

Bachelorarbeit

Entwicklung eines Tools zur manuellen Klassifikation von 3D Punktwolken auf Basis wissenschaftlicher Software

Felix Meixner

`felix.meixner@tuwien.ac.at`

Matr.Nr. 12027400

Datum: 7. Mai 2024

BetreuerInnen:

Mandlbürger, Gottfried; Univ.Prof. Dipl.-Ing. Dr.techn.
Otepka-Schremmer, Johannes; Senior Scientist Dipl.-Ing. Dr.techn.

Kurzfassung

In dieser Bachelorarbeit wird die Forschungsfrage beantwortet, ob es möglich ist eine einfache Computer Applikation (Tool) für die manuelle Klassifikation komplexer Szenen von Airborne Laser Scanning (ALS)-Punktwolken zu entwickeln. Aktuelle Tools haben nicht die Möglichkeit, nur Abschnitte der Punktwolke als vertikale Schnitte darzustellen. Das resultiert darin, dass für bathymetrische Anwendungen eine eindeutige Zuweisung der Punkte in Wasseroberfläche, Wassersäule und Gewässerboden nur schwer möglich ist, da die einzelnen Punkte voneinander verdeckt werden. Dieses Tool wird in der Programmiersprache Python mit dem Paket PyQt5 entwickelt. Die ALS-Datenverarbeitung erfolgt mit der Software Orientation and Processing of Airborne Laser Scanning (OPALS), welche auf der TU Wien entwickelt wurde. Genauer gesagt übernimmt der OPALS Data Manager (ODM) in Form des Python API pyODM das Datenmanagement. Mit dem Tool können Abschnitte (Sections) der Punktwolke entlang einer Achse als vertikale Schnitte dargestellt werden. Die Achse muss zuvor definiert und als Shapefile gespeichert werden. Die Punkte eines Abschnitts werden mit Hilfe eines Polygons, welches in der Größe variabel einstellbar ist aus dem ODM gelesen und dargestellt. Die Punkte können durch Markieren oder Anklicken klassifiziert werden. Die Darstellung der Section kann über integrierte Funktionen angepasst werden. Wenn eine Section klassifiziert wurde, kann das Polygon in Längsrichtung der Achse verschoben werden. Es existiert die Möglichkeit, die Klassen der nächsten Section über k Nearest Neighbours (kNN) vorherzusagen. Das Tool selbst lässt Möglichkeiten für zukünftige Erweiterungen oder Verbesserungen offen.

Inhaltsverzeichnis

1	Einleitung	4
2	Stand der Technik	5
2.1	Airborne Laser Scanning	5
2.2	Klassifizieren von Punktwolken	6
2.2.1	Überwachte Klassifikation	7
3	OPALS - Orientation and Processing of Airborne Laser Scanning	8
3.1	Module	8
3.2	ODM - Opals Data Manager	8
3.2.1	Räumliches Konzept	9
3.2.2	Attribute	10
3.3	pyDM Interface	10
4	Pre-Processing	11
4.1	Auswahl Entwicklungsumgebung	11
4.2	Erstellen einer Schummerung	11
4.3	Definition der Bezugsachse	11
5	Methoden	12
5.1	Ablaufdiagramm	12
5.2	Einlesen der Airborne Laser Scanning (ALS)-Daten und der Bezugsachse	13
5.2.1	Darstellung der Schummerung und der Bezugschse	13
5.2.2	Graphische Darstellung der aktuellen Section in der Übersichtsgrafik . . .	14
5.3	Teilmenge der Punktwolke aus dem OPALS Data Manager (ODM) laden	14
5.3.1	Erstellen der Abschnittsbegrenzung (Polygon)	15
5.3.2	Veränderung der Geometrie der Abschnittsbegrenzung	16
5.3.3	Teilmenge der Punktwolke und ihre Attribute	16
5.4	Darstellung der Punkte	17
5.4.1	Einfärbung der Punkte	18
5.4.2	Änderung der Ansicht	19
5.4.3	Variation der Punktgröße	20
5.4.4	Statusmessages	20
5.5	Klassifizieren der Punktwolke	21
5.5.1	Dynamischer Picker	21
5.5.2	Markieren von mehreren Punkten	21
5.5.3	Klassifizieren	22
5.6	Speichern und Reset der Klassifizierung	23
5.7	Navigation entlang der Bezugsachse	23
5.7.1	Wechseln in die nächste Section	24
5.7.2	Wechseln in die vorherige Section	25
5.7.3	Vorhersage der Klassen in der nächsten Section mittels kNN	25
6	Ergebnisse	26
6.1	Klassifizieren einer Punktwolke	26
6.1.1	Einlesen der ALS-Daten und der Bezugsachse	26
6.1.2	Laden und Darstellen der ersten Section aus dem ODM	27
6.1.3	Variation der Punktgröße	28
6.1.4	Einstellung von <i>Across</i> , <i>Along</i> und <i>Overlap</i>	29
6.1.5	Klassifizieren	29

6.1.6	Einfärbung der Punkte	30
6.1.7	Speichern und Reset der klassifizierten Punkte	31
6.1.8	Navigation entlang der Bezugsachse	31
6.1.9	Vorhersage mit kNN	32
6.1.10	Verschiedene Darstellungen	33
6.1.11	Endversion des Klassifikationstool	35
7	Diskussion	36
7.1	Eignung des Klassifikationstools	36
7.1.1	Klassifiziertvorgang	36
7.1.2	Verschieben entlang der Achse	37
7.2	Ausblick	37
	Abbildungsverzeichnis	39
	Tabellenverzeichnis	39
	Abkürzungen	40
	Literatur	40
	Anhang A - GitHub-Link zu Pythonskript	41

1 Einleitung

In den vergangenen Jahren ist der Einfluss der Menschheit auf unsere Umwelt stetig gestiegen. Um rasche Informationen über diesen Einfluss auf die Umwelt zu erhalten, können Fernerkundungsmethoden verwendet werden. Airborne Laser Scanning (ALS) ist ein Instrument, welches in Kombination mit Bildgebungssystemen Informationen über die Erdoberfläche liefert [5]. Für großflächige Erfassung von digitalen Geländemodellen ist in den vergangenen Jahren die Technologie ALS herangereift [7], sodass es zu einem der Standardverfahren für die topographische Geländeerfassung wurde [21]. Die Daten können für verschiedene Anwendungsbereiche verwendet werden, wie zum Beispiel zur Umweltüberwachung, topographischen Geländemodellierung, Vegetationshöhenbestimmung, zur Bestimmung von Überschwemmungsgebieten und viele mehr [5]. Um diese ALS Daten zu analysieren und um Berechnungen durchzuführen, wird zum einen eine Software benötigt, die die ALS-Daten verwaltet, zum anderen ist es nötig, die Punkte zur besseren Interpretation in Klassen einzuteilen. Es können Klassen wie Boden, Vegetation, Gebäude, Wasser und viele mehr vergeben werden. ALS hat die Eigenschaft, dass die Technologie mehrzielfähig ist. Das gilt für Laserscanner zur Topographieerfassung, die mit Wellenlängen im infraroten Bereich des elektromagnetischen Spektrums arbeiten und Reflexionen von der Vegetation und dem Boden liefern, genauso wie für bathymetrische Scannersysteme, die zusätzlich auch Wassersäulen durchdringen können. In der Laserbathymetrie ist eine Wassersäule in drei Teile unterteilt: Wasseroberfläche, Wassersäule und Gewässerboden. Wird nun die Punktwolke dreidimensional dargestellt, werden die Punkte des Gewässerbodens von jenen der Wasseroberfläche und der Wassersäule überdeckt. Aus diesem Grund ist eine Reduktion der Dimensionen der Punktwolke von 3D auf 2D notwendig. Die Darstellung der Punkte, welche auf 2D reduziert wurden, geschieht im Aufriss orthogonal zu einer vordefinierten Bezugsachse (zumeist eine Flussachse), welche als geometrischer Type eine Polylinie ist. Die aktuell zur Verfügung stehenden Tools wie Pointviewer¹ und CloudCompare² können eine solche Reduktion der Dimensionen nicht ausreichend gewährleisten.

In dieser Bachelorarbeit wird für die Verarbeitung der ALS-Daten die Software Orientation and Processing of Airborne Laser Scanning (OPALS) verwendet. OPALS ist eine Software für die Verarbeitung von sehr großen ALS Daten, welche am Forschungsbereich Photogrammetrie des Departments für Geodäsie und Geoinformation der TU Wien entwickelt wurde [14]. Die Datenverarbeitung von OPALS erfolgt entweder mit Modulen oder wie in dieser Bachelorarbeit über das Python API (pyODM). Das Datenmanagement der ALS-Daten übernimmt der OPALS Data Manager (ODM) [9], für die Organisation der Module steht ein einheitliches Software Framework zur Verfügung. [15]. Im Rahmen dieser Bachelorarbeit wird für die manuelle Klassifikation (manual labelling) der Punktwolke mittels PyQt [20], eine graphische Benutzeroberfläche (eng.: Graphical User Interface (GUI)), in Python erstellt. Die manuell klassifizierten Punkte können dann die Grundlage von automatischen Algorithmen bilden, die über Ansätze der künstlichen Intelligenz bzw. des tiefen Lernens (Deep Learning, DL) auf große Punktwolkendatensätze angewendet werden können. Die Hauptfunktion des Tools beruht auf dem pyODM. Mit diesem Klassifikationstool soll es möglich sein, die gesamte Punktwolke zu klassifizieren, indem schrittweise kleine übersichtliche Abschnitte aus dem ODM geladen werden. Welcher Abschnitt und wie viele Punkte es sind, wird mittels Polygonen, die orthogonal entlang der Bezugsachse aufgespannt werden, bestimmt.

Mit dieser Bachelorarbeit wird die Forschungsfrage beantwortet, ob es möglich ist, ein einfaches Tool zur manuellen Klassifikation von komplexen 3D Punktwolken aus Laserbathymetrie zu implementieren. Mit dem Tool soll es möglich sein, ALS-Daten im Aufriss entlang einer Achse

¹Pointviewer: https://pointcab-software.com/de/pointcab-share_kostenloser-punktwolken-viewer/
(Zugriff am 14.02.2024)

²CloudCompare: <https://www.cloudcompare.org/main.html> (Zugriff am 14.02.2024)

(zumeist eine Flussachse) darstellen. Die einzelnen Punkte sollen durch Anklicken einer Klasse (zum Beispiel Wasseroberfläche, Wassersäule, Boden) zugeordnet werden.

2 Stand der Technik

Dieses Kapitel fasst den aktuellen Stand der Technik für ALS zusammen, genauer welche technischen Möglichkeiten existieren, um für bathymetrische Anwendungen eine Unterscheidung zwischen Wasseroberfläche, Wassersäule und Gewässerboden zu erhalten. Zudem werden die aktuellen Möglichkeiten, um diese Daten manuell zu klassifizieren, aufgezählt.

2.1 Airborne Laser Scanning

Airborne Laser Scanning (ALS), ist eine aktive Fernerkundungstechnik, welche Infrarotimpulse über einen Laser sendet. Die ausgesandten Impulse werden von der Erdoberfläche zurückgestreut. Eine Fotodiode zeichnet das rückgestreute Echo auf. Gemessen wird die Zeit, die vom Aussenden bis zur Detektion eines Impulses vergeht. Daraus kann die geometrische Struktur der Erdoberfläche abgeleitet werden [22]. Das Prinzip von ALS wird in Abbildung 1 graphisch dargestellt.

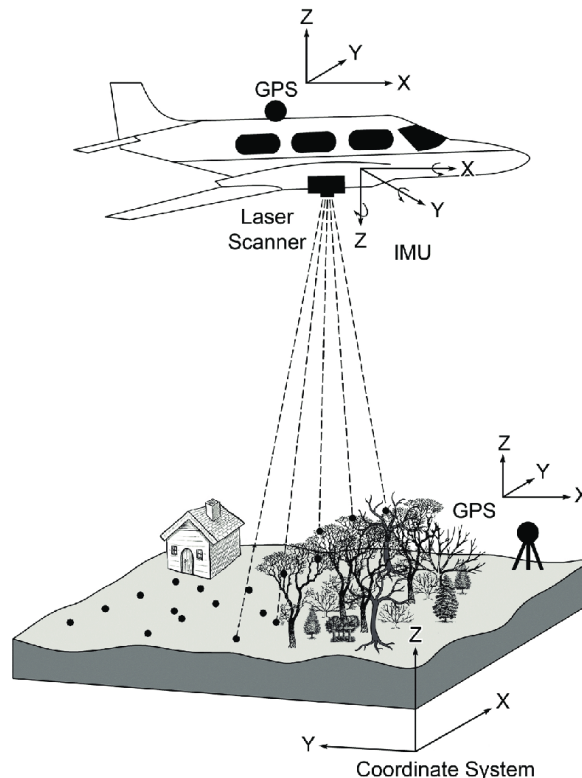


Abb. 1: ALS - System [17]

Für bathymetrische Anwendungen im Flachwasserbereich, wird heutzutage zum größten Teil nur noch auf ALS gesetzt. Die Laser, welche in der Bathymetrie zum Einsatz kommen, operieren im grünen Bereich des elektromagnetischen Spektrums, da nur in diesem Bereich Wasser durchdringbar ist. In der Bathymetrie ist eine flächendeckende Datenerfassung und eine sehr gute Höhengengenauigkeit für die Darstellung von Detailstrukturen mit ALS möglich. Die ausgesandten Laserimpulse sind in der Lage, die Vegetationsdecke zu durchdringen. Diese Eigenschaft ermöglicht eine Detektion des Bodens [3].

Das Messergebnis einer ALS-Messung ist eine Punktwolke. Um diese Punktwolke weiter zu verarbeiten, ist es notwendig, einige Prozessierungsschritte anzuwenden. In diesen Schritten wird

eine Qualitätskontrolle der Messung durchgeführt, es erfolgt eine erste Klassifizierung in Boden-, Wasser- und Nicht-Bodenpunkte und einige mehr [8].

Der Laserscanner wird für ALS entweder auf einem Flugzeug oder einem Helikopter befestigt, wie in Abbildung 1 zu sehen ist. Für ALS wird neben dem Laserscanner noch ein Global Navigation Satellite System (GNSS) und eine Integrated Measurement Unit (IMU) benötigt. Das GNSS misst die absolute Position des Flugobjektes auf der Erde. Die IMU ist für die Messung der Raumstellung des Messsystems notwendig. Moderne Sensoren, sogenannte Full Wave Laserscanner, messen das gesamte Echo des ausgesandten Impulses. Damit ist es möglich, Attribute wie Echoweite und Amplitude abzuleiten. Das hat positive Auswirkungen auf die Qualität der Daten [8].

Für bathymetrische Messungen liefern herkömmliche ALS-Systeme nicht ausreichend Informationen. Dieses Systeme operieren im nahen Infrarotbereich mit einer Wellenlänge von etwa 800 bis 1500nm. Nahes Infrarot kann die Wasseroberfläche nicht durchdringen, es kommt zu einer spiegelnden Reflexion und einer Absorption des Signals. Somit liefert Infrarot keine Information über die Wassertiefe und folglich auch nicht über den Gewässerboden [21]. Abhilfe wird geschaffen, in dem mit zwei verschiedenen Lasern gemessen wird. Zum einen wird die Wasseroberfläche mit einem Laser im nahen Infrarotbereich detektiert. Zum anderen wird die Gewassertiefe und der Gewässerboden werden mittels grünen Lasers, welcher eine kurze Wellenlänge hat, gemessen. Grünes Licht hat die Eigenschaft, Wasser zu durchdringen [10]. Das Messprinzip der Airborne LIDAR Bathymetry (ALB) ist in Abbildung 2 dargestellt.

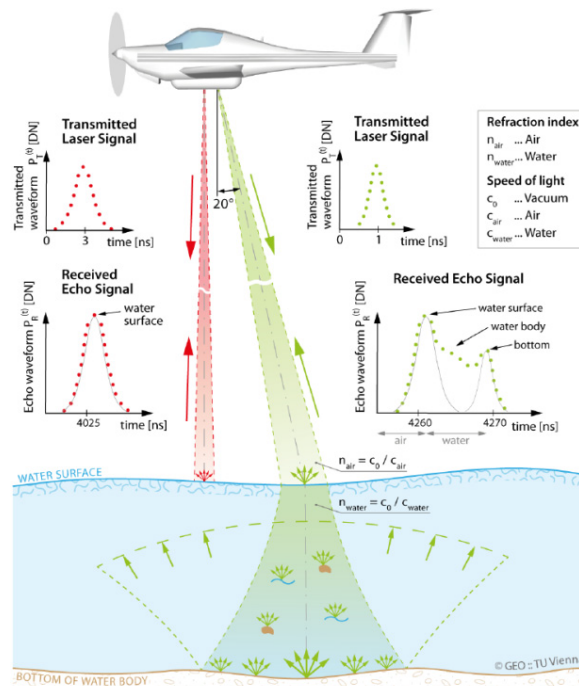


Abb. 2: Prinzipskizze ALS/ALB [10]

2.2 Klassifizieren von Punktwolken

Nach der ALS-Messung und der Durchführung der Qualitätskontrollen der Punktwolke ist der nächste Schritt die Klassifikation der einzelnen Punkte. Dabei stehen mehrere Möglichkeiten zur Verfügung. Zum einen existieren Möglichkeiten zur manuellen Klassifikation, zum anderen für die maschinengestützte Klassifikation. Diese Bachelorarbeit fokussiert sich zum größten Teil auf die manuelle Klassifikation. Es besteht auch die Möglichkeit einer maschinelle Vorhersage der Klassen

mittels k Nearest Neighbours (kNN). Im Falle des manuellen Klassifizierens von Punktwolken wird ein Punkt oder mehrere Punkte selektiert. Für die selektierten Punkte wird anschließend eine Klasse ausgewählt, welche den Punkten zugeteilt wird. Für das manuelle Klassifizieren von Punktwolken stellen Anbieter wie Cloudcompare oder Pointviewer entsprechende Tools zur Verfügung. Diese beiden Tools stellen die gesamte Punktwolke in 3D dar. Wenn die gesamte Punktwolke dargestellt wird, kann mit diesen beiden Tools eine Unterscheidung zwischen Wasseroberfläche, Wasserkörper und Gewässerboden in komplexen Situationen nur sehr schwierig erfolgen. Aus diesem Grund wird in dieser Bachelorarbeit ein Tool für die manuelle Klassifikation von ALS-Punktwolken entwickelt. In Tabelle 1 sind die ASPRS Standards LIDAR Klassen angeführt. Die Klassen 40 bis 45 sind gemäß Lewis Graham [6] benutzerdefiniert. Der Farbcode einer Klassen entspricht jenem, welcher im Modul View von OPALS verwendet wird [13].

Im Bereich der maschinellen Klassifikation kann zwischen überwachter Klassifikation und nicht-überwachter Klassifikation unterschieden werden. Da in dieser Bachelorarbeit ein Teil der Klassifikation mit kNN erfolgt und dieses Verfahren der überwachten Klassifikation zuzuordnen ist, wird in diesem Kapitel auf das überwachte Klassifizieren eingegangen.

Tab. 1: Standard LIDAR Punkt Klassen [6][S.98]

Wert der Klasse	Farbcode (r, g, b)	Beschreibung
0	210, 210, 210	unclassified
1	180, 180, 180	undefined
2	135, 70, 10	ground
3	185, 230, 120	low vegetation
4	145, 200, 0	medium vegetation
5	72, 128, 0	high vegetation
6	180, 20, 20	building
7	255, 255, 220	noise
8	220, 105, 20	model key point
9	0, 95, 255	water
10	100, 80, 60	rail
11	70, 70, 70	road surface
12	35, 35, 35	bridge deck
13	255, 250, 90	wire guard
14	255, 220, 0	wire conductor
15	235, 200, 60	transmission tower
16	190, 160, 50	wire connector
40	180, 180, 95	bathymetric point (e.g. seafloor or riverbed)
41	35, 0, 250	water surface
42	40, 220, 240	derived water surface
43	140, 80, 160	underwater object
44	90, 75, 170	IHO S-57 object
45	60, 130, 130	volume backscatter

2.2.1 Überwachte Klassifikation

In der überwachten Klassifikation, wie zum Beispiel kNN, liegen sogenannte Trainingsdaten vor. Diese Trainingsdaten sind bereits den richtigen Klassen zugeordnet, und dienen dem Klassifikator als Grundlage für die Parameter des Klassifizierers. Der Klassifikator wird anschließend auf Testdaten, welche in mehrere Klassen (Tabelle 1) eingeteilt sind, getestet. Der Testdatensatz ist dem Klassifikator vollkommen unbekannt. Die Merkmale des Testdatensatz müssen in deren

Eigenschaften gleich jenen der Trainingsdaten sein, um eine ausreichend gute Klassifikation durchzuführen [1]. Mit diesem Tool werden kleine Abschnitte aus der gesamten Punktwolke nacheinander geladen und klassifiziert. Die Trainingsdaten werden durch die aktuell klassifizierte Section repräsentiert und werden auf die nächste Section auf alle unklassifizierten Punkte angewandt. Das Testen wird von der jeweiligen Benutzer:in des Tools inhärent vorgenommen, indem das Ergebnis bewertet wird und gegebenenfalls werden einzelne Punkte umklassifiziert.

3 OPALS - Orientation and Processing of Airborne Laser Scanning

OPALS ist eine Software für die Verarbeitung von sehr großen ALS Daten, welche am Forschungsbereich Photogrammetrie des Departments für Geodäsie und Geoinformation der TU Wien entwickelt wurde [14]. Dafür wurden mehrere Hauptziele festgelegt. Eines dieser Hauptziele war, dass OPALS eine vollständige Bearbeitungskette von den Ausgangsdaten zu verschiedenen Produkten beinhaltet. Darüber hinaus sollte speziell für sehr große Datenmengen ein automatischer Arbeitsablauf implementiert werden, sowie die regelmäßige Erweiterung von Modulen der neuesten Forschungsergebnisse. Das letzte Hauptziel ist es, dass OPALS eine nachhaltige Plattform für wissenschaftliche Forschung bietet.

Um diese Ziele zu erreichen, wurde auf ein modulbasiertes Design gesetzt. Durch die Kombination von Modulen können benutzerdefinierte Prozessierungsketten realisiert werden. Die Module können entweder über die Programmiersprache Python, die Kommandozeile oder über das C++ API ausgeführt werden. Darüber hinaus kann über verschiedenste Skripte eine individuelle Prozesssteuerung erfolgen. Das Datenmanagement und die Administration der ALS-Daten und der Module übernimmt der ODM. Die Interaktivität wurde so weit als möglich reduziert [9], [15].

3.1 Module

Wie in Kapitel 3 erwähnt, beruht ein Großteil des Arbeitsablaufs von OPALS auf Modulen. Jedes Modul benötigt Eingangsdaten, diese sind notwendig, um die jeweiligen Algorithmen durchzuführen. Ebenso wie ein Modul Eingangsdaten übernimmt, gibt ein Modul auch berechnete Daten zurück. Je nach verwendetem Modul, können noch zusätzliche Parameter wie zum Beispiel gridSize, cellSize usw. übergeben werden [9], [15].

In Tabelle 2 werden die Module aufgelistet, die in dieser Bachelorarbeit über das Python API verwendet wurden.

Tab. 2: Angewendete OPALS Module [9]

Modulname	Beschreibung
Import	importiert die Punktwolkendaten in den ODM
Grid	basierend auf den Daten erfolgt eine Rasterinterpolation
Shade	Erstellung von getönten Rasterkarten

3.2 ODM - Opals Data Manager

Die Grundlagen des ODM sind in [14] beschrieben und werden im Folgenden kurz zusammengefasst.

Bei der Umsetzung von ALS-Projekten müssen sehr große Datenmengen analysiert und verarbeitet werden. Die Verarbeitung der Daten wird zumeist durch den jeweiligen Speicher limitiert. Aus diesem Grund wird der jeweilige Datensatz in mehrere kleine Datensätze geteilt. Die Anwendungen werden anschließend auf alle Datensätze angewandt, unter Berücksichtigung von einer gewissen

Überlappung zwischen den geteilten Daten. Im Falle der Verwendung eines topographischen Datensatzes ist die horizontale Ausdehnung deutlich größer als die vertikale. Der ODM kann eine oder mehrere Eingangsdateien übernehmen. Im Falle von mehreren Dateien werden diese in einen ODM importiert [14],[S.154].

3.2.1 Räumliches Konzept

Um die maximale Leistung des ODM auszunutzen, können räumlichen Suchanfragen von Punktdaten und von Polygondaten nicht in einem einzigen räumlichen Index dargestellt werden. In typischen ALS Projekten beinhalten die Datensätze mehr Punktdaten als Linien oder Polygone. Aus diesem Grund wendet der ODM verschiedene räumliche Indizes für Punkte und Polygone an [14][S.155].

Zuerst werden Punktdaten in Quadrate eingeteilt. Die Gesamtheit dieser Quadrate ergibt ein regelmäßiges Raster. Das Raster wird als Level 0 bezeichnet. Der ODM hält nur eine gewisse Anzahl an Kacheln im Speicher, welche für die jeweiligen Berechnungen benötigt werden. Eine graphische Veranschaulichung dieses Vorganges ist in Abbildung 3 dargestellt.

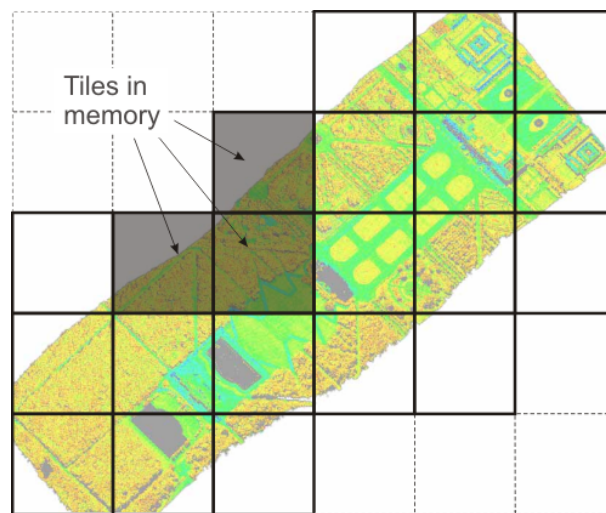


Abb. 3: Level 0 des ODM [13]

Der Rest der Daten bleibt im Hintergrund im ODM gespeichert. Für die Weiterverarbeitung der einzelnen Quadrate werden den Punkten innerhalb eines Quadrates mittels eines K-D-Baumes (Abbildung 4) ein eindeutiger Index zugewiesen. Für die Raumaufteilung verwendet ein K-D-Baum Hyperebenen³, welche an den Koordinatenachsen ausgerichtet sind [14][S.155].

³Hyperebene: „Ebene vom Anschauungsraum auf Räume beliebiger Dimension“ [4]

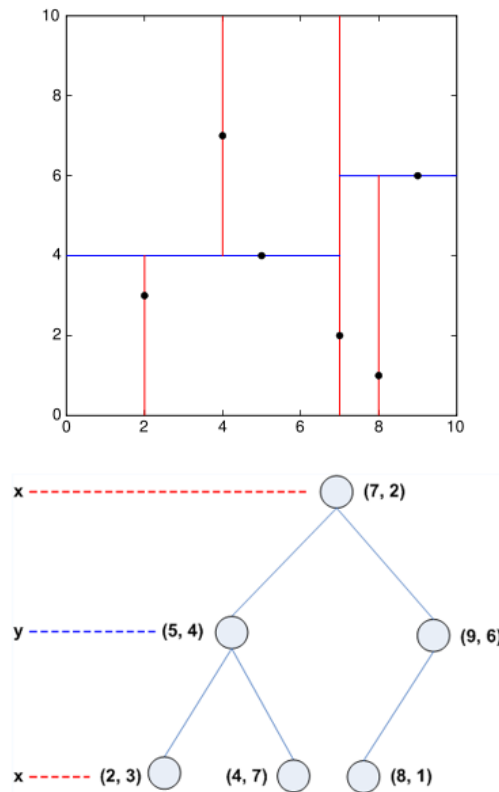


Abb. 4: KD-Baum [13]

Allen Daten, die nicht einem Punkt entsprechen, teilt der ODM einen zweiten unabhängigen Index zu. Zu diesen Daten zählen Oberflächenlinien, Flüsse und zwei-dimensionale Polygone (z.B. Gebäudeumrisse oder Vegetationsgrenzen) [14][S.155,156].

3.2.2 Attribute

Attribute spielen eine entscheidende Rolle für die Datenverarbeitung des ODM. Dabei wird zwischen zwei Arten unterschieden:

- Vordefinierte Attribute: Diese haben einen definierten Namen und Datentyp, sowie eine definierte Semantik. Dazu gehören zum Beispiel `opalsNormals` [14][S.156].
- Benutzerdefinierten Attribute: Diese besitzen keine Semantik und können entweder aus den Daten selbst stammen, oder mittels `opalsAddInfo` bestimmt werden. Attribute können während eines Prozesses beliebig verändert werden [14][S.156].

3.3 pyDM Interface

Das pyDM-Interface ist eine Einbindung der C++ ODM Bibliothek in die Programmiersprache Python. Das pyDM-Interface ist der C++ Bibliothek fast ident, und hat die gleichen grundlegenden Eigenschaften:

- Große geometrische Daten verwalten und speichern
- Attribute eines Objekts beliebig erweiterbar
- verschiedene räumliche Anordnung durch verschieden räumliche Indizes

- geometrische Objekte für Teilselektion filtern
- Effiziente Datenmanipulation

Das pyDM-Interface bietet Zugang zu den Objekten im ODM auf einer niedrigen Ebene. Weiters können Manipulationen oder die Verarbeitung von ODMs ähnlich wie in den OPALS-Modulen erfolgen. Mit dem pyDM können neue Prozesse auf kleinen Datensätzen sehr gut getestet werden. Für größere Datensätze steht das C++ API zur Verfügung, da dieses das Drei- bis Zehnfache schneller ist als Prozesse in Python. Im pyDM werden bestimmte Operationen für das Abrufen von Punkten oder Attributen mit NumPy Arrays [12] durchgeführt. NumPy erhöht die Performance von Python. Da Attribute verschiedenste Datentypen haben können, arbeitet der pyDM mit Wörterbüchern aus eindimensionalen NumPy Arrays [13].

4 Pre-Processing

4.1 Auswahl Entwicklungsumgebung

Für die Entwicklung eines Tools, dass das Klassifizieren von ALS-Daten ermöglicht, ist die Auswahl der Entwicklungsumgebung von entscheidender Bedeutung. Für diese Anwendung kam zum einen Plotly [16] und zum anderen PyQt [20] infrage. Plotly ist ein Python Modul, mit dem interaktive Plots erstellt werden können. Nach längerer Recherche konnten allerdings keine Möglichkeiten gefunden, die das Klassifizieren von Punkten ermöglicht. Aus diesem Grund fiel die Wahl für die Entwicklungsumgebung auf PyQt. Mittels PyQt lässt sich ein GUI erstellen, welches für die jeweilige Anwendung angepasst werden kann, so auch für die Klassifikation von Punktwolken. Die genaue Vorgangsweise wird im Kapitel 5 beschrieben. Die finale Version des GUI und wie dieses auf eine Punktwolke anzuwenden ist, wird in Kapitel 6 beschreiben.

4.2 Erstellen einer Schummerung

Eine Schummerung (eng.: Shading) ist eine sehr anschauliche Datenrepräsentation, um eine Punktwolke graphisch zu veranschaulichen. Dazu werden die OPALS-Module Import, Grid und Shade angewandt. Dass das Modul Import importiert die ALS-Punkte in den ODM. Anschließend wird eine Rasterinterpolation mittels des Grid Moduls mittels MovingPlanes Interpolation durchgeführt. Schlussendlich wird das Shading mit dem Shade Modul erzeugt [13].

4.3 Definition der Bezugsachse

Mit Hilfe des Shadings kann eine Bezugsachse, für bathymetrische Anwendungen zum Beispiel entlang einer Flussachse, definiert werden. Dazu ist es nötig, das Shading in einem GIS-Programm, wie zum Beispiel QGIS ⁴ zu öffnen. Nun kann die Bezugsachse, welche im späteren Verlauf genutzt wird, um zwischen einzelnen Querschnitten (Section) zu navigieren, eingezeichnet werden. Für die Anwendbarkeit der Bezugsachse im Klassifikationstool wird diese als Shapefile ⁵ abgespeichert.

⁴<https://www.qgis.org/de/site/> (Zugriff am 20.02.2024)

⁵Shapefile: „Ein Shapefile ist ein Vektordatenspeicherformat von Esri zum Speichern der Position, der Form und der Attribute von geographischen Features.“ [18]

5 Methoden

In Kapitel 5 werden der Aufbau und die Funktionsweise des GUI-basierten Klassifikationstools erläutert. Nach einer Vorstellung des Ablaufdiagramms sowie einer kurzen Erläuterung der wesentlichsten GUI Elemente in Kapitel 5.1, folgt in den Unterkapiteln 5.2 - 5.7 eine detaillierte Beschreibung der Prozesse, die für eine abschnittsweise manuelle Klassifikation einer Punktwolke erforderlich sind.

5.1 Ablaufdiagramm

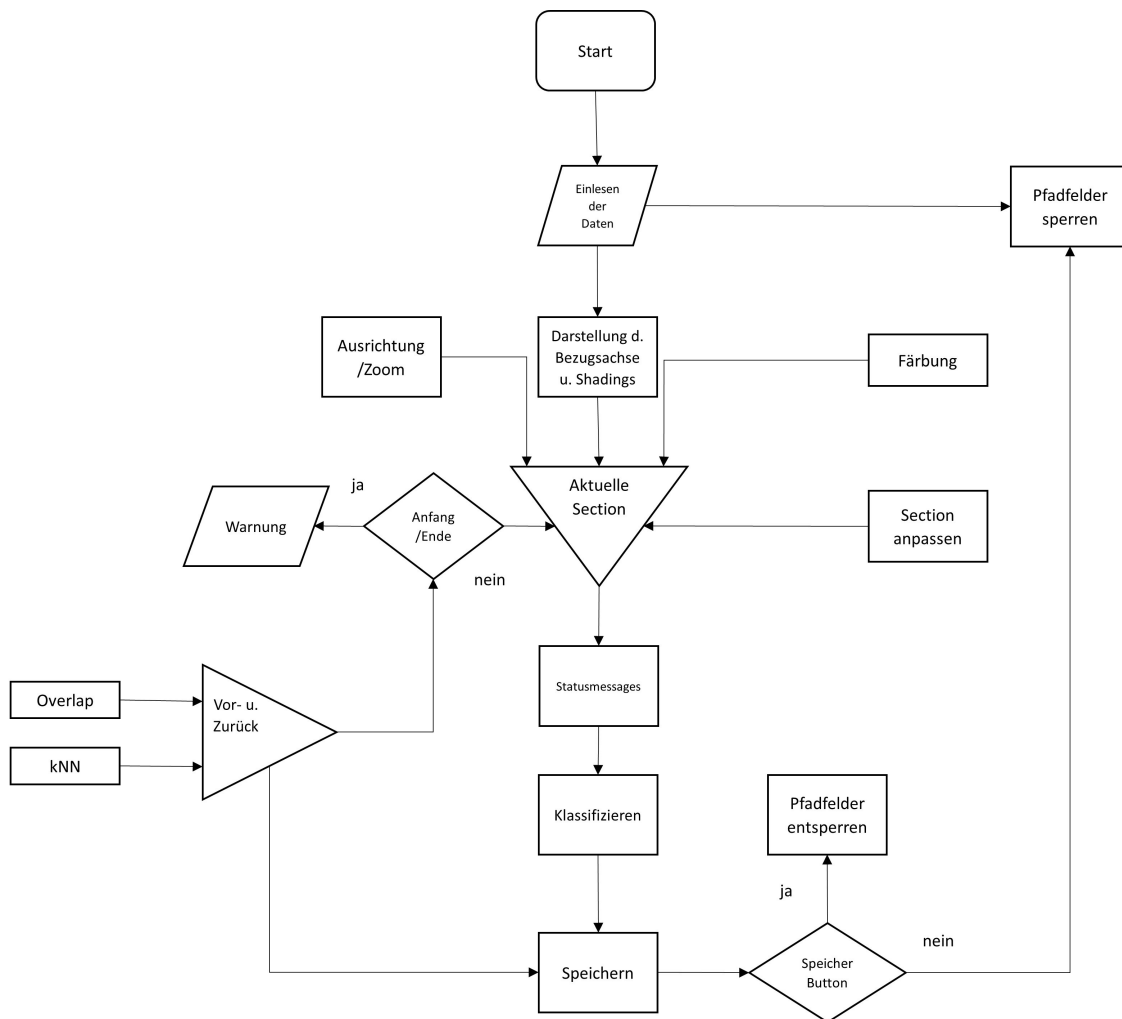


Abb. 5: Ablaufdiagramm

Nach dem das GUI gestartet und die Daten eingelesen wurden, werden in dem Fenster, das geöffnet wurde, zwei Grafikbereiche sowie Steuerelemente angezeigt. Im ersten Grafikbereich, welcher der Hauptgrafikbereich ist, wird die aktuelle Section als Aufriss dargestellt. In diesem Hauptgrafikbereich können die Punkte manuell klassifiziert werden. Im zweiten Grafikbereich wird die gesamte Punktwolke sowie die Bezugsachse und die aktuelle Section als Grundriss angezeigt. Zusätzlich befinden sich im GUI noch Steuerelemente. Unter dem Hauptgrafikbereich befindet sich ein Drop-Down-Menü zum Auswählen der gewünschten Klasse. Links neben der Klassenauswahl befindet sich jeweils ein Knopf um die nächste oder vorherige Section anzuzeigen. Unterhalb

dieser beiden Knöpfe befinden sich Eingabefelder. Mit diesen Eingabefeldern kann die Geometrie der Section und die Überlappung der Sections angepasst werden.

Unter dem Drop-Down-Menü für die Auswahl der Klassen befindet sich ein weiteres Drop-Down-Menü, mit welchem ausgewählt werden kann, ob eine Vorhersage der vorherigen oder nächsten Section erwünscht ist. Des weiteren existieren rechts neben den beiden Drop-Down Menüs zwei Knöpfe, um zwischen zwei Farbdarstellungen der aktuellen Section zu wechseln, und zwei Knöpfe, um Punkte zu klassifizieren. Dabei kann ebenfalls zwischen zwei Varianten gewählt werden. Die erste Variante verfügt über eine Pinsel-Funktion, mit der zweiten Variante kann ein Bereich, welcher zu klassifiziert ist, aufgezoogen werden. Unterhalb der beiden Knöpfe für die Farbdarstellung befindet sich ein weitere Knopf, mit welchem die aktuelle Section wieder orthogonal auf die Bezugsachse ausgerichtet werden kann.

Unterhalb des Grundrissgrafikbereiches befinden sich Textfelder in die die Dateipfad zu dem ALS-Daten und zu der Datei der Bezugsachse einzugeben ist. Unter den Textfeldern befindet sich ein Informationsfeld, in dem die wichtigsten Informationen der aktuellen Section zusammengefasst werden.

5.2 Einlesen der Airborne Laser Scanning (ALS)-Daten und der Bezugsachse

Die ALS-Daten werden für die anstehenden Berechnungen in den ODM geladen. Der ODM wird im Lese- und Schreibmodus geöffnet, um die klassifizierten Punkte abzuspeichern. Die Bezugsachse, entlang der die Klassifikation erfolgt, wird einen Linestring umgewandelt, um zwischen den einzelnen Sections wechseln zu können. Ein Linestring ist eine verschachtelte Liste, welche die x,y -Koordinaten der Stützpunkte der Bezugsachse enthält. Die Umwandlung des Shapefiles in den Linestring erfolgt mit der Import Methode des pyODM, welcher das Python API des ODM ist. `pyDm.Import()` ist eine spezifische Importklasse, die das Importieren von Shapefiles handhabt [13]. Anschließend werden die einzelnen Punkte der Achse als Koordinatenpaare in einer Liste gespeichert. Dieser gesamte Vorgang erfolgt in einer verschachtelten Schleife.

Program Code 1: Erstellen des Linestrings

```
1  imp = pyDM.Import.create(file, pyDM.DataFormat.auto)
2
3      pts = []
4      for obj in imp:
5          # loop over points
6          for i in range(obj.sizePoint()):
7              pt = obj[i]
8              pts.append([pt.x, pt.y])
```

5.2.1 Darstellung der Schummerung und der Bezugschse

Die Darstellung der Schummerung (Shading) der Punktwolke und der Achse ist im GUI mit einem Qt SVG-Widget umgesetzt. Dieses erlaubt SVG-Abbildungen darzustellen [20]. Für die Transformation der räumlichen Rasterdaten und Vektordaten der Puntwolke in das Koordinatensystem des GUI wird die Pythonbibliothek GDAL [2] angewandt. Weiters werden Höhe und Breite des Shadings berechnet. Der Koordinatenursprung des SVG-Widgets liegt in der linken oberen Ecke, hingegen der von OpenGL in der linken unteren Ecke liegt, wie in Abbildung 6 zu sehen ist.

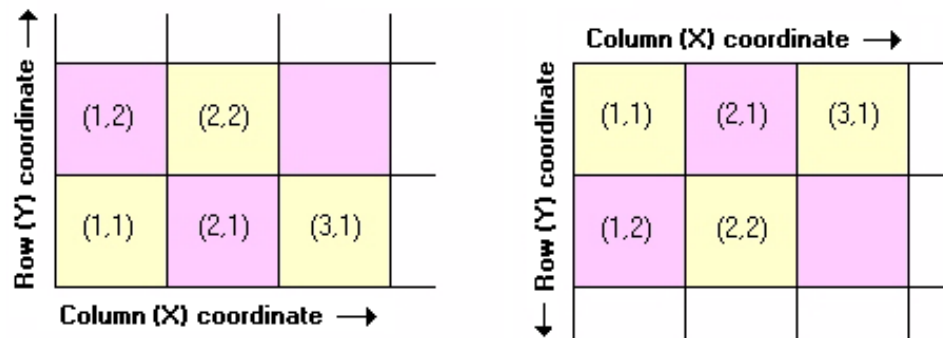


Abb. 6: Koordinatensysteme OpenGL (links) und PyQt (rechts)⁶

Für eine korrekte Darstellung wird die y -Koordinate des SVG-Widgets invertiert. Das Shading wird mit der *Add*-Methode und der *Image*-Methode von `svgwrite` [11] dargestellt. Der *Image*-Methode wird der Dateiname, die Ursprungskoordinaten des Widgets und die Abmessungen (Höhe und Breite) übergeben.

Die Achse wird als Linestring übergeben. Die Abschnitte der Achse, welche im Linestring gespeichert sind, werden jeweils einzeln dem SVG-Widget hinzugefügt. Für jeden Abschnitt wird ein Start- und ein Endpunkt festgelegt, zwischen denen eine Linie gezogen und dargestellt wird. Die einzelnen Abschnitte werden wie zuvor das Shading, mit Hilfe der *Add*-Methode dem SVG-Widget hinzugefügt. Sind alle Abschnitte graphisch abgebildet, ergeben diese die zuvor definierte Achse.

Program Code 2: Darstellung des Shadings und der Achse

```
1  svg.add(svg.image(href=self.shd_filename,insert=(minx,miny),size=(dx,dy)))
2
3  svg.add(svg.line(start=pt1, end=pt2, stroke='blue', stroke_width=2))
```

5.2.2 Graphische Darstellung der aktuellen Section in der Übersichtsgrafik

Neben dem Shading und der Achse wird auch der Umring (Polygon) (5.3.1), graphisch im SVG-Widget realisiert. Die Methode ist sehr ähnlich zu jener, die für die Darstellung der Achse angewandt wird.

Dem SVG-Widget werden die Eckpunkte des Polygons übergeben. Die Seiten des Polygons werden nacheinander im Widget dargestellt, indem jeweils zwei Punkte miteinander Verbunden werden, sodass ein Rechteck entsteht.

5.3 Teilmenge der Punktwolke aus dem OPALS Data Manager (ODM) laden

Um eine Punktwolke mit komplexer 3D Struktur klassifizieren zu können, ist es nötig, jeweils kleine Abschnitte (Sections) der Punktwolke als vertikale Schnitte darzustellen. Dafür muss der Bereich bestimmt werden, welcher aus dem ODM geladen werden soll. Dieser Bereich wird mit einem Polygon definiert.

⁶modifiziert aus: https://mirametrics.com/help/mira_pro_script_7/source/pixel_coordinate_definition.html (Zugriff am 20.02.2024)

5.3.1 Erstellen der Abschnittsbegrenzung (Polygon)

Für die genaue Positionierung des Polygons wird ein Fixpunkt in der Punktwolke benötigt. Dafür wird der zuvor erstellte Linestring der Polyline zur Hilfe genommen. Der Startpunkt (Fixpunkt in der Punktwolke) des ersten Polygons ist der erste Eintrag des Linestrings. Zusätzlich zu dem Fixpunkt in der Punktwolke ist die Richtung, in welche das Polygon aufgespannt wird, nötig. Für die Richtung wird ein Richtungsvektor (\vec{n}) bestimmt. Der Richtungsvektor wird zwischen zwei in Achsenrichtung benachbarten Punkten, mit den Koordinaten x_1, y_1 für den Startpunkt und x_2, y_2 für den nachfolgenden Punkt, aus dem Linestring berechnet und normiert.

$$\vec{n} = \begin{pmatrix} n_x \\ n_y \end{pmatrix} = \frac{\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} - \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \quad (1)$$

Der Richtungsvektor wird im späteren Verlauf auch für das Verschieben des Polygons verwendet. Die Richtung und der Startpunkt des Polygons sind bekannt. Die Koordinaten der Eckpunkte (P_1, P_2, P_3, P_4), zwischen welchen das Polygon aufgespannt wird, müssen noch bestimmt werden. Das Polygon soll so aufgespannt werden, dass es normal auf die Achse steht. Dafür wird der Richtungsvektor rotiert, sodass dieser orthogonal auf die Achse steht. Dabei werden die Koordinaten des Startpunkts vertauscht und bei einer Koordinate wird das Vorzeichen geändert.

$$\vec{r} = \begin{pmatrix} n_y \\ -n_x \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \end{pmatrix} \quad (2)$$

Für die Bestimmung des ersten Punktes, wird der Rotationsvektor (\vec{r}) um die Hälfte der gewünschten Breite (*Across*), vom Startpunkt aus verlängert. Der zweite Punkt wird in die gegengesetzte Richtung vom Startpunkt aus um denselben Faktor verlängert.

$$P_1 = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + \left(\begin{pmatrix} r_1 \\ r_2 \end{pmatrix} \cdot \frac{Across}{2} \right) \quad (3)$$

$$P_2 = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + \left(- \begin{pmatrix} r_1 \\ r_2 \end{pmatrix} \cdot \frac{Across}{2} \right) \quad (4)$$

Der dritte und vierte Punkt des Polygons werden mit Hilfe von P_1 und P_2 bestimmt. Dabei werden diese um die erforderliche Länge (*Along*) entlang des Richtungsvektors verschoben.

$$P_3 = P_2 + (Along \cdot \vec{n}) \quad (5)$$

$$P_4 = P_1 + (Along \cdot \vec{n}) \quad (6)$$

Sind die Eckpunkte des Polygons bekannt, kann mit Hilfe der Polygon Factory in OPALS das Polygon erzeugt werden [13]. Die vier Eckpunkte werden jeweils mit *addPoint* der Polygon Factory hinzugefügt. Das Polygon wird mit *closePart()* geschlossen.

Program Code 3: Erzeugung des Polygons

```

1  pf = pyDM.PolygonFactory()
2
3  def create_polygon(p1, p2, p3, p4):
4      pf.addPoint(p1[0, 0], p1[0, 1])
5      pf.addPoint(p2[0, 0], p2[0, 1])
6      pf.addPoint(p3[0, 0], p3[0, 1])
7      pf.addPoint(p4[0, 0], p4[0, 1])
8      pf.closePart()
9      return pf.getPolygon()
10
11 #create the polygon
12 polygon = create_polygon(p1, p2, p3, p4)

```

5.3.2 Veränderung der Geometrie der Abschnittsbegrenzung

Die Größe des Polygons kann zu jedem Zeitpunkt verändert werden. Dazu sind die Werte *Along* (Ausdehnung des Polygons parallel zur Achse) und *Across* (Ausdehnung des Polygons quer zur Achse) zu verändern. Werden die beiden Werte verändert, so erfolgt eine Neuberechnung der Eckpunkte nach demselben Schema wie in Kapitel 5.3.1. Die Änderung der Größe wird auch im SVG-Widget angezeigt.

5.3.3 Teilmenge der Punktwolke und ihre Attribute

Der Bereich, welcher aus dem ODM geladen werden soll, wurde mit dem Polygon bestimmt. Vor dem Laden und Darstellen der Punkte innerhalb des Polygons sind die Attribute, die von Interesse sind, festzulegen. Für dieses Tool werden die folgenden Attribute geladen:

- Id
- Classification
- `__manuallyClassified`

`__manuallyClassified` ist ein benutzerdefiniertes Attribut, das für die Vorhersage der Klassen in der nächsten Section mittels kNN benötigt wird. `__manuallyClassified` ist ein 8 Bit Integer als Datentyp. `__manuallyClassified` kann für einen Punkt drei Werte annehmen. *0* wenn der Punkt nicht klassifiziert ist. *1* wenn der Punkt bereits einer Klasse zugewiesen ist, und *2* wenn dem Punkt eine Klasse mittels kNN vorhergesagt wurde. Die Funktionsweise dieser automatischen Klassifikation wird in Kapitel 5.7 beschrieben. Das Laden der Attribute aus dem ODM wird mit der Layout Factory durchgeführt, welche ein Interface für die Erstellung von AddInfo-Layouts ist [13]. Der Layout Factory wird jedes Attribut einzeln hinzugefügt. Ist die Layout Factory angelegt, können die Punkte, die innerhalb des Polygons liegen, aus dem ODM geladen werden. Das Laden erfolgt mit Hilfe des integrierten *NumPy converters*, welcher die Punkte und die angelegten Attribute in NumPy Arrays umwandelt [13].

Das finale Ergebnis ist ein Wörterbuch, welches als Schlüssel die Koordinaten der geladenen Punkte und die Attributnamen aufweist. Die Werte sind die jeweiligen Numpy Arrays.

Program Code 4: Laden der Punkte

```

1 lf2 = pyDM.AddInfoLayoutFactory()
2 type, inDM = lf2.addColumn(dm, 'Id', True); assert inDM == True
3 type, inDM = lf2.addColumn(dm, 'Classification', True); assert inDM == True
4 type, inDM = lf2.addColumn(dm, self.manuallyClassified, True, pyDM.
    ColumnType.uint8)
5
6 self.layout2 = lf2.getLayout()
7
8 result = pyDM.NumpyConverter.searchPoint(self.odm, polygon, self.layout2,
    withCoordinates=True, noDataObj=0)
9 self.result = result

```

5.4 Darstellung der Punkte

Die Punkte werden als vertikale Schnitte dargestellt. Im Konkreten bedeutet das eine Reduktion der Dimensionen von 3D auf 2D. In dieser Anwendung wird bei der Darstellung der aktuellen Section die Tiefe der Punkte in Achsenrichtung nicht beachtet. Mit diesen Schnitten ist eine Unterscheidung zwischen Punkten der Wasseroberfläche, Wassersäule und Gewässerboden, gegebenenfalls anderer Klassen wie Boden, Vegetation, Gebäude etc. möglich. Die Darstellung erfolgt über das Draw-Widget, welches wie auch das Overview-Widget ein benutzerdefiniertes Widget in PyQt ist. Dem Widget werden alle geladenen Punkte samt ihrer Attribute übergeben. Die Koordinaten des ersten Punktes werden für die Berechnung eines Referenzpunkts (*self.StrechRefPt*) und für die Ermittlung des Richtungs- (*self.StrechVecA*) und Normalvektors (*self.StrechVecN*) benötigt. Beide Vektoren sind normiert. Diese drei Größen werden im weiteren Verlauf für die korrekte Darstellung der Section verwendet.

Program Code 5: Vorbereitung für die Darstellung

```

1 self.StrechRefPt = [coor1[0], coor1[1], 0]
2 self.StrechVecA = [coor2[0]-coor1[0], coor2[1]-coor1[1], 0]
3 len = math.sqrt(sum([math.pow(self.StrechVecA[i], 2) for i in range(3)]))
4 for i in range(3):
5     self.StrechVecA[i] /= len
6 self.StrechVecN = [-self.StrechVecA[1], self.StrechVecA[0], 0]
7 self.dataRefresh()

```

Ist diese Berechnung abgeschlossen, wird die *dataRefresh*-Methode des Widgets aufgerufen. Diese Methode aktualisiert die Darstellung der angezeigten Punkte und deren Eigenschaften. Dazu wird das Zentrum (*self.Center*) der Punkte der aktuellen Section berechnet, indem der Mittelpunkt der *x,y,z*-Koordinaten, mit Hilfe des Minimum und Maximum der aktuellen Punkte berechnet wird. Wurde das Zentrum ermittelt, wird aus diesem ein Skalierungsfaktor (*self.Scale*) berechnet. Das Zentrum und der Skalierungsfaktor wird im weiteren Verlauf für die korrekte Darstellung benötigt. In der *dataRefresh*-Methode werden zwei OpenGL-Listen [19] (*self.ptList*, *self.ptListIds*) angelegt. *self.ptList* wird für die Darstellung und Einfärbung der Punkte verwendet. Für das Auswählen einzelner Punkte wird auf die Liste *self.ptListIds* zurückgegriffen. Die beiden Listen werden mit den Funktionen *self.createColorList* und *self.createIdList* befüllt. Die Darstellung und Einfärbung der Punkte erfolgt in der *createColorList*-Methode.

Program Code 6: Aktualisieren der Darstellung

```

1  min, max = self.getDataExtends()
2
3  self.Center = [(min[i] + max[i]) / 2. for i in range(3)]
4
5  maxdist = 0
6  for i in range(3):
7      v = max[i] - self.Center[i]
8      if v > maxdist:
9          maxdist = v
10 if maxdist == 0:
11     self.Scale = 1.
12 else:
13     self.Scale = 1. / maxdist
14
15 self.ptList = glGenLists(1)
16 self.ptListids = glGenLists(3)
17
18 self.createColorlist()
19 self.createIdList()

```

Die Teilmengen der Punktwolke sollten als vertikale Schnitte, die orthogonal auf die Achse stehen, dargestellt werden. Der Rotationsvektor \vec{r} , welcher in 5.3.1 berechnet wurde, wird der Orthoview-Methode des Widgets übergeben. In dieser Methode wird der Rotationsvektor abermals rotiert, sodass dieser in die entgegengesetzte Richtung wie der Richtungsvektor \vec{n} in Kapitel 5.3.1 zeigt, und wird dem Camera-Modul übergeben. Dadurch bildet die Kamera den Abschnitt orthogonal entlang der Bezugsachse ab.

5.4.1 Einfärbung der Punkte

Für die Darstellung der Punkte im vertikalen Schnitt werden diese eingefärbt. In diesem Tool erfolgt das, indem *createColorList* aufgerufen wird, wenn die Daten in *dataRefresh* aktualisiert werden. In dieser Methode wird die zuvor angelegte OpenGL-Liste (*self.ptList*) mit den Punkten befüllt. Innerhalb der Liste existieren zwei Möglichkeiten der Einfärbung:

1. Einfärbung nach Klassen:

Dabei wird jeder Punkt nach seiner Klasse eingefärbt. Das Einfärben wird mit *glColor(c)* durchgeführt. Die Farbcodes der Klassen sind in Tabelle 1 aufgelistet. Wurde einem Punkt die richtige Farbe zugeteilt, werden die Koordinaten normiert. Die Normierung ist für eine besser Darstellung nötig und erfolgt in der Methode *_normalize* mit dem Referenzpunkt, Richtungs- und Normalvektor, die zuvor berechnet wurden.

Program Code 7: Einfärbung nach Klasse

```

1      c = [self.cmap[self.Data['Classification'][idx]][i] / 255 for i in
           range(3)]
2      coords = [self.Data["x"][idx], self.Data["y"][idx], self.Data["z"]
                 ][idx]]
3      glColor(c)
4      glVertex(self._normalize(coords))

```

2. Einfärbung nach Höhe:

Neben der Einfärbung nach Klassen besteht auch die Möglichkeit, die aktuelle Section in einer Höheneinfärbung darzustellen. Dafür wurden die Farben für die verschiedenen Höhen

mittels Look-up-Table bestimmt. Die 256 möglichen Farben wurden dafür auf eine Skala von 0-100 Prozent transformiert. Als Grundlage dient die OPALS-Standardpalette, welche in Tabelle 3 angeführt ist [13].

Tab. 3: Standard Farbpalette

Höhe [%]	Farbcode (r, g, b)
0	0, 153, 51
33	153, 230, 0
66	222, 222, 31
100	135, 87, 18

Nach Tabelle 3 wird demnach den niedrigsten Punkten (0% Höhe) der Farbcode (0, 153, 51) entsprechend Grün zugewiesen. Für die Punkte mit der Höhe 33%, 66% und 100% werden die jeweiligen Farben aus Tabelle 3 genutzt. Für Punkte anderer Höhen wird linear interpoliert. Zum Beispiel wird die zwanzigste mögliche Farbe den Punkten mit der Höhe von 7,8% zugewiesen:

$$p = \left(\frac{i}{256}\right) \cdot 100 = \left(\frac{20}{256}\right) \cdot 100 = 7,8125\% \quad (7)$$

Daher muss zwischen 0% und 33% interpoliert werden:

$$f = \frac{p - 0}{33 - 0} = \frac{7,8125 - 0}{33 - 0} = 0,2367 \quad (8)$$

Der Farbcode von 0% Höhe wird nun mittels diesem Faktor erhöht:

$$R = \frac{R_0 + f \cdot (R_{33} - R_0)}{255} = \frac{0 + 0,2367 \cdot (153 - 0)}{255} = 0.142 \quad (9)$$

Für G und B gilt dieselbe Berechnung mit den jeweiligen Farbwerten aus Tabelle 3. Nachdem alle 256 Farbwerte bestimmt und der Look-up-Table hinzugefügt wurden, wird jedem Punkt in Abhängigkeit seiner Höhe die passende Farbe zugewiesen. Haben alle Punkte eine Farbe erhalten, erscheinen die Punkte der Section in einem Grün über ein Gelb hin zu Braun.

5.4.2 Änderung der Ansicht

Die angezeigte Teilmenge des aktuellen vertikalen Schnittes kann in ihrer Darstellung auf drei unterschiedliche Möglichkeiten angepasst werden:

1. Rotation:

Wird eine Section aus dem ODM geladen, wird diese als vertikaler Schnitt (Reduktion von 3D auf 2D) dargestellt. Wegen der Reduktion der Dimensionen besteht die Möglichkeit, dass Punkte von anderen Punkten verdeckt werden. Aus diesem Grund kann die Section rotiert werden. Das Rotieren wird mit der *mouseMoveEvent*-Methode realisiert. Im Camera-Modul wird einer Rotationsmatrix die Position übergeben, ab der der Rotationsvorgang beginnt, und die aktuelle Position des Mauszeigers. Wird die Section rotiert, verändert die Kamera ihre Position, sodass die gewünschte Darstellung zu sehen ist. Die rotierte Position der Kamera kann in die ursprüngliche Position zurückgesetzt werden. Das Zurücksetzen läuft nach demselben Schema wie das Darstellen des vertikalen Schnitts (5.4) ab.

2. Zoom:

Eine weitere wichtige Funktion ist das Zoomen in die Punktwolke, um einen spezifischen Bereich größer abzubilden. Der Zoomvorgang geschieht in der *wheelEvent*-Methode. Wie

weit hinein bzw. heraus gezoomt werden soll, wird über die Drehbewegung des Mauseklasses bestimmt. Jede Drehbewegung wird in einen Differenzwinkel umgerechnet. Anschließend wird dem Kameramodul diese Winkeldifferenz übergeben, und die *dollyCameraForward*-Methode aufgerufen. In dieser Methode wird die Mauseklassesbewegung in die Distanz umgerechnet, wie stark der Zoom ausfällt.

3. Verschieben:

Die letzte Funktion für eine Anpassung der Darstellung, ist das Verschieben des vertikalen Schnittes. Das Verschieben erfolgt, indem die rechte Maustaste gedrückt wird. Um die veränderte Position zu messen, wird in der *mouseMoveEvent*-Methode, wenn die rechte Maustaste gedrückt wird, auch die aktuelle Position der Maus auf dem Widget detektiert, um die Differenz der Anfangsposition zu der aktuellen zu ermitteln. Die Differenz der Positionen wird der *translateSceneRightAndUp*-Methode des Kamera-Moduls übergeben. Diese Methode aktualisiert die Kameraposition.

5.4.3 Variation der Punktgröße

Für ein einfaches Picken der Punkte soll die Darstellungsgröße nach Belieben eingestellt werden können. In dem Widget, in welchem die aktuelle Section dargestellt wird, ist eine Standardpunktgröße von 1 eingestellt. Die Punktgröße kann beliebig über *glPointSize()* eingestellt werden. *glPointSize* übernimmt eine Integer Variable. Die Punktgröße wird der glListe *self.ptList* in *self.dataRefresh()* übergeben, bevor diese mit den einzelnen Punkten befüllt wird. Wird eine neue Punktgröße ausgewählt, wird die Darstellung aktualisiert.

5.4.4 Statusmessages

Wird ein Teil der Punktwolke geladen, ist es nützlich, wenn Informationen über den geladenen Abschnitt vorliegen. Informationen wie:

- **Anzahl der geladenen Punkte:**

Für diese Information wird die Länge des Arrays bestimmt, in welchem die x-Koordinaten der Punkte aufgelistet sind.

- **Klassifizierte Punkte:**

Um eine Information zu erlangen, wie viele Punkte bereits klassifiziert sind, wird über das Array, in welchem die Klassen der Punkte enthalten sind, eine Maske gelegt. Diese Maske initialisiert alle Einträge, die größer Null sind, mit *True*. Alle Einträge mit dem Wert *True* werden anschließend aufsummiert.

- **Unklassifizierte Punkte:**

Die Ermittlung der Punkte, welche unklassifiziert sind, erfolgt analog zu jener, welche bereits klassifiziert sind. Der wesentliche Unterschied liegt darin, dass die Maske all jene Punkte mit *True* initialisiert, welche den Wert Null (unklassifiziert) haben.

- **Histogramm der aktuellen Section:**

Wird eine Section geladen, wird diese analysiert und ein Histogramm erstellt. Das Histogramm gibt eine Auskunft darüber, wie viele Punkte in einer Klasse zugeordnet sind und wird als Wörterbuch ausgegeben.

- **Stationierung:**

Wird ein Abschnitt aus dem ODM geladen, wird als Statusmessage die Stationierung dieses Abschnittes auf der Bezugsachse ausgegeben. Wird der Abschnitt gewechselt, so ändert sich auch die Statusmessage der aktuellen Stationierung.

- **Anzahl der Punkte, denen eine Klasse vorhergesagt wurde:**

Wird eine Vorhersage der Klassen beim Wechseln eines Abschnittes gewünscht, wird die Summe der Punkte als Statusmessage ausgegeben, für welche eine Klasse vorhergesagt wurde.

5.5 Klassifizieren der Punktwolke

Um den Punkten der ALS-Daten eine Klasse zuzuordnen, sind in diesem Tool zwei Möglichkeiten implementiert. Zum einen ist es möglich, Punkte mit einem dynamischen Picker zu klassifizieren. Dynamisch deshalb, weil es möglich ist, mittels Mausexplorer die Größe des Bereiches zu variieren, welcher ein Mausklick abdeckt. Außerdem ist in diesem Picker eine Pinselfunktion integriert. Mit dieser Funktion können durch das Gedrücktthalten der linken Maustaste und durch Bewegung der Maus Punkte klassifiziert werden. Die zweite Möglichkeit ist es, mehrere Punkte, die derselben Klasse wie zum Beispiel Boden zuzuweisen sind, mittels Markierens gleichzeitig zu klassifizieren. Welcher Klasse die Punkte zugeordnet werden, wird über ein Drop-Down Menü ausgewählt. Das Drop-Down Menü wird, wenn das Klassifikationstool gestartet wird, zusammengebaut. Für das Erstellen des Drop-Down Menüs wird auf ein Wörterbuch zurückgegriffen, welches als Schlüssel die jeweiligen Werte der Klasse (Tabelle 1) enthält. Jeder Schlüssel hat eine Liste als Wert. Der erste Eintrag der Liste ist der Name der jeweiligen Klasse als String definiert. In dem zweiten Eintrag der Liste ist der RGB-Farbcode aus Tabelle 1 angelegt. Wird das Drop-Down Menü zusammengebaut, wird vor jedem Klassennamen ein Icon in der dazugehörigen Klassenfarbe platziert. Für das Klassifizieren der Punkte wird der Schlüssel der ausgewählten Klasse dem DrawWidget-Modul in der *PointClassification*-Methode übergeben.

5.5.1 Dynamischer Picker

Der dynamische Picker hat zu Beginn eine Größe von einem Pixel. Wird das Mausexplorer bewegt, wird die Fläche des Mauszeigers sowohl horizontal als auch vertikal in einem Schritt vergrößert. Dabei wird die *wheelEvent*-Methode der DrawWidgets-Class aufgerufen und der Winkel detektiert, um welchen sich das Mausexplorer bewegt. Eine Stufe des Mausexplorer entspricht einem Wert von 120. Die Anzahl und die Richtung der Radbewegungen werden, um die Größe des dynamischen Pickers zu bestimmen, gezählt. Um die genaue Position (x,y) des Mauszeigers herauszufinden, wird auch die *mouseMoveEvent*-Methode der DrawWidgets-Class angewandt. *mouseMoveEvent* ist aktiviert, solange sich der Mauszeiger auf dem Klassifikationsgrafikbereich befindet.

An der Stelle, an der sich der Mauszeiger befindet, wird ein Quadrat um den Zeiger aufgespannt. Die Größe des Quadrats ist abhängig davon, wie oft und in welche Richtung das Mausexplorer bewegt wurde. Das Quadrat wird graphisch im Widget dargestellt. Die Umsetzung erfolgt in der *paintgl*-Methode. Ist die passende Größe ausgewählt, so kann mittels Linksklick klassifiziert werden. Es werden alle Punkte innerhalb des Quadrats der ausgewählten Klasse zugeordnet. Eine weitere Eigenschaft dieses Pickers ist es, durch Gedrücktthalten der linken Maustaste und das Bewegen des aufgespannten Quadrates über Punkte diese wie mit einem Pinsel zu klassifizieren. Der Bereich, welcher klassifiziert wird, ist gleich dem aufgespannten Quadrat. Wie schlussendlich den jeweiligen Punkten die richtige Klasse zugeordnet wird, ist in 5.5.3 beschrieben.

5.5.2 Markieren von mehreren Punkten

Das Markieren von mehreren Punkten ist ähnlich dem Konzept des dynamischen Pickers. Der größte Unterschied liegt darin, dass das Rechteck durch Gedrücktthalten der linken Maustaste und durch Bewegung der Maus aufgezogen wird. Als Startpunkt wird die aktuelle Position des Mauszeigers am Widget gespeichert. Dies geschieht in der *mousePressEvent*-Methode. Solange die linke Maustaste betätigt und die Maus bewegt wird, verändert das Rechteck die Größe. Wird die linke Maustaste losgelassen, wird in der *mouseReleaseEvent*-Methode die Position

des Mauszeigers als Endposition des Rechteckes gespeichert und alle Punkte, die innerhalb des Rechteckes liegen, werden der ausgewählten Klassen zugeteilt. Wie auch der dynamische Picker in 5.5.1 wird auch das Rechteck mittels der *paintgl*-Methode graphisch dargestellt.

5.5.3 Klassifizieren

Ist ein Punkt oder ein Bereich ausgewählt, wird dieser der zuvor ausgewählten Klasse zugeteilt. Doch zuvor muss die Id des ausgewählten Punktes bestimmt werden. Die Bestimmung der Punkt-Id ist folgendermaßen aufgebaut. Im ersten Schritt werden die Seitenlängen des Rechteckes oder des Quadrats in Widget-Koordinaten in der *Picking*-Methode errechnet. Wird mit dem dynamischen Picker nur ein Pixel ausgewählt, entfällt die Berechnung.

Im nächsten Schritt werden die Seitenlängen sowie die Koordinaten der linken oberen Ecke des Vierecks oder im Fall eines einzelnen Pixels die Koordinaten des Cursors der *multiPtPicking*-Methode übergeben. Um all jene Pixel zu identifizieren, welche einen Messpunkt darstellen, wird über das ausgewählte Fenster iteriert und jedes, Pixel einzeln überprüft. Die Überprüfung erfolgt mit folgender Programmzeile:

Program Code 8: Analyse eines Pixels

```
1 posCol = glReadPixels(posX + x, self.heightInPixels - posY - y, 1, 1,
    GL_RGBA, GL_UNSIGNED_BYTE)
```

Die ersten beiden Parameter sind die x, y -Koordinaten des zu untersuchenden Pixels. Bei der Übergabe der y -Koordinate muss beachtet werden, dass in OpenGL die linke untere Ecke des Fensters als Ursprung definiert ist, im Gegensatz zu PyQt, wo der Ursprung in der linken oberen Ecke gesetzt ist, wie in Abbildung 6 dargestellt. Für die Ermittlung der y -Position des Pixels wird die y -Position der Maus von der Höhe des *DrawWidget* subtrahiert. Die nächsten zwei Parameter geben die Größe eines Pixels an. Mit den letzten Parametern wird angegeben, welche Information des Pixels zurückgegeben wird. In dieser Anwendung ist die Rückgabe ein NumPy Array [12], welches die RGBA-Information eines Pixels als unsigned Byte enthält. Für die Identifizierung des Punktes, wird der Hexadezimalcode in eine Dezimalzahl, welche der Punkt-Id entspricht mit folgender Gleichung 10 umgerechnet.

$$Id = R + G \cdot 256 + B \cdot 256 \cdot 256 - 1 \quad (10)$$

Zum Beispiel hat der Punkt mit dem Index 496 den folgenden RGBA-Hexcode: D5/01/00/FF.

Tab. 4: Hexadezimal in Dezimal Beispiel

	Hexadezimal	Dezimal
R	D5	213
G	01	1
B	00	0
A	FF	255

Der Wert, welcher an der zweiten Stelle gespeichert ist, wird mit 256 multipliziert, um die Punkte mit den Indizes über 255 zu identifizieren. Für den dritten Wert gilt ähnliches. Mit diesem Wert werden die Punkte mit den Indizes über 65.792 ($Id = R + G \cdot 256 = 256 + (256 \cdot 256) = 65.792$) identifiziert. Vom berechneten Index wird 1 subtrahiert, um festzustellen, ob ein Pixel ausgewählt wurde oder nicht, da der Hintergrund mit dem Index -1 vordefiniert ist.

Ist ein Pixel als Punkt identifiziert worden, wird die Punkt-Id einer Liste hinzugefügt. Alle Indizes in dieser Liste werden in dem Id-Array des ODM gesucht. An der Stelle, an der sich der

Index befindet, wird die Klasse des Punktes in die zuvor ausgewählte Klasse im Classification-Array geändert. In der *paintgl*-Methode des Widgets wird ein Depth-Buffer aktiviert. Dieser bewirkt, dass die Tiefe der Punkte nicht berücksichtigt wird. Das bedeutet, wenn sich zwei Punkte exakt hintereinander befinden oder sich Überlappen, wird nur der vorderste Punkt erkannt und klassifiziert. Der Depth-Buffer ist die gesamte Zeit über aktiv, sei es bei der Darstellung des aktuellen Abschnittes oder bei der Selektion der Punkte.

5.6 Speichern und Reset der Klassifizierung

Wurde die angezeigte Section klassifiziert, kann diese abgespeichert werden. Die Speicherung erfolgt über den Speicherknopf. Wird dieser betätigt, so werden alle aktuellen Attribute wieder in den ODM zurückgeschrieben. Ist der Schreibvorgang abgeschlossen, wird der ODM mit *odm.save()* gespeichert. Das Schreiben und das Speichern des ODM ist mit folgendem Code in der *changeAttributes*-Methode realisiert.

Program Code 9: Speichern des ODM

```
1 if np.array_equal(self.result['Classification'],self.checkClassification)
   == False:
2     self.setObj = {}
3     self.setObj['Id'] = self.result['Id']
4     self.setObj['Classification'] = self.result['Classification']
5     self.setObj[self.manuallyClassified] = self.result[self.
        manuallyClassified]
6     pyDM.NumpyConverter.setById(self.setObj, self.odm, self.layout2)
7     self.odm.save()
```

Das Speichern des ODM erfolgt nicht nur bei Betätigen des Speicherknopfs, sondern auch jedes Mal, wenn die Section gewechselt wird. Der NumPyConverter wandelt die Punkte mit ihren Attributen in NumPy Arrays um [13]. Die Attribute sind dieselben, die aus dem ODM geladen wurde.

- Id
- Classification
- *_manuallyClassified*

Wird diese Section zu einem späteren Zeitpunkt erneut aufgerufen, haben die Punkte in diesem Bereich die zuvor gespeicherten Attribute.

Wenn eine Section aus dem Daten Manager geladen wird, wird eine Deepcopy aller Attribute erzeugt. Diese Deepcopy wird benötigt, um ein Zurücksetzen der jeweiligen Section in den Ausgangszustand, der beim Laden der Punkte im jeweiligen Bereich vorliegt, zu ermöglichen. Das Zurücksetzen der Attribute erfolgt mit dem vorgesehenen Reset-Knopf. Wenn die aktuelle Klassifikation abgespeichert wird, erfolgt ein Löschen der Deepcopy. Ein Zurücksetzen in den Ausgangszustand ist ab diesem Zeitpunkt nicht mehr möglich. Die Deepcopy hat zudem noch die Funktion, dass vor jedem Speichervorgang überprüft wird, ob klassifiziert wurde. Wenn bei keinem Punkt eine Klasse geändert wurde, ist es auch nicht nötig, dass in den ODM zurückgeschrieben wird.

5.7 Navigation entlang der Bezugsachse

Wurde eine Section vollständig klassifiziert oder ist ein anderer Bereich, welcher vor oder nach dem aktuellen liegt, von größerer Bedeutung, kann das Polygon entlang der Achse verschoben

werden. Das Verschieben des Polygons erfolgt in den Methoden *previousSection* und *nextSection* über Stationierungen der Bezugsachse. Die Stationierungen werden mit Hilfe des *StationUtilities*-Moduls berechnet. Dazu wird der in Kapitel 5.2 erstellte Linestring dem *StationUtilities*-Modul übergeben. In diesem Modul werden in der *_update_values*-Methode alle Stationierungen und Richtungsvektoren der Bezugsachse bestimmt. Dafür werden immer zwei benachbarte Punkte des Linestrings herangezogen. Ein Startpunkt mit den Koordinaten x_1, y_1 und ein Endpunkt mit den Koordinaten x_2, y_2 . Für die Stationierungen der Bezugsachse wird die Distanz zwischen den beiden Punkten berechnet:

$$\text{Stationierung} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (11)$$

Nach der Berechnung der Stationierung der jeweiligen Punkte wird diese auf die Stationierung der vorausgegangenen Stationierung addiert und einer Liste hinzugefügt.

Die Richtungsvektoren zwischen den Punkten der Bezugsachse werden mittels Formel 1 berechnet. Die normierten Richtungsvektoren werden ebenfalls einer Liste hinzugefügt.

Program Code 10: Berechnung der Stationierungen und Richtungsvektoren der Bezugsachse

```

1  def _update_values(self):
2      self.stations = [0]
3      self.directions = []
4      currStat = 0
5      for idx in range(1, len(self.vertices)):
6          pt1 = self.vertices[idx - 1]
7          pt2 = self.vertices[idx]
8          dx = pt2[0] - pt1[0]
9          dy = pt2[1] - pt1[1]
10         slen = (dx ** 2 + dy ** 2) ** 0.5
11         if slen == 0:
12             raise Exception("Identical subsequent 2d points are not allowed")
13         dir = [dx/slen, dy/slen]
14         self.directions.append(dir)
15         currStat += slen
16         self.stations.append(currStat)
17     # duplicate last entry for direction (for simplified handling during
18     # interpolation)
19     self.directions.append(self.directions[-1])

```

Die durchgeführten Berechnungen werden in der Variablen *self.station_axis* im *ClassificationTool*-Modul gespeichert.

5.7.1 Wechseln in die nächste Section

Das Wechseln in die nächste Section erfolgt über die Methode *nextSection*. Dabei wird folgendermaßen vorgegangen:

1. Berechnung um wie weit das Polygon in Achsenrichtung (*Along*) verschoben werden muss. Wie weit das Polygon verschoben wird, hängt nicht nur von der Variablen *Along* ab, sondern auch davon, um wie viel Prozent das neue Polygon das vorherige Polygon überlappen soll. Die Überlappung (*Overlap*) wird in Prozent angegeben:

$$\text{Overlap} = \frac{\text{Prozent}}{100} \quad (12)$$

$$\Delta \text{Stationierung} = \text{Along} \cdot (1 - \text{Overlap}) \quad (13)$$

2. Der nächste Schritt besteht darin, dass die neue Stationierung berechnet wird. Dazu wird die alte Stationierung mit $\Delta Stationierung$ addiert:

$$Stationierung_{Neu} = Stationierung_{Alt} + \Delta Stationierung \quad (14)$$

3. Der dritte und letzte Schritt besteht darin, zu überprüfen in welchem Segment (mit dem Startpunkt S_1 und dem Endpunkt S_2) der Bezugsachse sich die neue Stationierung befindet. Dazu wird die neue Stationierung der *get_point_and_direction*-Methode des *StationUtilities*-Moduls übergeben. In dieser Methode wird mittels *bisect_left* der Index der Stationierung in der Liste, welche alle Stationierungen enthält, die links von der aktuellen Stationierung liegen, gesucht.

Mit diesem Index wird der passende Richtungsvektor aus der Liste aller Richtungsvektoren ausgewählt. Wurde der richtige Richtungsvektor ausgewählt, erfolgt die Berechnung des neuen Startpunktes des Polygons:

$$\begin{aligned} P_X &= S_{1,x} + Stationierung_{Neu} \cdot n_x \\ P_Y &= S_{1,y} + Stationierung_{Neu} \cdot n_y \end{aligned} \quad (15)$$

5.7.2 Wechseln in die vorherige Section

Das Wechseln in die vorherige Section funktioniert nach demselben Prinzip wie jenes für das Wechseln in die nächste Section aus Kapitel 5.7.1, und wird mit der Methode *previousSection* durchgeführt. Der einzige Unterschied besteht darin, dass die Berechnung von $\Delta Stationierung$ im zweiten Schritt mit folgender Gleichung durchgeführt wird:

$$Stationierung_{Neu} = Stationierung_{Alt} - \Delta Stationierung \quad (16)$$

5.7.3 Vorhersage der Klassen in der nächsten Section mittels kNN

Wenn alle Punkte der aktuellen Section klassifiziert wurden, kann ausgewählt werden, ob eine Vorhersage mit kNN von den Klassen der nächsten Section erfolgen soll.

Für eine Vorhersage, wird ein leeres *kdtree*-Objekt mit *PointIndexLeaf*, was ein räumliches Blatt mit einem Punktindex darstellt [13], angelegt. Dieses *kdTree*-Objekt wird mit den Punkten der vorherigen Section befüllt. Im nächsten Schritt werden die Parameter festgelegt, wie die Vorhersage durchgeführt werden soll:

Tab. 5: Parameter für *kdtree*-Objekt

Parameter	Wert
nnCount	1
searchPoint	Koordinaten eines Punktes der neuen Section
searchMode	nearest
maxSearchDist	-1

Mit dem Punkt, für welchen die Klasse vorhergesagt werden soll, wird das *kdtree*-Objekt nach seinem nächsten Nachbar durchsucht. Der maximale Suchradius wurde mit -1 definiert. Das bedeutet, dass die Suche so lange andauert, bis ein nächster Nachbar gefunden wurde. Wurde ein Nachbarn gefunden, wird dem Punkt die Klasse des Nachbarn zugewiesen. Es soll nur für jene Punkte eine Vorhersage getroffen werden, die noch keiner Klasse zugeordnet sind. Dazu wird für jeden Punkt abgefragt, ob die aktuelle Klasse gleich dem Wert Null entspricht. Wird einem Punkt eine Klasse vorhergesagt, wird in *__manuallyClassified* für diesen Punkt der Wert auf 2 gesetzt.

Somit ist ersichtlich, welchen Punkten eine Klasse vorhergesagt wurde. `__manuallyClassified` wird dem Layout des ODM beim Laden der Section mitgegeben.

6 Ergebnisse

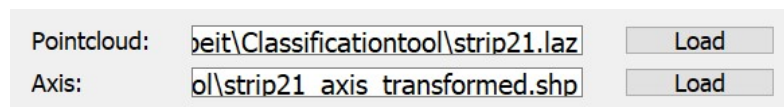
In diesem Kapitel werden die Ergebnisse dieser Bachelorarbeit präsentiert. Dabei wird darauf eingegangen, wie der Ablauf bei der Nutzung dieses Tools zur manuellen Klassifikation von ALS-Punktwolken ist. Das Tool selbst stellt das Hauptergebnis dieser Bachelorarbeit dar und ist in Abbildung 33 dargestellt.

6.1 Klassifizieren einer Punktwolke

Die Punktwolke, die verwendet wurde, um die Ergebnisse zu präsentieren, ist eine ALS-Messung vom Fluss Pielach in Niederösterreich. Wie in Kapitel 4 beschrieben, ist es notwendig, vor dem Beginn der Klassifikation Vorbereitungen zu treffen, z.B die Definition der Achse und das Erstellen des Shadings für die Übersicht. Die Achse, entlang derer klassifiziert wird, ist so gewählt, dass diese in der Mitte des Flusses verläuft. Dadurch kann der Fluss und die beiden angrenzenden Uferbereiche klassifiziert werden.

6.1.1 Einlesen der ALS-Daten und der Bezugsachse

Für das Einlesen der 3D Punkte und der Achse (Shapefile) ist der Dateipfad in die dafür vorgesehenen Textfelder einzufügen, wie in Abbildung 7 zu sehen ist. Nach dem Einlesen werden die Textfelder so lange gesperrt, bis die aktuelle Section gespeichert wurde. Wird in eine andere Section navigiert, werden die Textfelder abermals gesperrt.



The image shows a user interface with two rows of input fields. The first row is labeled 'Pointcloud:' and contains a text box with the path 'beit\Classificationtool\strip21.laz' and a 'Load' button to its right. The second row is labeled 'Axis:' and contains a text box with the path 'ol\strip21 axis transformed.shp' and a 'Load' button to its right.

Abb. 7: Einlesen der ALS-Daten und der Achse

Wurden die Daten und die Achse erfolgreich eingelesen, wird das Shading, die Achse (Blau) und das Polygon (Rot) im SVG-Widget (Abbildung 8) angezeigt. Für das Polygon sind die Standardwerte 20 (*Across*) und 5 (*Along*) eingestellt. Das heißt, dass im ersten Schnitt standardmäßig ein Datenbereich von 20 mal 5 dargestellt wird.

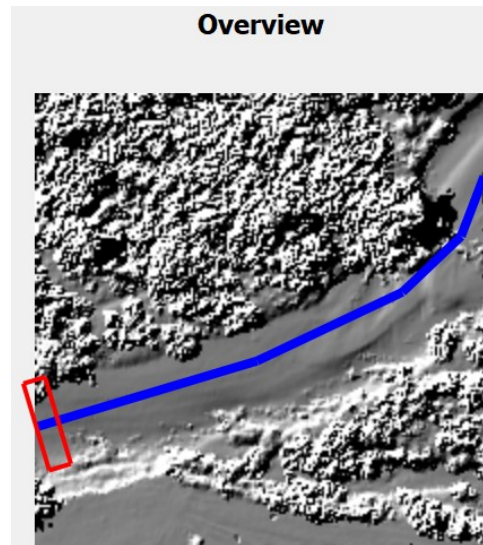


Abb. 8: Darstellung des Shadings, der Achse und des Polygons

6.1.2 Laden und Darstellen der ersten Section aus dem ODM

Nachdem die Achse und das Shading eingelesen wurde, werden mit dem Polygon (Abbildung 8 in Rot dargestellt) die Punkte der ersten Section aus dem ODM geladen und dargestellt (Abbildung 9). Die Punkte, die dargestellt sind, entsprechen jenen, die innerhalb des Polygons in Abbildung 8 liegen. In dem in Abbildung 9 abgebildeten vertikalen Schnitt ist eine Unterscheidung zwischen Wasseroberfläche (Dunkelblau), Wassersäule (Hellblau) und Gewässerboden (Ocker) deutlich zu sehen. Zudem sind noch Bodenpunkte in Braun sowie hohe Vegetation (zum Beispiel ein Baum) in Dunkelgrün und niedrige Vegetation (Sträucher oder ähnliches) in Hellgrün erkennbar.

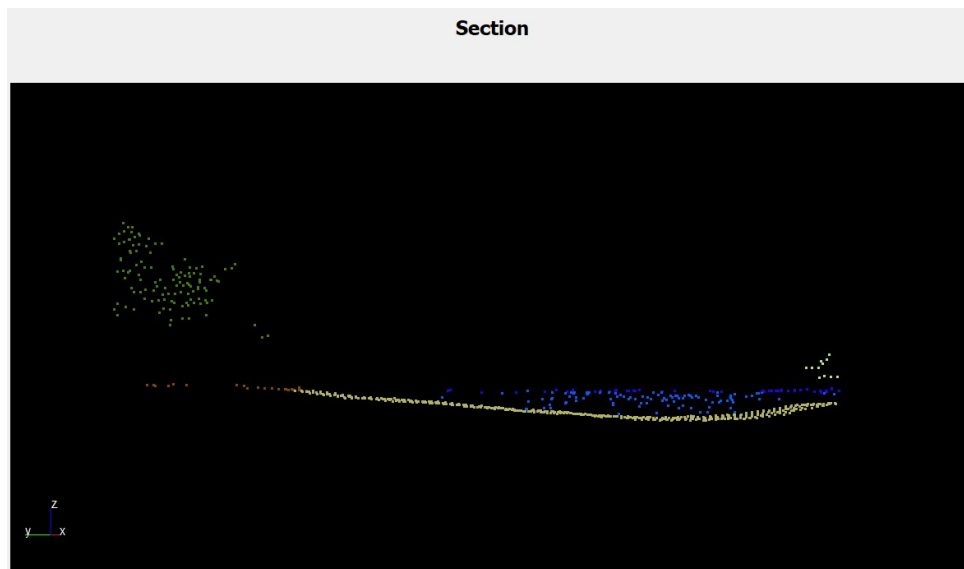


Abb. 9: Darstellung der ersten Section

Sind alle Punkte aus dem ODM geladen, werden Statusmitteilungen (Abbildung 10) über den aktuell angezeigten Bereich ausgegeben. In dieser Section wurden 679 Punkte aus dem ODM geladen. Davon wurde allen Punkten bereits eine Klasse zugeteilt.

```
Current station: 0.00 - 5.00  
Loaded: 679 Points  
Classified: 679 Points  
Unclassified: 0 Points  
Class histogram: {2: 26, 3: 9, 5: 117, 9: 81, 40: 390, 41: 56}
```

Abb. 10: Statusmessages

6.1.3 Variation der Punktgröße

Die in das Anzeigefenster geladenen Punkte, können in ihrer dargestellten Größe variiert werden. Das Einstellen der passenden Punktgröße erfolgt über den in Abbildung 11 abgebildeten Schieberegler. Dieser kann ganze Zahlenwerte zwischen eins und fünf annehmen.

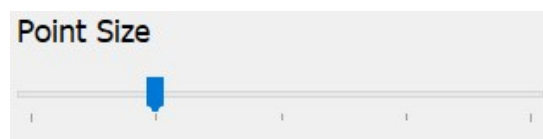


Abb. 11: Schieberegler für Punktgröße

In Abbildung 9 ist eine Punktgröße von zwei eingestellt. Punkte mit der Größe fünf sind in Abbildung 12 zu sehen.

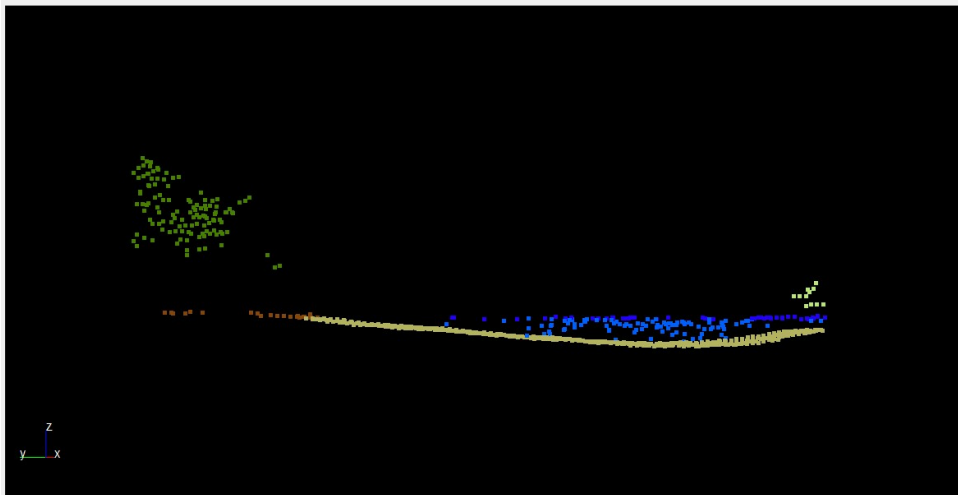


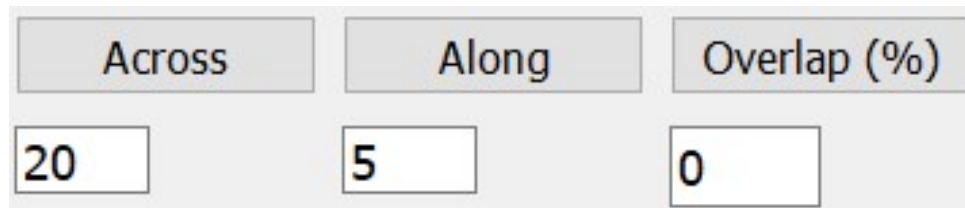
Abb. 12: Punkte mit eingestellter Punktgröße 5

Vergleicht man Abbildung 9 und Abbildung 12, so fällt auf, dass in Abbildung 12 Strukturen, wie zum Beispiel der Baum oder Strauch am linken Rand der Punktwolke, deutlich besser erkennbar sind. Neben dem Erkennen von Strukturen lassen sich auch die unterschiedlichen Teile des Gewässers (Wasseroberfläche, Wassersäule, Gewässerboden) besser unterscheiden. Der Hauptnutzen einer größeren Darstellung ist, dass die Punkte leichter ausgewählt werden können. Allerdings hat eine größere Darstellung der Punkte auch Nachteile, wie dass Punkte, die weiter hinten liegen, von anderen Punkten leichter verdeckt werden. Ein weiterer Nachteil ist, dass wenn Punkte überlappen, keine eindeutige Zuteilung zu einer Klasse möglich ist.

6.1.4 Einstellung von *Across*, *Along* und *Overlap*

Die Ausdehnungen des Polygons können zu jedem Zeitpunkt angepasst werden. Dazu sind neue Werte in den Feldern *Across* (Quer zu der Achse) und *Along* (Längs zu der Achse) einzugeben. Ein Beispiel ist in Abbildung 13 dargestellt. Als Standardwerte sind beim Programmstart 20 für *Across* und 5 für *Along* eingestellt. Eine Wertänderung erfolgt jeweils durch die Eingabe eines Wertes im numerischen Feld, gefolgt von der Bestätigung der Eingabe durch Drücken des darüber liegenden Knopfes.

Mit *Overlap* wird festgelegt, um wie viel Prozent das nächste Polygon mit dem aktuellen überlappt. Im Feld *Overlap* ist ein ganzzahliger Wert zwischen 0 und 100 einzutragen und zu bestätigen.



The image shows a user interface with three columns of controls. Each column has a label in a grey box at the top, a numerical input field in the middle, and a confirmation button at the bottom. The first column is labeled 'Across' and has the value '20'. The second column is labeled 'Along' and has the value '5'. The third column is labeled 'Overlap (%)' and has the value '0'.

Across	Along	Overlap (%)
20	5	0

Abb. 13: Anpassung der Polygoneigenschaften Across, Along, Overlap

6.1.5 Klassifizieren

Der nächste Schritt ist die Punkte der Section zu klassifizieren. Aus dem Drop-Down-Menü in Abbildung 14 ist die Klasse auszuwählen, welche vergeben werden soll. Soll Punkten eine andere Klasse zugeteilt werden, kann diese im Drop-Down-Menü gewechselt werden.

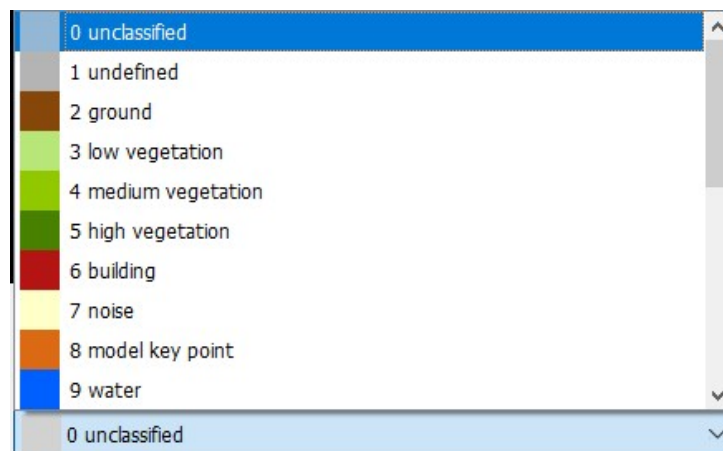


Abb. 14: Auswahl der Klassen mittels Drop-Down-Menü

Für das Klassifizieren von Punkten stehen in diesem Tool zwei Möglichkeiten zur Verfügung. Die erste Möglichkeit ist, einzelne Punkte mit dem dynamischen Picker (Abbildung 15) zu klassifizieren. Dieser Picker hat die Eigenschaft, dass der Bereich, welcher klassifiziert wird, in der Größe mit dem Mausekursor bestimmt werden kann. Abbildung 16 zeigt das mit dem Mausekursor vergrößerte Quadrat. Es werden alle Punkte, die innerhalb des Quadrats liegen, der gewünschten Klasse zugeteilt.



Abb. 15: Dynamischer Picker

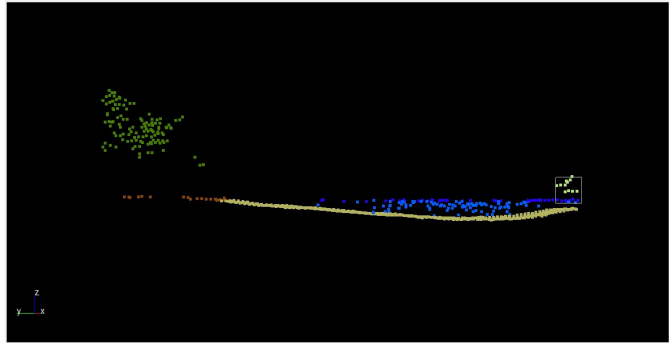


Abb. 16: Auswahl der Punkte mit dynamischem Picker

Die zweite Möglichkeit, Punkte zu klassifizieren ist, mehrere Punkte mit dem Markierungspicker (Abbildung 17) auszuwählen. Dabei wird ein Rechteck durch das Gedrückthalten der linken Maustaste aufgespannt. Alle Punkte, die innerhalb des Rechtecks liegen, werden klassifiziert, sobald die linke Maustaste losgelassen wird. Wie dieses Rechteck im Tool realisiert ist, wird in Abbildung 18 gezeigt.

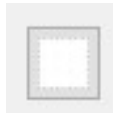


Abb. 17: Markierungspicker

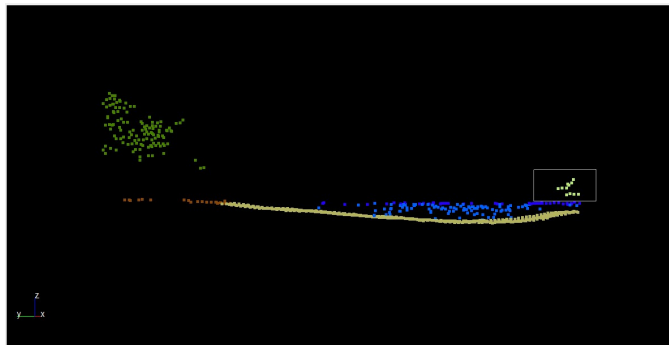


Abb. 18: Auswahl der Punkte mit Markierungspicker

Zu beachten ist, dass der dynamische Picker und der Markierungspicker nicht gleichzeitig auswählbar sind. Wird ein Picker aktiviert, so wird der andere Picker, falls dieser ausgewählt ist, deaktiviert.

6.1.6 Einfärbung der Punkte

Der aktuelle vertikale Schnitt kann in zwei Einfärbungen dargestellt werden. Die erste Einfärbung, welche als Standard eingestellt ist, ist die Einfärbung nach Klassen (Abbildung 19). Zum anderen ist eine Einfärbung nach Höhe, wie in Abbildung 20 dargestellt, möglich. Dabei werden die niedrigsten Punkte in Grün dargestellt und die höchsten in Braun. Zwischen minimaler und maximaler Höhe sind Farben, welche von Grün zu Gelb und schlussendlich zu Braun kontinuierlich verlaufen, eingestellt. Die Farben stammen aus der OPALS-Standardpalette [13].

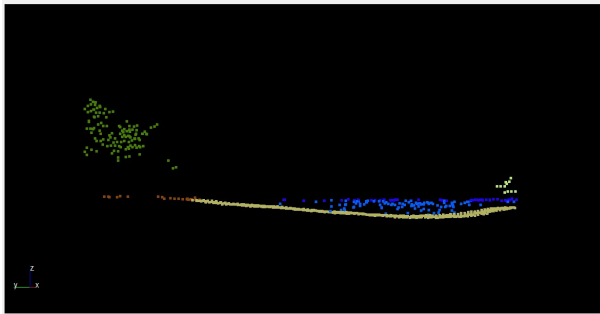


Abb. 19: Einfärbung nach Klassen

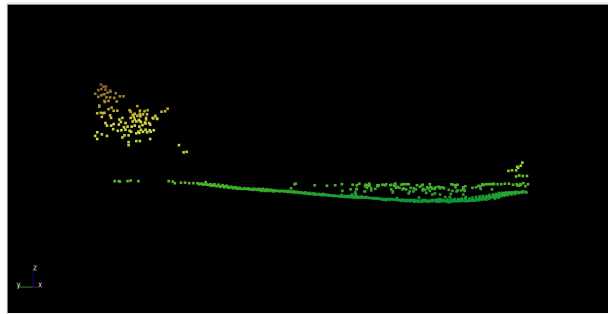


Abb. 20: Einfärbung nach Höhe

Wie die Punkte eingefärbt werden sollen, kann über zwei Knöpfe (Abbildung 21 und 22) ausgewählt werden. Gleich wie bei der Auswahl der Picker, ist es auch bei den Einfärbungen nicht möglich, beide Möglichkeiten gleichzeitig auszuwählen. Beim Wechsel zwischen den Darstellungen wird jeweils die nicht ausgewählte Darstellung deaktiviert. Die Einfärbung nach Höhe bezieht sich auf die Höhen der Punkte des aktuellen Bereichs und wird für jeden Schnitt neu berechnet.

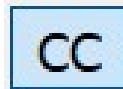


Abb. 21: Aktivieren der Einfärbung nach Klassen

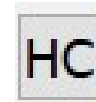


Abb. 22: Aktivieren der Einfärbung nach Höhe

6.1.7 Speichern und Reset der klassifizierten Punkte

Ist ein Bereich klassifiziert und sind die vergebenen Klassen sinnvoll, kann zu jedem Zeitpunkt der aktuelle Zustand gespeichert werden. Das Speichern erfolgt über den Save-Knopf in Abbildung 23. Wird während des Klassifizierens festgestellt, dass die vergebenen Klassen nicht sinnvoll sind, kann die Section in den ursprünglichen Zustand, der beim Laden vorlag, zurückgesetzt werden. Das Zurücksetzen erfolgt durch Betätigen des Reset-Knopfes aus Abbildung 23. Zu beachten ist, wenn die Section gespeichert wurde, ist ein Zurücksetzen in den Ausgangszustand nicht mehr möglich.



Abb. 23: Speichern und Zurücksetzen der klassifizierten Punkte

6.1.8 Navigation entlang der Bezugsachse

Nachdem ein Bereich vollständig klassifiziert wurde oder ein anderer Bereich entlang der Achse von größerem Interesse ist, kann mit den Knöpfen aus Abbildung 24 das Polygon in Längsrichtung der Achse verschoben werden. Wie weit das Polygon vor- oder zurück verschoben wird, bestimmt die Ausdehnung längs der Achse (Along). Wird das Polygon verschoben, so erfolgt ein Abspeichern der Section.



Abb. 24: Navigation in Längsrichtung der Achse

6.1.9 Vorhersage mit kNN

Vor der Verschiebung des Polygons kann über ein Drop-Down Menü (Abbildung 25) ausgewählt werden, ob eine Vorhersage der Klassen der nicht-klassifizierten Punkte der nächsten Section oder der vorherigen Section mittels kNN erfolgen soll. Es kann aus vier verschiedenen Möglichkeiten (*no prediction*, *predict next*, *predict previous*, *always predict*) ausgewählt werden. Wird *no prediction* ausgewählt, so erfolgt weder eine Vorhersage für die nächste Section noch für die vorherige Section. Bei der Auswahl der Möglichkeit *always predict* erfolgt eine Vorhersage für die nächste Section oder für die vorherige Section. Dies bestimmt die Nutzer:inn welche Navigationsrichtung gewählt wird. Ist *predict next* ausgewählt, so wird eine Vorhersage für die nächste Section erstellt. Bei *predict previous* wird eine Vorhersage für die vorherige Section berechnet.

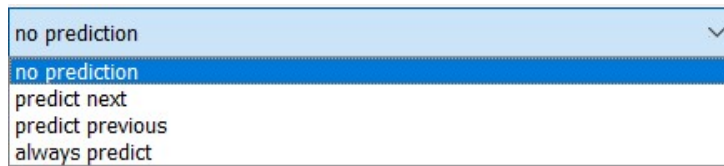


Abb. 25: Drop-Down Menü für kNN-Vorhersage

Wird keine Vorhersage der Klassen der nicht-klassifizierten Punkte im nächsten Abschnitt der Punktwolke gewünscht, werden die Punkte, wie diese im ODM gespeichert sind, geladen (Abbildung 26). Wenn einer der drei Fälle für eine Vorhersage, die in Abbildung 25 abgebildet sind, aktiviert ist, wird für all jene Punkte, die im nächsten Abschnitt nicht-klassifiziert sind, die jeweilige Klasse auf Basis des vorherigen klassifizierten Bereichs vorhergesagt. Die Vorhersage ist über eine kNN Suche realisiert. Es kommt vor, dass Punkten eine falsche Klasse vorhergesagt wird. In diesem Fall kann diesen Punkten mit einem der beiden Picker die richtige Klasse vergeben werden. Das Ergebnis einer solchen Vorhersage ist in Abbildung 27 abgebildet.

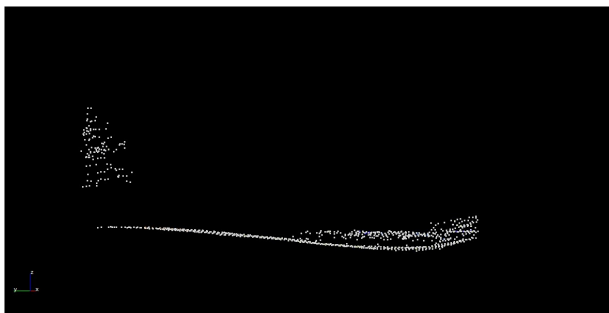


Abb. 26: Nächste Section ohne kNN Klassifizierung

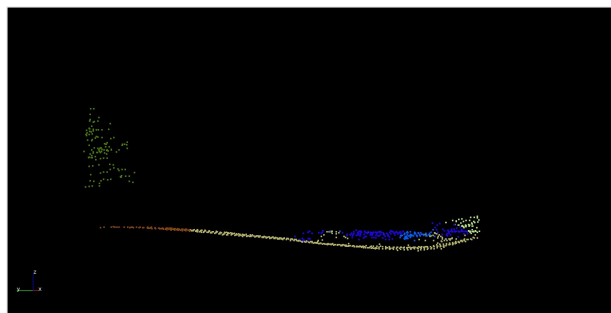


Abb. 27: Nächste Section mit kNN Klassifizierung

Bei einer Vorhersage der Klassen der nächsten Section wird zudem eine Statusmitteilung ausgegeben, in welcher mitgeteilt wird, für wie viele Punkte eine Vorhersage getroffen wurde (Abbildung 28).


```
Current station: 5.00 - 10.00
Loaded: 881 Points
Classified: 80 Points
Unclassified: 0 Points
Class histogram: {2: 76, 3: 52, 5: 118, 9: 66, 40: 420, 41: 149}
Class predicted: 801 Points
```

Abb. 28: Statusmitteilung für wie viele Punkte eine Klasse vorhergesagt wurde

6.1.10 Verschiedene Darstellungen

In Kapitel 5 wurden verschiedene Möglichkeiten angeführt, wie die Ansicht des aktuellen vertikalen Schnitts verändert werden kann. Die verschiedenen Möglichkeiten der Darstellung sind das Verschieben (Abbildung 29), das Vergrößern (Abbildung 30) und das Rotieren (Abbildung 32) der Punktwolke.

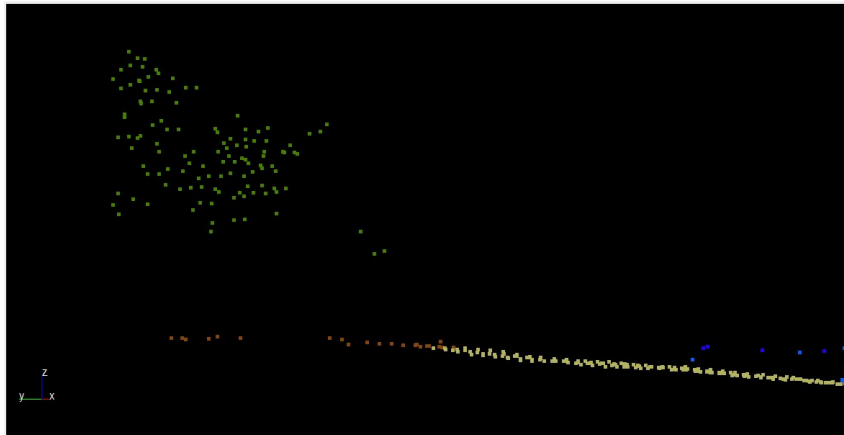


Abb. 29: Verschieben der Punktwolke

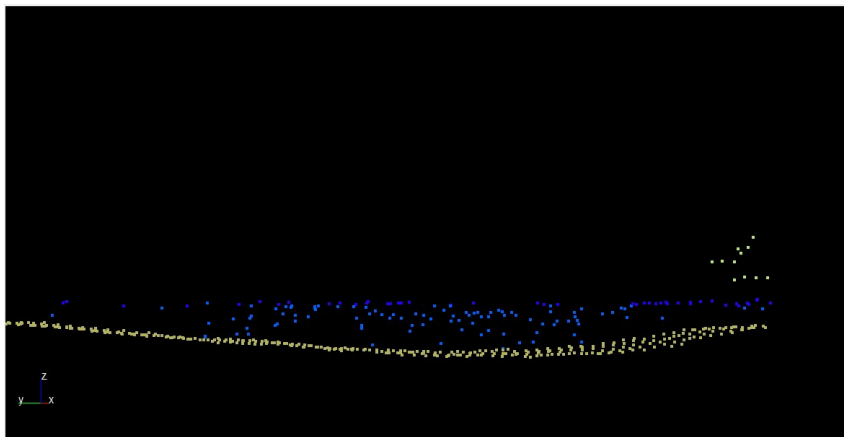


Abb. 30: Vergrößern der Punktwolke

Mit dem Knopf in Abbildung 31 wird die Section so gedreht, dass diese wieder orthogonal zur Achse ausgerichtet ist.

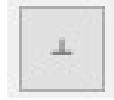


Abb. 31: Knopf für die orthogonale Ausrichtung der Section

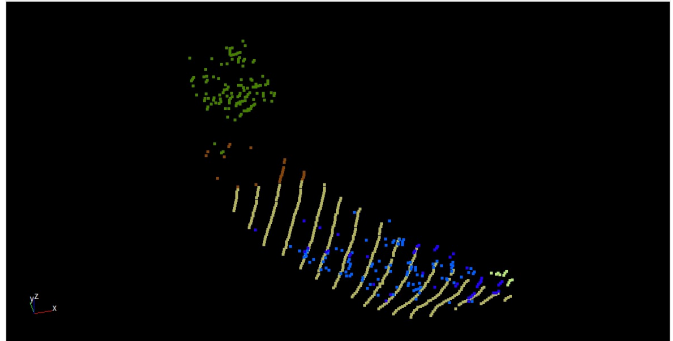


Abb. 32: Rotieren der Punktwolke

6.1.11 Endversion des Klassifikationstool

In Abbildung 33 ist das finale Tool zur Klassifizierung von Punktwolken abgebildet.

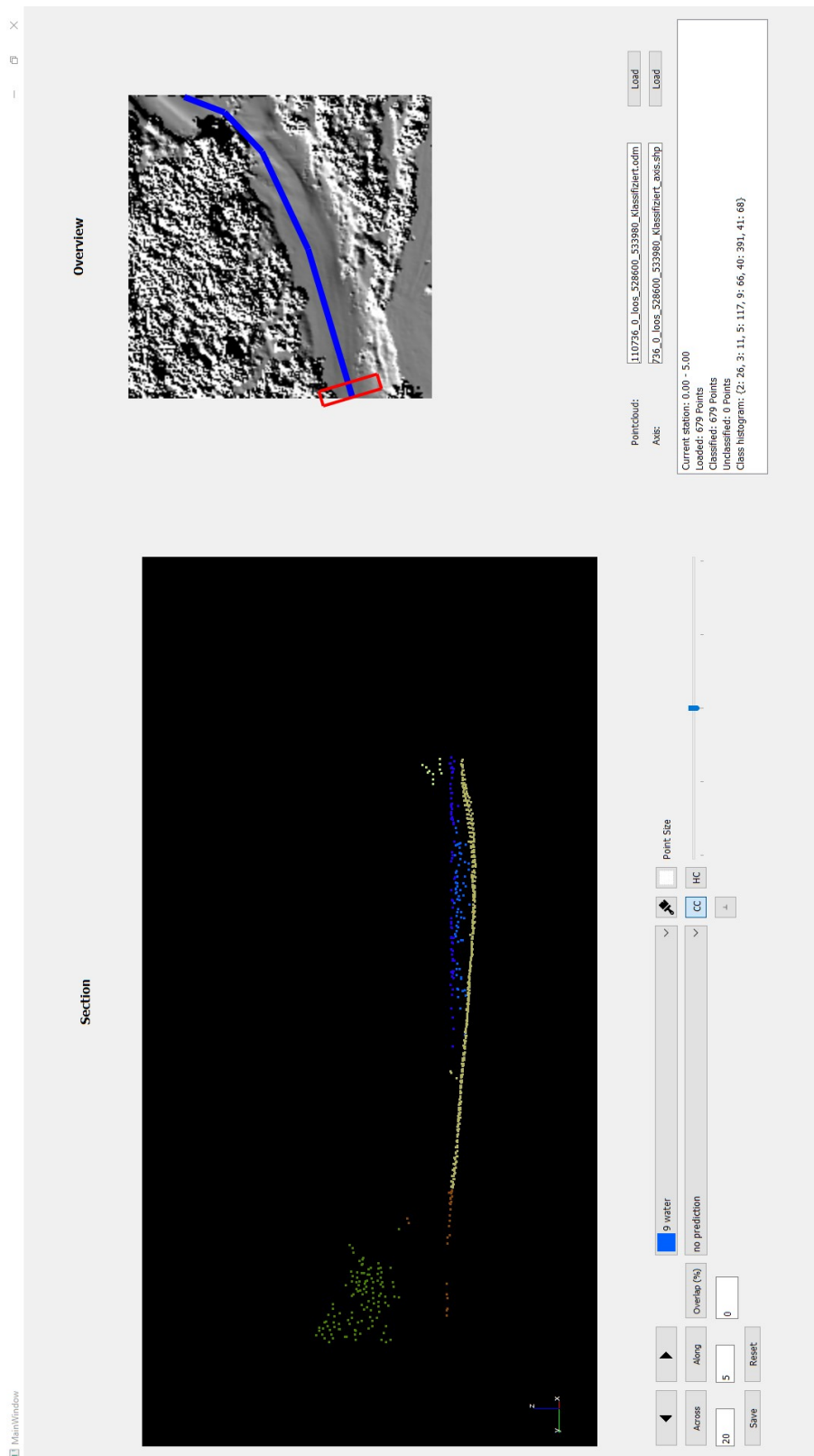


Abb. 33: Endversion des Tools für die Klassifizierung von Punktwolken

7 Diskussion

7.1 Eignung des Klassifikationstools

Mit diesem Klassifikationstool kann jede ALS-Punktwolke manuell klassifiziert werden. Im Vergleich zu gängigen Klassifikationsprogrammen wie zum Beispiel Pointview oder Cloud Compare, schafft es dieses Tool kleine Abschnitte der Punktwolke als vertikale Schnitte darzustellen. Diese Möglichkeit zur Darstellung ist von sehr großer Bedeutung für die Bathymetrie. Dadurch kann ein Gewässer in seine Bereiche Wasseroberfläche, Wassersäule und Gewässerboden eingeteilt und klassifiziert werden, zu sehen in den Abbildungen 12 und Abbildung 30.

Die Abschnitte, welche als vertikale Schnitte dargestellt werden, werden bestimmt durch die Lage der Achse. Zumeist wird die Achse entlang einer Flussachse, Straße oder dergleichen verlaufen. Zum anderen sind die vertikalen Schnitte von der Position und Größe des Polygons abhängig. Das ist sowohl ein Vorteil als auch ein Nachteil. Ein Vorteil, weil der Bereich, welcher aus der Punktwolke geladen wird, sehr gut für die jeweilige Anwendung eingrenzt werden kann, und nur der Bereich, welcher vom Polygon eingegrenzt ist, dargestellt wird. Ein Nachteil, weil wenn die gesamte Punktwolke klassifiziert werden soll, muss entweder das Polygon in der Querausdehnung so vergrößert werden, dass das Polygon alle Messpunkte erreicht. Eine andere Möglichkeit wäre, eine neue Achse zu definieren und entlang dieser zu klassifizieren. Da man sich in bathymetrischen Anwendungen hauptsächlich für das Monitoren und für Messungen von Gewässern interessiert, reicht eine Achse entlang der Gewässermittlinie aus. Durch Darstellung der Abschnitte als vertikale Schnitte können auch andere Klassen den Punkten eindeutiger zugeteilt werden. Zum Beispiel ist der Übergang von Gewässerboden zu trockenem Boden genauer feststellbar.

Die Größe des Polygons ist in diesem Tool von wichtiger Bedeutung, denn diese bestimmt, wie viele Punkte in einem Abschnitt dargestellt werden. Liegt eine geringe Ausdehnung in Längs- und Querrichtung der Achse vor, werden weniger Punkte dargestellt. Das hat einen positiven Einfluss auf die Ladezeit der Punkte. Da dieses Tool in Python entwickelt wurde, und Python zu den langsameren Programmiersprachen zählt, darf die Ladezeit nicht außer Acht gelassen werden. Ein weiterer Vorteil einer kleinen Geometrie, vor allem aber die Ausdehnung in Längsrichtung des Polygons ist es, dass die Wahrscheinlichkeit von Punkten, die sich gegenseitig verdecken sehr gering ist. Ein kleines Polygon hat allerdings auch Nachteile. Eine eindeutige Einteilung der Punkte in ihre Klassen kann in diesem Fall schwierig werden. Wird nur ein kleiner bzw. schmaler Streifen dargestellt, können Punkte, die einem Baum, Strauch oder ähnlichem zugehören, nicht eindeutig dieser Klasse zugeteilt werden, da es nicht eindeutig ist, ob es sich wirklich um einen Baum oder Strauch handelt. Eine Vorhersage der Klassen im nächsten Bereich wird, wenn schmale Streifen eingestellt sind, ungenauer, als wenn breitere Streifen vorliegen. Die Einfärbung der Punkte nach Höhe des aktuellen vertikalen Schnittes ist bis auf die Berechnungsdauer unabhängig von der Geometrie des Polygons.

7.1.1 Klassifizierungsvorgang

Für das Klassifizieren der Punkte wurden in diesem Tool zwei Möglichkeiten implementiert. Die Abmessungen des Polygons haben auf das Klassifizieren nur in Hinblick auf die Möglichkeit, dass Punkte von anderen Punkten verdeckt werden, einen Einfluss. Das Klassifizieren von einzelnen Punkten durch Anklicken ist in diesem Tool sehr zuverlässig. Einzig das Klassifizieren von verdeckten Punkten erfolgt in seltenen Fällen nicht genau genug. Diese Problemstellung kann zumeist durch Vergrößern des Bereichs gelöst werden. Wird ein größerer Bereich gewählt, welcher klassifiziert wird, besteht der größte Vorteil darin, dass mehrere Punkte durch einen Klick klassifiziert werden. Da in so einem Fall jedes Pixel in dem jeweiligen Bereich geprüft wird, dauert das Klassifizieren unterschiedlich lange. Das gilt sowohl für den dynamischen Picker als auch für den Markierungspicker.

Werden die beiden Picker miteinander verglichen, so haben beide Vor- und Nachteile. Mit dem

Markierungspicker können sehr gut Bereiche klassifiziert werden, die sehr lange und schmal sind. Bereiche wie zum Beispiel Bodennunkte oder Gewässerbodennunkte. Ist es notwendig, Punkte zu klassifizieren in einem Bereich wo zwei verschiedene Klassen ineinander verschwimmen und keine eindeutige Grenze zu sehen ist, erweist sich der Markierungspicker als zu ungenau. In so einem Fall wird mittels dynamischem Picker klassifiziert. Der dynamische Picker ermöglicht es, einzelne Punkte durch Anklicken zu klassifizieren oder ein Quadrat aufspannen und mit diesem zu klassifizieren. Entweder einen Bereich durch einmaliges Klicken zu klassifizieren oder durch das Gedrückthalten der linken Maustaste wie mit einem Pinsel zu klassifizieren. Für das Quadrat des dynamischen Pickers gilt dasselbe wie für den Bereich, der mittels Markierungspicker abgedeckt wird. Je größer der zu klassifizierende Bereich ist desto mehr Laufzeit wird für das Zuordnen der richtigen Punkte der Klasse benötigt.

7.1.2 Verschieben entlang der Achse

Um die Punktwolke zu klassifizieren, muss das Polygon verschoben werden. Das Verschieben und Ausrichten des Polygons erfolgt in diesem Tool entlang einer Bezugsachse (Achse). Die Achse hat je nach Verlauf eine bestimmte Anzahl an Stützpunkten. Die Anzahl der Stützpunkte bestimmt, wie kontinuierlich der Verlauf der Bezugsachse ist, und somit auch wie kontinuierlich die Ausrichtung des Polygons erfolgt. Der Verlauf der Bezugsachse hängt vom Verlauf des Flusses oder der Straße ab, entlang der die Bezugsachse gezogen wird. Ein kontinuierlicher Verlauf ist wünschenswert, da die Ausrichtung des Polygons am Verlauf der Bezugsachse berechnet wird. Hat die Bezugsachse an manchen Stellen eine sehr starke Richtungsänderung, wie zum Beispiel an Flussschlingen, so wird das Polygon an diesen Stellen sehr sprunghaft ausgerichtet. Die Problematik an einer sprunghaften Ausrichtung ist, dass es unmöglich ist, alle Punkte in diesem Bereich mit dem Polygon zu erfassen und zu klassifizieren.

Diese Problematik kann mit Hilfe von Splines gelöst werden. Bei Splines werden Ecken von Polylinien in einen glatten Kurvenverlauf umgewandelt. Mit einer glatten Kurve können alle Punkte in diesem Bereich mit dem Polygon erfasst und klassifiziert werden. In der aktuellen Version, die mit dieser Bachelorarbeit realisiert wurde, kann dieses Problem reduziert werden, indem in solchen Bereichen der Punktwolke mehrere Stützpunkte der Bezugsachse definiert werden. Dadurch wird die sprunghafte Richtungsänderung ein wenig gedämpft.

7.2 Ausblick

Die zu Beginn gestellte Forschungsfrage, ob es möglich ist ein einfaches Tool zur Klassifikation von ALS-Punktwolke zu entwickeln, konnte zufriedenstellend beantwortet werden. Mit dem vorliegenden Tool können 3D Punktwolken klassifiziert werden. Diese Bachelorarbeit bietet eine gute Grundlage, um das Tool weiterzuentwickeln und zu verbessern. Da der Hauptgrafikbereich, in welchem die vertikalen Schnitte dargestellt werden digital ist, kann eine Übersichtsgrafik, in welcher die Schummerung und Achse dargestellt sind, in der Weiterentwicklung von diesem Tool digitalisieren werden. Dadurch wird es möglich werden, den Verlauf der Achse direkt in dem Tool zu definieren. Es ist dann nicht mehr notwendig, in ein externes Programm zu wechseln, um den Verlauf zu definieren. Neben der direkten Definition der Bezugsachse kann auch der Startpunkt des Polygons, durch Klicken auf die Bezugsachse festgelegt werden.

Der Verlauf der Bezugsachse kann, wie in 7.1.2 diskutiert, sehr sprunghaft sein, was zu Problemen führt. In einer Weiterentwicklung des Tools kann diese Problemstellung durch Splines behoben werden. Dabei werden sprunghafte Richtungsänderungen der Bezugsachse in glatte kontinuierliche Kurven umgewandelt. Die Punkte, die als vertikaler Schnitt dargestellt werden, können unterschiedlich eingefärbt werden, zum einen in die Farben der jeweiligen Klasse und zum anderen in einem Höhenverlauf. Für eine bessere Einordnung der Punkte, welche nicht eindeutig einer Klasse zuzuordnenden sind, kann es hilfreich sein, weitere Farbdarstellungen

zu implementieren. Eine Möglichkeit wäre die Einfärbung nach Echonummer oder Echokategorie. Denn es ist sinnvoll zwischen First-, Intermediate- und Lastecho zu unterscheiden. Ein ausgesandtes Lasersignal kann mehrmals zurückgestreut werden, wenn es an unterschiedlichen Objekten reflektiert wird, wie zum Beispiel bei Vegetation. Das Firstecho stammt zum Beispiel von einer Baumkrone, das Intermediateecho von den darunter liegenden Ästen und Blättern und das Lastecho stammt typischerweise vom Boden. Mit einer Einfärbung nach Echonummer kann eine bessere Unterscheidung zwischen hoher Vegetation, niedriger Vegetation oder Boden gemacht werden. Eine weitere Möglichkeit ist das Einfärben nach der Amplitude der detektierten Signale.

Mit der aktuellen Version können 3D Punktwolken sehr gut klassifiziert werden. In der Weiterentwicklung können weitere Features und Funktionen eingebaut werden, die das Klassifizieren vereinfachen und weitere Informationen über die jeweilige Punktwolke liefern.

Abbildungsverzeichnis

1	ALS - System [17]	5
2	Prinzipskizze ALS/ALB [10]	6
3	Level 0 des ODM [13]	9
4	KD-Baum [13]	10
5	Ablaufdiagramm	12
6	Koordinatensysteme OpenGL (links) und PyQt (rechts)	14
7	Einlesen der ALS-Daten und der Achse	26
8	Darstellung des Shadings, der Achse und des Polygons	27
9	Darstellung der ersten Section	27
10	Statusmessages	28
11	Schieberegler für Punktgröße	28
12	Punkte mit eingestellter Punktgröße 5	28
13	Anpassung der Polygoneigenschaften Across, Along, Overlap	29
14	Auswahl der Klassen mittels Drop-Down-Menü	29
15	Dynamischer Picker	30
16	Auswahl der Punkte mit dynamischem Picker	30
17	Markierungspicker	30
18	Auswahl der Punkte mit Markierungspicker	30
19	Einfärbung nach Klassen	31
20	Einfärbung nach Höhe	31
21	Aktivieren der Einfärbung nach Klassen	31
22	Aktivieren der Einfärbung nach Höhe	31
23	Speichern und Zurücksetzen der klassifizierten Punkte	31
24	Navigation in Längsrichtung der Achse	32
25	Drop-Down Menü für kNN-Vorhersage	32
26	Nächste Section ohne kNN Klassifizierung	32
27	Nächste Section mit kNN Klassifizierung	32
28	Statusmitteilung für wie viele Punkte eine Klasse vorhergesagt wurde	33
29	Verschieben der Punktwolke	33
30	Vergrößern der Punktwolke	33
31	Knopf für die orthogonale Ausrichtung der Section	34
32	Rotieren der Punktwolke	34
33	Endversion des Tools für die Klassifizierung von Punktwolken	35

Tabellenverzeichnis

1	Standard LIDAR Punkt Klassen [6][S.98]	7
2	Angewendete OPALS Module [9]	8
3	Standard Farbpalette	19
4	Hexadezimal in Dezimal Beispiel	22
5	Parameter für kdtree-Objekt	25

Abkürzungen

OPALS Orientation and Processing of Airborne Laser Scanning

ALS Airborne Laser Scanning

ALB Airborne LIDAR Bathymetry

ODM	OPALS Data Manager
DHM	digitales Höhenmodell
DOM	digitales Oberflächenmodell
DGM	digitales Geländemodell
GUI	Graphical User Interface
GNSS	Global Navigation Satellite System
IMU	Integrated Measurement Unit
kNN	k Nearest Neighbours

Literatur

- [1] A. P. Charaniya, R. Manduchi und S. K. Lodha. „Supervised parametric classification of aerial lidar data“. In: *2004 Conference on Computer Vision and Pattern Recognition Workshop*. IEEE. 2004, S. 30–30.
- [2] Frank Warmerdam, Even Rouault, and others. *GDAL*. URL: <https://gdal.org/index.html> (Zugriff am 21.02.2024).
- [3] B. Höfle, W. Mücke, M. Dutter, M. Rutzinger und P. Dorninger. „Detection of building regions using airborne LiDAR—A new combination of raster and point cloud based GIS methods“. In: *GI-Forum 2009-International Conference on Applied Geoinformatics, Salzburg*. accepted. 2009.
- [4] *Hyperebene*. URL: <https://de.wikipedia.org/wiki/Hyperebene> (Zugriff am 06.01.2024).
- [5] S. Jany. „Monitoring mittels Airborne Laserscanning—Projektbeispiele über 20 Jahre aus der Bundesrepublik Deutschland“. In: (2020). DOI: doi.org/10.15488/9346.
- [6] Lewis Graham. *The LAS 1.4 Specification*. URL: https://www.asprs.org/wp-content/uploads/2010/12/LAS_Specification.pdf (Zugriff am 26.02.2024).
- [7] H. J. Lüthy. *Entwicklung eines Qualitätsmodells für die Generierung von Digitalen Geländemodellen aus Airborne Laser Scanning*. Bd. 95. 95. ETH Zurich, 2008.
- [8] G. Mandlbürger, B. Höfle, C. Briese, C. Ressler, J. Otepka, M. Hollaus und N. Pfeifer. „Topographische Daten aus Laserscanning als Grundlage für Hydrologie und Wasserwirtschaft“. In: *Österreichische Wasser- und Abfallwirtschaft* 61.7 (2009), S. 89–97.
- [9] G. Mandlbürger, J. Otepka, W. Karel, W. Wagner und N. Pfeifer. „Orientation And Processing Of Airborne Laser Scanning Data (OPALS) - Concept And First Results Of A Comprehensive Als Software“. In: *IAPRS XXXVIII* (2009), S. 55–60. ISSN: 1682-1750.
- [10] G. Mandlbürger, M. Pfennigbauer und N. Pfeifer. „Analyzing near water surface penetration in laser bathymetry—A case study at the River Pielach“. In: *ISPRS annals of the photogrammetry, remote sensing and spatial information sciences* 2 (2013), S. 175–180.
- [11] Manfred Moitzi. *svgwrite*. URL: <https://pypi.org/project/svgwrite/> (Zugriff am 21.02.2024).
- [12] *NumPy*. URL: <https://numpy.org/devdocs/> (Zugriff am 23.01.2024).
- [13] OPALS. *Orientation and Processing of Airborne Laser Scanning data*. URL: <https://opals.geo.tuwien.ac.at/html/stable/index.html> (Zugriff am 05.01.2024).

- [14] J. Otepka, G. Mandlbürger und W. Karel. „THE OPALS DATA MANAGER - EFFICIENT DATA MANAGEMENT FOR PROCESSING LARGE AIRBORNE LASER SCANNING PROJECTS“. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* I-3 (2012), S. 153–159. DOI: 10.5194/isprsannals-I-3-153-2012. URL: <https://isprs-annals.copernicus.org/articles/I-3/153/2012/>.
- [15] N. Pfeifer, G. Mandlbürger, J. Otepka und W. Karel. „OPALS – A framework for Airborne Laser Scanning data analysis“. In: *Computers, Environment and Urban Systems* 45 (2014), S. 125–136. ISSN: 0198-9715. DOI: <https://doi.org/10.1016/j.compenvurbsys.2013.11.002>. URL: <https://www.sciencedirect.com/science/article/pii/S0198971513001051>.
- [16] *Plotly*. URL: <https://plotly.com/> (Zugriff am 23.01.2024).
- [17] A. Roman und T. Ursu. „Multispectral satellite imagery and airborne laser scanning techniques for the detection of archaeological vegetation marks“. In: Dez. 2016, S. 141–152. ISBN: 978-606-543-787-6.
- [18] *Shapefiles*. URL: <https://enterprise.arcgis.com/de/portal/latest/use/shapefiles.htm> (Zugriff am 29.04.2024).
- [19] The Khronos® Group Inc. 2024. *OpenGL Overview*. URL: <https://www.khronos.org/api/opengl> (Zugriff am 23.02.2024).
- [20] The Qt Company Ltd. *Qt for Python*. URL: <https://doc.qt.io/qtforpython-6/index.html> (Zugriff am 22.01.2024).
- [21] M. Vetter, B. Höfle, G. Mandlbürger und M. Rutzinger. „Ableitung von flusssohlenmodellen aus Flussquerprofilen und Integration in Airborne Laserscanning Geländemodelle mit GRASS GIS“. In: *Angewandte Geoinformatik* (2008), S. 382–391.
- [22] W. Wagner, A. Ullrich, V. Ducic, T. Melzer und N. Studnicka. „Gaussian decomposition and calibration of a novel small-footprint full-waveform digitising airborne laser scanner“. In: *ISPRS journal of Photogrammetry and Remote Sensing* 60.2 (2006), S. 100–112.

Anhang A - GitHub-Link zu Pythonskript

In dem nachfolgendem Git-Hub Repository wurde das Klassifikations Tool entwickelt:
<https://github.com/FelixMeix/classificationtool>

In diesem Git-Hub Repository ist die aktuellste Version des Tools zu finden:
https://github.com/TUW-GEO/opals_PointCloudLabeler