Technische Universität München
Ingenieurfakultät Luftfahrt, Raumfahrt und Geodäsie
Studiengang Geodäsie und Geoinformation
Lehrstuhl für Geoinformatik
Univ.-Prof. Dr. rer. nat. Thomas H. Kolbe
In Kooperation mit Audi AG

Bachelorarbeit

Identifikation von geometrischen Abweichungen zwischen semantischen Stadtmodellen und Punktwolken

Verfasser: Alexander Gawronski
Matrikelnummer: 03686261
Bearbeitungszeitraum: 02.11.2020 – 08.02.2021
Kontakt: alexander.gawronski@tum.de

Betreuer: Univ.-Prof. Dr. rer. nat. Thomas H. Kolbe
M.Sc. M.Sc. Benedikt Schwab

Leicht aktualisierte Version

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Bachelors Thesis selbständig und ohne unzulässige fremde Hilfe angefertigt habe. Die verwendeten Literaturquellen sind im Literaturverzeichnis vollständig aufgeführt.

München, den 08.02.2021

_____

Kurzfassung

In dieser Arbeit ist ein Verfahren zur Analyse geometrischer Abweichungen in Stadt- und Straßenmodellen erstellt worden. Untersucht werden Abweichungen, die bei Modellflächen in Bezug auf Punktwolkendaten auftreten, auf deren Grundlage das Modell erstellt worden ist. Exemplarisch werden hierfür alle Wände und Türen eines Testbereiches aus mehreren Gebäudemodellen, sowie die Straßenmarkierungen eines weiteren Testbereichs, anhand von Metriken auf mögliche Abweichungen analysiert. Das Ergebnis der Analysen wird jeweils in einem Modell mit eingefärbten Flächen dargestellt, wobei sich anhand der Farbwahl erkennen lässt, um welche Art von Abweichung es sich hierbei handelt. Im Rahmen dieses Arbeitstextes wird auf verwandte Arbeiten und thematische Grundlagen eingegangen, sowie der Aufbau und die Implementierung des mithilfe der Feature Manipulation Engine (FME) entwickelten Programms vorgestellt. Anschließend wird das Ergebnis anhand eines Testdatensatzes vorgestellt und bewertet. Anhand der Ergebnisse werden sowohl Schwachstellen der Implementierung als auch der vorliegenden Daten aufgezeigt. Außerdem wird auf Verbesserungsmöglichkeiten und auf denkbare Erweiterungen des Programms hingewiesen.

Inhaltsverzeichnis

1. Einleitung

1.1 Motivation

Das Automobil hat seit seiner offiziellen [1] Erfindung einen steten technischen Fortschritt zugunsten des Fahrkomforts, vor allem aber der Sicherheit [2] erfahren. Eine Schlüsselkomponente hat sich dennoch seitdem nicht verändert: Gesteuert wird das Auto bis heute von einem Menschen. Mit der zunehmenden Rechenleistung und immer kompakteren Bauweise moderner Computersysteme scheint das selbstfahrende und mitdenkende Auto immer weiter von einer blassen Abbildung einer möglichen Zukunft zu einer konkreten Realität zu werden [3].

Doch wie wird Fahrzeugautonomie definiert und worin unterscheiden sich ihre Detailgrade? Zur Beantwortung dieser Frage hat die *Society of Automotive Engineers* im Jahre 2016 den Fünf-Stufenplan der Fahrzeugautonomie veröffentlicht, der Richtlinien für die internationalen Bemühungen zur Entwicklung von autonomen Fahrzeugen und Fahrerassistenzsystemen festhält [4]:

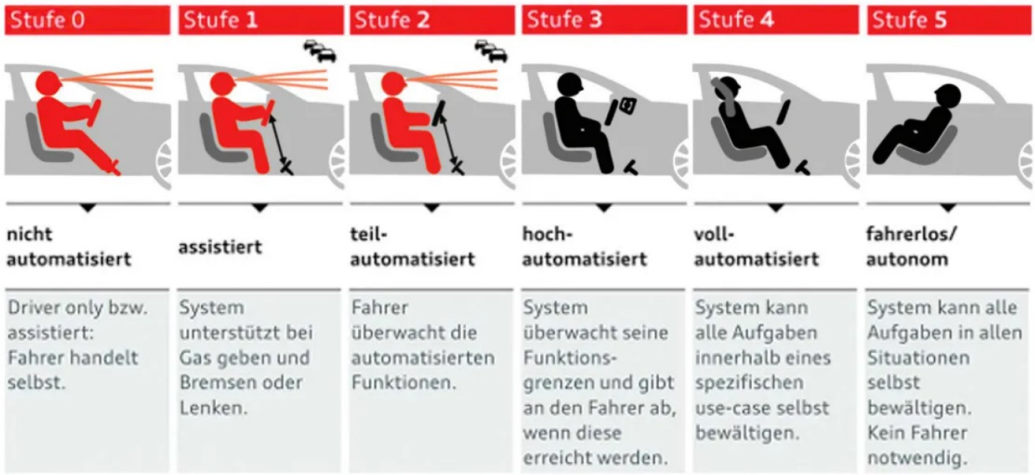

Abbildung 1 Der Fünf-Stufenplan der Fahrzeugautonomie [5]

Nach aktuellem Stand sind die Stufen 0 und 1 mit am weitesten verbreitet. Assistenzsysteme, wie etwa der Einparkassistent in Fahrzeugen des deutschen Automobilherstellers Volkswagen, bei denen beispielsweise das Fahrzeug die Lenkbewegungen während des Einparkvorganges übernimmt oder Notfallbremssysteme zum Verhindern von Auffahrunfällen zählen gemäß den Richtlinien des Plans zur Stufe eins. Für darüberliegende Stufen gibt es nach aktuellem Stand[1] nur wenige Praxisbeispiele, die zudem nur in bestimmten Ländern zugelassen sind. So verfügen etwa die Fahrzeuge des US-amerikanischen Automobilherstellers Tesla [6] oder des US-amerikanischen Unternehmens Waymo [7] über einen Autonomisierungsgrad, der zum Teil der dritten Stufe entspricht [6]. Modelle höherer Automatisierungsgrade sind heute aber – vor allem aus ethischen Gründen [8] – noch nicht marktreif und bedürfen weiterhin umfassender Forschung.

Abbildung 2 Fahrzeuge mit zusätzlichen Sensoren der Firma Waymo [7]

[1] Januar 2021

Im Zeitalter der Digitalisierung bieten sich für die Entwicklung dieser Systeme Simulationen an. So können über rein computergestützte Verfahren nützliche Informationen zur Weiterentwicklung bestehender Systeme und zur Entwicklung ganz neuer Ansätze gesammelt werden, bevor diese in die Realität umgesetzt werden. Semantische Stadtmodelle liefern hierbei ein wichtiges Werkzeug zur Entwicklung der benötigten Simulationsumgebungen, da hier neben der räumlichen Information in Form von geometrischen Objekten auch weitergehende und nicht-geometrische Objektmerkmale für umfassende räumliche und thematische Analysen enthalten sind. Zur Entwicklung einer sinnvoll nutzbaren Simulationsumgebung ist ein genaues geometrisches Abbild der Realität aber grundlegende Voraussetzung. Abweichungen in der Geometrie können sich hier auf die Messergebnisse in den Simulationen auswirken und die Entwicklungszeit negativ beeinflussen. Es ist also ratsam, das Stadtmodell schon bei der Modellierung aus Laserscandaten auf mögliche Abweichungen in der Geometrie hin zu untersuchen.

1.2 Zielsetzung

Ziel dieser Arbeit ist ein mit der *Feature Manipulation Engine* (FME) entwickeltes Programm, mit dem der Nutzer ein von Hand erstelltes Stadtmodell im CityGML-Format auf geometrische Abweichungen zu einer LiDAR-Punktwolke, die als Grundlage für das Modellieren des Stadtmodells gedient hat, hin zu überprüfen. Die einzelnen Flächen des Modells sind im Endergebnis farblich zu kennzeichnen, damit auf einem Blick deutlich wird, bei welchen Objektflächen Abweichungen auftreten. Mit dem Programm soll es auch möglich sein, aktuellere Laserscans per Punktwolke mit dem Modell zu vergleichen, um sehen zu können, ob sich Änderungen durch eine neue Messsituation ergeben. Im Arbeitsumfang ist zudem ein zweites Programm anzufertigen, mit dem Abweichungen zwischen einem Straßenmodell und den dazugehörenden Punktdaten untersucht werden können. Hier liegt der Fokus auf der Modellierung von Straßenmarkierungen. Der Nutzer kann mit dem gebotenen Programmumfang gezielt die Flächen identifizieren und bei Bedarf nachbessern. Außerdem bietet eine zusätzliche MATLAB-Anwendung eine Übersicht über einzelne Ergebnisse der metrischen Analyse in Form geeigneter Plots.

2. Verwandte Arbeiten aus der angewandten Wissenschaft

2.1 Semantic-based Geometry Refinement of 3D City Models for Testing Automated Driving

Im Rahmen seiner Masterarbeit [9] "*Semantic-based Geometry Refinement of 3D City Models for Testing Automated Driving"* hat Olaf Wysocki ein Verfahren zur Verbesserung von bestehenden Straßenraummodellen entwickelt, in dem die bestehenden Modelle durch Punktwolkendaten ergänzt und verbessert werden. Verwendet wurden hierfür Testdaten im Bereich der Ingolstädter Innenstadt in Form von Gebäudemodellen in mehreren Detailstufen, sowie ein Straßenraummodell, jeweils im CityGML-Format. Die Methode ist in mehreren Workspaces mit der Feature Manipulation Engine entwickelt worden und umfasst zudem ein detailliertes Konzept zur Verbesserung von Gullideckeln im Straßenmodell anhand der Punktdaten. Die entwickelten Workspaces in FME boten für die hier vorliegende Arbeit einen Leitfaden für Konventionen bei der Entwicklung von FME-Workspaces und die Verarbeitung von Laserscandaten.

2.2 Straßenraummodellierung mittels Mobile Mapping in OpenDRIVE und CityGML sowie geeignete Visualisierungsmethoden

Theresa Coduro befasst sich in Ihrer Masterarbeit [10] "*Straßenraummodellierung mittels Mobile Mapping in OpenDRIVE und CityGML sowie Entwicklung geeigneter Visualisierungsmethoden"* mit der Umwandlung und geeigneten Visualisierung von Laserscandaten, in Form eines Stadtmodells im CityGML-Format. Entwickelt werden die hierfür notwendigen Prozessierungen in FME, die Visualisierung des Ergebnisses erfolgt mithilfe eines 3DCityDB Web-Map-Clients. Die Arbeit liefert gemeinsam mit der in Kapitel 2.1 vorgestellten Literatur einen thematischen Rahmen und Hilfestellung für den Einstieg in das Themengebiet Modellierung und den Umgang mit semantischen Stadtmodellen.

3. Technische und mathematische Grundlagen

3.1 Laserscanning und fahrzeuggetragene Messsysteme

3.1.1 Laser und das Lasermessprinzip

Als Laser wird durch spezielle elektrische Bauteile aktiv erzeugte, kohärente, gerichtete, und stark gebündelte elektromagnetische Strahlung bezeichnet [11]. Für die Emission dieses speziellen Lichts ist in modernen Messsystemen, wie beispielsweise Tachymetern, die so genannte Lumineszenzdiode zuständig: Durch einen Anregungsstrom tritt an bei dieser Halbleiterdiode elektromagnetische Strahlung aus, die anschließend gebündelt und ausgesandt wird [12]. Das verwendete Material bestimmt dabei den Spektralbereich dieser emittierten Strahlung.

Messen lassen sich mit Laser Distanzen über Laufzeitmessungen. Mit der gemessenen Dauer t zwischen Aussendung des Laserimpulses und Empfang der vom Messobjekt reflektierten Antwort und der Lichtgeschwindigkeit c ergibt sich die Distanz s aus der halben Laufzeit – denn die Gesamtlaufzeit umfasst Hin- und Rückweg [12]:

$$s = \frac{c}{2} * t$$

Diese Formel gilt aber nur für den Fall der Messung im Vakuum [11] [12]. Um auf die tatsächliche Distanz zum Zeitpunkt der Messung zu kommen, ist eine meteorologische Korrektur anzubringen. Üblicherweise geben die Hersteller von Messsystemen hierfür eine Bezugsatmosphäre vor und erlauben durch die Eingabe meteorologischer Messdaten zum Zeitpunkt der Distanzmessungen die direkte interne Korrektur der gemessenen Distanz.

3.1.2 Wechselwirkung von elektromagnetischer Strahlung mit Materie

Elektromagnetische Strahlung interagiert mit Materie. Diese Interaktion setzt sich dabei zusammen aus den drei Anteilen Reflektion, Transmission und Absorption bei Kontakt mit Materie. Die Summe der drei Anteile hat dabei nach dem Energieerhaltungssatz immer den Wert 1 [13]. Eine wichtige Aussage über ein betrachtetes Objekt liefert die Intensität, da jedes Material seine eigene spektrale Signatur besitzt [13]. Darunter versteht man den Verlauf der Intensität im Spektralbereich.

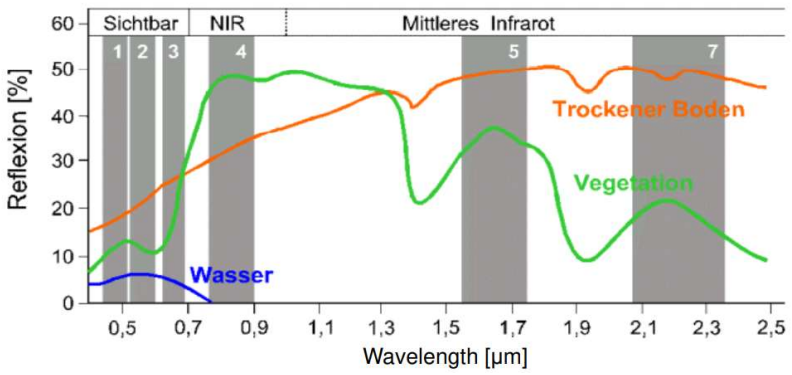

Abbildung 3 Charakteristische spektrale Kennlinienverläufe [13]

Dies gilt allerdings auch für die Interaktion mit der Atmosphäre und den darin enthaltenen Partikeln. Für die Intensität der Strahlung ergibt sich deshalb: Die Intensität I nimmt gegenüber der Intensität I_0 bei der Emission für eine zurückgelegte Distanz s ab und ergibt sich dann zur aktuellen Intensität:

$$I = I_0 * e^{-\frac{1}{\lambda^4} * s}$$

Die Intensität nimmt also abhängig von der verwendeten Wellenlänge über die Strecke s hinweg ab. Die Abschwächung wird aber bei zunehmender Wellenlänge geringer, sodass die erreichbare Reichweite für Lasermesssysteme im nahen Infrarotbereich höher ist als im kurzwelligeren Bereich des sichtbaren Spektralbereichs[2].

3.1.3 Aufnahmeverfahren Laserscanner und mobile Lasermesssysteme

Mithilfe eines Laserscanners ist es möglich, ein Objekt mit dreidimensionaler Ausdehnung durch ein dicht angelegtes Raster aus gemessenen Punkten in Form einer Punktwolke aufzunehmen. Die Koordinierung der einzelnen Messpunkte im dreidimensionalen Raum erfolgt durch die Messung der Schrägdistanz sowie den dazugehörenden Horizontal- und Vertikalwinkeln (vgl. Abbildung 4).

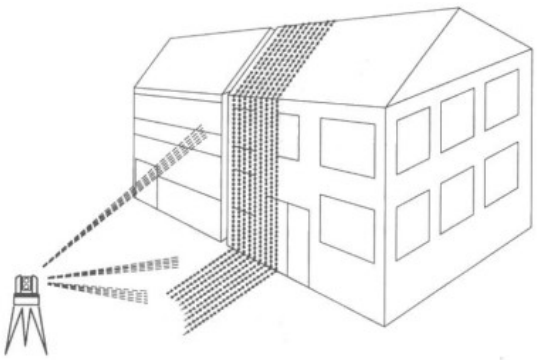

Abbildung 4 Darstellung eines klassischen Messaufbaus mit einem Laserscanner [12]

Die Distanz wird entweder aus der Laufzeitmessung oder einem Phasenvergleich bestimmt. Beim Phasenvergleich wird das empfangene Signal mit einem im Gerät verbliebenen Anteil des emittierten Signals verglichen. Die ausgesendete elektromagnetische Lichtwelle wird im Gerät in einen Referenzanteil und Sendeanteil aufgeteilt. Dadurch, dass beide Anteile aus demselben Signal stammen, lässt sich aus dem Unterschied der Phase zwischen Referenz und empfangenen Signalanteil der Unterschied beider Signale als Phasenverschiebung feststellen. Für handelsübliche Laserscanner ist heute durch die Verwendung eines etwa komplexeren Signals aus Träger- und Modulationswelle die Feststellung äußerst feiner Phasenunterschiede bis in den Millimeterbereich möglich [12], zudem benötigt das Phasenmessverfahren keine Zeitmessung mehr, wodurch die Genauigkeit der Messergebnisse auch nicht mehr von der Güte des Zeitmessinstruments abhängt.

Die Horizontal- und Vertikalwinkel ergeben sich aus dem Abgriff der Einstellung von im Scanner verbauten Ablenkspiegeln. Der Laserstrahl wird zum Aussendezeitpunkt über zwei bewegliche Spiegel abgelenkt, dadurch kann die Aufnahmeszene vom Standpunkt aus über ein weites Blickfeld hinweg betrachtet und aufgezeichnet werden. Die Verdrehung der Spiegel gibt dann Auskunft über

[2] Dieser Umstand erklärt zudem, weshalb blaues Laserlicht beim Airborne Laserscanning oder bei der Laser-Bathymetrie bisher keine Anwendung findet [55]

die gesuchten beiden Winkel der Horizontal- und Vertikalausrichtung des Messpunktes [12]. Moderne Scannersysteme schaffen durch diese beiden Messmethoden je nach Gerät eine Aufnahmefrequenz von bis zu 10.000 Punkten pro Sekunde, wobei jeder Punkt seine drei Lagekoordinaten und als vierten Wert die empfangene Intensität besitzt [12].

Der Vorteil eines statischen Laserscanners liegt darin, dass sich mit bekannter Position und Ausrichtung des Standpunktes direkt die Koordinaten je Punkt in der Wolke in einem übergeordneten System messen lassen. Der Übergang vom statischen zum mobilen Lasermesssystem erfordert allerdings eine zusätzliche Ausrüstung. Zum einen wird eine spezielle Art von Scanner verwendet: das ablenkende Element im Scanner rotiert hierbei um eine horizontale Achse und nimmt damit ein zweidimensionales Profil als vertikale Ebene auf. Das dreidimensionale Abbild des Messobjektes entsteht dann aus der Fortbewegung des Messfahrzeugs. Essenziell ist hierbei die zusätzliche Aufzeichnung der Position und Orientierung des Scanners über den gesamten Zeitraum der Messung durch zusätzliche Sensoren. Die Position kann durch den Einsatz von GNSS [14][3]-Antennen aufgezeichnet werden, die auf dem Fahrzeug und an festgelegten Orten im Messgebiet zu positionieren sind [15]. Mithilfe von bekannten Festpunkten, die während der Messung mit aufgezeichnet werden, lässt sich so im dGNSS[4]-Verfahren die Position genau bestimmen. Desweiteren ist die Ausrichtung des Scanners während der Messung mithilfe von INS-Systemen[5] notwendig [15]. Es ist dennoch zu beachten, dass das Ergebnis in Form einer Punktwolke Lücken aufweisen kann, wenn sich zum Zeitpunkt der Messung Hindernisse vor dem eigentlichen Messobjekt befinden. Dieser Verlust an Information ist besonders in mobilen Messkampagnen nicht zu vernachlässigen.

Die in dieser Arbeit verwendeten Punktwolken sind das Ergebnis von Laserscans aus Mobile Mapping Kampagnen der in Holzkirchen und Pittsburgh ansässigen Firma 3D Mapping Solutions [16] . Gemäß der Beschreibung des eigenen Messsystems verfügen ihre Messfahrzeuge über je einen Scanner, der zur Erfassung der Intensität neben der Punktkoordinierung bis zu 5000 Punkte pro Sekunde auf eine maximale Messdistanz von 100 Metern mit einer Lageabweichung von etwa einem halben Millimeter aufzeichnet [16]. Für das vorliegende Programm wird allerdings eine absolute Lagegenauigkeit von 2 bis 3 cm verwendet.

[3] GNSS: Global Navigation Satellite System: Satellitengestütztes System zur Positionierung und Navigation auf und über der Erdoberfläche. Beispiel: Global Positioning System (GPS) der USA

[4] dGNSS: differenzielles GNSS-Verfahren: Position aus relativer Position des Messfahrzeugs zu den Referenzantennen und Referenzpunkten im Messgebiet

[5] INS: Inertial Navigation System. Zeichnet die Ausrichtung des Messinstruments während der Messung auf

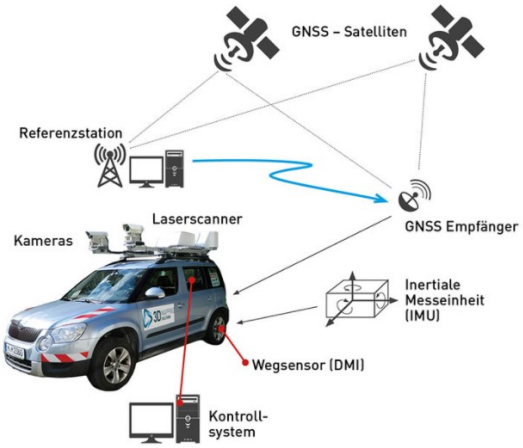

Abbildung 5 Das Zusammenspiel mehrerer Verfahren zur Gewährleistung der Genauigkeit für Messkampagnen am Beispiel der Firma 3D Mapping Solutions GmbH [16]

3.2 Punktwolke

Als Punktwolke bezeichnet man eine dichte Ansammlung von dreidimensionalen, individuellen Punkten [17]. Neben der räumlichen Information, die sich aus der Verteilung der Punkte ergibt, liefern moderne Messsysteme auch eine radiometrische Information in Form der Intensität je Punkt. Somit lassen sich auch Informationen zum gescannten Objekt aus der Interaktion von elektromagnetischer Strahlung mit dem Aufnahmeobjekt ermitteln, wie etwa Beschaffenheit und Material der Objektoberfläche [18] [13]. Punktwolken bilden als Menge diskreter räumlicher Messpunkte meist die Grundlage dreidimensionaler nichtdiskreter Modelle, wie beispielsweise Oberflächenmodellen in Form von Dreiecksvermaschungen nach Delaunay-Triangulation [17].

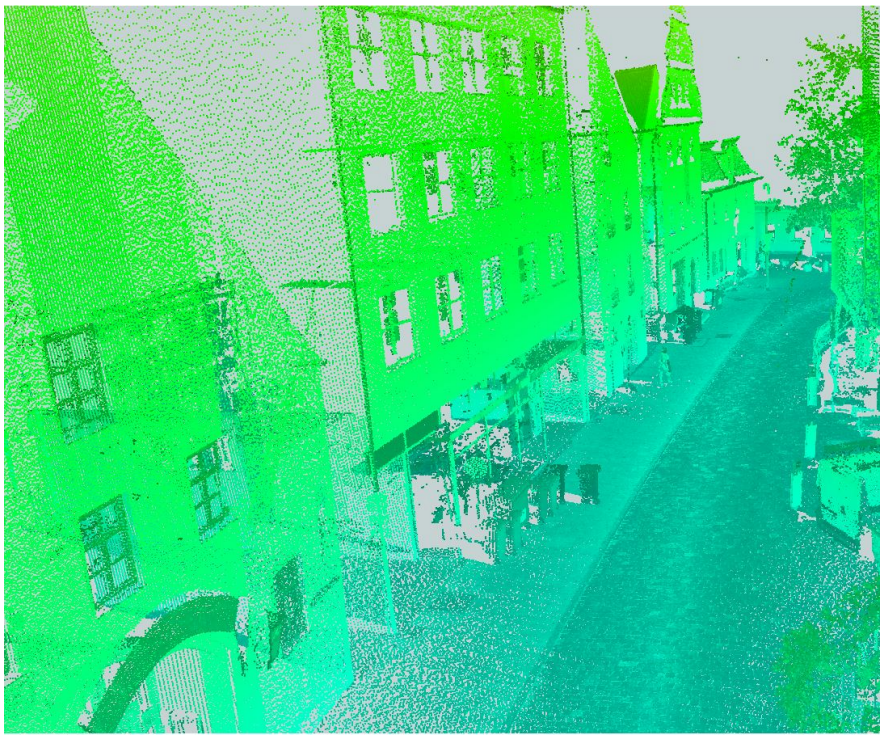

Abbildung 6 Beispiel einer Punktwolke aus einer Mobile Mapping Befahrung. Durch die hohe Punktdichte lassen sich nicht nur große Objekte wie Mülltonnen (Mitte), sondern auch Details wie die Pflastersteine der Straße (Rechts) erkennen. Zu erkennen sind dabei auch Lücken bei Verdeckung (Bereich hinter den Mülltonnen)

3.3 Semantisches Stadtmodell

3.3.1 Das Stadtmodell

Als Stadtmodell wird die möglichst maßstabsgetreue Abbildung der Verteilung der einzelnen Elemente einer Stadt bezeichnet. Es handelt sich also um ein räumliches Objekt und damit Abbild der Realität, Gebäude werden im Bezug zu den Nachbargebäuden maßstabsgetreu verortet. Im analogen Fall liefert ein Stadtmodell so dem Betrachter einen Blick aus der „Vogelperspektive“ und findet beispielsweise im Rahmen des Tourismus oder für historische Ausstellungen in grober Genauigkeit und Detaillierung Anwendung[6]. Für die Wissenschaft bietet der digitale Fall aber in Form eines semantischen Stadtmodells vor allem für Simulationen von Umwelteinflüssen oder für räumliche Analysen Anwendung.

Abbildung 7 Gebäudemodell in LoD3 mit zusätzlichem Straßenraummodell. Die Szene enthält den gleichen Straßenabschnitt wie in Abbildung 2 nur von einem anderen Standpunkt aus (vgl. hierzu: Abbildung 2 Links mit Abbildung 3 Bildmitte: beiges Gebäude mit Laternen und Fahnenmästen)

Stadtmodelle erfordern je nach Anwendungsfall unterschiedliche Modellierungen. Unterschieden werden zwei Modelltypen: Zur reinen Visualisierung werden VR-Modelle verwendet, die sich heutzutage automatisch aus Scannerdaten erzeugen lassen. Um aber auf Grundlage des Modells räumliche Analysen durchführen zu können, wird ein Urban Information Model – kurz UIM – gebraucht. Letzterer Fall bildet das für die Wissenschaft interessantere Modell, da hier neben der Geometrie auch thematische und funktionale Aspekte enthalten sind. Der Fokus der Modellierung liegt hier nicht bei der Visualisierung, sondern bei der Erzeugung auf Grundlage thematischer Informationen und Interpretationen [19]. Es werden fünf Detailstufen, die so genannten Level of Detail (LoD) unterschieden [19]:

- LoD 0: Regionalmodell
    - Das Modell umfasst ein digitales Geländemodell in 2.5D; Dieses Modell besteht aus einem zweidimensionalen Raster mit Gitterpunkten P (x, y), wobei jeder Gitterpunkt

[6] Feststellung des Autors

noch einen zusätzlichen Höhenwert besitzt. Der Höhenwert allein reicht aber nicht für ein dreidimensionales Modell aus und wird daher als 2.5-dimensional bezeichnet

- LoD 1: Stadt-/Bauwerksmodell
  - Das Modell ist dreidimensional, besteht aber noch aus Gebäuden in Form von „Klötzchen“ ohne Dachform
- LoD 2: Stadt-/Bauwerksmodell
  - Das einfache Klötzchenmodell wird durch ein detaillierteres Stadtmodell mit unterschiedlichen Dachstrukturen und Anbauten verfeinert und individualisiert
- LoD 3: Stadt-/Bauwerksmodell
  - In der vierten Detailstufe sind die Gebäude als detaillierte Architekturmodelle abgebildet. Gebäude umfassen jetzt individuelle Merkmale wie spezielle Gebäudeinstallationen
- LoD 4: Innenraummodell
  - Das bisher eher nur konzeptionell verfasste Gebäudemodell im Level of Detail 4 sieht eine detaillierte Modellierung des Gebäudeinneren vor. Dieser Detailgrad findet heute nur für bestimmte Gebäude Anwendung, etwa im Design von Architekturstudien. Ein vollständiges Stadtmodell im LoD 4 einer realen Stadt kann es aber realistisch betrachtet nicht geben, da dies einen starken Einschnitt in die Grundrechte des Menschen und den Verlust der Privatsphäre bedeuten würde.

3.3.2 Semantik, semantisches Stadtmodell

Der Begriff Semantik stammt aus dem Griechischen und bezeichnet die *Lehre von der Bedeutung der Wörter und ihrer Wandlungen* [20]. Im Sinne der Geoinformatik bezeichnet die Semantik die Bedeutung von thematischen, funktionalen und logischen Aspekten in dreidimensionalen Stadt- und Landschaftsmodellen [19]. Ein semantisches Stadtmodell ist demnach ein Modell, dass neben der räumlichen Verteilung der einzelnen Bestandteile des Stadtbildes auch zu den individuellen Elementen gehörende Metainformationen und Merkmale umfasst und so die Analyse von verschiedenen Metriken im Kontext des Stadtmodells für jedes kleinste Bestandteil des Modells ermöglicht. Einsatzbeispiele seien hier etwa Simulationen zur Analyse und Planung von möglichen Hochwasserschutzmaßnahmen für die Stadt Passau nach dem Jahrhunderthochwasser von 2013 [21] [22] oder als Grundlage zur Energiepotentialanalyse für die gezielte Installation von Solarpanelen (vgl. Abbildung 8) [22].

Abbildung 8 Links: Analyse des Sonnenpotentials von Hausdächern [22]

Das Modell ist hierarchisch gegliedert, es handelt sich also nicht um ein einziges, zusammenhängendes Objekt. Stattdessen setzt sich das Modell selbst aus einer Summe an Gebäuden zusammen, welche wiederum selbst aus mehreren Objekten zusammengesetzt sind. So setzt sich zum Beispiel ein Stadtmodell im LoD3 als eine Summe an Gebäuden zusammen, die selbst wiederum aus einer Summe an Wänden, Fenstern, Türen, Dach- und Bodenflächen, sowie individuellen Installationen wie Balkonen oder Reklametafeln, bestehen.

3.4 CityGML

Geodaten finden heute in vielen verschiedenen Bereichen Anwendung. Dabei ist es allerdings wichtig, dass die Daten – unabhängig vom Anwendungszweck – vom Computer eindeutig verstanden und gelesen werden können. Dies ist nur durch einen festen Standard möglich, der soft- und hardwareunabhängig eingesetzt werden kann. CityGML stellt dabei den Standard für räumliche Daten [23] dar, welcher durch das *Open Geospatial Consortium*[7] (kurz: OGC) – der auf Geodaten spezialisierte Mitgliedorganisation des *World Wide Web Consortiums* (W3C)[8] – entwickelt und definiert wurde. Dieses auf Geodaten spezialisierte Definitionsgerüst, das auf der *Extensible Markup Language* XML basiert, dem grundlegenden Gerüst zum Datenaustausch über das Internet, ermöglicht die effiziente und preisgünstige Beschreibung, Instandhaltung und Aktualisierung von Geodaten [23].

XML- und damit auch CityGML-Dateien folgen dabei einem hierarchischen Aufbau [24]:

Um einzelne Elemente zu beschreiben, wird der dafür verwendete Abschnitt im Dokument durch auszeichnende Tags definiert. Ein solcher Abschnitt beginnt mit einem Start-Tag und endet mit einem End-Tag. Innerhalb dieses abgekapselten Bereichs werden Attribute und Inhalte für das Objekt definiert. Inhalte sind dabei einfache Zeichenketten oder Attribute. Zuletztgenannte bestehen als Paar aus Attributname und einem dazu gehörenden Attributwert. Verbunden sind Attributname und Attributwert durch ein Gleichheitszeichen, der Attributwert wird mit Anführungszeichen umschlossen. Es ist aber auch möglich, Kommentare im Dokument anzugeben; dazu wird der Kommentarabschnitt mit „<! --“ geöffnet und mit „-->“ wieder geschlossen. Außerdem können im XML-Dokument direkt Verarbeitungsanweisungen für spezielle Verarbeitungssoftware formuliert werden. So können damit etwa spezielle Formatierungshinweise für die passende Visualisierungssoftware formuliert werden, die diese Software beim Laden des Dokuments direkt als Anweisung versteht und diese automatisch umsetzen kann – vorausgesetzt es wurden auch Anweisungen verwendet, die für diese Software gedacht sind.

Voraussetzung der breiten Einsetzbarkeit von auf Auszeichnungssprachen basierenden Dateiformaten ist die ausnahmslose Wohlgeformtheit von XML-artigen Dokumenten wie die im CityGML-Format: Es gelten folgende Grundsätze [24]:

1. Jeder Start-Tag braucht ein entsprechenden End-Tag. Innerhalb dieses festgelegten Paars dürfen nur Inhalte einer tieferliegenden hierarchischen Ebene liegen. Daraus folgt:

[7] Gegründet als Open GIS Consortium
[8] Standardisierungsgremium für das Internet

2. Voneinander verschiedene Elemente dergleichen hierarchischen Ebene dürfen sich nicht überlappen. Bevor also ein neues Element der gleichen Ebene begonnen werden darf, muss das Vorgänger-Element abgeschlossen sein.

3. Attributwerte müssen in Anführungszeichen geschrieben werden.

4. Elementbezeichner und Attributnamen müssen XML-Namen sein. Für XML-Namen gilt:

- XML-Namen müssen immer mit einem Buchstaben, Ideogramm oder Unterstrich beginnen. Alle Namen, die dabei mit „XML-“ – unabhängig von Groß oder Kleinschreibung – beginnen, sind aber bereits reserviert.
- Erlaubt sind alle Buchstaben des Alphabets in Groß- und Kleinschreibung, nicht-Englische Buchstaben wie Umlaute oder Ideogramme, die Ziffern 0 bis 9, sowie die Sonderzeichen Unterstrich (_), Minus (-) und Punkt (.).
- Andere Sonderzeichen wie der Doppelpunkt haben in XML eine feste Bedeutung und können in XML-Namen durch Ersatzzeichen definiert und eingebaut werden: so lässt sich etwa das kaufmännische Und-Zeichen „&“ durch das Ersatzzeichen „&“ verwenden.

5. Ein Element darf keine zwei Attribute mit demselben Namen besitzen. Zu beachten ist dabei, dass XML zwischen Groß- und Kleinschreibung unterscheidet

6. Kommentare und Prozessierungsanweisungen dürfen nicht in den Tags stehen, sondern müssen in einer tiefer liegenden Hierarchie-Ebene formuliert werden.

7. In Textinhalten dürfen keine Sonderzeichen mit feststehender Bedeutung für XML wie etwa das kaufmännische Und-Zeichen „&“ vorkommen; nur in Form der für sie festgelegten Ersatzzeichen können diese Sonderzeichen verwendet werden.

8. Das XML-Dokument muss exakt ein Element auf der obersten Hierarchie-Ebene haben. Jeglicher nachfolgender Inhalt ist Teil und eine Detaillierung dieses obersten Elements.

Diese Grundsätze gelten für XML-basierte Formate im Allgemeinen. Spezialisierte Ausführungen wie etwa CityGML werden dann durch eigene festgelegte Namensräume ausgezeichnet. Das ermöglicht es, Daten verschiedenster Formate miteinander kombinieren zu können, da sie zwar auf dem Grundgerüst des XML-Standards beruhen, sich selbst aber durch ihren eigenen Namensraum von den anderen unterscheiden. Auf die Definition des Namensraumes eines bestimmten Formats wird dabei immer zu Beginn eines XML-Dokuments in Form eines *Uniform Resource Identifier* (URI) verwiesen. Diese URI, etwa in Form einer weltweit eindeutigen URL[9], gibt eine für den Computer wichtige Auskunft über den verwendeten Namensraum und ermöglicht ihm das Interpretieren des Dokuments. Für CityGML gelten hier je nach Objekttyp eigene Definitionen, die online[10] nachlesbar sind.

Die CityGML-Version 2.0.0 stellt seit ihrer Veröffentlichung im März 2012 den aktuellen Standard der OGC für semantische Stadtmodelle in Modellierung, Speicherung und Austausch [25] [26]. Die Nachfolgerversion CityGML 3.0.0 befindet sich derzeit[11] noch in der Bewertungsphase durch das OGC

[9] Eindeutige Internetadresse
[10] Siehe: http://schemas.opengis.net/citygml/ und [26]
[11] Stand: Januar 2021

und wird nach Abschluss dieser Bewertung die Vorgängerversion um neue und überarbeitete Inhalte ergänzen beziehungsweise ersetzen [27].

3.5 FME

Die Feature Manipulation Engine – kurz FME – der kanadischen Firma *Safe Software* bietet eine recht intuitive Arbeitsumgebung für das Einbinden und Weiterverarbeiten von Geodaten verschiedenster Datenformate [28]. Safe Software bietet hierbei drei Ausführungen von FME, die jeweils an ihre Zielgruppen angepasst ein verschieden umfangreiches Repertoire an Möglichkeiten bieten: FME Desktop, FME Server und FME Cloud.

Während die beiden zuletzt genannten Ausführungen eher an Unternehmen mit dem Ziel automatisierter server-basierter Geodatenanalysen mit kommerzieller Anwendung gerichtet sind, bietet sich FME Desktop zur Entwicklung einfacherer, reproduzierbarer Analysestrukturen [29] [30]. Zudem bietet diese Version auch unterschiedliche Lizenzen, unter anderem zu Forschungs- und Bildungszwecken als Mehrfachlizenz für verschiedene Lehrinstitute [31], was sich für den Zweck dieser Bachelorarbeit am besten eignet.

FME Desktop bietet eine Vielzahl von Einsatzmöglichkeiten: Von der einfachen Übersetzung zwischen zwei von über 450[12] unterstützten Datenformaten [32] [33] bis hin zu umfangreichen Analysestrukturen lässt sich mit FME beinahe jede Problemstellung im Hinblick auch auf Geodaten lösen. Dazu bietet FME ein Spektrum an Transformern, also eine Bibliothek von fast 500 festintegrierten [34] sowie zahlreichen von der Community entwickelten [35] Funktionen.

Der Umfang von FME Desktop unterteilt sich hauptsächlich in zwei Anwendungen [36]: die FME Workbench und der FME Data Inspector. Die FME Workbench bietet eine Arbeitsumgebung zur Verarbeitung und Weiterentwicklung der Datensätze mithilfe der Transformer in einem dafür angelegten Workspace. Mit dem FME Data Inspector dagegen können die Daten betrachtet, aber nicht verändert werden. Zwar bietet diese Anwendung nützliche Tools zum Betrachten der Daten, wie etwa ein digitales Maßband für grobe Abmessungen im Kartenbild, das tabellarische Datengefüge kann damit aber nicht verändert werden.

Zum besseren Verständnis sind im Folgenden für FME wichtige und wiederkehrende Begriffe kurz vorgestellt und erläutert:

- FeatureType: Als FeatureType bezeichnet man den Oberbegriff einer speziellen Klasse von Hauptbestandteilen innerhalb eines Geodatensatzes. Alle Mitglieder (=Features) dieser Klasse haben eine Gemeinsamkeit, die sie von anderen unterscheidet, beispielsweise ihre Geometrie oder Rolle: So lassen sich die Bestandteile eines Gebäudes etwa in die Klassen Wandflächen, Dachflächen, Böden, Fenster, Türen oder sonstige Installationen unterteilen. Jede dieser Klassen stellt in FME ein eigenes FeatureType dar.
- Attributes: Mithilfe der FeatureTypes lässt sich also ein Geodatensatz in Klassen unterteilen. Jedes Feature, also jedes Einzelteil innerhalb solch einer Klasse, besitzt selbst aber Merkmale, die es individualisieren. Diese individuellen Merkmale werden in FME als Attribute bezeichnet. Unterschieden werden kann hier zwischen formatspezifischen und

[12] Stand Dezember 2020

benutzerdefinierten Attributen. Am Beispiel des Gebäudes kann eine Wand etwa Merkmale wie Farbe oder Material besitzen und unterscheidet sich damit zu anderen Wänden.

- Transformer: Möchte der Nutzer spezielle Merkmale eines Features analysieren, die aber zu Beginn nicht direkt im Datensatz gegeben sind, so muss dieses Attribut erst berechnet werden. FME stellt dafür ein breites Repertoire an Funktionen – den so genannten Transformern – bereit, mit denen die Ausgangsdaten auf verschiedenste Art und Weise verarbeitet und neue Attribute erzeugt werden können. Mithilfe von so genannten Bookmarks lassen sich Transformer im Workspace farblich einrahmen und gruppieren.

Sollte trotz des umfangreichen Funktionenangebots für eine speziellen Anwendungsfall keine bereits existierende Lösung in FME existieren, ist es möglich die gewünschten Funktionen durch eigene Implementierungen in den Programmiersprachen Python oder R zu ergänzen. Dazu verfügt FME über einen so genannten Interpreter, mit dem Anweisungen in den besagten Programmiersprachen für FME verständlich übersetzt werden. So ist es etwa möglich, direkt auf Feature-Ebene eigene Funktionen auszuführen, gleich einem Transformer.

3.6 Python

Bei Python handelt es sich um eine hauptsächlich imperative[13] Programmiersprache mit dem Ziel hoher Flexibilität in Anwendung und Nutzung, sowie eine zugleich einfach verständliche Syntax [37]. Entwickelt wurde diese Sprache vom niederländischen Softwareentwickler Guido van Rossum am CWI[14], dem nationalen Forschungsinstitut für Mathematik und Informatik der Niederlande [38]. Python eignet sich zum einen durch seine einfache und gut lesbare Syntax für Programmier-einsteiger, kann aber auch für äußerst komplexe Anwendungen plattformunabhängig und flexibel angewendet werden [37]. Vorteil der Flexibilität von Python ist zudem die mögliche Erweiterung der bereits umfangreichen Standardbibliothek durch mit der Python API entwickelte Funktionspakete von Drittanbietern. Das ermöglicht dem Nutzer selbst komplexe Operationen in einem kurzen und übersichtlichen Programm zu formulieren. Die hier für diese Arbeit verwendete Version 3.6.8 ist am 24. Dezember 2018 veröffentlicht worden und ist die aktuell letzte offiziell veröffentlichte Version von Python 3.6.

Im Zuge dieser Arbeit sind neben der Standardbibliothek[15] folgende Zusatzbibliotheken für Python verwendet worden:

- PyntCloud [39]: Pyntcloud ist eine Funktionenbibliothek von David de la Iglesia Castro[16], die speziell für den Umgang und die Verarbeitung von Punktwolkendaten konzipiert ist [40]. Unter anderem bietet Pyntcloud eine Implementierung einer RANSAC-Operation, mit der einzelne Wolkenpunkte danach, ob sie in einer gemeinsamen Ebene liegen, untersucht werden können (siehe auch Kapitel 4.2.2). Die identifizierten Punkte können dann aus der Gesamtwolke herausgefiltert und für weitere Operationen verwendet werden [41]. Die

[13] Imperative Programmierung: Vorgabe einer festen Abfolge von Befehlen, die der Computer Schritt für Schritt in der gegebenen Reihenfolge Zeile für Zeile abarbeitet.
[14] Centrum Wiskunde en Informatica
[15] Hier im speziellen: Die Inhalte ‚csv' (Arbeiten mit CSV-Dateien) und ‚math' (Basisbibliothek mathematischer Funktionen von Python)
[16] https://github.com/daavoo

Funktionen von Pyntcloud selbst greifen unter anderem auf Zusatzbibliotheken wie pandas[17] oder NumPy zu. Bei der Installation der Pyntcloud-Bibliothek ist aber darauf zu achten, kompatible Bibliotheksversionen zu verwenden, da die aktuelle Version[18] von Pyntcloud zum Teil veraltete Versionen dieser Zusatzbibliotheken mit herunterlädt, die in dieser Form nicht miteinander kompatibel sind[19].

- NumPy [42]: Die Bibliothek NumPy bietet ein breites Spektrum an Funktionen für numerisch mathematische Prozesse. So ermöglicht sie zum Beispiel vektorielle und matrixbasierte Operationen und ist Grundlage zahlreicher anderer Zusatzbibliotheken für Python.

3.7. MATLAB

MATLAB, Akronym für *Matrix Laboratory,* ist ein Programm mit eigener Programmiersprache der Firma MathWorks, das speziell zur direkten Implementierung von matrizenbasierten mathematischen Algorithmen entwickelt wurde [43]. Zusammen mit dem Zusatzprodukt Simulink bietet MATLAB eine umfangreiche Bibliothek an Funktionen zum Umgang mit matrix- und vektorbasierten Daten sowie zur Visualisierung in aussagekräftigen Diagrammen und Graphen und ist damit ein wichtiges Werkzeug für ein breites Spektrum wissenschaftlicher und ingenieurswissenschaftlicher Disziplinen. MATLAB kann zudem wie Python mit einer Vielzahl an Toolboxen für die verschiedensten Anwendungen erweitert werden, wie etwa zur digitalen Bildverarbeitung, für Deep Learning oder etwa zur Simulation chemisch-physikalischer Prozesse [43].

3.8 Mathematische Grundlagen

3.8.1 Stochastik: Konfidenzintervall

Werden Messwerte betrachtet, so ist auch zu beachten, dass jeder Messwert aus verschiedenen Gründen fehlerbehaftet sein kann. Schließt man Einflüsse systematischer Fehler, wie etwa ein falsch eingestelltes Messinstrument, bei einem gemessenen Datensatz aus, so gilt dennoch, dass jeder Wert einer statistischen Verteilung folgend vom wahren Wert abweicht. Für den ingenieurwissenschaftlichen Bereich folgen Messungen ohne systematische Einflüsse meist der so genannten Normalverteilung. Die in dieser Arbeit betrachteten Größen werden unter der Annahme, dass die Messungen frei von systematischen Fehlern sind, als normalverteilt angenommen.

Für normalverteilte Größen gibt es dann die Möglichkeit, einen Bereich der Verteilung zu finden, in dem mit einer Sicherheitswahrscheinlichkeit S der tatsächliche Wert angenommen werden kann. Anhand dieses Bereiches, der das Konfidenzintervall darstellt, ist es möglich, Datensätze gleicher Natur – also Messungen, die mit derselben Messmethode aufgenommen und frei von systematischen Einflüssen sind – mit diesem Intervall auf Abweichungen hin zu untersuchen.

Im Rahmen dieser Arbeit werden explorativ ermittelte Intervallgrenzen gesetzt. Der mathematisch geeignetere Ansatz zur Bildung des Konfidenzintervalls folgt den nachfolgenden Formeln [44]:

1. Wenn die Standardabweichung σ der Messwerte bekannt ist gelten folgende Grenzen des Intervalls [45]:

[17] Pandas: Bibliothek zur Performance-optimierten Analyse umfangreicher Dateien in tabellenbasierten Formaten wie CSV oder Excel
[18] Stand 14. Januar 2021
[19] Im Anhang sind die verwendeten Versionen jeder Zusatzbibliothek aufgelistet

$$\left[\bar{x} - z_{1-\frac{\alpha}{2}} * \frac{\sigma}{\sqrt{n}};\ \bar{x} + z_{1-\frac{\alpha}{2}} * \frac{\sigma}{\sqrt{n}}\right]$$

Mit

- Dem Mittelwert der n Messwerte $\bar{x}$
- Dem Irrtumsniveau $\alpha = 1 - S$
- Das $1 - \frac{\alpha}{2}$ Quantil der Standardnormalverteilung[20]
- Der bekannten Standardabweichung σ

2. Ist die Standardabweichung der Messung nicht bekannt, so gilt stattdessen [46]:

$$\left[\bar{x} - t_{1-\frac{\alpha}{2};n-1} * \frac{q}{\sqrt{n}};\ \bar{x} + t_{1-\frac{\alpha}{2};n-1} * \frac{q}{\sqrt{n}}\right]$$

- Geschätzte Standardabweichung $q = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$
- $t_{1-\frac{\alpha}{2};n-1}$: Das $1 - \frac{\alpha}{2}$-Quantil der t-Verteilung mit $n - 1$ Freiheitsgraden[21]

Zur Visualisierung statistischer Verteilungen bietet sich bei MATLAB die Verwendung eines Boxplots an. Die wichtigsten Bestandteile sind dabei wie folgt:

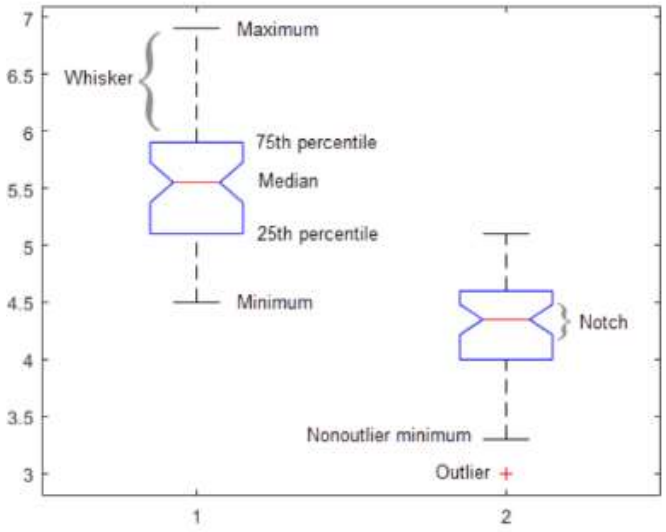

Abbildung 9 Schematischer Aufbau eines Boxplots [47]

Die Hauptkomponente des Boxplots stellt die namengebende Box dar, die den Prozentbereich von 25% bis 75% der Verteilung angeben. Durch eine rote Linie beziehungsweise Einkerbungen wird der Median angegeben. Dabei handelt es sich um den Wert, der sich an mittlerer Stelle befindet, wenn man die einzelnen Datenpunkte der Größe nach ordnet – also nicht zu verwechseln mit dem Mittelwert einer Verteilung. Die gesamte Bandbreite eines Datensatzes wird durch die so genannten Whiskers verdeutlicht. Diese reichen von der Box bis zum Maximum bzw. Minimum des Datensatzes. Messwerte, die sich nicht im Bereich zwischen 25% und 75% der Verteilung befinden, werden als Ausreißer gekennzeichnet [47]. MATLAB bietet hier die Möglichkeit, verschiedene Symbole zur

[20] Quantile online einsehbar unter z.B.: https://www.math.tugraz.at/mathc/wsp/normalverteilungstabelle.pdf
[21] Online einsehbar und ausrechenbar unter: http://eswf.uni-koeln.de/glossar/tvert.htm

Markierung dieser Ausreißer zu verwenden. Das ist besonders hilfreich zur Darstellung verrauschter, umfangreicher Datensätze.

3.8.2. RANSAC: Random Sample Consensus

Der Random Sample Consensus – kurz RANSAC – bietet eine Möglichkeit mit Datensätzen umzugehen, die aufgrund von starkem Messrauschen (Ausreißer) bei herkömmlichen Ausgleichungsansätzen zu einem stark verfälschten Ergebnis führen würden [48]. Der iterative Algorithmus, mit dem sich besagte Ausreißer identifizieren und ausschließen lassen, findet in einer Vielzahl an Disziplinen Anwendung, zum Beispiel etwa in der Photogrammmetrie zur automatischen Bildauswertung [18]. Dieses Verfahren dient oftmals zur Vorverarbeitung der Daten für eine anschließende Ausgleichung im klassischen Sinne, wie etwa im Gauß-Markov-Modell [48].

Der RANSAC funktioniert nach dem folgenden Schema [48] [18]:

Schritt 1: Je nach gesuchtem Modell – also der mathematischen Beschreibung des gesuchten Objekts im Datensatz – wird eine feste Anzahl an Messwerten aus dem Datensatz zufällig ausgewählt und die Modellparameter für diese Auswahl berechnet.

Schritt 2: Anschließend wird der Abstand aller anderen Messwerte zu diesem gerade ermittelten Modell berechnet und die Anzahl aller Messwerte gezählt, dessen Abstand geringer als die gewählte Toleranz des RANSAC-Verfahrens ist.

Schritt 3: Wird eine für das Verfahren festgelegte Mindestanzahl an Messwerte im Toleranzbereich gefunden, so wird der Modellvorschlag mit den dazu passenden Messwerten abgespeichert und das Verfahren mit einer neuen zufälligen Wahl an Messwerten wiederholt.

Über diese drei Schritte wird so lange iteriert, bis ein Abbruchkriterium, wie etwa eine Maximalanzahl an Wiederholungen, erreicht worden ist. Aus allen bis dahin ermittelten möglichen Modellversionen stellt die eine mit den meisten dazu passenden Messwerten die Lösung und damit einen von Ausreißern bereinigten Datensatz dar.

Für dieses Verfahren ist aber auch zu beachten, dass das Ergebnis nicht immer eindeutig und mitunter fehlerhaft sein kann. Mögliche Fehlereinflüsse können die Wahl einer zu groben oder zu feinen Toleranz sein, sowie eine zu geringe Anzahl an Wiederholungen. Außerdem kann sich ein Durchlauf A von einem anschließenden Durchlauf B aufgrund der zufälligen Wahl der Messwerte in Rechenschritt 1 im Ergebnis unterscheiden.

3.8.3. PCA: Principal Components Analysis

Die Principal Component Analysis (PCA), im Deutschen auch als Hauptkomponentenschätzung bekannt, wird hauptsächlich zur Reduktion von Dimensionen verwendet und zur Visualisierung von bestimmten Verteilungen und Ausdehnungen, wie etwa zur Betrachtung der Streuung in Datensätzen [49]. Das Verfahren ist ein weit verbreitetes Mittel, um komplexe Daten und Zusammenhänge mit einfachen Mitteln darstellen und verständlich machen zu können.

Der Ablauf der PCA lässt sich wie folgt beschreiben [50]: Gegeben sei ein Datensatz für n Messobjekte mit jeweils m verschiedenartigen Messungen. Der Datensatz allein ist aufgrund der m Messungen je Messobjekt nicht einfach darstellbar und mitunter unverständlich. Um eine Aussage über die Verteilung und Zusammenhänge zwischen den m Messwerten machen zu können, ist es

notwendig, die Daten auf eine für den Menschen sinnvollere Anzahl an Dimensionen zu reduzieren. Folgende Graphik [50] fasst die notwendigen Schritte übersichtlich zusammen:

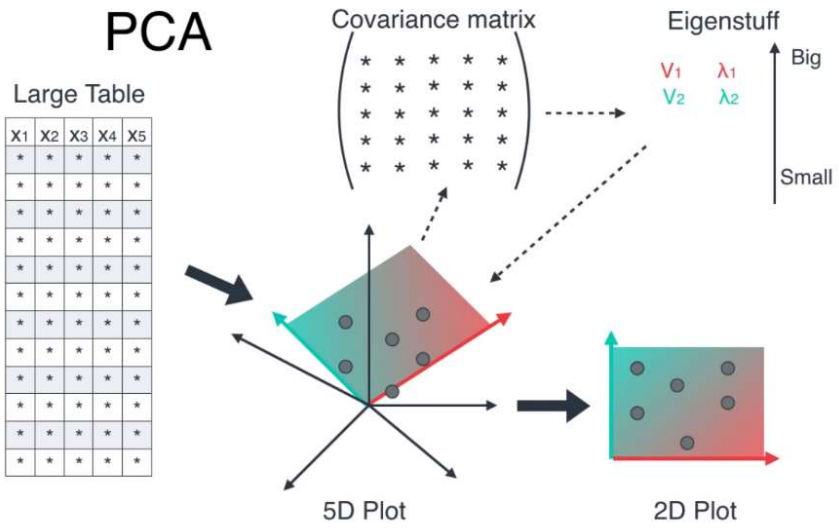

Abbildung 10 Ablaufschema der Principal Component Analysis [50]

Im ersten Schritt ist also für die Datenmatrix die zugehörige Kovarianzmatrix zu berechnen. Hierbei gilt: Die Kovarianzmatrix einer Datenmatrix mit m Spalten und n Zeilen ist selbst eine quadratische Matrix mit m Spalten und Zeilen. Für das Matrixelement σ_{ij} der Kovarianzmatrix K_{xx} gilt:

$$\sigma_{ij} = \begin{cases} F\ddot{u}r\ i = j \rightarrow Var(x) = \dfrac{\sum_{i=1}^{n}(x - \bar{x})^2}{n-1} \\ F\ddot{u}r\ i \neq j \rightarrow Cov(x,y) = \dfrac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n-1} \end{cases}$$

($\bar{x}, \bar{y}$ Mittelwerte)

Außerdem gilt, dass die Kovarianzmatrix symmetrisch und positiv definit ist, also gilt $\sigma_{ij} = \sigma_{ji}$.

Im folgenden zweiten Schritt werden die Eigenwerte und Eigenvektoren der Kovarianzmatrix berechnet. Für eine quadratische und symmetrische Matrix [m x m] gilt dann: es gibt m reelle Eigenwerte mit je einem Eigenvektor, wobei die Eigenvektoren zueinander orthogonal stehen. Die Eigenvektoren spannen damit einen eigenen Vektorraum auf. Die Eigenwerte können anschließend der Größe nach sortiert werden. Je größer der Eigenwert, desto ausgeprägter ist die Streuung der Daten in die Richtung des zugehörigen Eigenvektors. Die Verteilung kann nun durch die Wahl von p<m Eigenvektoren im von den Eigenvektoren aufgespannten Raum dargestellt werden, beispielsweise als zweidimensionaler Graph, dessen Koordinatenachsen die zwei Eigenvektoren mit den größten Eigenwerten bilden. Dargestellt wird dann die Projektion der Messwerte auf die Ebene zwischen den zwei Vektoren.

4. Methodik

4.1. Theoretische Konzeption und Umsetzung

Vor der eigentlichen Implementierung eines umfangreichen Programms ist es sinnvoll, sich – unabhängig von der eigentlichen Syntax der späteren Implementierung – Gedanken zum Aufbau und Ablauf der einzelnen Programmabschnitte zu machen. Diese theoretische Konzeption beginnt dabei erst recht grob in Form eines Grundgerüst aus Hauptabschnitten und wird nach und nach innerhalb der Hauptabschnitte weiter konkretisiert. Für das im Rahmen dieser Arbeit entwickelte Programm ist zu Beginn der Bearbeitungszeit folgendes Konzept entwickelt worden:

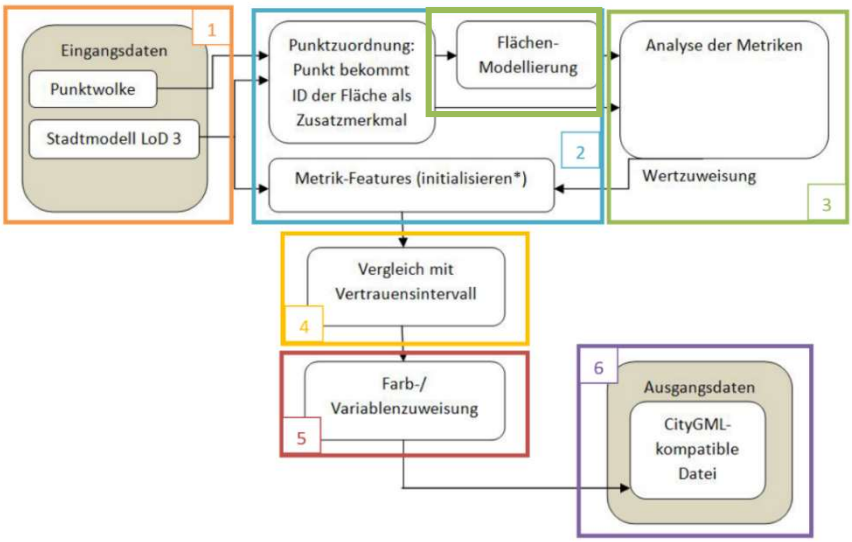

Abbildung 11 Theoretisches Konzept zur Programmstruktur

Das Programm besteht aus sechs Hauptabschnitten[22]. Als erstes sind die Daten einzulesen. Da die Daten in verschiedenen Formaten vorliegen, ist hier jeweils ein passendes Einleseverfahren zu entwerfen und anzuwenden. Anschließend sind die Daten im zweiten Hauptabschnitt einer Vorverarbeitung, dem so genannten Preprocessing, zu unterziehen. Ziel dieses Programms ist die Untersuchung geometrischer Beziehungen zwischen einer Punktwolke und einem aus Flächen und Körpern bestehendem Stadtmodell. Je Objektfläche soll aber nicht die gesamte Wolke betrachtet werden, sondern nur der Teil, der auch wirklich mit dieser Fläche zu tun hat. Teil der Vorverarbeitung ist dafür ein Abschnitt, der jedem Punkt in der Punktwolke die passende Fläche zuordnet, etwa mit der ID der Fläche. Durch das Ausschneiden und Identifizieren der relevanten Abschnitte der Eingangspunktwolke wird zudem die Datenmenge reduziert, was zu einem schnelleren Programm beiträgt. Bei den Objektflächen ist zudem jeweils ein neues Attribut je Metrik anzulegen und womöglich zu initialisieren, damit hier das für die einzelne Fläche zugehörige Ergebnis der jeweiligen Metrik nach der Berechnung eingetragen werden kann.

Nach der Zuordnung der Punktwolkenbereiche zu den Objektflächen werden im dritten Abschnitt die geometrischen Merkmale in Form der Metriken berechnet. Für jede Metrik wird hier das Mittel über

[22] Vergleiche farbliche Kennzeichnung mit Nummerierung in Abbildung 11

alle Punkte in der zugeordneten Punktwolke berechnet. Bei der Berechnung ist vor allem zu beachten, dass es Punkte vor und hinter der Objektfläche geben kann. Wenn sich also ein Punkt bezogen auf eine Wandfläche im Inneren eines Gebäudes befindet, so ist beispielsweise die berechnete Distanz zu der Fläche mit einem negativen Vorzeichen zu versehen. Der Mittelwert über alle der Fläche zugeordneten Punkte wird dann beim für jede Metrik angelegten Attribut abgespeichert. Für manche Metriken kann es notwendig sein, das Gesamtverhalten der Punkte gegenüber der Fläche zu untersuchen. Hierfür ist eine ausgleichende Ebene in die Punkte zu schätzen und das Programm mit einem entsprechenden Algorithmus zu erweitern.

Diese im dritten Abschnitt berechneten Mittel je Metrik werden im darauffolgenden vierten Abschnitt für jede Modellfläche auf ein festgelegtes Vertrauensintervall hin getestet. Wenn der Wert für eine Metrik außerhalb dieses Toleranzbereichs liegt, so wird die Fläche farblich gekennzeichnet. Dieses Einfärben der Objektfläche findet dann im fünften Abschnitt des Programms statt.

Der sechste und damit letzte Abschnitt dient zum Verfassen des Endprodukts: dem eingefärbten Stadtmodell im CityGML-Format. Diese Datei ist an einem festgelegten Ort abzuspeichern, damit der Nutzer sie sich mit einem Visualisierungsprogramm im Anschluss ansehen kann. Je nachdem ist es denkbar, hier mehrere Dateien auszugeben: jeweils eine für jede Metrik. Dann lässt sich nicht nur sehen, dass eine Fläche auffällig vom Toleranzbereich abweicht, sondern auch direkt, um welche Art von Abweichung es sich dabei handelt.

Im Laufe der Entwicklungszeit hat sich dieser theoretische Entwurf fortwährend weiterentwickelt, doch der grundlegende Aufbau ist erhalten geblieben. Die sieben Programmteile gliedern sich in vier Themenbereiche:

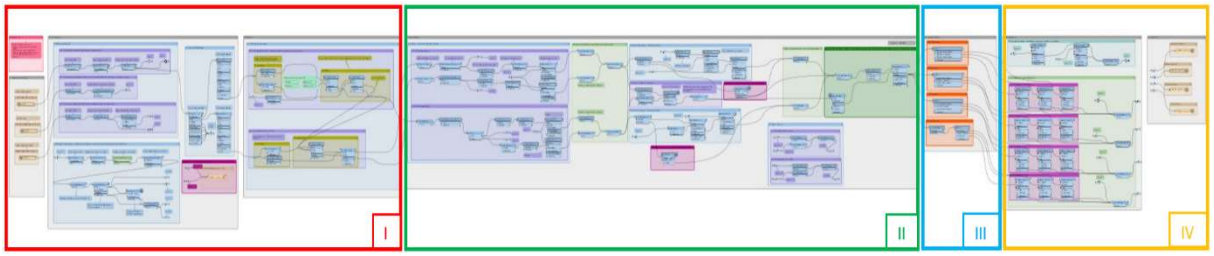

Abbildung 12 Grobe Übersicht über die FME Workbench zur Analyse von Gebäudeflächen (römische Nummerierung: die Themenbereiche)

Programmabschnitte 1 bis 3 befassen sich – zusammengefasst zum Themenbereich I – mit dem Einlesen und Vorverarbeiten der Daten. Dabei werden in Abschnitt 1 die Datensätze eingelesen und im zweiten Abschnitt allgemeinen Vorverarbeitungsschritten unterzogen. Im dritten Abschnitt wird die Analyse für jede betrachtete Objektfläche vorbereitet, indem die Punktwolke in mehrere kleine Punktwolken aufgeteilt wird und die Teilwolken den Modellflächen zugeordnet werden.

Im zweiten Themenbereich und damit Programmabschnitt 4 werden für die Objektflächen die passenden Metriken angewendet und als Kennwert den Flächen zugeordnet. Jede Metrik ergibt damit einen konkreten Wert für die individuelle Fläche.

Der im vierten Programmabschnitt errechnete Wert je Metrik wird im darauffolgenden fünften Programmabschnitt, also dritten Thementeil gegen festgelegte Grenzwerte getestet. Damit können die Flächen je Metrik in mehrere Klassen unterteilt und im vierten Hauptteil und damit sechsten Programmabschnitt für jede Klasse eingefärbt werden. Im siebten Programmabschnitt wird für jede

Metrik ein eingefärbtes Stadtmodell in Form einer CityGML-Datei geschrieben, das Ergebnis kann dann mithilfe des FME Inspectors betrachtet werden.

4.2. Aufbau im Detail

4.2.1. Themenbereich 1: Einlesen und Vorverarbeiten der Eingangsdaten

4.2.1.1. Einlesen

Um die Daten bearbeiten zu können, sind die Daten im richtigen Format einzulesen und in das Programm einzubinden. In FME wird hierfür ein so genannter Feature Type Reader verwendet. Damit dieser die Daten korrekt einlesen kann, sind zuvor noch das Format und der Speicherort der Daten anzugeben. Werden mehrere Dateiformate verwendet, so ist für jedes Format ein eigener Feature Type Reader anzulegen. Im Rahmen dieser Arbeit werden drei verschiedene Formate verwendet:

Das Stadtmodell in Form von einzelnen Gebäudemodellen befindet sich im CityGML-Format. Jedes Gebäudemodell setzt sich dabei aus bestimmten Komponenten zusammen: Den so genannten Feature Types. Bei der Konfiguration eines neuen Readers ist es hier möglich, sich entweder für jeden Feature Type einen individuellen Startpunkt angeben zu lassen oder einen für den Dateityp gemeinsamen Startpunkt, der alle Feature Types des Datensatzes vereint. Im letzteren Fall lässt sich der einzelne Feature Type nur nach einer Filterung ansprechen, allerdings wird die Programmstruktur dadurch übersichtlicher. Für den Workspace dieser Arbeit wird aus eben diesem Ziel der Übersichtlichkeit ein Feature Type Reader im „Merge“ Modus verwendet.

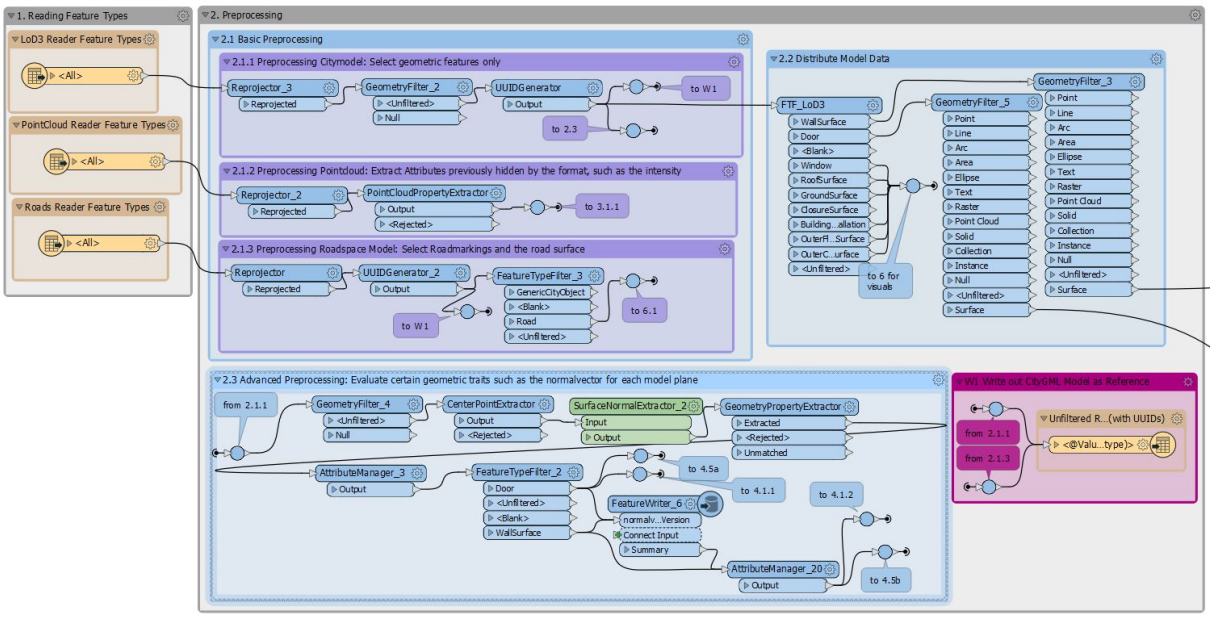

Abbildung 13 Übersicht Programmabschnitte Einlesen und Vorverarbeiten

Auch das Straßenmodell liegt im CityGML-Format vor und wird mit einem dafür konfigurierten Feature Type Reader eingelesen. Hier ist aber zu erwähnen, dass dieses Modell im Kontext der Analyse der Gebäudemodelle nur der Vollständigkeit halber zur Visualisierung im letzten Programmabschnitt verwendet wird. Zur Analyse der Straßenmarkierungen ist ein eigener Workspace konzipiert worden, da das Testgebiet für die Gebäude keine analysierbaren Straßenmarkierungen beinhaltet. Da dieser Workspace aber einem ähnlichen Aufbau folgt, werden nur die auszeichnenden Unterschiede zum jeweiligen Programmabschnitt beschrieben.

Die Punktwolke wird im LAS-Format eingelesen, dieses liefert im Gegensatz zum Gebäude- oder Straßenmodell nur ein Punktwolken-Feature Type. Daher besteht hier kein sichtbarer Unterschied zwischen den beiden möglichen Darstellungsoptionen.

4.2.1.2. Preprocessing Teil 1: Grundlegende Vorverarbeitung

Die Daten sind nun also eingelesen und stehen zur Verarbeitung zur Verfügung. Abbildung 14 zeigt einen Ausschnitt der Gebäude- und Punktdaten, nachdem diese erfolgreich eingelesen worden sind. Doch bevor mit den Daten sinnvoll gearbeitet werden kann, sind grundlegende Schritte zur Vorverarbeitung notwendig[23]. Der erste notwendige Schritt für alle drei Datentypen ist das Schaffen eines gemeinsamen Bezugsrahmens. Verwendet wird hier das System EPSG[24]: 32632, wobei diese Codierung für die Zone 32N des UTM-Koordinatensystems steht [51]. Der Blick auf die Daten direkt nach dem Einlesen zeigt zwar bereits, dass sie sich bereits in einem gemeinsamen Bezugsrahmen zu befinden scheinen. Durch diesen Schritt wird jedoch sichergestellt, dass sich die Datensätze tatsächlich im gleichen System befinden. Ohne gemeinsamen Bezug wäre eine räumliche Analyse schließlich nicht möglich.

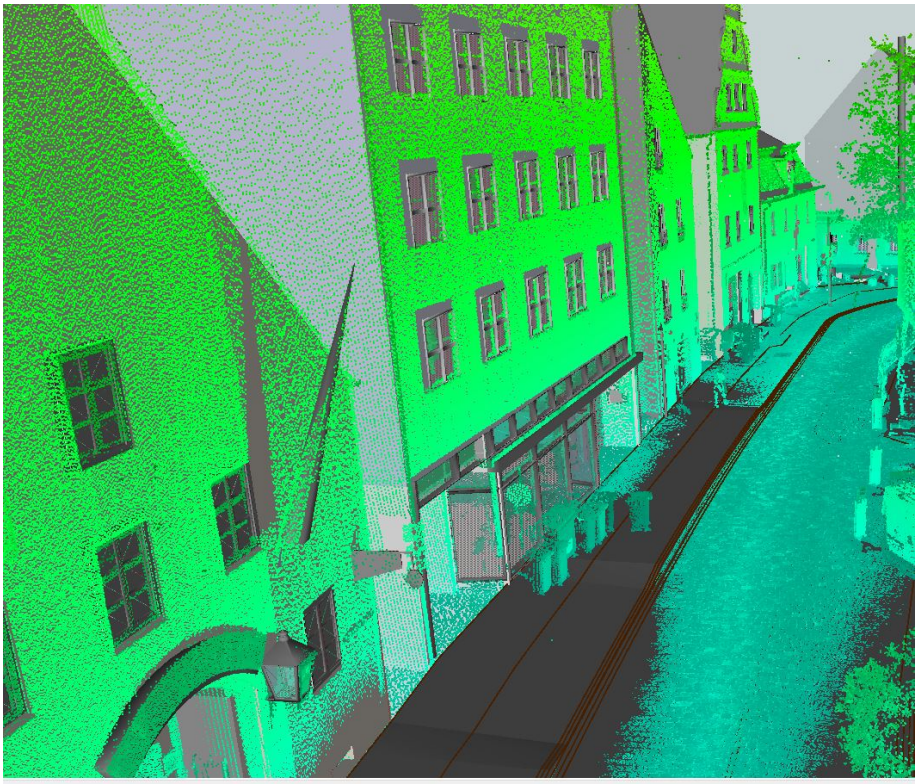

Abbildung 14 Stadt- und Straßenmodell zusammen mit der Punktwolke nach dem Einlesen

Da im Rahmen dieser Arbeit geometrische Abweichungen untersucht werden, werden anschließend geometrischen Features selektiert und mithilfe des so genannten UUIDGenerator-Transformers für jede Geometrie im Gebäudemodell eine einzigartige Kennung erstellt. Dieser Schritt ist notwendig, da sich die von FME automatisch generierten Kennungen der einzelnen Elemente im Laufe des Programms zu verändern scheinen. Die per Transformer erstellten IDs umgehen dies und

[23] Programmabschnitt 2.1
[24] EPSG: European Petroleum Survey Group Geodesy

ermöglichen in späteren Schritten die Zuordnung zwischen den Objektflächen und den für die einzelne Fläche berechneten Analyseergebnissen. Dieser Schritt erfolgt auch für das Straßenmodell und erfüllt speziell im Workspace zur Untersuchung dieses Straßenmodells denselben Zweck.

Für das Stadtmodell – in Form der einzelnen Gebäudemodelle – ist damit der grundlegende Vorverarbeitungsteil abgeschlossen. Hier verzweigt sich nun die Netzarchitektur in drei Pfade (Abbildung 13 Bookmark 2.1.1 Ende). Bevor für die Gebäude der zweite große Vorverarbeitungsteil beginnt, werden die bisher in einen Strom enthaltenen einzelnen FeatureTypes auf dem ersten Pfad mittels eines FeatureTypeFilters aufgefächert. Die in diesem Programm implementierten Metriken werden auf alle Türen und Hauswände des Stadtmodells angewendet. Bei dieser Auftrennung werden also diese beiden FeatureTypes „Door“ und „WallSurface“ von den anderen getrennt. Die Restlichen werden bereits zum Abschnitt 6 für die Visualisierung weitergeleitet, das gleiche geschieht mit dem dafür herausgefilterten FeatureType „Road“ des Straßenmodells im Anschluss auf dessen UUID-Generierungsschritt. Der für die Wände und Türen folgende Programmabschnitt setzt einen festen Geometrietyp „Surface“ voraus. Um das zu gewährleisten wird also kurz vor der Weiterleitung noch per GeometryFilter-Transformer sichergestellt, dass nur solche Geometrien weitergeleitet werden.

Der zweite Pfad des Gebäudemodells ist mit einem FeatureTypeReader in Programmabschnitt W1 verbunden, mit dem auch ein zweiter Pfad des für das Straßenmodell zuständigen Vorverarbeitungsbereichs verbunden ist. Hier wird ein vollständiges, zusammenhängendes Modell aus dem Straßen- und Stadtmodell herausgeschrieben. Dies ist insofern sinnvoll, um nach dem Durchlauf einen zusammenhängenden Referenzdatensatz mit den neuen Flächenkennungen vorliegen zu haben, falls auf die Analyse – außerhalb des in dieser Arbeit entwickelten Programms – weitere Verarbeitungen dieses Datensatzes folgen sollen.

Der dritte Pfad nach der grundlegenden Vorverarbeitung der Gebäudemodelle geht zum Programmabschnitt 2.3 *Advanced Preprocessing*. Hier werden zusätzliche geometrische Informationen für die Gebäudeflächen ermittelt, wie beispielsweise der Mittelpunkt oder Normalenvektor einer Objektfläche, da diese Werte in nachfolgenden Programmabschnitten gebraucht werden. Die Komponenten der Normalenvektoren werden zudem mittels eines FeatureWriter-Transformers in eine eigene CSV-Datei geschrieben, die für den mit Python formulierten Programmabschnitt 4.2 – beschrieben in Kapitel 4.2.2.2 – gebraucht wird.

Die Datensätze sind somit grundlegend vorverarbeitet und bereit für weitergehende Schritte. Der einzige Vorverarbeitungsschritt für die Punktwolkendaten – neben dem Setzen des Bezugssystems – ist die Extraktion von Zusatzinformationen, die bis dahin in FME nicht angezeigt wurden. Das liegt vor allem daran, dass Punktwolken hier nicht als eine Anzahl von vielen, einzelnen Punkten betrachtet werden, sondern als ein Punktwolkenobjekt. Damit ermöglicht FME einen effizienten Umgang mit diesem oft sehr umfangreichen Datentyp. Besonders aber zur Analyse von etwa der Intensität von einzelnen Punkten sind die im Hintergrund liegenden Informationen relevant. Daher sind diese hier zu extrahieren. Für die Gebäudemodelle sind hier nur die Koordinaten relevant, bei der Analyse des Straßenmodells – speziell den dort untersuchten Straßenmarkierungen – wird aber zusätzlich die Intensität benötigt und darum mit extrahiert.

4.2.1.3. Preprocessing Teil 2: Buffering und Clipping

Nach der grundlegenden Vorverarbeitung der Eingangsdaten beginnt die Vorbereitung der Metrikenberechnung in Abschnitt 4 (Kapitel 4.2.2). Bisher liegen die Gebäudemodelle und die Punktwolke zwar im gleichen Bezugssystem, allerdings fehlt der Bezug, welche Ausschnitte der Gesamtpunktwolke zu den einzelnen Gebäudeflächen gehören. Daher ist die Eingangspunktwolke jetzt in kleinere Teilwolken aufzuteilen. Das wird erreicht, indem Abschnitte der Punktwolke gesucht werden, die sich in einem festgelegten Bereich genau vor oder hinter einer Objektfläche befinden.

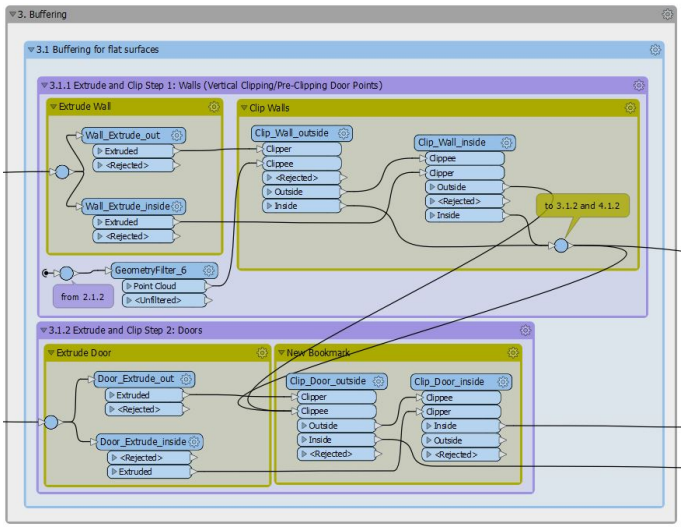

Abbildung 15 Programmabschnitt Buffering und Clipping

Benötigt werden Pufferkörper mit einer gerichteten Ausdehnung. Klassische puffererzeugende Transformer von FME, wie beispielsweise der Bufferer, sind in diesem Fall nicht geeignet, da der hier erzeugte Körper das gepufferte Objekt vollständig umgibt, anstatt sich nur in eine bestimmte Richtung auszudehnen. Für die Türen und Wände werden daher gerichtete Pufferkörper mithilfe des Extruder-Transformers erzeugt. Mit diesem Transformer kann aus einer flächenhaften Geometrie ein Körper mit räumlicher Ausdehnung vom Geometrietyp „solid“ erzeugt werden, indem die Fläche kopiert und um einen festen Wert in eine festgelegte Richtung verschoben wird. Der dazwischen liegende Raum wird dann zu einem abgeschlossenen Körper verbunden. Für die Türen und Wände wird die Kopie dabei in Richtung des Normalenvektors verschoben, der Körper erstreckt sich damit also genau vor beziehungsweise hinter die Gebäudefläche, je nach Vorzeichen. Da Bereiche einer Teilpunktwolke sowohl vor als auch hinter der Modellfläche liegen können, werden zwei Pufferzonen je Modellfläche erzeugt: eine für den Bereich einen Meter vor der Gebäudefläche, also außerhalb des Gebäudes, und eine für Punkte innerhalb des Gebäudes. Der für das Gebäudeinnere gedachte Puffer ragt um einen Meter in das Gebäude hinein, die Ausdehnungsweite beträgt hier als -1 Meter.

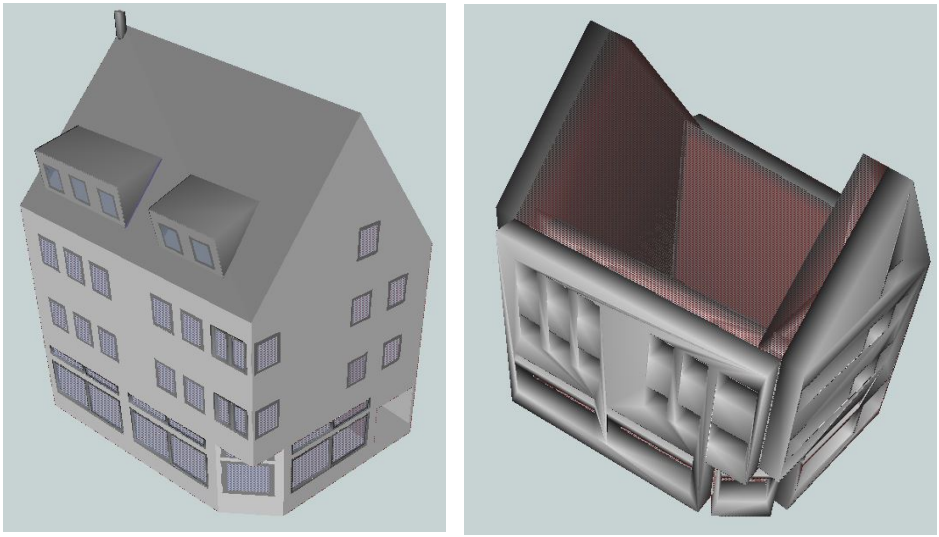

Abbildung 16 Gebäude vor und nach Pufferung der Wände und Türen nach außen und innen. Sonstige Gebäudeflächen sind vor der Pufferbildung durch Filterung entfernt worden

Im Anschluss an die Bildung der Pufferkörper (vgl. Abbildung 16) können die Teilwolken der Gesamtwolke ausgeschnitten werden, dies erfolgt mithilfe des Clipper-Transformers. Dieser benötigt zwei Eingangsdaten: einen *Clipper* und einen *Clippee*. Der Clipper ist dabei das Objekt, für das untersucht werden soll, welche Teile des Clippee-Objekts innerhalb und welche außerhalb davon liegen. Im vorliegenden Fall bilden die gepufferten Körper die Clipper und die Punktwolke den Clippee. Wird für eine Objektfläche dadurch herausgefunden, welcher Teilbereich der Punktwolke innerhalb des Pufferkörpers liegt, so werden diesem Teilbereich die auszeichnenden Attribute wie etwa die Kennung (UUID) als neue Information zugewiesen. Erfolgt das für alle betrachteten Objektflächen reduziert sich zudem die Datenmenge für die folgenden Prozesse, da nur noch Teile der großen Eingangspunktwolke gebraucht werden und der Rest vernachlässigt werden kann.

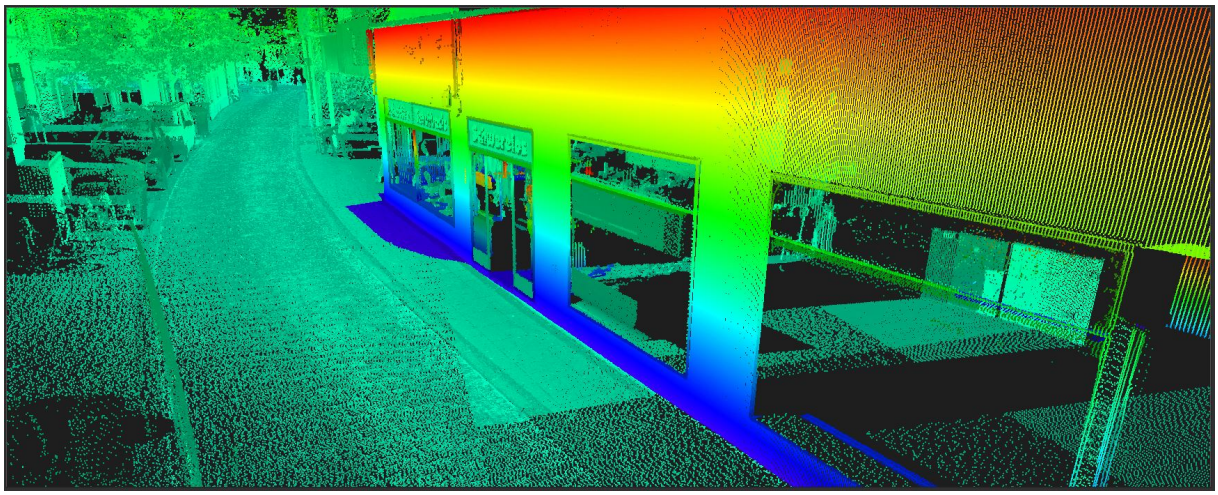

Abbildung 17 Ergebnis der Clipping-Operation. Hier auffällig gekennzeichnet: Wolkenbereich innerhalb eines Pufferkörpers

Dieser Abschnitt des Programms beansprucht den größten Teil der Laufzeit des Programms. Um die Laufzeit etwas zu reduzieren wird eine bestimmte Reihenfolge angewendet, in der die einzelnen Pufferkörper mit der Punktwolke geschnitten werden. Zuerst werden die Körper der Wandflächen mit der Wolke geschnitten. Die dabei übrigbleibende Restwolke wird dann an die Türen weitergeleitet und mit den Pufferkörpern dieser geschnitten. Damit sind für jede Wandfläche und für

jede Tür jeweils zwei Teilwolken gefunden und zugeordnet worden. Für das Straßenraummodell erfolgt dieser Schritt analog dazu im dafür konzipierten Workspace, wobei hier zuerst die relevanten Markierungsarten selektiert werden, bevor daraus die zwei Pufferkörper je Markierung[25] erzeugt werden.

4.2.2. Themenbereich 2: Formulierung und Berechnung der Metriken

4.2.2.1. Vorstellung der erdachten Konzepte

Die Vorverarbeitung liefert nun mehrere kleine Punktwolken, denen bestimmte Flächen der Gebäude im Stadtmodell zugeordnet sind. Ausgehend von den Punkten in diesen kleineren Punktwolken können nun mehrere geometrische Aspekte untersucht und in Form von Metriken formuliert werden.

- Konzept 1: Punkt-Flächen-Distanz (PTS[26]-Distanz):
  Bei den vorliegenden Geometrien in Form von Punkten in einer Punktwolke und Gebäudemodellen, die sich aus einzelnen Flächen zusammensetzen, erscheint die Betrachtung des Abstands beider Datensätze am naheliegendsten. Der Abstand wird realisiert durch die Betrachtung der Abstände jedes einzelnen, der Fläche zugeordneten Punktes der Teilpunktwolke. Für den Abstand eines Punktes zu einer Fläche ist aber wichtig zu konkretisieren, dass hier der kürzest mögliche Abstand gemeint ist, also die Distanz zwischen Punkt und Fläche in Richtung des Normalenvektors. Zur Berechnung der Distanz $d_{P\perp F}$ eines Punktes P zu einer Fläche F werden neben den Koordinaten des Punktes P noch die Koordinaten des Flächenmittelpunktes N sowie die Komponenten des Normalenvektors $\vec{n}$ der Fläche F benötigt. Sind diese gegeben, so lässt sich die Berechnung des gesuchten Abstandes über den Satz des Pythagoras für ebene Dreiecke und durch Ergänzung trigonometrischer Beziehungen formulieren [52]: Gesucht ist eigentlich der Abstand des Punktes P zu seiner Projektion P^* auf der Fläche F in Richtung des Normalenvektors. Die Punkte P, N, P^* spannen dann ein rechtwinkliges Dreieck mit den Seiten $a = \overline{NP^*}, b = \overline{PP^*}, c = \overline{NP}$ auf, für das eingesetzt nach dem Satz des Pythagoras gilt:

  $$b = \sqrt{c^2 - a^2}$$

  Da der Punkt P^* allerdings nicht bekannt ist, kann die Strecke $a = \overline{NP^*}$ über den Normalenvektor $\vec{n}$ berechnet werden, da dieser mit dem Verbindungsvektor $\vec{c} = \overrightarrow{NP}$ den Winkel α einschließt, mit dem sich a über den Pythagoras mit dem Betrag von $\vec{n}$ und $\vec{c}$ errechnen lässt:

  $$b = \sqrt{|c|^2 - [\sin(\alpha) * |c|]^2}$$

  Mit $\alpha = \cos^{-1}\left(\frac{\langle \vec{n}, \vec{c} \rangle}{|\vec{n}| * |\vec{c}|}\right), \langle \vec{n}, \vec{c} \rangle = x_n * x_c + y_n * y_c + z_n * z_c$

  Als zusammenhängende Formel ergibt sich also die Punkt-Flächen-Distanz zu:

$$d_{P\perp F} = \sqrt{(|\vec{c}|)^2 - \left[|\vec{c}| * \sin\left(\cos^{-1}\left(\frac{[x_N * (x_N - x_P)] + [y_N * (y_N - y_P)] + [z_N * (z_N - z_P)]}{|\vec{n}| * |\vec{c}|}\right)\right)\right]}$$

  Mit

[25] Da Straßenmarkierungen als horizontale Flächen anzunehmen sind: Oberhalb bzw. unterhalb der Modellfläche mit einer Weite von $\pm$ 1 Meter

[26] PTS: Point-to-Surface

$$P = \begin{pmatrix} x_P \\ y_P \\ z_P \end{pmatrix}$$
$$N = \begin{pmatrix} x_N \\ y_N \\ z_N \end{pmatrix}$$
$$|\vec{c}| = \sqrt{(x_N - x_P)^2 + (y_N - y_P)^2 + (z_N - z_P)^2}$$
$$|\vec{n}| = \sqrt{n_x^2 + n_y^2 + n_z^2} = 1$$

In FME lässt sich die gesuchte Distanz recht einfach mithilfe des NeighborFinder-Transformers ermitteln. Dieser sucht für jeden einzelnen Punkt die am nächsten liegende Objektfläche und gibt unter anderem auch diesen kürzesten Abstand als Ergebnis an. Zur Anwendung dieses Transformers ist es aber notwendig, die Punktwolken aufzulösen und die Wolkenpunkte als individuelle dreidimensionale Punkte aufzufassen. Hierfür wird vor der Auflösung der Punktwolke durch den PointCloudCoercer-Transformer noch die Punktwolke auf eine maximale Anzahl an Punkten ausgedünnt. Hier ist die Maximalzahl auf 300 gleichmäßig verteilte Punkte gesetzt. Es ist zu beachten, dass dieser Schritt zur Reduktion der Laufzeit durchgeführt wird, aber möglicherweise Einfluss auf das Ergebnis der Distanzanalyse haben könnte. Im vorliegenden Fall wird dieser Einfluss aber wissentlich zur Laufzeitoptimierung vernachlässigt, da der dafür angewendete PointCloudThinner-Transformer für eine gleichmäßige Verteilung der übrigbleibenden Punkte sorgt, wodurch der Einfluss als sehr gering eingeschätzt werden kann.

- Konzept 2: Flächen-Ebenen-Distanz (STP[27]-Distanz):
  Ein ähnlicher Ansatz lässt sich wählen, wenn statt einzelnen Punkten eine durch die Punkte gelegte Ebene betrachtet wird und der Abstand dieser Ebene zur Modellfläche berechnet wird. Hierfür ist zum einen die in die Punkte passende Ebene zu schätzen und daraufhin die Ebenengleichung in Hesse 'scher Normalenform zu berechnen. Damit kann der Abstand des Objektflächenmittelpunktes zu der geschätzten Ebene bestimmt werden. Da die Umsetzung dieser Metrik allerdings ein vergleichbares Ergebnis wie die PTS-Distanz bei deutlich komplexerer Implementierung liefert, ist dieses Konzept nicht implementiert worden.

- Konzept 3: Flächen-Ebenen-Ausrichtung:
  Eine wesentlich eindeutigere Aussage liefert dagegen die Betrachtung der Ausrichtung der Objektfläche gegenüber einer in die Punkte geschätzten Ebene. Die Ausrichtung unterteilt sich hier in die zwei Winkel Orientierung und Inklination. Beide Winkel ergeben sich als Winkel zwischen dem Normalenvektor der Objektfläche und der bestangepassten Ebene durch die Punkte. Die Orientierung beschreibt dabei den Winkel zwischen den beiden Normalenvektoren als Projektion auf eine Horizontalebene parallel zur x-y-Ebene des Koordinatensystems. Die Inklination dagegen beschreibt den Winkel zwischen den beiden besagten Vektoren gegenüber der z-Achse. Beide Winkel lassen sich aus der Projektion der Normalenvektoren auf geeignete Ebenen betrachten:

[27] STP: Surface-to-Plane

Mit dem Normalenvektor der Objektfläche $\overrightarrow{n_{Ob}} = \begin{pmatrix} n_{Ob_x} \\ n_{ob_y} \\ n_{ob_z} \end{pmatrix}$ und dem Normalenvektor der bestangepassten Ebene $\overrightarrow{n_{Eb}} = \begin{pmatrix} n_{Eb_x} \\ n_{Eb_y} \\ n_{Eb_z} \end{pmatrix}$ ergeben sich die Orientierung γ und die Inklination ζ:

$$\gamma = \tan^{-1}\left(\frac{n_{Eb_x}}{n_{Eb_y}}\right) - \tan^{-1}\left(\frac{n_{Ob_x}}{n_{Ob_y}}\right), \zeta = \tan^{-1}\left(\frac{n_{Eb_z}}{\sqrt{n_{Eb_x}^2 + n_{Eb_y}^2}}\right) - \tan^{-1}\left(\frac{n_{Ob_z}}{\sqrt{n_{Ob_x}^2 + n_{Ob_y}^2}}\right)$$

Die Umsetzung in FME ist hier aber nicht trivial, da der Umfang des Transformer-Angebots keine Methode zum Schätzen einer bestangepassten Ebene in eine Punktwolke beinhaltet. Hier ist nun auf eine eigens entwickelte Implementierung dieser Methodik in Python zurückzugreifen. In FME ist eine solche zusätzliche Implementierung mithilfe des PythonCaller-Transformers möglich, der auf den in FME internen oder einen vom Nutzer ausgewählten externen Interpreter für die Sprache Python zugreift.

In der Implementierung werden zwei Arten von Daten benötigt, für die zu Beginn die Dateipfade anzugeben sind. Gebraucht werden dabei zum einen der Pfad zum Verzeichnis, in dem in Programmschritt W1 die Normalenvektoren als CSV-Datei hinterlegt worden sind. Zum anderen wird der Pfad zum Verzeichnis gebraucht, in dem die ausgedünnte Punktwolke für jede Objektfläche nach Programmabschnitt W3 als XYZ-Datei abgespeichert wurden. Die folgenden Befehlsketten werden iterativ für jede im Verzeichnis abgelegte Punktwolkendatei durchgeführt.

Zuerst wird die aktuellen Punktwolkendatei geöffnet und ihr Name als lokale Variable abgespeichert. Die Punktwolkendateien tragen nämlich die eindeutige Kennung der passenden Objektfläche im Namen, sodass darüber im Anschluss an den Python-basierten Abschnitt die Analyseergebnisse den Objektflächen in FME zugeordnet werden können.

Nach Abgriff der Kennung wird die Punktwolkendatei mithilfe von Funktionen der Zusatzbibliothek Pyntcloud gelesen und anschließend mit einer RANSAC-basierten Methode von Ausreißern befreit. Die Funktion *add_scalar_field(„plane_fit“,max_dist=1e-3)* erweitert die tabellarische Struktur der Punktwolkendatei dabei um eine Spalte “is_plane“, die das Ergebnis der RANSAC-Analyse wiedergibt. Alle Punkte, die gemäß Algorithmus bei einem maximalen Normalabstand von einem Millimeter in einer gemeinsamen Ebene liegen, bekommen an dieser Stelle den Wert „1“ zugewiesen, alle außerhalb liegenden Punkte den Wert „0“. Das ermöglicht die Filterung zwischen den passenden Punkten und den Ausreißern. Anschließend wird jeder Punkt der Wolke als eigener Array formuliert und eine nx3 – Matrix für alle n passenden Punkte aufgestellt. Dies erfolgt mit Funktionen der NumPy-Bibliothek. Dieser Schritt ist notwendig, um darauffolgend mithilfe einer Hauptkomponentenschätzung den Normalenvektor der bestangepassten Ebene zu erhalten. Hier wird auf eine Implementierung der besagten PCA zurückgegriffen [53]. Dieses Verfahren ist äußerst effizient, hat aber den Nachteil, dass es keinerlei Information über die Genauigkeit der geschätzten Ebene wiedergibt, wie sie eine Ausgleichung im klassischen Sinne liefern würde. Für vertikale Ebenen ist eine klassische Ausgleichung aber nur über ein nicht-lineares und äußerst komplexes Funktionalmodell möglich; zugunsten der schnelleren Laufzeit wird daher auf die Hauptkomponentenschätzung zurückgegriffen.

Die PCA-Analyse liefert als Ergebnis drei Eigenvektoren; die zwei mit den größten Eigenwerten davon spannen die am besten zu den Punkten passende Projektionsebene auf. Diese ist zugleich die gesuchte bestangepasste Ebene, zu der der dritte Eigenvektor durch Normierung den Normalenvektor darstellt. Mit diesem Normalenvektor der Ausgleichsebene und dem zu der Objektfläche dazugehörenden Normalenvektor könne jetzt Orientierung und Inklination nach der oben vorgestellten Formulierung berechnet werden. Dabei ist zu beachten, dass manchmal nicht der eigentliche Normalenvektor der Ebene gefunden wird, sondern ein orthogonal dazu stehender Vektor. Nach der Berechnung der beiden Winkel sind diese auf den Bereich zwischen $\pm 45°$ zu normieren.
Die beiden Winkel werden anschließend – zusammen mit der Kennung der zugehörigen Objektfläche – als neue Zeile in ein CSV-Dokument geschrieben. Nach Durchführung der soeben beschriebenen Schritte liefert dieses eine Dokument dann die Ergebnisse, die in FME den Objektflächen zugeordnet werden können.

- Konzept 4: Punktdichte reine Wand versus Wand mit Fenstern
  Die Idee dieses Konzepts liegt darin, aus der einer Objektfläche zugeordneten Menge an Punkten die Punktdichte je Flächeneinheit zu ermitteln und diesen Wert gegen einen Referenzwert zu testen. Der Referenzwert stammt von einer Fläche mit homogener, repräsentativer Punktdichte, sodass es möglich ist, aus dem Vergleich herauszufinden, wie viel Prozent der Objektfläche zum Beispiel von Fenstern bedeckt sind. Trifft der Strahl des Laserscanners auf Glas, wird dieser je nach Winkel entweder durchgelassen oder voll reflektiert, sodass im Allgemeinfall kein reflektierter Anteil dieser Flächen beim Empfänger eingeht und somit Fenster Lücken in der Punktwolke darstellen. Wird nun die tatsächliche Punktdichte der Objektfläche abzüglich der in dieser Objektfläche enthaltenen Gesamtfläche an Fenstern gegenüber der Referenzpunktdichte gestellt, so lässt sich herausfinden, ob beispielsweise die Gesamtfläche der Fenster zu groß oder zu klein ist, also ein möglicher Hinweis auf fehlende bzw. ungenau modellierte Fenster. Dieser Ansatz ist aber allein schon aufgrund von Verdeckung von Gebäudeflächen, etwa durch parkende Fahrzeuge oder Vegetation, nicht sinnvoll umsetzbar. Zudem wird die Punktdichte durch die Art und Weise, wie mobile Messsysteme mit Laserscannern die Punktwolken aufnehmen abhängig vom Blickwinkel variieren. Ohne weiteres ist dieses Konzept also nicht umsetzbar.

- Konzept 5: Punktdichte als Kennwert guter Messabdeckung
  Dennoch lässt sich mithilfe der Punktdichte eine Aussage zur Modellierung treffen. Die Punktdichte gibt eine Aussage darüber, wie gut eine bestimmte Objektfläche beim Scan von Wolkenpunkten abgedeckt wurde. Hierfür wird in Programmabschnitt 4.1.1 bzw. 4.1.2 vor dem Ausdünnen der Punktwolken und Aufteilen in individuelle Punkte für jede Objektfläche die Anzahl an Punkten festgehalten und an Programmabschnitt 4.3.1 weitergeleitet. Dort werden die Punkanzahlen, die zuvor noch separat für den Bereich vor bzw. hinter der Objektfläche festgehalten wurden, zu einer Gesamtanzahl je Fläche zusammengefasst. Mit den in den Programmabschnitten 4.5a und 4.5b errechneten Flächeninhalten der Objektflächen kann das Verhältnis Der Anzahl an Punkten gegenüber der Fläche in Quadratmetern zur Punktdichte $\frac{Punkte}{m^2}$ gebildet werden. Dieser Wert für jede Objektfläche wird später mit der mittleren Punktdichte des gesamten Datensatzes verglichen. Es wird also für jede einzelne Wand und Tür die Punktdichte berechnet und anschließend das Mittel

davon über alle Türen und Wände gebildet. Das Verhältnis der individuellen Punktdichte gegenüber der mittleren Gesamtpunktdichte ist Gegenstand dieser Metrik.

- Konzept 6: Flächenkrümmung:
  Interessant, aber ebenso nicht ohne weiteres umsetzbar ist die Betrachtung möglicher Krümmungen aus den Punktdaten gegenüber den modellierten Gebäudeflächen. Der Nutzen dieses Konzepts ist aber für die Modellierung einer Simulationsumgebung sehr wahrscheinlich zu gering, verglichen mit den eigentlichen Anwendungsfällen in Deformationsanalysen und im Hinblick auf die Gesamtkosten der Umsetzung dieses Konzepts, vor allem bezogen auf die Rechenzeit des Programms.

- Konzept 7: Flächenversatz
  Mithilfe dieses Konzepts wäre es möglich, herauszufinden, ob die Objektfläche in der richtigen Höhe, an der richtigen Position und in der korrekten Ausdehnung im Modell vorkommt. Hierzu ist es aber notwendig, die Eckpunkte der Modellfläche eindeutig Punkten der Punktwolke zuzuordnen. Auch dies ist mit ohne weiteres rein aus den Punkten erzielbar.

- Konzept 8: Intensitätsanalyse
  Die bisher beschriebenen Metriken sind vor allem auf die betrachteten Gebäudeelemente, also Wände und Türen bezogen. Für das Straßenmodell, dort speziell alle Fahrbahnmarkierungen, liefert die Intensität der empfangenen Signalantwort aber eine viel relevantere Aussage. Es ist anzunehmen, dass sich eine Fahrbahnmarkierung gegenüber dem Asphalt des Fahrbahnbelags durch eine deutlich höhere Intensität auszeichnet. Grund hierfür ist der höhere Rückstrahlanteil der weißen Fahrbahnmarkierung gegenüber dem dunklen Asphalt. Nachdem also für die Markierungen die einzelnen Teilwolken ausgeschnitten worden sind, wird zu jeder Markierung die mittlere Intensität festgestellt. Dieser eine Wert je Markierung wird anschließend gegen einen festgelegten Schwellwert getestet.

- Konzept 9: Position und Ausrichtung von Straßenobjekten
  Ein interessantes Konzept wäre die Bestimmung der Positionierung und Ausrichtung von Straßenelementen wie Ampeln, Schildern oder Laternen aus den Punktdaten. Hierfür wäre es aber notwendig, die Punktwolke passend einem Objekt zuzuweisen. Die Position lässt sich dabei nur für Objekte mit einer Hauptausdehnung, also einfache Laternen oder Straßenschilder, bestimmen. Komplexere Objekte wie Ampeln für mehrspurige Kreuzungen mit mehreren Ampelkästen, die über die Fahrbahn hinausragen, können damit jedoch nicht aussagekräftig analysiert werden.

Die auf der nachfolgenden Seite abgebildete Tabelle (Abbildung 18) listet die erarbeiteten und soeben beschriebenen Metriken übersichtlich auf und gibt auch Aufschluss darüber, welche davon im Rahmen dieser Arbeit umgesetzt wurden.

Nr.	Metrik Bezeichnung	Implementiert
1	PTS-Distanz	Ja
2	STP-Distanz	Nein
3	Flächen-Ebenen-Ausrichtung	Ja
4	Punkdichte reine Wand vs. Wand mit Fenstern	Nein
5	Punktdichte als Kennwert guter Messabdeckung	Ja
6	Flächenkrümmung	Nein
7	Flächenversatz	Nein
8	Intensitätsanalyse	Ja
9	Position und Ausrichtung von Straßenobjekten	Nein

Abbildung 18 Übersicht entwickelte und davon umgesetzte Metriken

4.2.2.2. Umsetzung

Implementiert wurden die Konzepte *PTS-Distanz*, *Flächen-Ebenen-Ausrichtung*, *Punktdichte als Kennwert guter Messabdeckung* sowie die *Intensitätsanalyse* für die Straßenmarkierungen. Die nicht implementierten Metriken ließen sich entweder mit den vorliegenden Daten nicht umsetzen oder erwiesen sich als zu komplex und zeitaufwendig, auch hinsichtlich der Laufzeit des Programms. Mit den bereits implementierten Metriken lässt sich aber schon eine gute Aussage über auftretende Abweichungen treffen. Das Vorgehen zur Implementierung der Gebäude-bezogenen Metriken wird in Kapitel 4.2.2.1 bei der jeweiligen Metrik beschrieben

Hauptsächlicher Unterschied zwischen dem Workspace für das Straßenmodell gegenüber dem Workspace für das Stadtmodell besteht in der hier verwendeten Metrik. Dafür wird nach dem Clipping je Markierung die Punktwolke in individuelle Punkte aufgelöst und für die Intensität der StatisticsCalculator-Transformer angewendet. Dieser liefert für jede Markierung ein Minimum, Maximum, Mittelwert, Median, sowie statistische Kenngrößen wie die dazu gehörende Standardabweichung. Die in Kapitel 4.2.2.1 beschriebene Implementierung in Python befindet sich mit Kommentaren im Anhang.

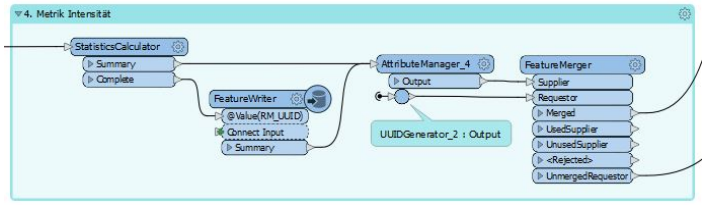

Abbildung 19 Aufbau der Metrik zur Intensitätwertsanalyse der Straßenmarkierungen

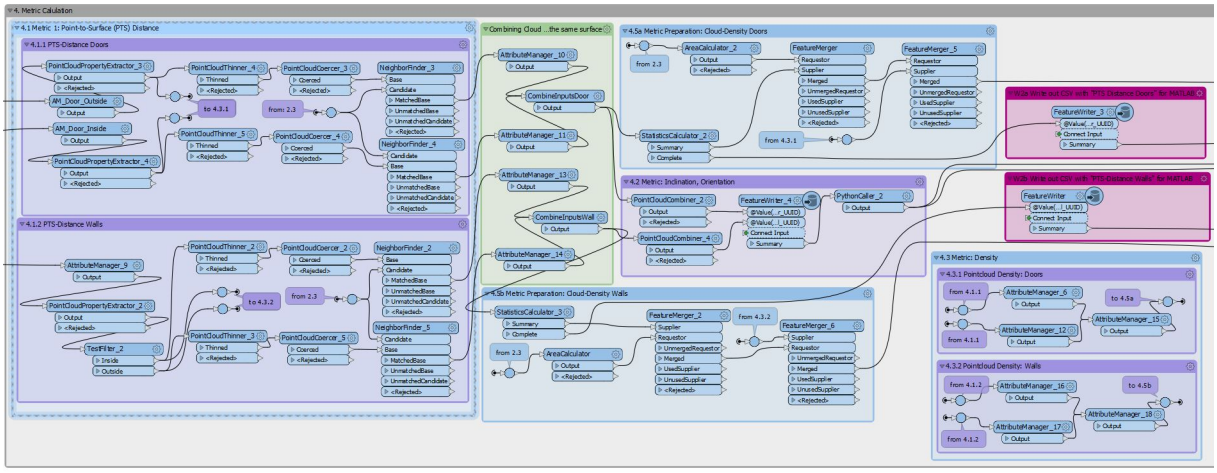

Abbildung 20 Programmabschnitt Metrikenberechnung

Die meisten Berechnungen finden im Metrik-Teil des Programms mithilfe von StatisticCalculator-Transformern und AttributManager-Transformern statt. Beide zählen gemäß Safe Software zu den am meisten verwendeten Transformern in FME [34] und liefern vielseitige Einsatzmöglichkeiten. So wird der AttributManager in diesem Programm häufig zur Vereinigung mehrerer Programmpfade in einen gemeinsamen verwendet. Zwar gibt es für solche Operationen eigene Transformer, FME ist aber so konzipiert, dass es für eine Problemstellung unzählige Lösungsansätze gibt [34].

4.2.3. Themenbereich 3: Statistisches Filtern

Die Umsetzung des statistischen Filterns erfolgt in FME mithilfe des TestFilter-Transformers. Getestet werden hier die Ergebnisse der Metriken gegen festgelegte Schwellwerte. Die Schwellwerte sind explorativ oder mithilfe eines MATLAB-Skriptes festgelegt worden. Explorativ wurde dabei ein oder mehrere Schwellwerte festgelegt und mithilfe des MATLAB-Skriptes verifiziert. Das Skript liefert hierfür hilfreiche Grafiken in Form von Plots und Berechnungen von Grenzen eines Konfidenzintervalls. Bei den Schwellwerten wurde vor allem darauf geachtet, dass sie sich betragsmäßig innerhalb der Grenzen der errechneten Intervallgrenzen für eine Sicherheit von 95% befinden. Das Testen findet vollständig im Programmabschnitt 5 statt.

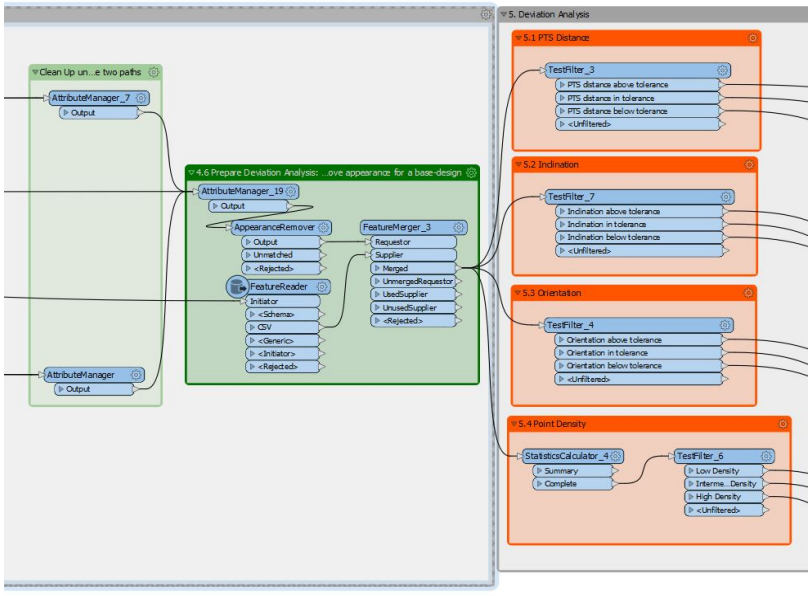

Abbildung 21 Programmabschnitt Pfadunion und statistisches Filtern

Für die einzelnen Metriken ergeben sich folgende festgelegte[28] Schwellwerte:

- Für die PTS-Distanz -die im Folgenden als cm-genau betrachten wird – wäre der ideale Wert eine Distanz von exakt 0 cm. Bei einer Punktgenauigkeit von etwa 2 bis 3cm ergeben sich damit Intervallgrenzen von rund ± 5cm. Der Test gibt also drei mögliche Ausgänge an: die betragsmäßige PTS-Distanz liegt bei einem Wert zwischen null und fünf Centimetern innerhalb der Toleranz. Alle außerhalb liegenden Distanzen weisen auf eine mögliche Verschiebung der Modellfläche hin. Das Vorzeichen gibt dabei an, in welche Richtung diese Verschiebung geht: bei einem positiven Vorzeichen ist die Distanz zu groß, die Modellfläche ist also zu weit ins eigentliche Gebäude Innere versetzt. Bei einem negativen Vorzeichen gilt das entsprechende Gegenteil.
- Für die Inklination und Orientierung gelten ebenso jeweils drei mögliche Ergebnisklassen. Hier ist der Schwellwert bei beiden Tests auf $\pm 1°$ gesetzt, dieser Wert ist dabei rein explorativ gewählt. Die Berechnung der Winkel ergibt sich aus einer Differenz zwischen dem Winkel der Soll-Richtung anhand des geschätzten Ebenennormalenvektors und dem Winkel der Ist-Richtung anhand des Objektflächen-Normalenvektors. Liegt das Ergebnis außerhalb der Vertrauensgrenzen, so ist die Objektfläche in geodätischer[29] Zählrichtung bei einem negativen Vorzeichen im Uhrzeigersinn, bei einem positiven Vorzeichen gegen den Uhrzeigersinn entlang der z-Achse verdreht. Die Inklination ergibt sich aus der gleichen Differenz, nur diesmal wird der Winkel von der horizontalen Ebene aus gezählt. Ein positives Vorzeichen weist auf eine Verkippung in der Richtung von der Vertikalen zur Horizontalen hin, bei einem negativen Vorzeichen gilt der entsprechende Gegensatz
- Für die Punktdichte wird das Verhältnis der Punktdichte je Objektfläche zu der mittleren Punktdichte über alle betrachteten Objektflächen gebildet. Der Bereich mittlerer Dichte liegt dabei zwischen 0.8 und 1.2, also den Relationen von 80% und 120% von einzelner Dichte bezüglich der mittleren Gesamtdichte. Auch dieser Test liefert drei Klassen in Form der niedrigen, mittleren und hohen relativen Punktdichte
- Für die Analyse der Straßenmarkierungen wird der Mittelwert der Intensität je individuelle Markierung gegen einen Schwellwert getestet, der sich aus der Betrachtung der Verteilung der erfassten Intensitätswerte für den Testbereich ergibt. Für eine sichere, statistische Aussage wäre noch eine Information über die Präzision der Intensitätsdetektion des Scanners notwendig. Es ergeben sich die drei möglichen Testergebnisse durch die Klasse niedriger mittlerer Intensität, der mittleren Intensität im mittleren Spektrum und der Klasse mit hoher Intensität.

4.2.4 Themenbereich 4: Visualisierung

Das Testen gegen Schwellwerte in Programmabschnitt 5 ist – wie in Kapitel 4.2.3 beschrieben – so konzipiert, dass sich pro Metrik drei mögliche Wertklassen ergeben. Diese sind nun farblich zu kennzeichnen. Dafür werden alle vorherigen Erscheinungsparameter der Gebäudemodelle und des Straßenmodells durch den AppearanceRemover-Transformer entfernt und anschließend mittels des AppearanceSetter-Transformers neu eingefärbt. Nicht untersuchte Flächen erhalten hierbei einen neutralen Grauton, vom Straßenmodell werden nur die Flächen der Fahrbahn mit abgebildet, hierfür

[28] Ausnahme: Die mittlere Punktdichte des gesamten Datensatzes, zu dem je Fläche das Dichteverhältnis gebildet wird, errechnet sich abhängig vom verwendeten Datensatz und ist damit nicht als fester Wert vorgeschrieben.

[29] Lagesystem mit vertikaler X-Achse (Hochwert) und horizontaler Y-Achse (Rechtswert)

wird ein deutlich dunklerer Grauton zum Kontrast gewählt. Die Metriken werden je nach Testergebnis unterschiedlich eingefärbt, hauptsächlich werden hierfür die Grundtöne Rot, Grün und Blau verwendet; zur Visualisierung der Punktdichte gegenüber der mittleren Gesamtdichte wird statt des blauen Farbtons ein gelber Farbton für den Bereich nahe der mittleren Dichte verwendet.

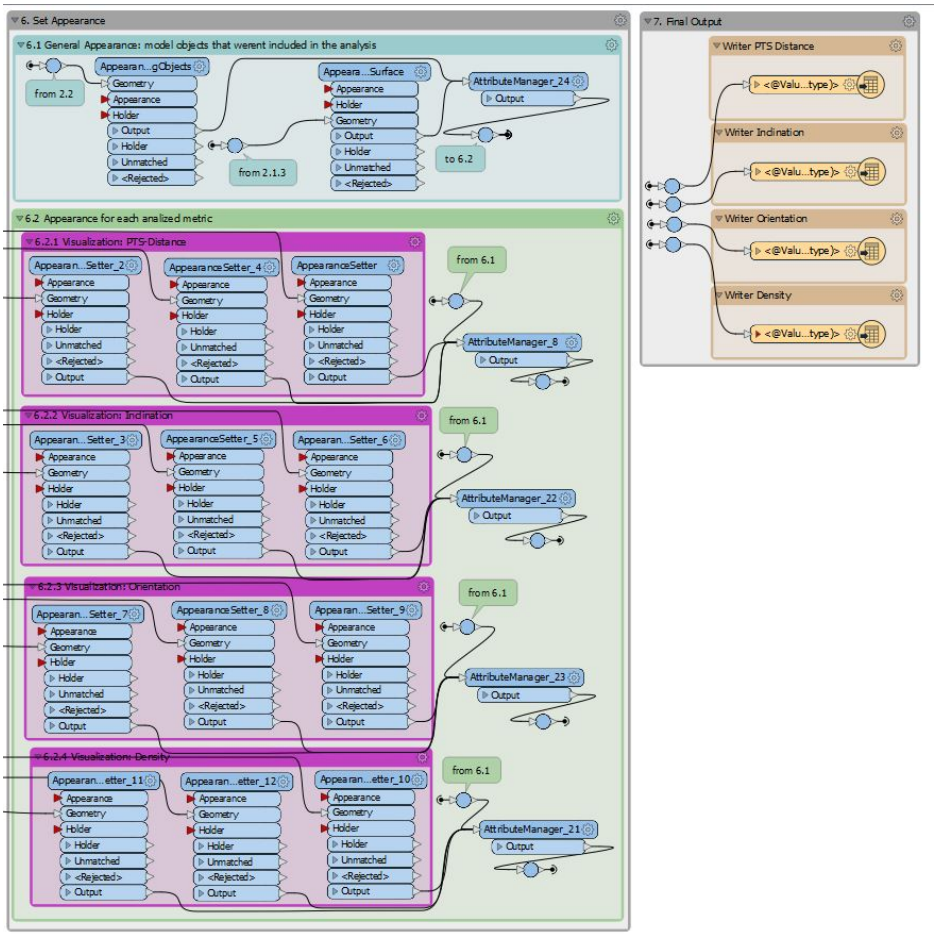

Abbildung 22 Programmabschnitt Farbvergabe und Ausgabe der Ergebnisdateien

Die Farbkombinationen ergeben hier folgende Aussage:

- Metrik: PTS-Distanz:
  - Blau: Die mittlere Punkt-Flächen-Distanz hat ein negatives Vorzeichen und ist betragsmäßig außerhalb der Toleranz. Die Modellfläche liegt also gemäß der Berechnung zu weit im eigentlichen Gebäude Inneren versetzt.
  - Grün: Die mittlere Punkt-Flächen-Distanz liegt im Toleranzbereich
  - Rot: Die errechnete Distanz ist zu hoch. Die Modellfläche liegt also gemäß der Berechnung zu weit im eigentlichen Gebäude Inneren versetzt.
- Metriken: Flächen-Ebenen-Orientierung und -Inklination
  - Blau: Der Normalenvektor der Objektfläche ist mathematisch betrachtet für die Orientierung gegen den Uhrzeigersinn vom Normalenvektor der geschätzten Ebene weg verdreht. Bei der Inklination weist ein negatives Vorzeichen auf eine Verkippung nach unten, bezogen auf den Normalenvektor der geschätzten Ebene, hin.
  - Grün: Orientierung und Inklination liegen innerhalb der Toleranz
  - Rot: Die Verdrehung stellt den jeweiligen Gegensatz zum negativen Vorzeichen dar

- Metrik: Punktdichte
    - Rot: Die mittlere Dichte der Fläche liegt unter dem Durchschnitt. Die Fläche ist also mit geringer Punktdichte vertreten
    - Gelb: Die mittlere Dichte der Fläche ist weniger als 20% vom Durchschnitt entfernt, damit ist die Fläche mittelmäßig gut vertreten
    - Grün: Die mittlere Dichte der Fläche liegt über dem Durchschnitt. Damit ist die Fläche überdurchschnittlich gut erfasst worden.
- Metrik: Intensität bei Fahrbahnmarkierungen
    - Rot: Intensität ist außerhalb der Toleranz und zu niedrig. Das weist möglicherweise auf eine verschobene Position der Modellfläche hin
    - Gelb: Intensität ist im mittleren Toleranzbereich. Es kann weder eine korrekte noch falsche Positionierung festgestellt werden
    - Grün: Intensität ist höher als die Toleranz: Die Position der Fahrbahnmarkierung kann als bestätigt bezeichnet werden.

5. Vorstellung und Diskussion der Analyseergebnisse des Testbereichs

Jetzt kann die Analyse für den Testbereich durchgeführt werden. Dieser umfasst für das Stadtmodell 13 ausgewählte Gebäudemodelle und einer Punktwolke und ein Straßenmodell mit zugehöriger Punktwolkenabdeckung in einem gesonderten Bereich. Die Gebäude sind dabei so gewählt, dass sie in einer gemeinsamen Punktwolke liegen, um die Datenmenge möglichst gering zu halten. Zudem sind einige Gebäude, die eigentlich im betrachteten Bereich liegen würden, aufgrund von möglichen Inhomogenitäten in der Modellierung ausgeschlossen worden. Bei diesen Gebäuden würde das angewendete Clipping-Verfahren versagen. Um diese Gebäude wieder in den Datensatz einbinden zu können, muss diese Fehlerquelle erst behoben werden. Dies findet aber nicht im Rahmen dieser Arbeit statt, sodass besagte Gebäude zweckmäßig aus dem Testbereich ausgeschlossen werden. Bei dem betrachteten Testbereich handelt es sich um den Straßenabschnitt Schäffbräustraße und Sauerstraße in der Ingolstädter Altstadt. Da dieser Bereich aber keine Straßenmarkierungen aufweist, wird zum Test der Markierungen zusätzlich die große Kreuzung der Nürnberger Straße mit der Theodor-Heuss-Straße betrachtet:

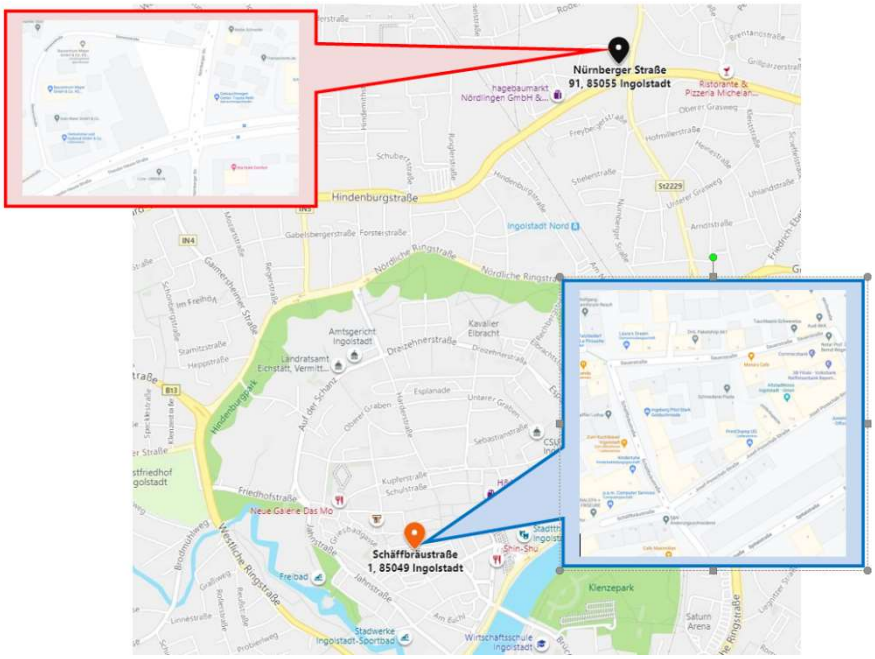

Abbildung 23 Die beiden Testbereiche mit Onlinekartendiensten verortet

Mit der zur Analyse verwendeten Hardware[30] braucht das Programm zur Analyse der Gebäudemodelle etwa dreieinhalb Stunden, die Analyse der Straßenmarkierungen etwa zwei Stunden. Einen Großteil davon nimmt jeweils der Schritt zur Pufferzonenbildung und dem Clipping in Anspruch (jeweils ca. 85% der Gesamtlaufzeit[31]). Sobald dieser umfangreichste Programmabschnitt aber vollendet worden ist, werden innerhalb weniger Minuten die einzelnen Analysen und das gefilterte Visualisieren abgeschlossen. FME gibt am Ende eines Durchlaufes zusätzliche Informationen über den insgesamten Verlauf. Neben der Gesamtlaufzeit stehen hier noch Informationen über die Anzahl gelesener und geschriebener FeatureTypes, sowie, ob die Translation erfolgreich war oder fehlgeschlagen ist.

Die nachfolgenden Abbildungen zeigen das Modell aus einer festen Perspektive mit der zur Metrik jeweils gehörenden Einfärbung. Für den Kontext enthält das Ergebnis noch die Fahrbahnflächen des Straßenmodells für diesen Bereich. Es ist darauf hinzuweisen, dass bei einigen Gebäuden die Rückwände fehlen. Das diese sind im Scan nicht enthalten gewesen und verfälschen durch ihr Fehlen nur das ästhetische Wirken des Ergebnisses, nicht aber die thematische Aussage. Zusätzlich sind die Analyseergebnisse auch in Form von MATLAB-Plots abgebildet, um eine Aussage über die statistische Verteilung der Werte treffen zu können.

[30] Systemspezifikationen siehe Anhang
[31] Abgeschätzt anhand dreimaliger Durchführung mit Minimaldatensatz: ein einzelnes Gebäudemodell

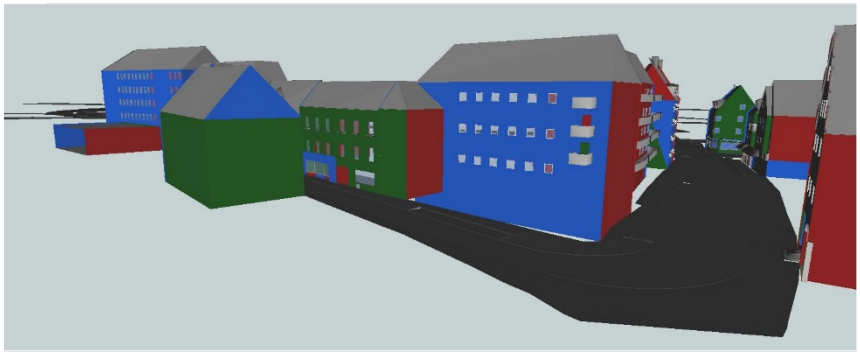

Abbildung 24 Ergebnis der Analyse der Punkt-Flächen-Distanz

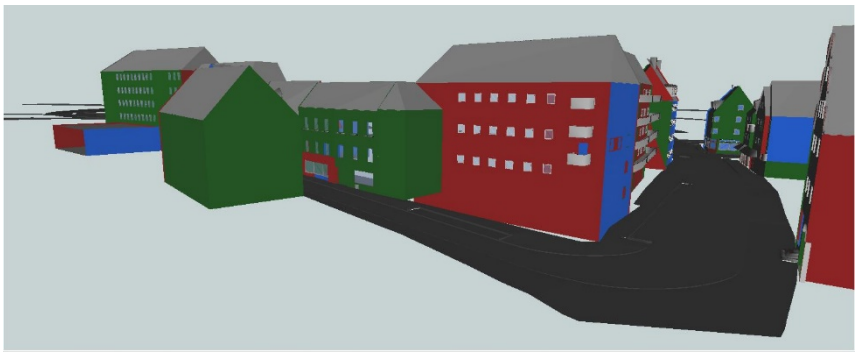

Abbildung 25 Ergebnis der Flächen-Ebenen Analyse: Orientierung

Abbildung 26 Ergebnis der Flächen-Ebenen Analyse: Inklination

Abbildung 27 Ergebnis der Punktdichte Analyse

Das Ergebnis zeigt: Die Analysen liefern sichtbare und verständliche Ergebnisse. Allerdings sind diese keinesfalls fehlerfrei. Betrachtet man beispielsweise das in den Abbildungen 24 bis 27 sichtbare Gebäude in der Bildmitte, welches an der Abzweigung des Fahrtweges gelegen ist, so fällt auf, dass

die Werte aller analysierten Metriken deutlich von den jeweiligen Toleranzbereichen abweichen. Ein Blick auf die Werte verrät zudem, wie stark vor allem die Punkt-Flächen Distanz und die Punktdichte davon abweichen. So zeigt der Blick auf die zum Betrachter zugewandte Hauswand eine mittlere Distanz von -21.8 cm mit detektierten Maximal- und Minimaldistanzen bei etwa ± 1 Metern. Besonders die mittlere Punktdichte von etwa $\frac{552}{m^2}$ fällt deutlich als Abweichung auf, verglichen mit der mittleren Punktdichte über das gesamte Messgebiet von über $\frac{3700}{m^2}$.

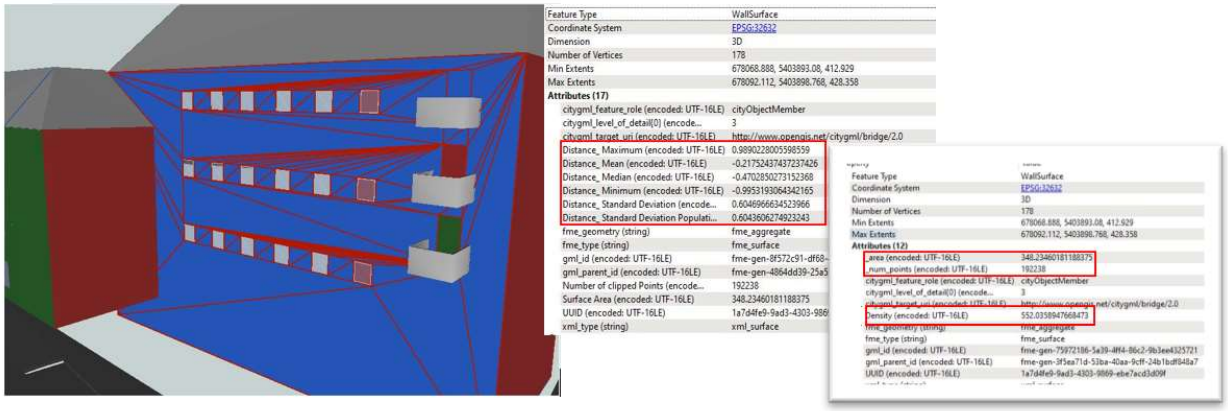

Abbildung 28 Abbildung einer auffälligen Wand mit Zusatzinformationen

Die Ursache zeigt sich durch die Überlagerung des eingefärbten Modells mit der Punktwolke aus der Scanbefahrung. Zum Zeitpunkt der Messung war das Gebäude fast vollständig eingerüstet. Die Daraus entstandene Verdeckung hat nicht nur zu einer auffällig geringen Punktdichte geführt, sondern verfälscht auch das Ergebnis der anderen Metriken. Der Einfluss wird besonders im Hinblick auf die Seitenwand des links davon abgebildeten Nachbargebäudes deutlich. Die Messpunkt erreichen nur die Gerüstkonstruktion und dringen kaum bis zur eigentlichen Hauswand durch. Das führt zu einer mittleren PTS – Distanz von 20.4 cm, allerdings mit einem Maximum von 99.5cm und einem Minimum von knapp -3cm. Das Ergebnis suggeriert dadurch, die Modellfläche wäre zu weit im Gebäude Inneren modelliert worden. Indem aber zusätzlich die mittlere Dichte betrachtet wird, fällt hier bei einem Wert von etwa $\frac{\mathbf{463}}{m^2}$ eine deutliche Abweichung auf.

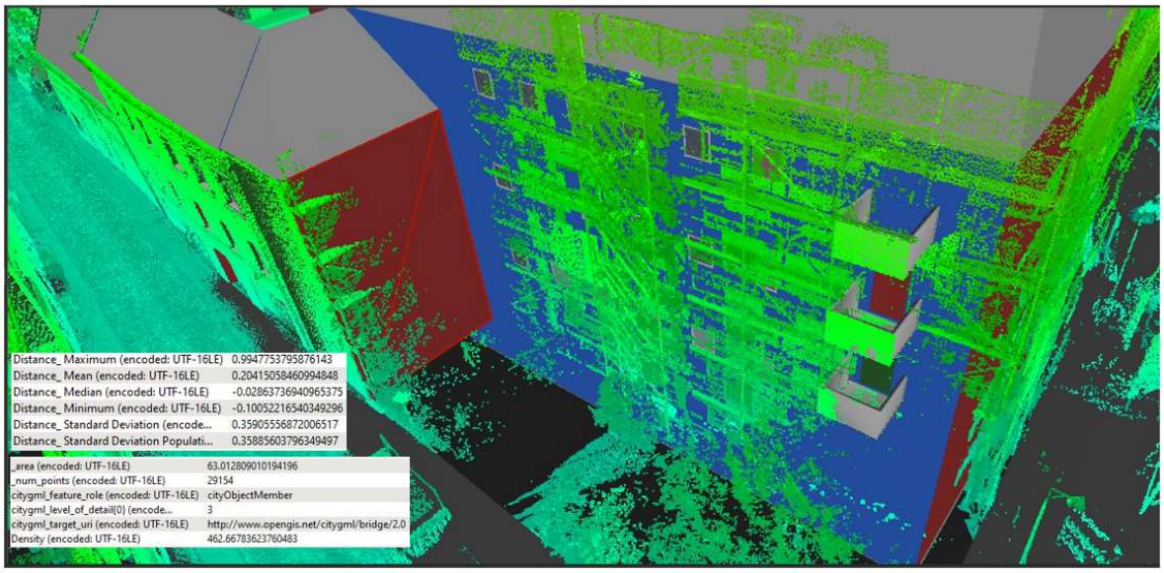

Abbildung 29 Abgeschattete Gebäudeflächen während der Laserscan-Befahrung. Deutlich erkennbar: die Gebäude sind eingerüstet. Links unten: Auszüge aus den zu der markierten Wand gehörenden Attributen

Es zeig sich also, dass die Metriken einzeln insgesamt keine wirklich absoluten Aussagen über die Genauigkeit der Modellierung treffen können. Erst in der passenden Kombination von mehreren Metriken ist es möglich, eine begründete Aussage treffen zu können. Die Kombination PTS-Distanz und Punktdichte liefert dabei die deutlichste Aussage: Die Distanz allein reicht nicht aus, eine deutliche Aussage über eine Abweichung im Modell zu treffen. Wird aber der zur Fläche gehörende Dichtewert zudem betrachtet, so ergibt die Kombination aus hoher Punktdichte und PTS Distanz eine glaubwürdige Aussage: Hier kann es sich außerhalb der Toleranz um eine Fehlmodellierung handeln, beziehungsweise bei einer Distanz innerhalb der Toleranz um eine passende Modellierung.

Durch die Aufteilung der Ausrichtung der Objektfläche in die zwei Ausrichtungskomponenten Orientierung und Inklination ist es einfacher, im Ergebnis mögliche Fehlmodellierungen zu entdecken. Leider ist darauf hinzuweisen, dass der verwendete Algorithmus nicht immer brauchbare Ergebnisse liefert. Die Werte der Inklination sind dabei noch am vertrauenswürdigsten. Für Wände und Türen wird schließlich angenommen, dass diese vertikale Flächen darstellen. Der Normalenvektor wird in dem Fall also in der dritten Komponente für die Z-Richtung einen Wert sehr nahe null aufweisen. Das macht es einfacher, eventuelle Falschbezeichnungen im PCA-Teil des Pythonskriptes auszuschließen. Das Ergebnis (Abbildung 26) macht deutlich, wie gut die Modellflächen vertikal verfasst worden sind. Bis auf wenige Fälle mit Abweichungen unter $\pm 2°$ liegen alle Wände und Türen im Rahmen der Toleranz – obwohl auch hier wieder die durch das Gerüst verdeckte Wandfläche auffällt. Das lässt sich dadurch begründen, dass das Gerüst als dreidimensionales Objekt in seiner Ausdehnung nicht unbedingt perfekt vertikal dasteht. Die Orientierung dagegen schwankt deutlich im Wertebereich. Das liegt daran, dass hier Mehrdeutigkeiten nach der Hauptkomponentenanalyse auftreten können, wenn der falsche der drei Eigenvektoren als Normalenvektor verwendet wird. Zudem versagt das Verfahren in Sonderfällen bei Wolken mit zwei nahezu identischen Ausdehnungen oder in Fällen wie in Abbildung 29. Die dabei resultierenden Vektoren lassen sich durch Auslöschung der Mehrdeutigkeiten auf einen normierten

Bereich[32] eingrenzen, allerdings schwanken die Werte stark in diesem Bereich. Deutlich wird dies wieder für die besagte vom Gerüst verdeckte Fläche. Die detektierte Orientierung beläuft sich auf 25.88°, was bei einer Toleranz von 1° für die Orientierung zu einem deutlichen Unterschied führt. Im Laufe der Entwicklung dieser Arbeit hat sich diese unschöne Eigenschaft der Umsetzung des dritten Konzepts verdeutlicht. Fixieren lässt sich diese methoden-bedingte Fehlberechnung mit dem in Kapitel 4.2.2.1 beschriebenen alternativen Ansatz.

Diese Sachlage wird in Abbildung 30 besonders deutlich: Es lässt sich gut erkennen, dass die Inklination nahe 0° herum verteilt liegt, wohingegen der Wertebereich der Orientierung den Gesamtbereich von $\pm 45°$ erfüllt. Aussagekräftig sind hier neben dem Histogramm auch die mitabgebildeten Boxplots. Diese Form der Visualisierung von Berechnungsergebnissen wird vor allem zur Darstellung statistischer Verteilungen verwendet.

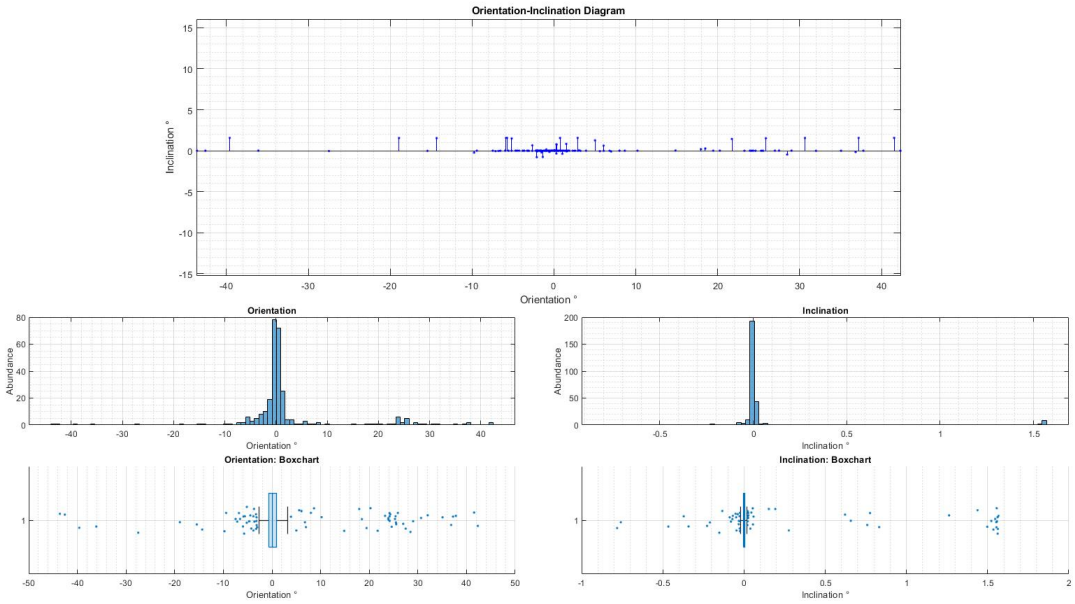

Abbildung 30 Visualisierung der numerischen Ergebnisse der Metriken Orientierung und Inklination

Bei der Inklination liegt der Prozentbereich von 25% bis 75% der Verteilung sehr nahe bei 0°, die Ausreißer sind – verglichen mit dem Umfang der Daten – dünn gesät. Anders bei der Orientierung: Der Boxplot zeigt nicht nur einen deutlich größeren Wertebereich, sondern auch Perzentile bei etwa $\pm 2°$. Die im Programm gesteckten Grenzen erscheinen daher mit $\pm 1°$ ziemlich eng. Es empfiehlt sich also, für ein aussagekräftigeres Ergebnis den zuvor beschriebenen alternativen Ansatz umzusetzen.

Die MATLAB-Implementierung liefert neben der Visualisierung der beiden Winkel auch Grafiken für die PTS-Distanzen und Intensitätsverteilungen bei den Straßenmarkierungen. Für die Türen und Wände sind dies folgende Grafiken: Während die Türen weniger Ausreißer im Wertebereich aufweisen, scheint der Bereich von 25-75% der Verteilung bei den Wänden ein besseres Ergebnis zu liefern: Hier sind die beiden Grenzen recht gleichmäßig um den Median verteilt. Das Histogramm liefert zudem jeweils die Bestätigung der Annahme, die Werte seien normalverteilt durch den

[32] Siehe 4.2.2.1 Konzept 3

charakteristischen Verlauf, ähnlich der Wahrscheinlichkeitsdichtefunktion, die auch bekannt ist als die so genannte Gauß-Glocke

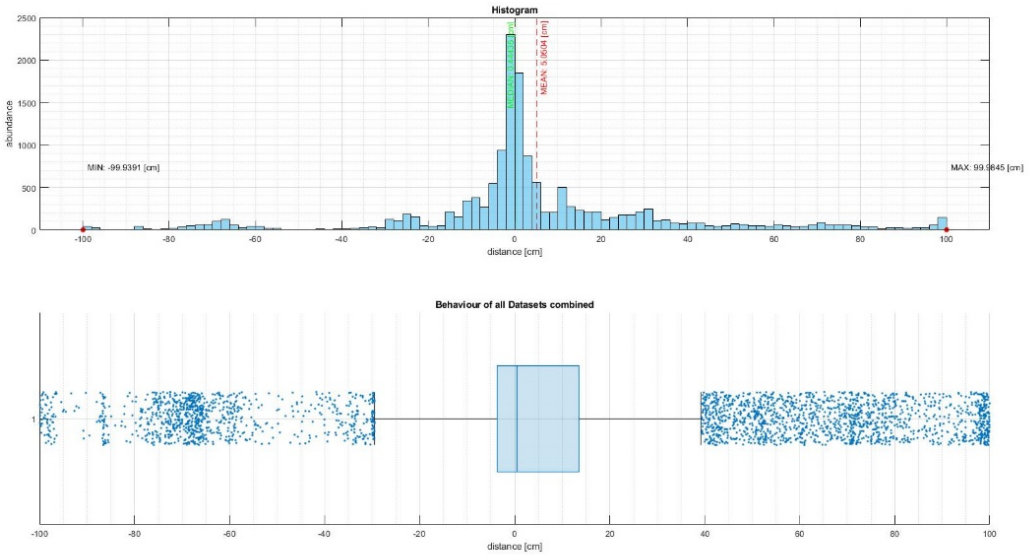

Abbildung 31 Histogramm und Boxchart für die Türen

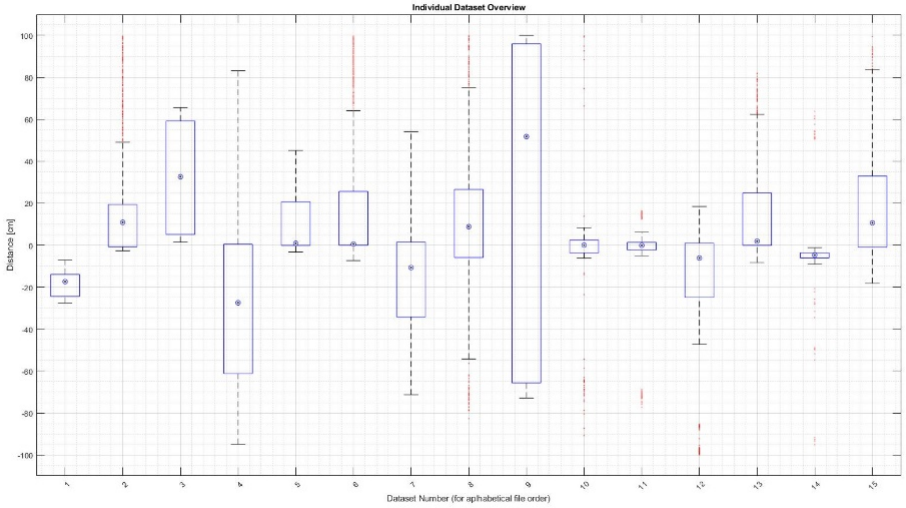

Abbildung 32 Boxplot für die Türen im Testbereich

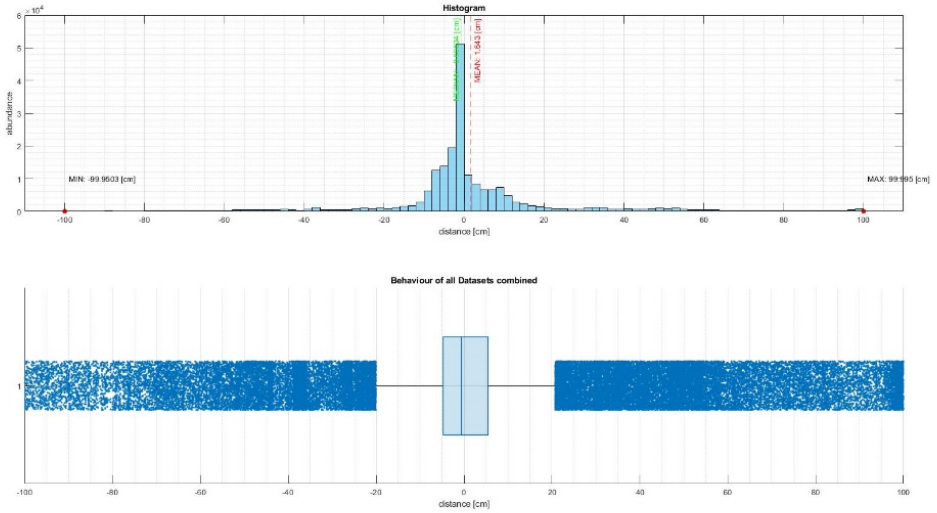

Abbildung 33 Histogramm und Boxchart für die Wandflächen

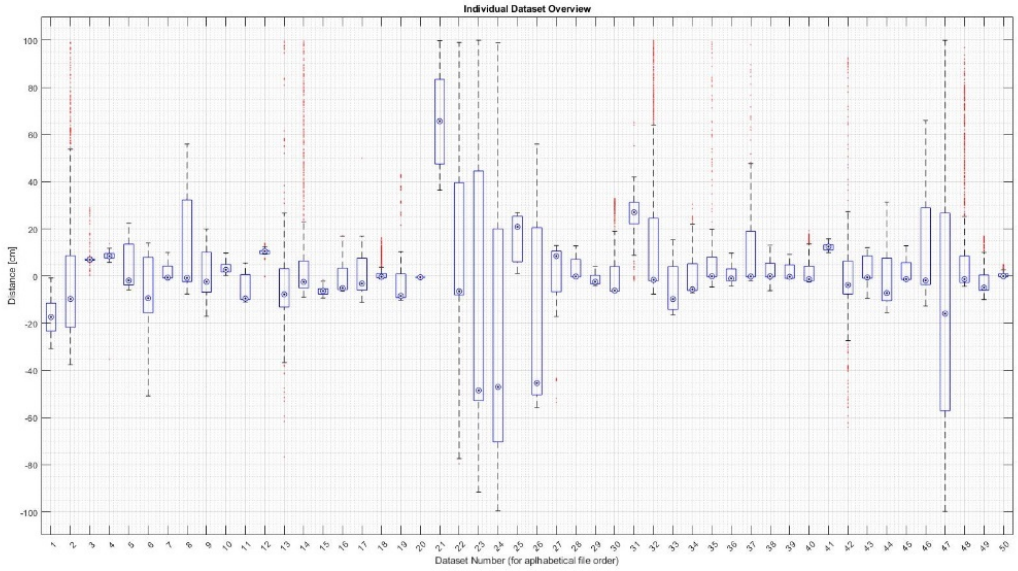

Abbildung 34 Boxplot über eine Auswahl von 50 Wänden im Testbereich

Der Blick auf den in Abbildung 34 dargestellten Boxplot zu einer Auswahl von 50 Wandflächen des Testbereichs zeigt zum einen deutlich, dass einzelne Wände deutlich vom Erwartungswert von 0cm Distanz abweichen. Es wird aber auch deutlich, wie homogen die Werte je Wand verteilt sind. Auffällig ist dabei Wand Nummer 24, die durch wenige Punkt repräsentiert worden zu scheint: der Wertebereich erstreckt sich im Vergleich zu den anderen deutlich über den gesamten Wertebereich von -1 bis 1 Metern. Der Prozentbereich von 25-75% der Verteilung reicht aber ebenfalls sehr weiträumig von +20cm bis -70cm, wobei der Median bei kurz vor -50cm liegt. Die Wand scheint also durch verhältnismäßig wenige Punkte repräsentiert worden zu sein. Tatsächlich handelt es sich hier um die in Abbildung 28 untersuchte, vom Baugerüst verdeckte Hauswand. Aus dem Boxplot der Distanzen lässt sich also tatsächlich ein Zusammenhang zur Punktdichte der Objektfläche herstellen.

Nach diesem Einblick in das Ergebnis der Gebäudemodellanalysen bietet sich noch die Betrachtung des zusätzlich untersuchten Straßenmodells der Marktkaufkreuzung nordöstlich der Innenstadt. Metrisch ermittelt worden ist die mittlere detektierte Intensität der vorhandenen Straßenmarkierungen. Nach farblicher Codierung ergibt sich folgendes Bild:

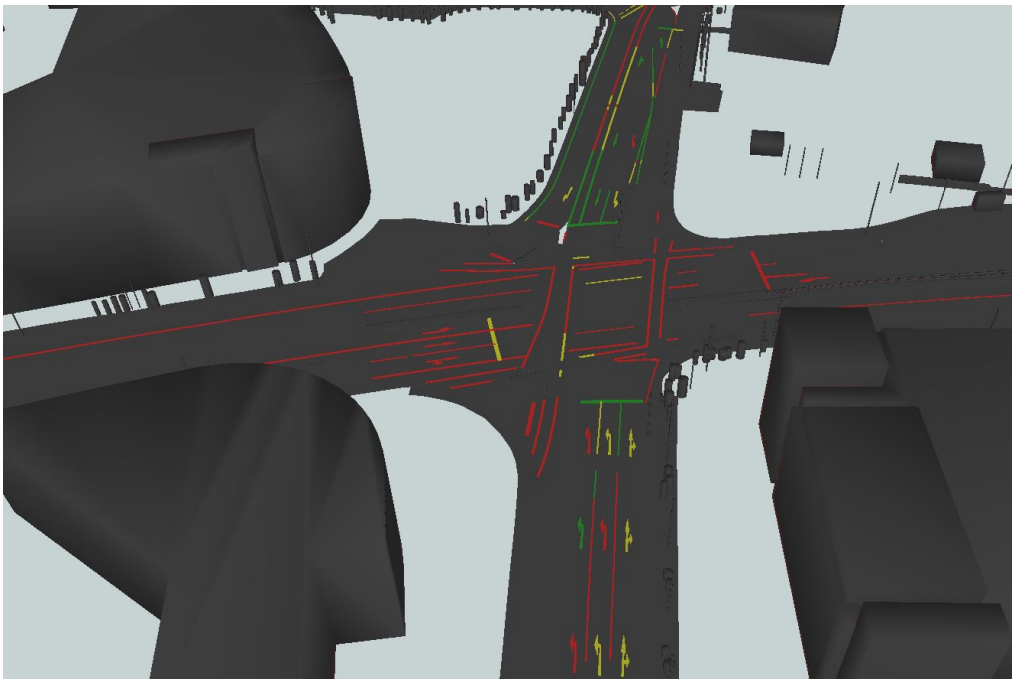

Abbildung 35 Ausschnitt aus dem analysierten Straßenabschnitt

Zur Darstellung in dieser Aufnahme sind die Markierungen etwas nach oben versetzt worden, da sonst Teile durch andere Flächen, etwa der Fahrbahn, verdeckt würden. Der Blick auf das Ergebnis zeigt sehr viele Markierungen mit einem zu geringem mittleren Intensitätswert. Es fällt auf, dass besonders die Markierungen auf der vom Betrachter aus kreuzenden Straße außerhalb der Toleranz liegen. Das Ergebnis vermittelt so den Eindruck einer schlechten Modellierung oder einer zu strengen und unpassenden Toleranz. Aufschluss liefert hier aber der zusätzliche Blick auf das Ergebnis in Verbindung mit den zugrundeliegenden Punktdaten.

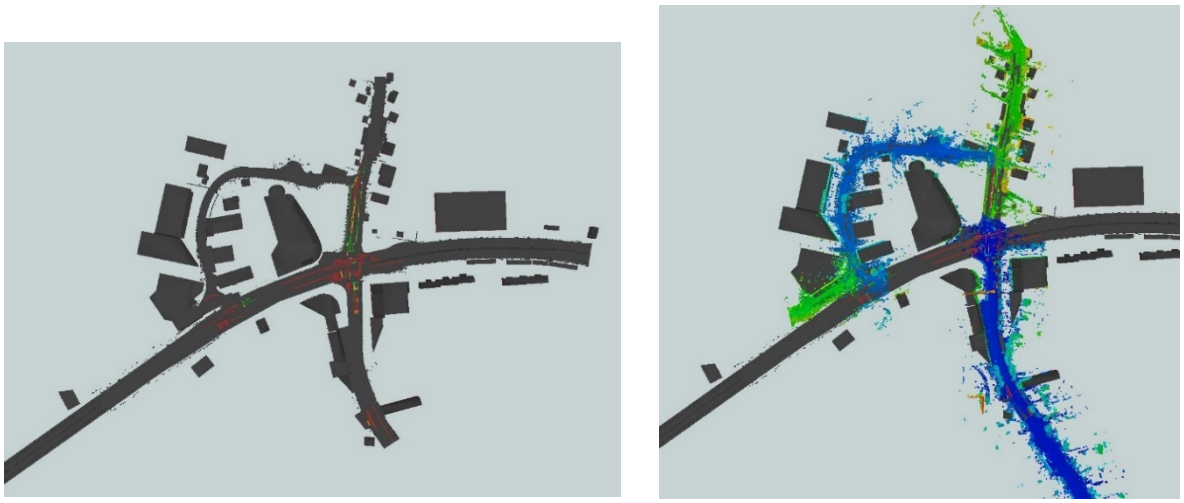

Abbildung 36 Der modellierte Straßenbereich (links) mit zusätzlichem Blick auf den Abdeckungsbereich der Punktwolken aus der Scanbefahrung (rechts).

Die Übersicht des gesamten Messgebiets des Straßenmodells zeigt, dass manche Straßenabschnitte zwar im Modell vorkommen, offensichtlich aber durch keine Punktwolke abgedeckt werden. Das Messfahrzeug hat gut erkennbar über eine Nebenstraße einen Teilbereich des Modells umfahren, weshalb hier keine beziehungsweise nur sporadisch verstreute Punkte vorliegen. Zudem zeigt ein anderer Blickwinkel, dass einige Markierungen aufgrund der Verkehrslage zum Zeitpunkt der Messung durch Fahrzeuge oder anderweitige Objekte wie Ampeln oder Verkehrsschilder im Scanbereich verdeckt wurden.

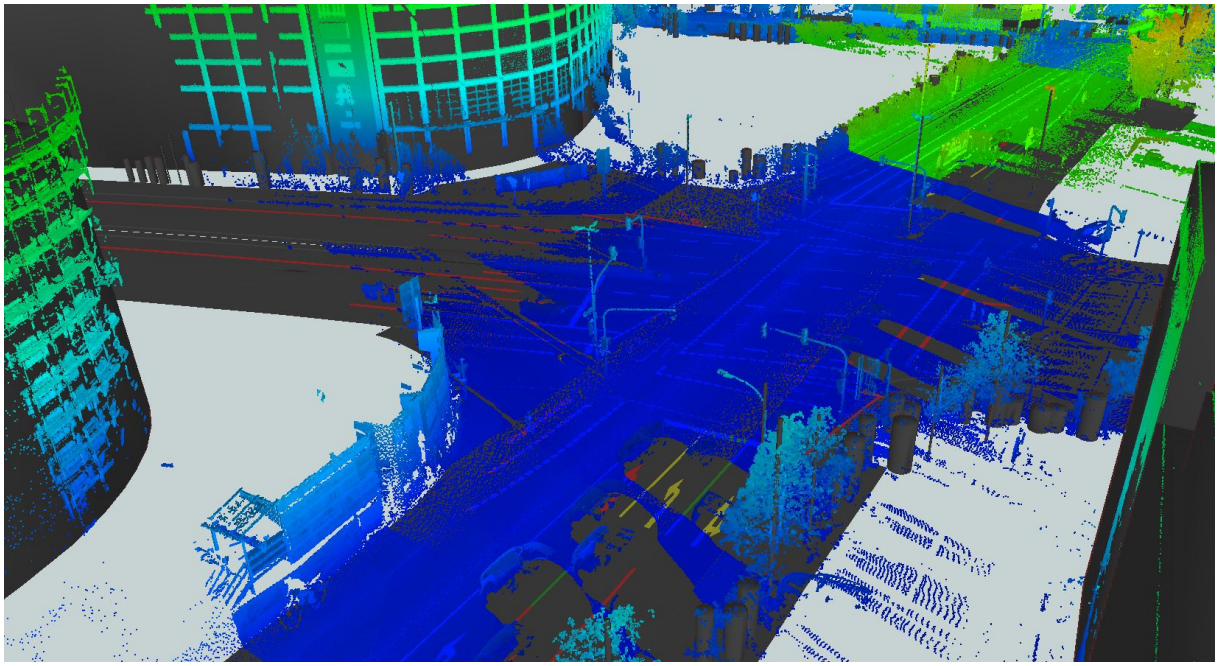

Abbildung 37 Die Messsituation aus einer anderen Perspektive. Gut erkennbar: Abschattungen durch andere Verkehrsteilnehmer und Verkehrszeichen sowie nicht befahrene Bereiche im Testgebiet

Für eine bessere Aussage zur Qualität der Modellierung dieses Straßenbereichs sind weitere Aufnahmen notwendig, die dann im Optimalfall frei von Verdeckungen – zumindest durch andere Verkehrsteilnehmer – sind.

Nichtsdestotrotz lassen sich die ermittelten Intensitäten betrachten. Es zeigt sich – unter Berücksichtigung des Einflusses durch oben beschriebene Verdeckungen – die Markierungen liefern selbst bei voller Einmessung kein homogenes Rückstrahlergebnis, sondern variieren durchaus. Das kann beispielsweise an Defekten oder Verschmutzung der jeweiligen Markierung liegen.

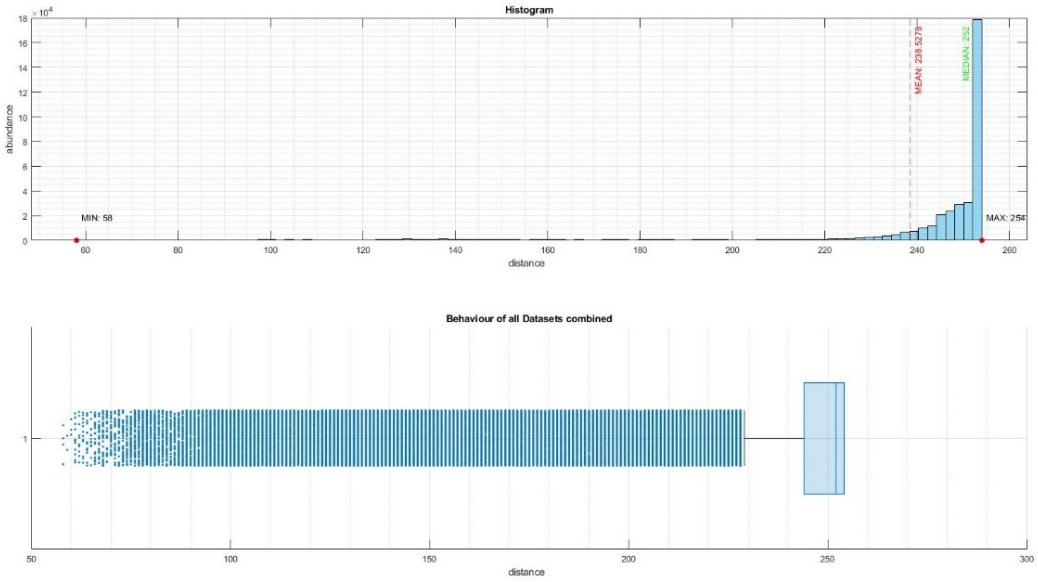

Abbildung 38 Histogramm und BoxChart für die betrachteten Intensitätswerte

Die Abbildungen 35 und 37 verraten aber auch einen systematischen Einfluss der Modellierung auf die Intensitätsanalyse. Gestrichelte Linien, wie die Leitlinie bei mehrspurigen Straßen (vgl. Abbildung 37) werden nicht je Strich einzeln im Modell angegeben, sondern mit mehreren zu einem Objekt zusammengefasst. Das reduziert zwar die Datenmenge, führt aber dazu, dass sich in einem Markierungsobjekt dadurch nicht nur Markierungen, sondern auch Abschnitte der Fahrbahn in Form von Asphalt befinden. Dadurch entsteht ein bimodales Intensitätsbild: es gibt für die Markierung zwei unterschiedliche Intensitätsbereiche – für die Analyse wird aber je Markierungsfläche nur ein Wert errechnet, der dadurch das Mittel beider Intensitätsbereiche ergibt. Das führt schon im vornherein zu einem auffällig niedrigen Intensitätswert.

Abschließend noch einmal der Blick auf die gewählten Schwellwerte. Das MATLAB-Skript liefert nach Abschluss der Erstellung der Plots noch das Ergebnis der Berechnung der Konfidenzintervalle für die PTS-Distanzen und die Intensität. Die Schwellwerte sind zu Beginn explorativ gesetzt worden, also nach Betrachtung der Ergebnisse der Metriken. Damit sind diese aber noch nicht statistisch gesichert. Mit der Aussage der Berechnungen von MATLAB bestätigen sich aber diese Werte als passend. Die Konsole liefert folgende Ergebnisse.

```
Door PTS Distances:
Calculated Boundaries:
 [ 5.002240 | 5.098602 ]
Wall PTS Distances:
Calculated Boundaries:
 [ 1.629947 | 1.656130 ]
Road Marking Intensity:
Calculated Boundaries:
 [ 238.524581 | 238.530933 ]
```

Abbildung 39 Auszug aus der Ausgabe der MATLAB – Konsole

Diese Werte bestätigen die explorativ gewählten Intervallgrenzen. Für Orientierung und Inklination allerdings fehlt aufgrund der gewählten Implementierung – wie in Kapitel beschrieben – eine gesicherte Aussage über die Genauigkeit der Schätzung des Normalenvektors aus den Punktdaten.

6. Fazit und Ausblick

6.1. Fazit

Im Rahmen dieser Arbeit ist also ein Programm entwickelt worden, mit dem sowohl ein Ausschnitt eines Stadtmodells, bestehend aus mehreren Einzelgebäudemodellen, als auch ein Straßenraummodell auf mögliche Abweichungen zwischen Modell und Referenzpunktwolke untersucht und die Ergebnisse nachvollziehbar dargestellt werden können. Damit entspricht das Ergebnis der ursprünglichen Zielsetzung der Arbeit. Als Einschränkungen sind folgende Aspekte anzuführen: Der verwendete Testbereich für das Stadtmodell umfasst nicht den gesamten, zur Verfügung stehenden Datensatz. Grund hierfür ist zum einen die schiere Datenmenge, die zu einer sehr langen Rechenzeit und hohen Speicherkapazität führt. Zum anderen schlägt für manche Gebäudemodelle das in Kapitel 4.2.1.3 beschriebene Clipping-Verfahren zur Auftrennung der Eingangspunktwolke in mehrere kleinere Teilwolken fehl. Gemäß der Fehlermeldung im FME Workspace verfügen diese Gebäude über nicht unterstützte Geometrietypen. Dies ließe sich leicht abfangen, indem nur bestimmte, unterstützte Geometrien zugelassen werden. Allerdings führt das hier nicht zur Lösung der Geometrieinhomogenität und der Fehler tritt dennoch auf.

Nichtsdestotrotz liefert das Programm einen guten Überblick über die Qualität der Modellierung, hilft bei der fortwährenden Verbesserung der Modelle und stellt somit eine nützliche Grundlage für weiterführende Analysen.

6.2. Ausblick

Obwohl das Programm brauchbare Informationen über im Modell auftretende geometrische Abweichungen liefert, bieten sich einige mögliche Ansätze zur Erweiterung und Optimierung an.

Die in Kapitel 4.2.2.1 vorgestellten Konzepte für mögliche Metriken, mit denen geometrische Abweichungen formuliert werden könnten, sind nur zum Teil umgesetzt worden (vgl. Abbildung 18). Einige von den nicht implementierten Konzepten könnten aber nützliche Ergebnisse zur Bewertung der Modellierung des vorliegenden Stadt- und Straßenmodells liefern. Besonders für den Straßenraum nützlich ist sicher die Erweiterung um Metriken, die Kenngrößen für Objekte wie Ampeln oder Verkehrszeichen liefern, die gerade für Verkehrssimulationen von großer Bedeutung sind.

Wie sich in Kapitel 5 gezeigt hat, sind die Schwellwerte für die statistischen Tests im Rahmen dieser Arbeit hauptsächlich explorativ entstanden. Eine möglich Optimierung wäre hier die automatisierte Ermittlung der Schwellwerte anhand eines Testdatensatzes, um im Anschluss damit einen größeren Bereich analysieren zu können. Durch die Automatisierung ließe sich so besser passende Aussage über die geometrische Genauigkeit des Datensatzes treffen, da die Schwellwerte selbst eine charakteristische Eigenschaft dieses Datensatzes darstellen.

Hinsichtlich der Laufzeit ist es auch denkbar, das Programm zu verbessern. Besonders bieten sich hier die Abschnitte des Clippings und der PTS-Distanzberechnung an. Das Clipping macht den Großteil der Gesamtlaufzeit aus, das Programm würde bei einer Reduktion dieser besser eingesetzt werden können. Falls sich ein Weg finden lässt, die Laufzeit deutlich zu reduzieren und die Distanz ohne

Ausdünnung der Punktwolke berechnen zu können, bietet das Programm einen guten Ansatzpunkt für beispielsweise automatisierte Verfahren zur Korrektur geometrischer Abweichungen.

Aus geodätischer Sicht sei hier nochmal das Verfahren zur Schätzung vertikaler Ebenen betont. Das bisher implementierte Verfahren auf Grundlage der Principal Component Analysis liefert zwar ein sehr schnelles, aber statistisch wenig aussagekräftiges Ergebnis. Es empfiehlt sich, etwa durch Kombination von PCA, RANSAC und einem Ausgleichungsansatz auf Basis der Hesse ‘schen Normalenform den Algorithmus zur Schätzung der vertikalen Ebenen durch die noch ausstehende Information über die Güte der Schätzung zu erweitern.

Abbildungsverzeichnis

Literaturverzeichnis

[1] S. Weißenborn, „Welt.de," Welt.de, 27 November 2009. [Online]. Available: https://www.welt.de/motor/article5347845/Der-Erfinder-des-Autos-hiess-Cugnot-nicht-Benz.html. [Zugriff am 07 Oktober 2020].

[2] Verband der Automobilindustrie e.V., „VDA - Innovation und Technik," Aperto AG, 2020. [Online]. Available: https://www.vda.de/de/themen/innovation-und-technik/zeitstrahl/zeitstrahlinnovationen.html. [Zugriff am 07 Oktober 2020].

[3] H. Bardt, „Ökonomische Trends - Deutsche Autoindustrie und autonomes Fahren," 2016. [Online]. Available: https://www.econstor.eu/bitstream/10419/156429/1/873462130.pdf. [Zugriff am 31 Januar 2021].

[4] Society of Automotive Engineers, „SAE J3016 automated-driving graphic," SAE International, 2021. [Online]. Available: https://www.sae.org/news/2019/01/sae-updates-j3016-automated-driving-graphic. [Zugriff am 31 Januar 2021].

[5] S. Weißenbron, „In fünf Schritten zum Roboter-Auto," *ACE Lenkrad,* p. 13, 01 Dezember 2020.

[6] Tesla Deutschland, „Model 3 | Tesla Deutschland," Tesla Deutschland, 2021. [Online]. Available: https://www.tesla.com/de_de/model3. [Zugriff am 31 Januar 2021].

[7] Waymo LLC, „Home - Waymo," Waymo LLC, 2021. [Online]. Available: https://waymo.com/. [Zugriff am 4 Februar 2021].

[8] D. Keim, „Künstliche Intelligenz - Chancen und Herausforderungen," 07 November 2019. [Online]. Available: https://www.evtheol.uni-muenchen.de/aktuelles/nachrichten/archiv/2019/vortrag-ki-7_11_19/ki_flyer-vortrag-7_11_19.pdf. [Zugriff am 07 Oktober 2020].

[9] O. Wysocki, Semantic-based Geometry Refinement of 3D City Models for Testing Automated Driving, München: Medien- und Publikationsserver Technische Universität München, 2020.

[10] T. Coduro, „mediaTUM - Medien- und Publikationsserver," 2018. [Online]. Available: https://mediatum.ub.tum.de/doc/1451954/1451954.pdf. [Zugriff am 01 Februar 2021].

[11] D. (Hg.), Physik Oberstufe - Ausgabe A, Hannover: Hermann Schroedel Verlag KG, 1975.

[12] T. Wunderlich, „Grundlagen der Vermessungskunde 2 für Geodäten und Umweltingenieure - TUM Moodle," 2018. [Online]. Available:

https://www.moodle.tum.de/pluginfile.php/1389388/mod_resource/content/1/Vermessungskunde_2_VO_SS2018.pdf. [Zugriff am 24 Januar 2021].

[13] M. Schmitt, „Photogrammmetrie und Fernerkundung 4: Einleitung und Motivation," 20 April 2020. [Online]. Available: https://www.moodle.tum.de/pluginfile.php/2193097/mod_resource/content/3/PF4_Kap1_Einleitung_online.pdf. [Zugriff am 24 Januar 2021].

[14] U. Hugentobler, „Satellitengeodäsie 1 - Einführung | TUM Moodle," 2019. [Online]. Available: https://www.moodle.tum.de/pluginfile.php/1809508/mod_resource/content/1/SatGeod1_Kap1.pdf. [Zugriff am 30 Januar 2021].

[15] T. Wunderlich, „Kinematische Geodäsie - TUM Moodle," 2020. [Online]. Available: https://www.moodle.tum.de/pluginfile.php/2132632/mod_resource/content/0/VO_KinGeodaesie.pdf. [Zugriff am 24 Januar 2021].

[16] 3D Mapping Solutions GmbH, „3D Mapping - Home," 3D Mapping Solutions GmbH, 2021. [Online]. Available: https://www.3d-mapping.de/. [Zugriff am 31 Januar 2021].

[17] U. Stilla und M. Hebel, Photogrammmetrie und Fernerkundung 3: Registrierung von LASER-Punktwolken, München: TUM-Vorlesungsskript, 2019.

[18] C. (. Heipke, Photogrammetrie und Fernerkundung - Handbuch der Geodäsie, Berlin: Springer Spektrum, 2017.

[19] T. H. Kolbe, „Geoinformatik 2: 3D - Stadt - und Landschaftsmodelle - TUM Moodle," 27 Januar 2020. [Online]. Available: https://www.moodle.tum.de/pluginfile.php/2096338/mod_resource/content/0/Geoinformatik%202%20-%2009%20-%203D-Stadt-%20und%20Landschaftsmodellierung.pdf. [Zugriff am 24 Januar 2021].

[20] Lexikographisches Institut München, Die neue deutsche Rechtschreibung, Gütersloh: Bertelsmann Lexikon Verlag, 1996.

[21] RegioWiki Niederbayern, „Hochwasser 2013 (Passau)- RegioWiki Niederbayern," 20 August 2018. [Online]. Available: https://regiowiki.pnp.de/wiki/Hochwasser_2013_(Passau)#:~:text=Das%20Hochwasser%202013%20in%20Passau,Stra%C3%9Fen%20und%20Gassen%20in%20Kan%C3%A4le.. [Zugriff am 24 Januar 2021].

[22] T. H. Kolbe, T. Kutzner und S. Nguyen, „Einführung in die Informatik für Geodäten 1 - TUM Moodle," Oktober 2017. [Online]. Available: https://www.moodle.tum.de/pluginfile.php/1296001/mod_resource/content/1/Informatik%201%20-%2000%20-%20Begr%C3%BC%C3%9Fung.pdf. [Zugriff am 24 Januar 2021].

[23] Open Geospatial Consortium, „CityGML | OGC," Open Geospatial Consortium - OGC, 2021.

[Online]. Available: https://www.ogc.org/standards/citygml. [Zugriff am 23 Januar 2021].

[24] T. H. Kolbe, „Informatik 2: Datenaustausch mit XML -TUM Moodle," 2018. [Online]. Available: https://www.moodle.tum.de/pluginfile.php/1408151/mod_resource/content/2/Informatik%202%20-%2009%20-%20Datenaustausch%20mit%20XML.pdf. [Zugriff am 23 Januar 2021].

[25] Technische Universität München, „CityGML 3.0 - Lehrstuhl für Geoinformatik," 2020. [Online]. Available: https://www.lrg.tum.de/gis/projekte/citygml-30/. [Zugriff am 23 Januar 2021].

[26] G. Gröger, T. H. Kolbe, C. Nagel und K.-H. Häfele, „CityGML Specification," 04 April 2012. [Online]. Available: https://mediatum.ub.tum.de/doc/1145731/1145731.pdf. [Zugriff am 30 Januar 2021].

[27] T. Kutzner, K. Chaturvedi und T. H. Kolbe, „CityGML 3.0: NEw Functions Open Up New Apllications | SpringerLink," 26 Februar 2020. [Online]. Available: https://link.springer.com/article/10.1007/s41064-020-00095-z. [Zugriff am 23 Januar 2021].

[28] Safe Software, „About Safe Software," Safe Software, 2020. [Online]. Available: https://www.safe.com/about/. [Zugriff am 22 Dezember 2020].

[29] Safe Software, „FME Server | Data Integration and Automation | Safe Software," Safe Software, 2020. [Online]. Available: https://www.safe.com/fme/fme-server/. [Zugriff am 22 Dezember 2020].

[30] Safe Software, „FME Cloud | Hosted Data Integration | Safe Software," Safe Software, 2020. [Online]. Available: https://www.safe.com/fme/fme-cloud/. [Zugriff am 22 Dezember 2020].

[31] Safe Software, „Education Subscription | FME | Safe Software," Safe Software, 2020. [Online]. Available: https://www.safe.com/pricing/education-subscription/. [Zugriff am 22 Dezember 2020].

[32] Safe Software, „FME Desktop | Data Integration and Automation | Safe Software," Safe Software, 2020. [Online]. Available: https://www.safe.com/fme/fme-desktop/. [Zugriff am 22 Dezember 2020].

[33] Safe Software, „FME Integrations Gallery," Safe Software, 2020. [Online]. Available: https://www.safe.com/integrate/. [Zugriff am 22 Dezember 2020].

[34] Safe Software, „FME Transformer Gallery," Safe Software, 2020. [Online]. Available: https://www.safe.com/transformers/. [Zugriff am 22 Dezember 2020].

[35] Safe Software, „FME Hub," Safe Software, 2020. [Online]. Available: https://hub.safe.com/?page=1&page_size=10&order=relevance. [Zugriff am 22 Dezember 2020].

[36] Safe Software, „FME Desktop Basic . GitBook," 2019. [Online]. Available: https://s3.amazonaws.com/gitbook/Desktop-Basic-2019/index.html. [Zugriff am 30 Januar

2021].

[37] P. Kaiser und J. Ernesti, „Rheinwerk Computing :: Python 3 - Die Programmiersprache Python," 2020. [Online]. Available: http://openbook.rheinwerk-verlag.de/python/02_001.html#u2. [Zugriff am 14 Januar 2021].

[38] Centrum Wiskunde & Informatica, „About Centrum Wiskunde & Informatica - CWI Amsterdam," Centrum Wiskunde & Informatica, 2021. [Online]. Available: https://www.cwi.nl/about. [Zugriff am 14 Januar 2021].

[39] D. de la Iglesia Castro, „Welcome to pyntcloud! - pyntcloud 0.1.3 documentation," 13 Oktober 2020. [Online]. Available: https://pyntcloud.readthedocs.io/en/latest/index.html#. [Zugriff am 14 Januar 2021].

[40] Python Software Foundation, „pyntcloud . PyPI," Python Software Foundation, 2021. [Online]. Available: https://pypi.org/project/pyntcloud/. [Zugriff am 14 Januar 2021].

[41] D. de la Iglesia Castro, „Scalar Fields - pyntcloud 0.1.3 documentation," 2020. [Online]. Available: https://pyntcloud.readthedocs.io/en/latest/scalar_fields.html#scalar-fields. [Zugriff am 14 Januar 2021].

[42] NumPy, „NumPy," NumPy Steering Council, 2020. [Online]. Available: https://numpy.org/about/. [Zugriff am 14 Januar 2021].

[43] The MathWorks Inc., „MATLAB - MathWorks -MATLAB & Simulink," The MathWorks Inc., 2021. [Online]. Available: https://de.mathworks.com/products/matlab.html. [Zugriff am 14 Januar 2021].

[44] U. Stilla und L. Hoegner, Vorlesung Ausgleichungsrechnung: Statistische Tesverfahren, München: TUM Vorlesungsskript, 2018.

[45] Lennart, „Konfidenzintervalle für den Erwartungswert einer Normalverteilung (Varianz ist bekannt) - YouTube," 26 Juni 2019. [Online]. Available: https://www.youtube.com/watch?v=GF1A89pZ29s&list=PLE2xr43Rbzg47nzxDaMJgOlFZXPhiok2r&index=3. [Zugriff am 23 Januar 2021].

[46] Lennart, „Konfidenzintervall für den Erwartungswert einer Normalverteilung (Varianz ist unbekannt) - YoutTube," 03 Juli 2019. [Online]. Available: https://www.youtube.com/watch?v=5-0vWr5kQAU&list=PLE2xr43Rbzg47nzxDaMJgOlFZXPhiok2r&index=2. [Zugriff am 23 Januar 2021].

[47] MathWorks Inc., „Visualize summary statistics with box plot - MATLAB boxplot - MathWorks Deutschland," MathWorks Inc., 2021. [Online]. Available: https://de.mathworks.com/help/stats/boxplot.html. [Zugriff am 31 Januar 2021].

[48] U. Stilla und L. Hoegner, Vorlesung Ausgleichungsrechnung: RANSAC, München: TUM-Vorlesungsskript, 2018.

[49] J. Gasteiger und T. Engel, „Vernetzte Chemie: PCA," Universität Erlangen, 15 April 2004. [Online]. Available: https://www2.chemie.uni-erlangen.de/projects/vsc/chemoinformatik/erlangen/datenanalyse/pca.html. [Zugriff am 23 Januar 2021].

[50] L. Serrano, „Principal Component Analysis (PCA) - YouTube," 10 Februar 2019. [Online]. Available: https://www.youtube.com/watch?v=g-Hb26agBFg. [Zugriff am 24 Januar 2021].

[51] MapTiler, „WGS84/UTM zone 32N - EPSG:32632 | epsg.io," MapTiler, 2019. [Online]. Available: https://epsg.io/32632. [Zugriff am 31 Januar 2021].

[52] I. N. Bronštejn, H. Mühlig, G. Musiol und K. A. Semendjajew, Taschenbuch der Mathematik, Haan-Gruiten: Verlag Europa-Lehrmittel Noruney, Vollmer GmbH & Co. KG, 2016.

[53] D. de la Iglesia, „python - Plane fitting in a 3d point cloud - Stack Overflow," Stack Overflow, 04 August 2016. [Online]. Available: https://stackoverflow.com/questions/38754668/plane-fitting-in-a-3d-point-cloud. [Zugriff am Februar 2021].

[54] MathWorks, „Was ist RANSAC? - MATLAB & Simulink," The MathWorks Inc., 2021. [Online]. Available: https://de.mathworks.com/discovery/ransac.html. [Zugriff am 23 Januar 2021].

[55] L. Hoegner, „Gewässerfernerkundung und Laserbathymetrie - TUM Moodle," 2020. [Online]. Available: https://www.moodle.tum.de/pluginfile.php/2231628/mod_resource/content/1/tum_pf4_20_02.pdf. [Zugriff am 24 Januar 2021].

Anhang

Python-Zusatzbibliotheken

Package	Version
cycler	0.10.0
kiwisolver	1.3.1
matplotlib	3.3.3
mpmath	1.1.0
numpy	1.19.3
pandas	1.1.5
Pillow	8.1.0
pip	20.3.3
pyntcloud	0.1.3
pyparsing	2.4.7
python-dateutil	2.8.1
pytz	2020.4
scikit-spatial	5.2.0
scipy	1.5.4
setuptools	40.6.2
six	1.15.0
sympy	1.7.1

Systemspezifikationen

Aktuelles Datum/Zeit:	Donnerstag, 4. Februar 2021, 17:10:27
Computername:	AMGPC001
Betriebssystem:	Windows 10 Pro 64-Bit-Version (10.0, Build 19041)
Sprache:	Deutsch (Gebietsschema: Deutsch)
Systemhersteller:	To Be Filled By O.E.M.
Systemmodell:	To Be Filled By O.E.M.
BIOS:	BIOS Date: 06/11/18 16:10:34 Ver: 04.06.05
Prozessor:	Intel(R) Core(TM) i5-4570 CPU @ 3.20GHz (4 CPUs), ~3.2GHz
Speicher:	16384MB RAM
Auslagerungsdatei:	8708 MB verwendet, 23996 MB verfügbar
DirectX-Version:	DirectX 12

Name:	NVIDIA GeForce GTX 660
Hersteller:	NVIDIA
Chiptyp:	GeForce GTX 660
DAC-Typ:	Integrated RAMDAC
Gerätetyp:	Vollanzeigegerät
Gesamtspeicher ca.:	10158 MB
Anzeigespeicher (VRAM):	1998 MB
Gemeinsam genutzter Speicherbereich:	8160 MB

Abbildung Python-Implementierung

```
import os
import csv
import glob
import fme
import fmeobjects
import numpy as np
import math
from pyntcloud import PyntCloud
from fmeobjects import (FMELogFile,FME_WARN)#FME native logs
#=============================================================================================================================
# Concept for PCA and best_fitting_plane implementation from:
# https://stackoverflow.com/questions/38754668/plane-fitting-in-a-3d-point-cloud
#=============================================================================================================================
#Definition for Principal Components Analysis PCA
def PCA(data, correlation = False, sort = True):
    mean = np.mean(data, axis=0)
    data_adjust = data - mean
    #: the data is transposed due to np.cov/corrcoef syntax
    if correlation:
        matrix = np.corrcoef(data_adjust.T)
    else:
        matrix = np.cov(data_adjust.T)
    eigenvalues, eigenvectors = np.linalg.eig(matrix)
    if sort:
        #: sort eigenvalues and eigenvectors
        sort = eigenvalues.argsort()[::-1]
        eigenvalues = eigenvalues[sort]
        eigenvectors = eigenvectors[:,sort]
    return eigenvalues, eigenvectors
#=============================================================================================================================
# Fit plane into the Pointcloud using the PCA algorithm above
def best_fitting_plane(points, equation=False):
    w, v = PCA(points)
    #: the normal of the plane is the last eigenvector
    normal = v[:,2]
    #: get a point from the plane
    point = np.mean(points, axis=0)
    if equation:

        a, b, c = normal
        d = -(np.dot(normal, point))
        return a, b, c, d
    else:
        ev1=v[:,0]
        ev2=v[:,1]
        ev3=v[:,2]
        return point, normal
#=============================================================================================================================
# Definition for calculating the signed orientation
def oriAndIncli(a,b):
#Calculates angle for 3d vectors projected to the horizontal plane in radians CURRENTLY DEGREE
# Orientation is the angle from vector a to vector b
# Positive sign: counterclockwise (mathematical direction to count angles) -> b on the left side of a
# Negative Sign: clockwise -> b on the right side of a
    raddeg=180/math.pi
    ori=math.atan2(a[0],a[1])-math.atan2(b[0],b[1])
    lengthA=math.sqrt(a[0]**2+a[1]**2)
    lengthB=math.sqrt(b[0]**2+b[1]**2)
    incl=math.atan2(a[2],lengthA)-math.atan2(b[2],lengthB)
    if 0<=abs(ori)*raddeg<=45:
        ori=ori*raddeg
    elif 45<abs(ori)*raddeg<=135:
        if ori<0:
            ori=ori*raddeg+90
        else:
            ori=ori*raddeg-90
    elif 135<abs(ori)*raddeg<=225:
        if ori<0:
            ori=ori*raddeg+180
        else:
            ori=ori*raddeg-180
    elif 225<abs(ori)*raddeg<=315:
        if ori<0:
            ori=ori*raddeg+270
        else:
            ori=ori*raddeg-270
    return ori, incl
```

```
#=============================================================================================================================
# Define multiple variables
#
# n: [int], represents number of files that have been read
# a: [int], represents the number of files that have been written
# path: [string], the path to the directory where FME writes out the xyz files
#                 that contain the clipped pointclouds (*)
# outpath: [string], the path to the directory where the results of the Python calculations
#                    will be located (*)
# normalvectorpath: [string], the path to the directory where the normalvectors have been written out
#                   before
#
# IMPORTANT: replace 'path' with the location of the folder 'OutputPointClouds'
#            replace 'outpath' with the location of the folder 'ResultPython'
#            replace 'normalvectorpath' with the location of the folder 'OutputNormal'
#=============================================================================================================================
h=i=j=k=0
path="C:/Users/Alexander/Desktop/Daten BA/OutputLocation/OutputPointClouds/"
outpath="C:/Users/Alexander/Desktop/Daten BA/OutputLocation/ResultPython/"
normalvectorpath="C:/Users/Alexander/Desktop/Daten BA/OutputLocation/OutputNormal/"
# Print-Section (limits the Python-Section within the Log-File)
print("===============\n| PYTHON SECTION INITIALIZED |\n===============\n. .\nReading from path: ",path)
# Start reading in the files in the given directory
# Inspired by: https://realpython.com/working-with-files-in-python/


with open(outpath+'PythonResult.csv','w',newline='') as outfile:
    k=k+1
    outfilewriter=csv.writer(outfile)
    outfilewriter.writerow(['ID','Orientation','Inclination'])
    #outfilewriter.writerow(['Orientation','Inclination'])
    with os.scandir(path) as entries:
        for entry in entries:
            with open(normalvectorpath+'normalvectors_UUID_Version.csv') as vecfile:
                reader=csv.reader(vecfile)
                # First, identify the current file and safe the Cloud identifier for later use
                file=entry.name
                i=i+1;

                if file !='.xyz':
                    j=j+1
                    dot_index=file.rfind('.')
                    id=file[:dot_index]
                    cloud=PyntCloud.from_file(path+file,sep=" ",header=0,names=["x","y","z"])
                    is_plane=cloud.add_scalar_field("plane_fit",max_dist=1e-3)
                    foundInliers=cloud.points.loc[cloud.points["is_plane"]==1]
                    #Points that have been identified by the RANSAC are written out to a file per Pointcloud
                    #foundInliers.to_csv(outpath+id+'.xyz',index=False, header=True)
                    points=np.c_[np.array(foundInliers['x']),np.array(foundInliers['y']),np.array(foundInliers['z'])]
                    p,n=best_fitting_plane(points)
                    #p,n, ev1,ev2,ev3=best_fitting_plane(points)
                    for row in reader:
                        if id in row[0]:

                            #outfilewriter.writerow([id,nv[0],nv[1],nv[2],n[0],n[1],n[2]])
                            #Format: X-component Normalvector Object = Y-component Normalvector Plane
                            #        Y-component Normalvector Object = Y-Component Normalvector Plane
                            #        Z-Component Normalvector Object = Z-Component Normalvector Plane
                            compa=[float(n[0]),float(n[1]),float(n[2])]
                            #compa1=[float(ev1[0]),float(ev1[1]),float(ev1[2])]
                            #compa2=[float(ev2[0]),float(ev2[1]),float(ev2[2])]
                            #compa3=[float(ev3[0]),float(ev3[1]),float(ev3[2])]
                            compb=[float(row[1]),float(row[2]),float(row[3])]
                            #ori=orientation(compa1,compa2,compa3,compb)
                            #ori=orientation(compa,compb)
                            #incl=inclination(compa,compb)
                            ori,incl=oriAndIncli(compa,compb)
                            outfilewriter.writerow([id,ori,incl])
                            #outfilewriter.writerow([ori,incl])
                else:
                    print('The following file was not read:')
                    print(file)
                    h=i
                    print(". .")

# ================
# End of the calculations.
```

```
# Final message
# ================
print(". .")
print("============================================================")
print('SUMMARY:')
print("Number of read files: ",i)
print("Number of lines written in CSV (apart from the header): ",j)
print("Number of files created: ",k)
print("============================================================")
print(". .")
print(". .")
print("==================")
print('| PYTHON SECTION CLOSED |')
print("==================")
#=============================================================================================================================

# Template Class Interface:
# When using this class, make sure its name is set as the value of
# the 'Class or Function to Process Features' transformer parameter
class FeatureProcessor(object):
# Magic happens HERE
# Create Pyntcloud-algorithm HERE and expose with setAttributes() created new Attributes
#(the result of the Pyntcloud process)

    def __init__(self):
        pass
    def input(self,feature):
        feature.getAttribute('_dataset')
        self.pyoutput(feature)
    def close(self):
        pass
```

MATLAB-Code

```
%{
  ============================================
  |                                          |
  |  MATLAB script to visualize the metrics  |
  |                                          |
  ============================================
%}
clc, clear, close all
%% 1. Read in data
%        first: specify the path to the folder 'OutputLocation'
%        make sure to change the path to the directory on your system
dataFolder='C:\Users\Alexander\Desktop\Daten BA\OutputLocation';
fprintf("\nProgram initiated.\n");
[d_dist,d_loc,w_dist,w_loc,m_intense,m_loc,orientation,inclination]=importData↙
(dataFolder);
%% 2. Visualize Distance Data
PlottingTheData(d_dist*100,'Door Distances',d_loc,'[cm]');
disp('Figure for Door-Distance Visualization completed.');
PlottingTheData(w_dist*100,'Wall Distances',w_loc,'[cm]');
disp('Figure for Wall-Distance Visualization completed.');
PlottingTheData(m_intense,'Road Marking Intensity',m_loc,' ');
%% 3. Visualize Angular Data
PlotAngularData(orientation,inclination);
disp('Figure for Inclination and Orientation inspection completed.');
%% 4. Statistics
sigma=3; %cm
quant=1.95996;%for 95%
cmddist=d_dist*100;
cmwdist=w_dist*100;
fprintf('Door PTS Distances:\n');
interval(cmddist,sigma,quant);
fprintf('Wall PTS Distances:\n');
interval(cmwdist,sigma,quant);
fprintf('Road Marking Intensity:\n');
interval(m_intense,1,quant);


```

```
function [dat1,loc1,dat2,loc2,dat3,loc3,dat4,dat5] = importData(thepath)
%Custom function to read in the data found in certain folders
%at the given location 'path'. If the path does not exist this
%function will inform you about that and cancel the run

%{
The following folders are expected to be found at the given path:
-OutputStatistics (subdirectories Door, Wall, Marking)
    -> Metrics PTS-Distance, Intensity
-ResultPython
    -> Metrics Orientation, Inclination

This function will return three vectors per metric
-METRICNAME: nx1 vector with each individual given value
-METRICLOCS: mx1 vector with the indices for METRICNAME that specify
 the beginning of a new dataset (to identify individual origin)
%}
%FileFormat: CSV
if ~isfolder(thepath)
    locateMessage = sprintf('Please specify the correct path to the Folder↙
"OutputLocation".\nYou may copy and paste the path into the following dialog↙
window.');
    uiwait(warndlg(locateMessage));
else
     fprintf("Folder found. \n%↙
s\n================================================================\n",thepath);
end
tic
dsf='\OutputStatistics\Doors';%DoorSubFolder
doorFiles=dir(fullfile(append(thepath,dsf),'*.csv'));
door_distance=[]; door_locs=[];
for dfc=1:length(doorFiles)
        dtemp=readtable(fullfile(doorFiles(dfc).folder,doorFiles(dfc).name));
        door_distance=[door_distance;dtemp.x_distance];
        door_locs=[door_locs;length(door_distance)];
end
wsf='\OutputStatistics\Walls';%WallSubFolder
wallFiles=dir(fullfile(append(thepath,wsf),'*.csv'));
wall_distance=[]; wall_locs=[];
for wfc=2:length(wallFiles)
    wtemp=readtable(fullfile(wallFiles(wfc).folder,wallFiles(wfc).name));
    wall_distance=[wall_distance;wtemp.x_distance];
    wall_locs=[wall_locs;length(wall_distance)];
end
msf='\OutputStatistics\Markings';%MarkingsSubFolder
markFiles=dir(fullfile(append(thepath,msf),'*.csv'));
mark_intensity=[]; mark_locs=[];
for mfc=1:length(markFiles)
    mtemp=readtable(fullfile(markFiles(mfc).folder,markFiles(mfc).name));
    mark_intensity=[mark_intensity;mtemp.intensity];
    mark_locs=[mark_locs;length(mark_intensity)];
end
dat1=door_distance; dat2=wall_distance; dat3=mark_intensity;
loc1=door_locs; loc2=wall_locs; loc3=mark_locs;
asf='\ResultPython';
```

```
53 angleFiles=dir(fullfile(append(thepath,asf),'*.csv'));
54 ori=[]; incl=[];
55 for afc=1:length(angleFiles)
56     atemp=readtable(fullfile(angleFiles(afc).folder,angleFiles(afc).name));
57     ori=[ori;atemp.Orientation];
58     incl=[incl;atemp.Inclination];
59 end
60 dat4=ori; dat5=incl;
61 clc
62 fprintf('====\nDATA IMPORT COMPLETED.\nFrom "%s": Read files: %d\nFrom "%s": Read
files: %d\nFrom "%s": Read files: %d\nFrom "%s": Read files: %d\n',append(thepath,
dsf),length(doorFiles),append(thepath,wsf),length(wallFiles),append(thepath,msf),
length(markFiles),append(thepath,asf),length(angleFiles))
63 toc
64 end
65
```

```
 1 function PlottingTheData(data,type,dloc,unit)
 2 %Function to visualize the datasets
 3 % Input: data: nx1 vector with the metric values
 4 %        type: Type of Dataset, e.g 'Distances Doors', 'Intensity Markings'
 5 %        dloc: mx1 vector with the dataset limits for 'data'
 6 %        unit: value units, e.g '[cm]'
 7 figure('Name',append('Visualizing: ',type))
 8 % 2.1.1 Histogram about entire dataset
 9 subplot(2,2,[1 2])
10 histogram((data),100,'FaceColor','#4DBEEE');
11 hold on; grid on; grid minor; xlabel(append('distance ',unit)); ylabel↙
('abundance');
12 dPOIs=[min(data) max(data); 1 1];
13 minlabel=append('MIN: ',num2str(min(data)),' ',unit);
14 maxlabel=append('MAX: ',num2str(max(data)),' ',unit);
15 meanlabel=append('MEAN: ',num2str(mean(data)),' ',unit);
16 medlabel=append('MEDIAN: ',num2str(median(data)),' ',unit);
17 scatter(dPOIs(1,:),dPOIs(2,:),'r','filled');
18 text(dPOIs(1,:)+1,[length(data)*0.05 length(data)*0.05],{minlabel, maxlabel});
19 xline(mean(data),'r--',meanlabel);
20 xline(median(data),'g:',medlabel,'LabelHorizontalAlignment','left');
21 title('Histogram');
22 % 2.1.2 Boxplot for entire dataset collection
23 subplot(2,2,[3 4])
24 title('Behaviour of all Datasets combined')
25 hold on;
26 bc=boxchart(data);
27 grid on; grid minor;
28 bc.JitterOutliers='on';
29 bc.MarkerStyle='.';
30 bc.Orientation='horizontal';
31 xlabel(append('distance ',unit));
32 % 2.1.3 Boxplot per dataset
33 figure('Name',append('Boxplot: An Overview [',type,']'))
34 g=[];
35 gdat=[];
36 if length(dloc) >50
37      rand=sort(randperm(length(dloc),50));
38      ddloc=rand;
39 else
40     ddloc=dloc;
41 end
42 for i= 1:length(ddloc)
43     if i==1
44         gline=repmat({num2str(i)},dloc(1),1);
45         gdatline=data(1:dloc(i));
46         g=[g;gline];
47         gdat=[gdat;gdatline];
48     else
49         holder=(dloc(i)-dloc(i-1));
50         gdatline=data(dloc(i-1)+1:dloc(i));
51         gline=repmat({num2str(i)},holder,1);
52         g=[g;gline];
53         gdat=[gdat;gdatline];
54     end
```

```
55 end
56 boxplot(gdat,g,'MedianStyle','target','OutlierSize',1,'Symbol','r.');
57 hold on; grid on; grid minor;
58 xlabel('Dataset Number (for aplhabetical file order)')
59 xtickangle(45)
60 ylabel(append('Value ',unit));
61 title('Individual Dataset Overview');
62 clear dPOIs g gline holder i maxlabel meanlabel medlabel minlabel
63 end
64
65
```

```
 1 function PlotAngularData(orientation,inclination)
 2 %Visualize the Orientation and Inclination of the given dataset
 3 figure('Name','Inclination and Orientation: Overview')
 4 subplot(4,4,[1 8])
 5 stem(orientation,inclination,'b.'); hold on; grid on; grid minor;
 6 axis equal; xlabel('Orientation °'); ylabel('Inclination °');
 7 title('Orientation-Inclination Diagram')
 8 subplot(4,4,[9 10])
 9 histogram(orientation,100);
10 hold on; grid on; grid minor;
11 title('Orientation');
12 xlabel('Orientation °');
13 ylabel('Abundance');
14 subplot(4,4,[11 12])
15 histogram(inclination,100);
16 hold on; grid on; grid minor;
17 title('Inclination');
18 xlabel('Inclination °');
19 ylabel('Abundance');
20 subplot(4,4,[13 14])
21 bo=boxchart(orientation);
22 title('Orientation: Boxchart');
23 hold on; grid on; grid minor;
24 bo.JitterOutliers='on';
25 bo.MarkerStyle='.';
26 bo.Orientation='horizontal';
27 xlabel('Orientation °');
28 subplot(4,4,[15 16])
29 bi=boxchart(inclination);
30 title('Inclination: Boxchart');
31 hold on; grid on; grid minor;
32 bi.JitterOutliers='on';
33 bi.MarkerStyle='.';
34 bi.Orientation='horizontal';
35 xlabel('Inclination °');
36 disp('End of angular visualization');
37 end
38
39
```

```
 1 function interval(data,sigma,quant)
 2 % Function to calculate the lower and upper boundary of the confidence
 3 % interval of a set of normal distributed data if sigma is known
 4 % data: nx1 vector containing the data for the boundary calculation
 5 % sigma: standard deviation
 6 % quant: quantil for the intended certainty
 7 n=length(data);
 8 lEnd=mean(data)-quant*(sigma/sqrt(n));
 9 hEnd=mean(data)+quant*(sigma/sqrt(n));
10 fprintf("Calculated Boundaries:\n [ %f | %f ]\n",lEnd, hEnd);
11 end
12
13
```