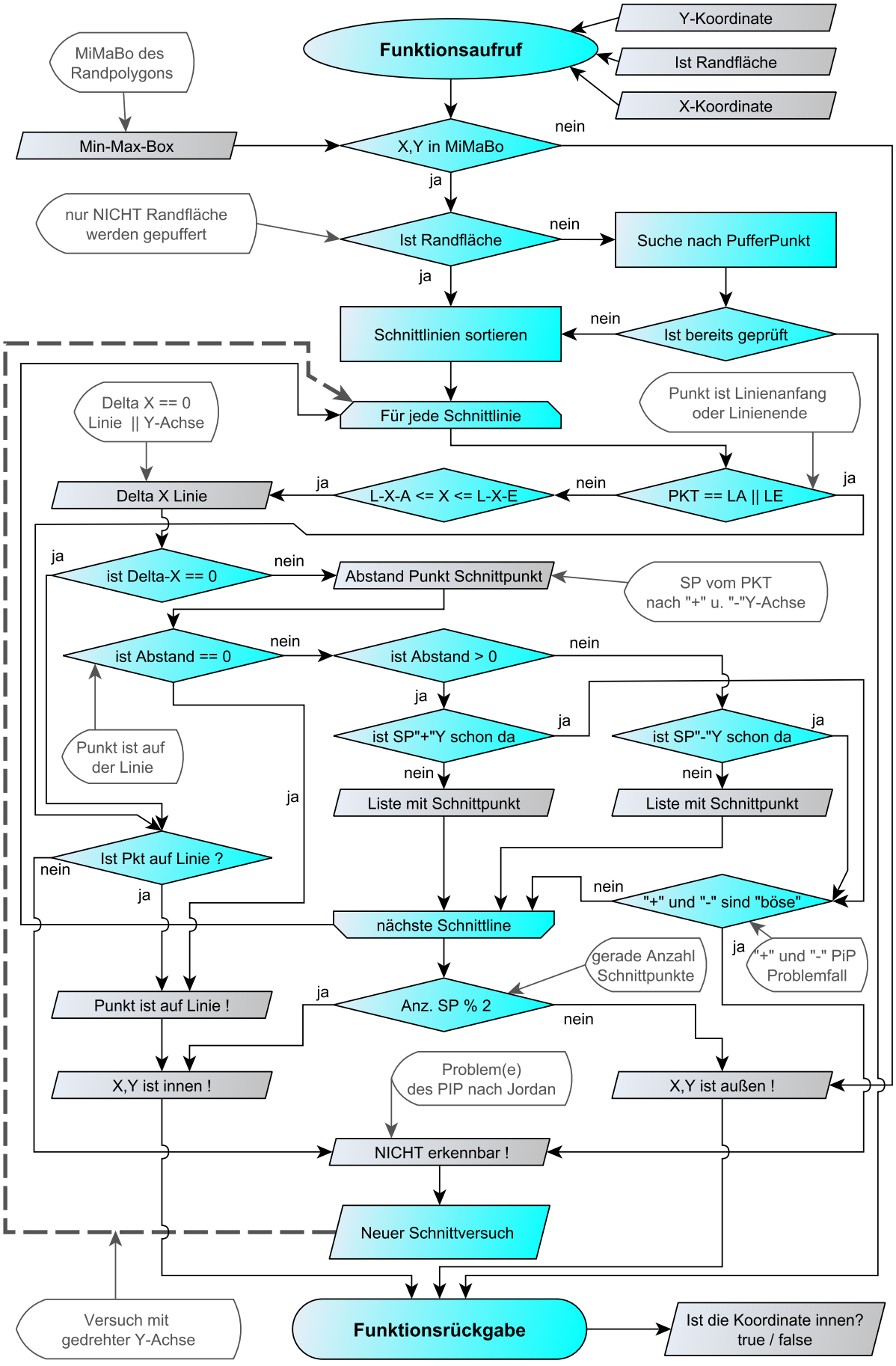


Abbildung A.4: Ablaufdiagramm der "Punkt-im-Polygon"Funktion

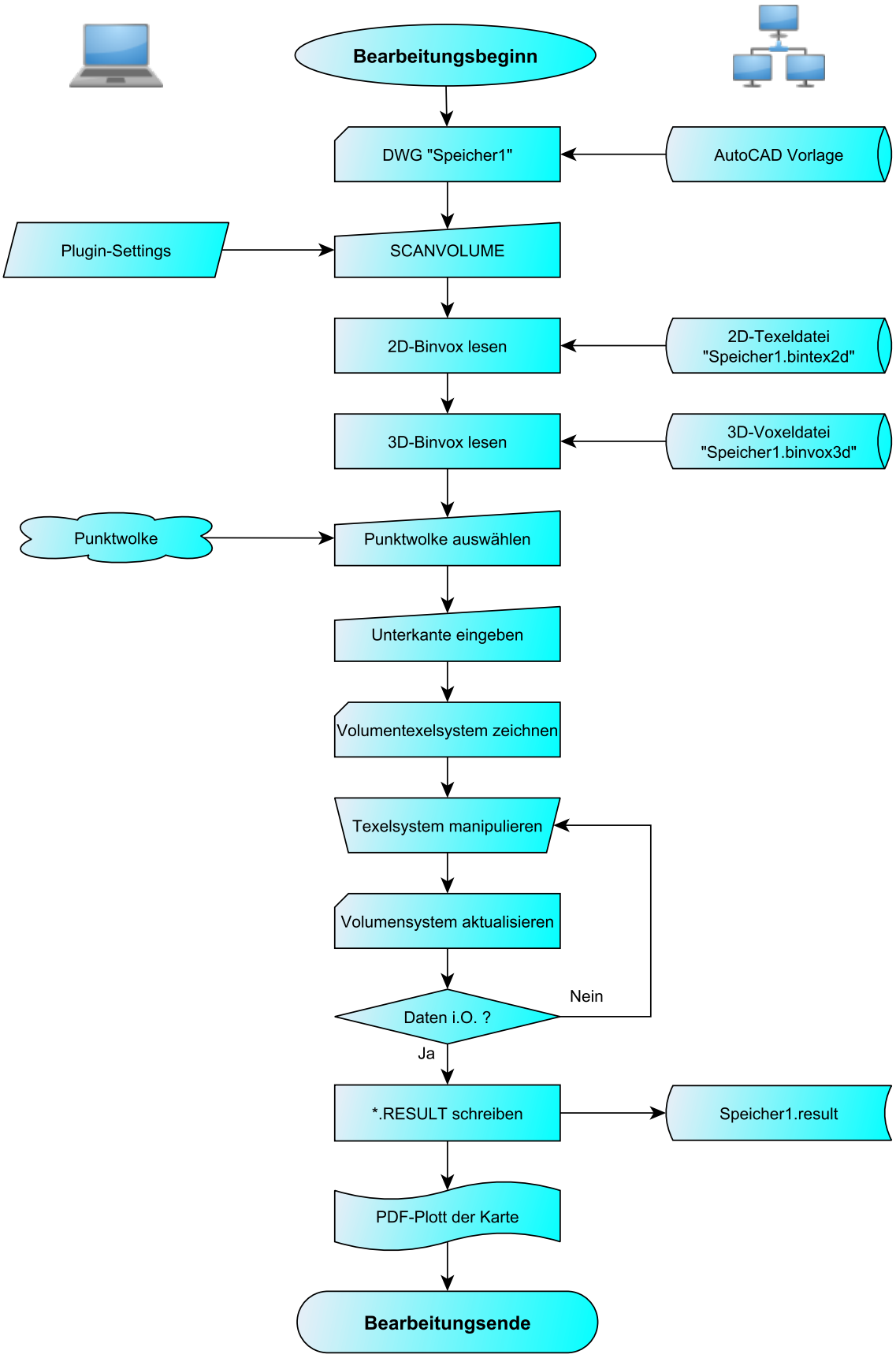


Abbildung A.5: Ablaufdiagramm einer Scanauswertung

Auflistung aller, speziell für dieses Projekt erzeugter Klassen:

Klassenname	Beschreibung
KSRO_ScanVolume - Projekt (ca. 2000 Zeilen Quellcode)	
AcPluginScanVolumen	Die Plugin-Arbeitsklasse, diese wird von AutoCAD gelesen und beinhaltet die aufrufbaren AutoCAD-Befehle.
AppSettingsScanVolume	Datenklasse zum Einstellungs- / Parameter-Handling.
InfoFromThisDLL	Toolklasse, enthält jede DLL, gibt das Informationen zu dem DLL-Assembly zurück.
	Klassen zum Handling der MVVM-Struktur des Einstellungs-Änderung-Fenster ...
SettingsControl	Controler-Klasse, diese wird aufgerufen und kapselt den »Show()«-Aufruf.
SettingsModel	Modelklasse, speichert Position, Fenster- und Schriftgröße.
SettingsViewModel	ViewModel-Klasse, stellt als Getter und Setter Properties und Commands für das View zur Verfügung.
SettingsView	XAML-Klasse mit der Fensterstruktur.
	Klassen für die Volumenberechnung ...
PointFileFormat	Einstellung der möglichen Punktwolkendateien OBJ und PTS.
VolumeGrid	Klasse des Volumenmodells.
VolumeGridCell	Klasse einer Volumensäule der Volumenmodells.
VolumeMesh	Klasse zur Netzbildung und Interpolation des Z-Wertes an der X-Y Koordinate.
clDrawOnAutoCAD	Kapselung aller Funktionen mit direktem DWG-Zugriff.

Tabelle A.1: Übersicht der erstellten Klassen im Hauptprojekt

Klassenname	Beschreibung
Core_VermessungsTools - Projekt (ca. 2200 Zeilen Quellcode)	
	Klassen zum Arbeiten mit dem STL-Flächenmodell ...
enPointMustInside	Enumeration mit den möglichen Festlegungen, wie viele Punkte innen liegen müssen. (OneOfFoure bis AllOfFour und OneOfEight bis AllOfEight)
STLReader	Klasse zum Lesen einer STL-Datei und zur Datenhaltung eines aus einem AutoCAD-Solid erzeugtem STL-Flächenmodell.
STLTriangle	Klasse zur Datenhaltung eines STL-Flächendreiecks.
STLCutLine	Stellt eine Schnittlinie mit einem STL-Flächendreieck dar.
STLCutPolygon	Stellt das Polygon einer Schnittberechnung zwischen X-Y-Ebene und STL-Flächenmodell dar, incl. Schnittberechnung.
STLCutPoints	Datenhaltung für die Schnittpunktpufferung während der Voxelberechnung.
STLCutYList	Datenhaltung für Schnittpunkte bei Y-Achsen paralleler Schnittachse in der Punkt-im-Polygon Funktion.
STLCutXYList	Datenhaltung für Schnittpunkte bei gedrehter Schnittachse in der Punkt-im-Polygon Funktion.
	Klassen für das Texelsystem und das Voxelsystem ...
Texelizer2D	Klasse zum Erzeugen eines Texel-Modells aus Kreis oder Polygon.
TexelCell2D	Datenklasse einer Texel-Zelle.
LinearQuadTree	Umsetzung des »linearen«-Quadtree im Texelsystem.
BoundaryTexel	Erweiterung des LinearQuadTree um einen »Randtexel« zu modellieren.
Voxelizer3D	Klasse zum Erzeugen,Speichern und Lesen des Voxelsystems.

Tabelle A.2: Übersicht der erstellten Klassen in der Library

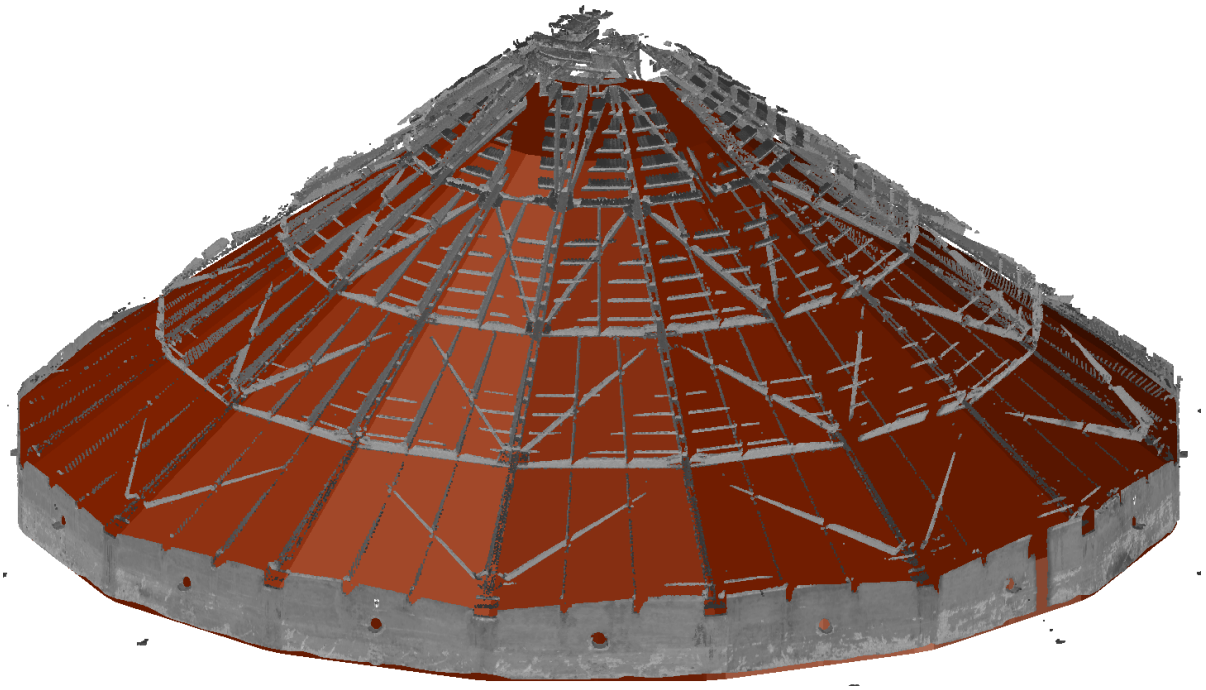


Abbildung A.6: Sammelkörper eines Rundspeichers in der Punktwolke

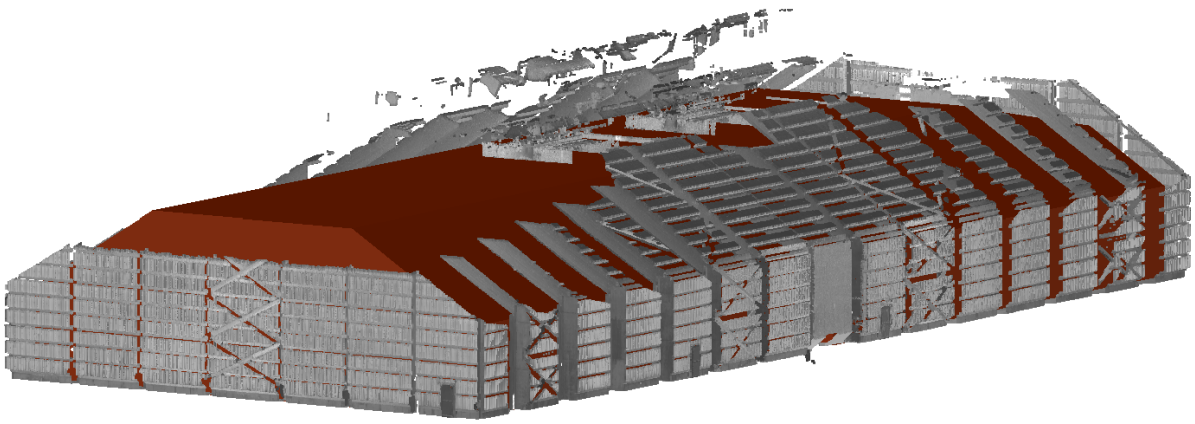


Abbildung A.7: Sammelkörper eines Rechteckspeichers in der Punktwolke

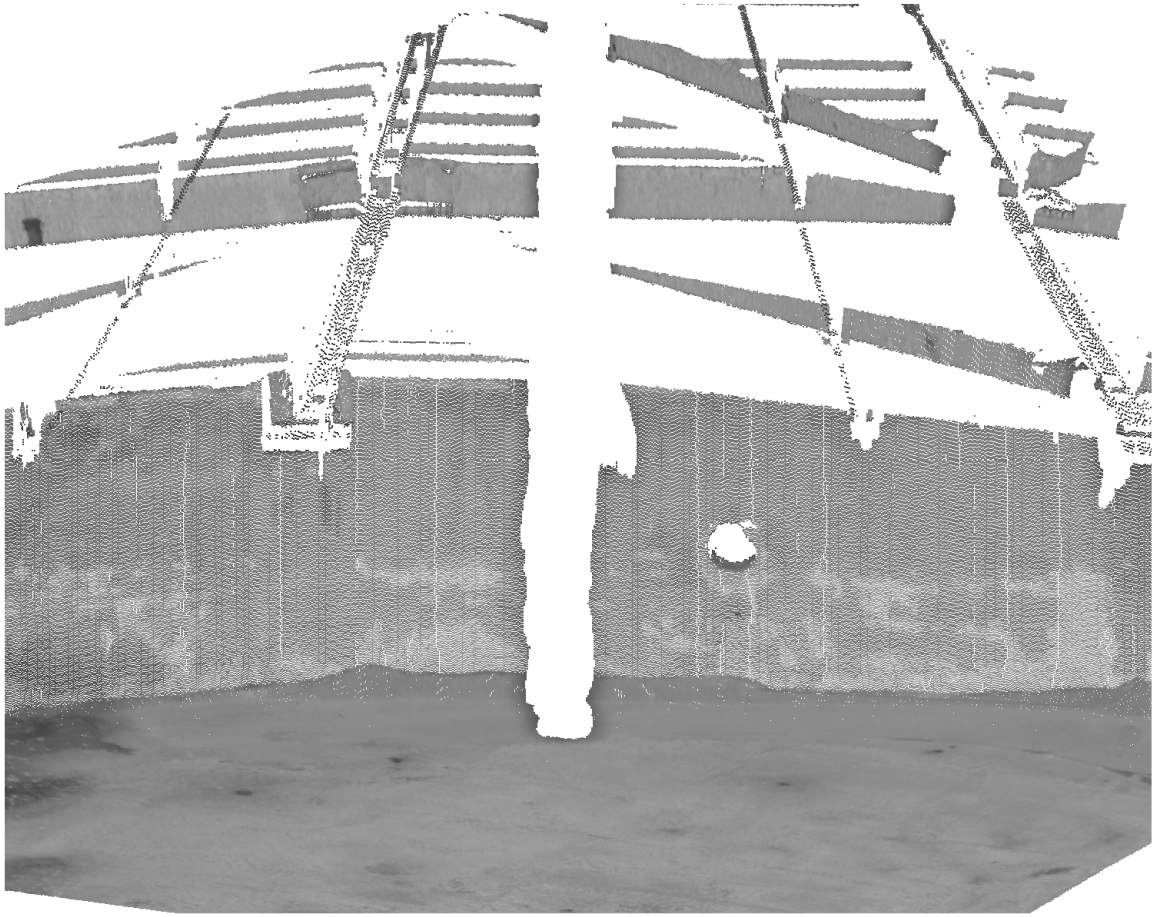


Abbildung A.8: Scanschatten durch ein Einspeicherrohr

Abbildung A.9: Fahrzeug in der Punktwolke

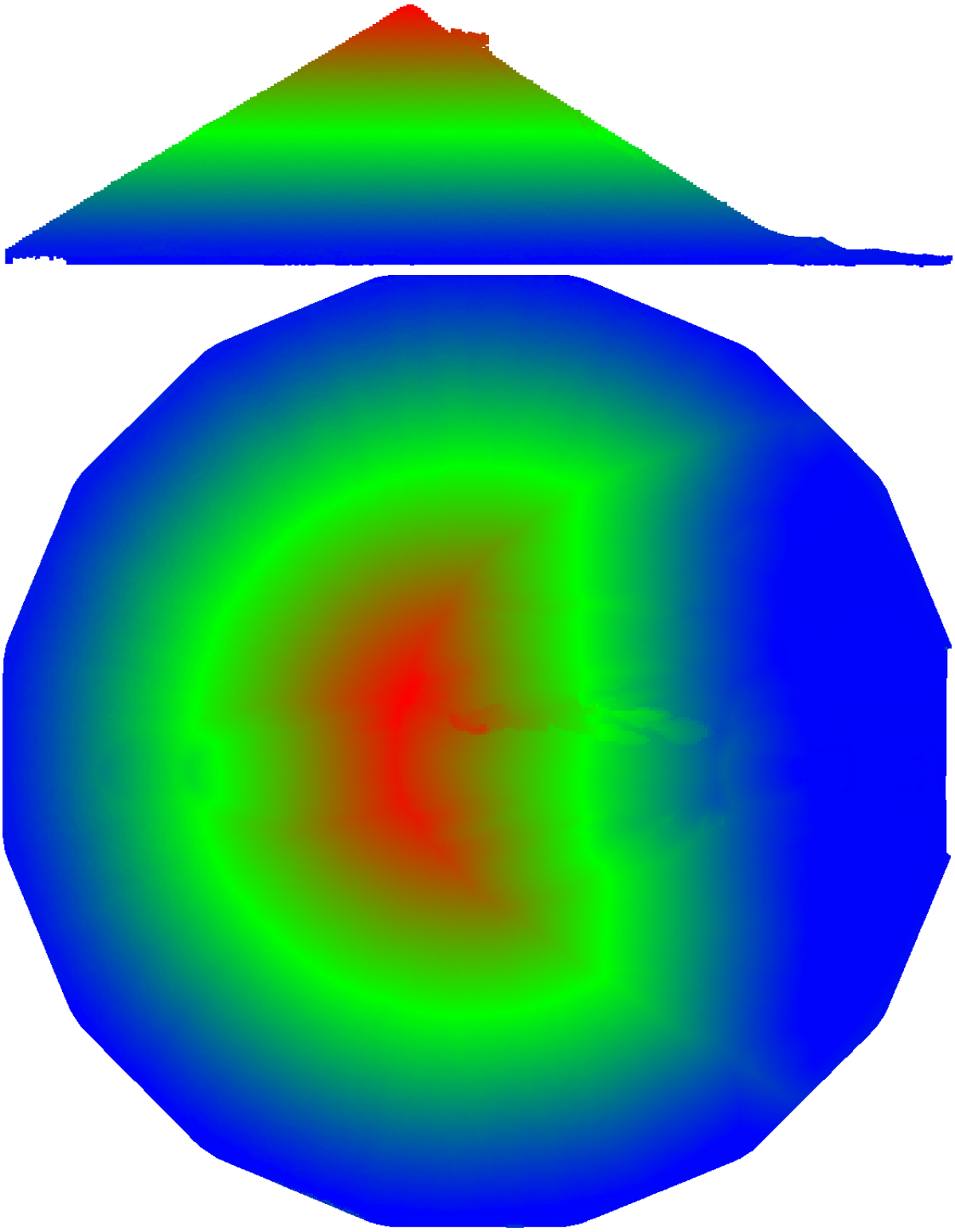


Abbildung A.10: Darstellung des Rundspeicher-Volumenmodells, oben Profil, unten Draufsicht

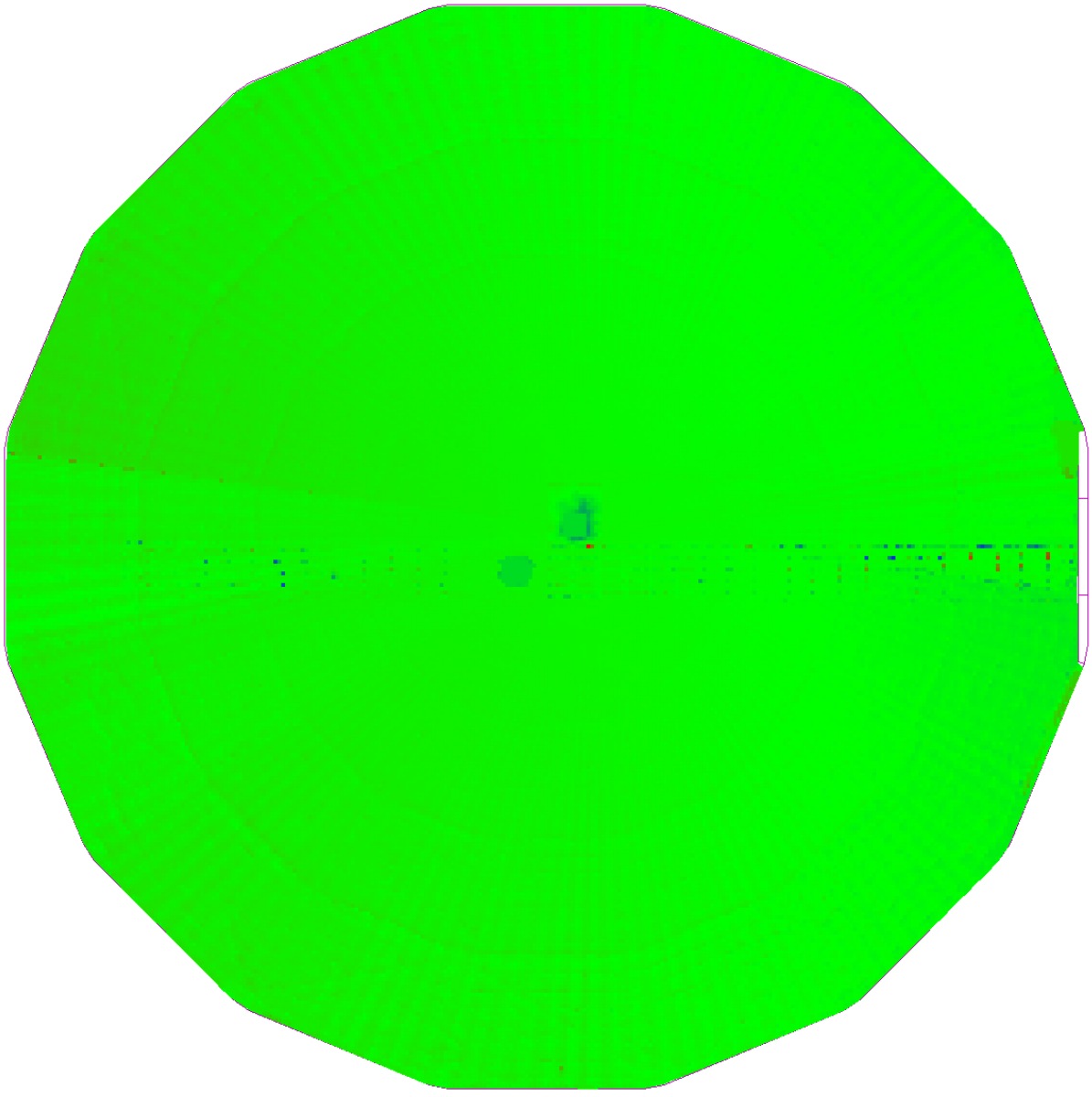


Abbildung A.11: Höhendarstellung des Vergleiches der Volumen aus 44.4 Mio. Punkte vs. 11.1 Mio. Punkte, höchster Punkt (rot) +11 cm, tiefster Punt (blau) -8 cm

Eigenständigkeitserklärung

Erklärung über die eigenständige Erstellung der Arbeit

Hiermit erkläre ich, Mario Rosenbohm, dass ich die vorgelegte Arbeit mit dem Titel »Automatisierte Bestimmung von Schüttgutvolumen aus Punktwolken« selbständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie alle wörtlich oder sinngemäß übernommenen Stellen in der Arbeit als solche und durch Angabe der Quelle gekennzeichnet habe. Dies gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet.

Mir ist bewusst, dass die Hochschule für Technik und Wirtschaft Dresden Prüfungsarbeiten stichprobenartig mittels der Verwendung von Software zur Erkennung von Plagiaten überprüft.

Loitsche, 15.05.2020

Mario Rosenbohm