TODOs

Masterarbeit

Berechnung charakteristischen Eigenschaften von botanischen Bäumen mithilfe von 3D-Punktwolken.

Name:	**Anton Wetzel**
E-Mail:	anton.wetzel@tu-ilmenau.de
Matrikelnummer:	60451
Studiengang:	Informatik Master
Betreuer:	**Tristan Nauber**
E-Mail:	tristan.nauber@tu-ilmenau.de
Professor:	**Prof. Dr.-Ing. Patrick Mäder**
E-Mail:	patrick.maeder@tu-ilmenau.de
Fachgebiet:	Data-intensive Systems and Visualization Group
Datum:	27.11.2023

TECHNISCHE UNIVERSITÄT
ILMENAU

Inhaltsverzeichnis

1. Glossar

Todo: Glossar

1.1. Punktwolken

Koordinatensystem ist eine Menge von Achsen, mit den eine Position genau beschrieben werden kann. Im Normalfall werden kartesische Koordinaten verwendet, welche so orientiert sind, dass die x-Achse nach rechts, die y-Achse nach oben und die z-Achse nach hinten zeigt.

Punkt ist eine dreidimensionale Position, welcher zusätzlichen Informationen zugeordnet werden können.

Punktwolke ist eine Menge von Punkten. Für alle Punkte sind die gleichen zusätzlichen Informationen vorhanden.

Normale ist ein normalisierter dreidimensionaler Vektor, welcher die Orientierung einer Oberfläche von einem Objekt angibt. Der Vektor ist dabei orthogonal zur Oberfläche, kann aber in das Objekt oder heraus gerichtet sein.

1.2. Scanner

Arial ...

Terrestrial ...

... ...

1.3. Datenstrukturen

Voxel ist ein Würfel im dreidimensionalen Raum. Die Position und Größe vom Voxel kann explicit abgespeichert oder relative zu den umliegenden Voxeln bestimmt werden.

Tree ist eine Datenstruktur bestehend aus Knoten, welche wiederum Kinderknoten haben können. Die Knoten selber können weitere Informationen enthalten.

Octree ist ein Tree, bei dem ein Knoten acht Kinderknoten haben kann. Mit einem Octree kann ein Voxel aufgeteilt werden. Jeder Knoten gehört zu einem Voxel, welcher gleichmäßig mit den Kinderknoten weiter unterteilt wird. Eine Veranschaulichung ist in Abbildung 1 gegeben.

Quadtree ist ein Tree, bei dem ein Knoten vier Kinderknoten haben kann. Statt eines Voxels bei einem Octree, kann ein Quadtree ein zweidimensionales Quadrat unterteilen. Eine Veranschaulichung ist in Abbildung 2 gegeben.

Leaf-Knoten ist ein Knoten, welcher keine weiteren Kinderknoten hat. Für Punktwolken gehört jeder Punkt zu genau einem Leaf-Knoten.

Branch-Knoten ist ein Knoten, welcher weitere Kinderknoten hat.

Root-Knoten ist der erste Knoten im Tree, alle anderen Knoten sind direkte oder indirekte Kinderknoten vom Root-Knoten.

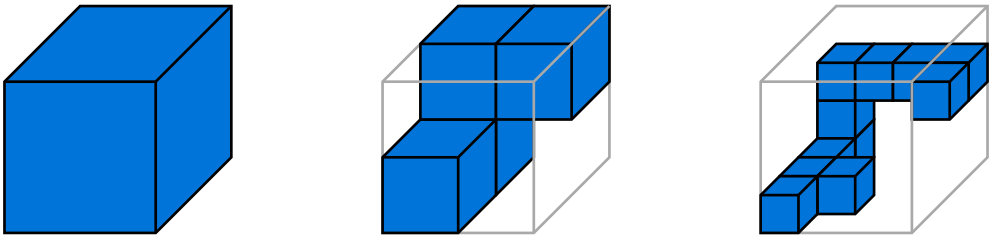


Abbildung 1: Unterschiedliche Stufen von einem Octree.

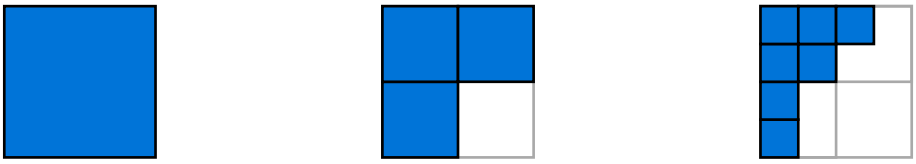


Abbildung 2: Unterschiedliche Stufen von einem Quadtree.

1.4. Akronyme

Todo: Akronyme

I. Überblick

1. Punktwolke

- Menge von Punkten
- mindestens Position

2. Testdaten

Der benutze Datensatz [1] beinhaltet 12 Hektar Waldfläche in Deutschland, Baden-Württemberg. Die Daten sind mit Laserscans aufgenommen, wobei die Scans von Flugzeugen, Drohnen und vom Boden aus durchgeführt wurden. Dabei entstehen 3D-Punktwolken, welche im komprimiert LAS Dateiformat gegeben sind.

Der Datensatz ist bereits in einzelne Bäume unterteilt. Zusätzlich wurden für 6 Hektar die Baumart, Höhe, Stammdurchmesser auf Brusthöhe und Anfangshöhe und Durchmesser der Krone gemessen. Mit den bereits bestimmten Eigenschaften können automatisch berechnete Ergebnisse validiert werden.

3. Ablauf

1. Diskretisieren
   - in 5cm große Voxel unterteilen
   - kann vollständig im Hauptspeicher sein
2. Segmente bestimmen
   - Top Down
   - Quadtree
   - nearest mit ...m max distance
3. Segmentieren (nochmal)
   - je nach Voxel zum Segment ordnen
4. Segmente analysieren
   - Beimeigenschaften
   - Punktgreigenschaften
5. Segmente abspeichern
6. gemeinsamer Octree erstellen
   - LOD für Visualisierung

Todo: Mehr Überblick

4. Stand der Technik

Punktwolken können mit unterschiedlichen Lidar Scanverfahren aufgenommen werden. Aufnahmen vom Boden oder aus der Luft bieten dabei verschiedene Vor- und Nachteile [2]. Bei einem Scan von Boden aus, kann nur eine kleinere Fläche abgetastet werden, dafür mit erhöhter Genauigkeit, um einzelne Bäume genau zu analysieren [3]. Aus der Luft können größere Flächen erfasst werden, wodurch Waldstücke aufgenommen werden können, aber die Datenmenge pro Baum ist geringer [4].

Nach der Datenerfassung können relevante Informationen aus den Punkten bestimmt werden, dazu gehört eine Segmentierung in einzelne Bäume [5] und die Berechnung von Baumhöhe oder Kronenhöhe [4].

Ein häufiges Format für Lidar-Daten ist das LAS Dateiformat [6]. Bei diesem werden die Messpunkte mit den bekannten Punkteigenschaften gespeichert. Je nach Messtechnologie können unterschiedliche Daten bei unterschiedlichen Punktwolken bekannt sein, aber die Position der Punkte ist immer gegeben. Aufgrund der großen Datenmengen werden LAS Dateien häufig im komprimierten LASzip Format [7] gespeichert. Die Kompression ist Verlustfrei und ermöglicht eine Kompressionsrate zwischen 5 und 15 je nach Eingabedaten.

LASTools [8] ist eine Sammlung von Software für die allgemeine Verarbeitung von LAS Dateien. Dazu gehört die Umwandlung in andere Dateiformate, Analyse der Daten und Visualisierung der Punkte. Durch den allgemeinen Fokus ist die Software nicht für die Verarbeitung von Waldteilen ausgelegt, wodurch Funktionalitäten wie Berechnungen von Baumeigenschaften mit zugehöriger Visualisierung nicht gegeben sind.

II. Berechnung

1. Separierung in Bäume

1.1. Diskretisieren

1. Punkte laden
2. Position diskretisieren (5cm) und zugehöriger Voxel berechnen
3. alle Voxel, welche Punkte enthalten abspeichern

1.2. Segmente bestimmen

1. Boden Bestimmen
   - tiefster Voxel in ...m × ...m
2. von oben nach unten in Scheiben segmentieren
   - für jeden Voxel den nächsten bereits Segmentieren Voxel mit Maximaldistanc ...m bestimmen
   - wenn kein Voxel gefunden, neues Segment anfangen
   - wenn Voxel gefunden gleiches Segment verwenden

1.3. Segmentieren

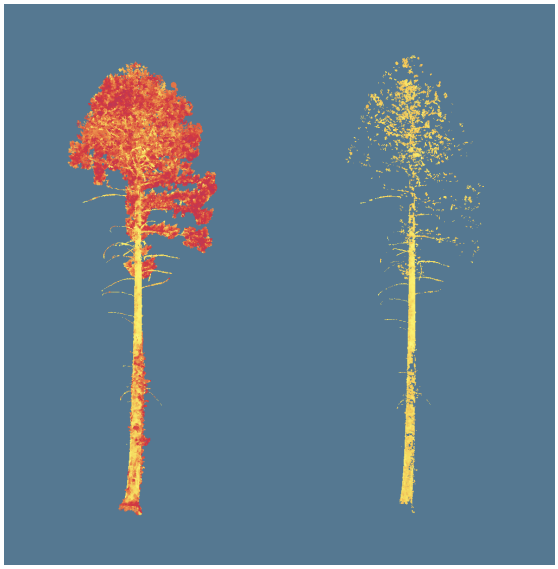
1. Punkt nochmal laden
2. Berechnen zu welchen Voxel der Punkt gehört
3. Segment vom Voxel zum Punkt zuordnen
4. Punkte im gleichem Segment zusammenfassten zu einer Punktwolke

2. Baumeigenschaften

Alle Punkte in einem Segment (Baum) sind verfügbar

2.1. Krümmung

1. Hauptkomponentenanalyse
   - λ_i mit $i \in \mathbb{N}_0^2$ und $\lambda_i > \lambda_j$ wenn $i > j$
2. $c\frac{3\lambda_2}{\lambda_0+\lambda_1+\lambda_2}$
   - $c \in [0, 1]$



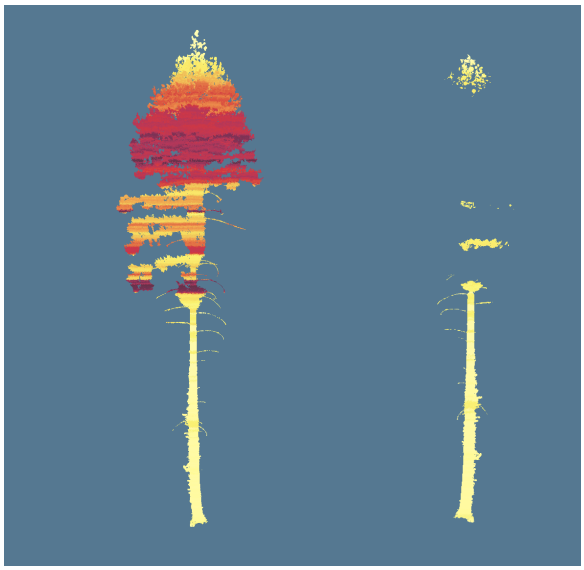
Todo: Background Option für weißen Hintergrund für Bilder

2.2. Punkthöhe

1. $h = \frac{p_y - y_{\min}}{y_{\max} - y_{\min}}$
   - $h \in [0, 1]$

2.3. Varianz in Scheibe

1. 5 cm Scheiben
2. geometrischen Schwerpunkt berechnen
3. Varianz v berechnen
4. $x = \frac{v_i}{v_{\max}}$
   - $x \in [0, 1]$



Todo: Mehr Baumeigenschaften

3. Segmentierung von einem Baum

Todo: Baumeigenschaften + ? → Segmente

4. Eigenschaften für Visualisierung

4.1. Normale

1. Hauptkomponentenanalyse
2. Eigenvektor für λ_2

4.2. Punktgröße

1. Durchschnittliche Abstand zu umliegenden Punkten
2. Ausgleichsfaktor?

5. Baumart

Todo: Segmente + Eigenschaften + ? → Klassifizierung?

- out of scope?
- neural?

III. Meshing

Todo: Meshing

IV. Visualisierung

1. Technik

- Rust
- WebGPU (wgpu)
- native Window (website?)
- LAS/LAZ

2. Punkt

2.1. Dreieck

Als Basis für einen Punkt wird ein Dreieck gerendert. Das Dreieck hat die Eckpunkte $(-1.73, -1)$, $(1.73, -1)$ und $0, 2$, wodurch der Kreis mit Radius 1 und Zentrum $(0, 0)$ vollständig im Dreieck liegt.

Todo: Warum die Ecken, schöner aufschreiben

Für jeden Punkt wird mit der Position p, Normalen n und Größe s die Position der Eckpunkte vom Dreieck im dreidimensionalen Raum bestimmt. Dafür werden zwei Vektoren bestimmt, welche paarweise zueinander und zur Normalen orthogonal sind.

Für den erste Vektor a wird mit der Normalen $n = \left(n_x, n_y, n_z\right)$ das Kreuzprodukt $a = \left(n_x, n_y, n_z\right) \times \left(n_y, n_z, -n_x\right)$ bestimmt. Weil $|n| > 0$ ist, sind $\left(n_y, n_z, -n_x\right)$ und n nicht gleich. a muss noch für die weiteren Berechnungen normalisiert werden.

Für den zweiten Vektor b wird das Kreuzprodukt $b = n \times a$ bestimmt. Weil das Kreuzprodukt zweier Vektoren orthogonal zu beiden Vektoren ist, sind n, a und b paarweise orthogonal.

Todo: Bild Vektoren und Kreuzprodukt

Die Vektoren a und b spannen eine Ebene auf, welche orthogonal zu n ist. Für den Eckpunkt i vom Dreieck mit den Koordinaten (x, y), wird die Position $p_i = p + a * x * s + b * y * s$ berechnet werden.

2.2. Vergleich zu Quad

Todo: Vergleich Quad oder Dreieck als Basis

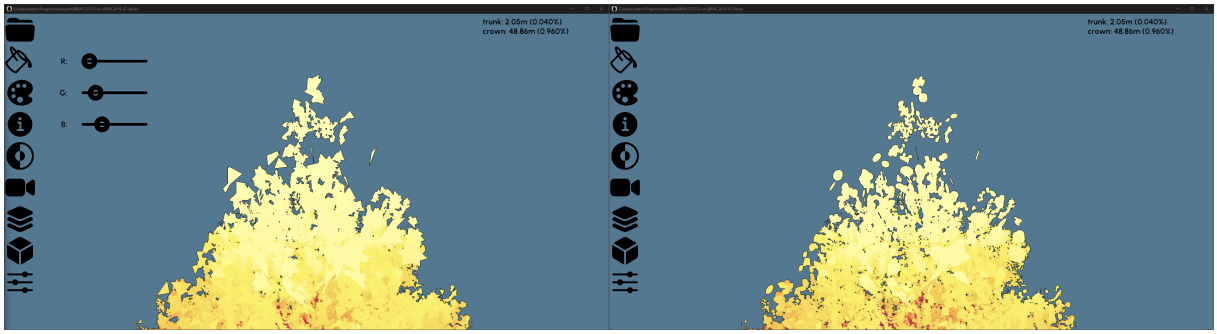
- triangle als Basis
  - weniger Verticies (3 zu 6)
  - mehr Fragment (4.0 zu 5.19615242270663)
  - schneller als Quad

2.3. Instancing

Weil für alle Punkte das gleiche Dreieck als Basis verwendet wird, muss dieses nur einmal zur Grafikkarte übertragen werden. Mit **Instancing** wird das gleiche Dreieck für alle Punkte verwendet, während nur die Daten spezifisch für einen Punkt sich ändern.

2.4. Kreis

Die Grafikpipeline bestimmt alle Pixel, welche im transformierten Dreieck liegen. Für jeden Pixel kann entschieden werden, ob dieser im Ergebnis gespeichert wird. Dafür wird bei den Eckpunkten die untransformierten Koordinaten abgespeichert, dass diese später verfügbar sind. Für jeden Pixel wird von der Pipeline die interpolierten Koordinaten berechnet. Nur wenn der Betrag der interpolierten Koordinaten kleiner 1 ist, wird der Pixel im Ergebnis abgespeichert.

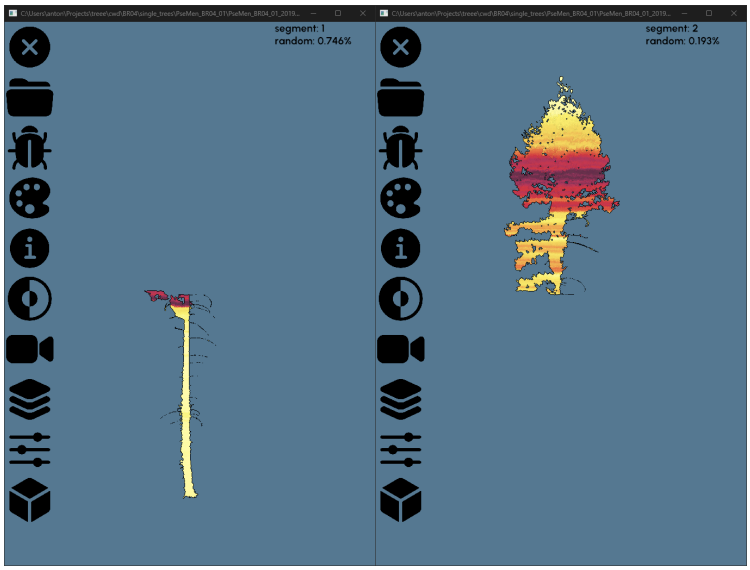


Todo: Bilder Crop

3. Dynamische Eigenschaft

- eigenschaften als 32 bit unsigned integer
- look up table für farbe basierend auf eigenschaftwert

4. Subpunktwolken (Bäume)



Note: Oben/Unten Teilung in 2 Segmente für Debug

4.1. Selektion (Raycast)

- von root bis leaf
- bestimme intersection mit knoten
- wenn leaf mit intersection gefunden
  - lade Segmente, welche im Leaf liegen
  - bestimme Abstand Ray-Punkt
  - wenn kleiner als Radius hit
  - Segment mit Punkt mit geringstem Abstand Ergebnis

4.2. Anzeige

- segmente seperate abgespeichert
  - keine LOD Stufen, nur Originalpunkte
  - maxbuffersize?

5. Eye-Dome-Lighting

Um die Punktwolke auf Anzuzeigen, werden die Punkte aus dem dreidimensionalen Raum auf den zweidimensionalen Monitor projiziert. Dabei gehen die Tiefeninformationen verloren. Mit der Rendertechnik **Eye-Dome-Lighting** werden die Kanten von Punkten hervorgehoben, bei denen die Tiefe sich stark ändert.

Todo: Vergleichsbild

Beim Rendern von 3D-Scenen wird für jeden Pixel die momentane Tiefe vom Polygon an dieser Stelle gespeichert. Das wird benötigt, dass bei überlappenden Polygonen das nähere Polygon an der Kamera angezeigt wird. Nachdem die Scene gerendert ist, wird

mit den Tiefeninformationen für jeden Pixel der Tiefenunterschied zu den umliegenden Pixeln bestimmt.

Todo: Farb- und Tiefenbild

Je größer der Unterschied ist, desto stärker wird der Pixel im Ergebnisbild eingefärbt. Dadurch werden Kanten hervorgehoben, je nachdem wie groß der Tiefenunterschied ist. Für den Effekt kann die Stärke und die Farbe angepasst werden.

Todo: Stark und Schwache Effekt Bild

6. Detailstufen

Je nach Scannertechnologie und Größe des abgetasteten Gebietes kann die Punktwolke unterschiedlich viele Punkte beinhalten. Durch Hardwarelimitierungen ist es nicht immer möglich alle Punkte gleichzeitig anzuzeigen, während eine interaktive Wiedergabe gewährleistet ist.

Besonders für weit entfernte Punkt ist es nicht notwendig, alle Punkte genau wiederzugeben. Deshalb wird für weit entfernte Punkte eine vereinfachte Version angezeigt. Diese besteht aus weniger Punkten und benötigt dadurch weniger Ressourcen, bietet aber eine gute Approximation der ursprünglichen Daten.

Todo: Vergleich alle Punkt und vereinfachte Versionen

Für die gesamte Punktewolke wird ein Octree mit den Punkten erstellt. Der zugehörige Voxel vom Root-Knoten wird so gewählt, dass alle Punkte im Voxel liegen. Rekursiv wird der Voxel in acht gleichgroße Voxel geteilt, solange in einem Voxel noch zu viele Punkte liegen. Bei dem Octree gehört jeder Punkt zu genau einem Leaf-Knoten.

Für jeden Branch-Knoten wird eine Punktwolke berechnet, welche als Vereinfachung der Punkte der zugehörigen Kinderknoten verwendet werden kann. Dafür wird de Algorithmus aus Abschnitt 6.1 verwendet.

Beim anzeigen wird vom Root-Knoten aus zuerst geprüft, ob der momentane Knoten von der Kamera aus sichtbar ist. Für die Knoten wird mit den Algorithmen aus Abschnitt 6.2 entschieden, ob die zugehörige vereinfachte Punktwolke gerendert oder der gleiche Algorithmus wird für die Kinderknoten wiederholt wird.

6.1. Berechnung der Detailstufen

Todo: Kopiert vom Fachpraktikum

Die Detailstufen werden wie bei „Fast Out-of-Core Octree Generation for Massive Point Clouds“ [9] von den Blättern des Baumes bis zur Wurzel berechnet. Dabei wird als Eingabe für einen Knoten die Detailstufen der direkten Kinder verwendet. Als Anfang werden alle originalen Punkte in einem Blatt als Eingabe benutzt.

Dadurch haben zwar Berechnungen der gröberen Detailstufen für Knoten näher an der Wurzel nur Zugriff auf bereits vereinfachte Daten, dafür müssen aber auch viel weniger Punkte bei der Berechnung betrachtet werden. Solange die Detailstufen eine gute Vereinfachung der ursprünglichen Punkte sind, kann so der Berechnungsaufwand stark verringert werden.

Der Voxel, welcher zu dem Knoten gehört, wird in gleich große Zellen unterteilt. Für jede Zelle mit Punkten wird ein repräsentativer Punkt bestimmt. Dafür wird für die Zelle die Kombination aller Eingabepunkte, welche in der Zelle liegen berechnet. Die Anzahl der Zellen ist dabei unabhängig von der Größe des ursprünglichen Voxels, wodurch bei gröberen Detailstufen durch den größeren Voxel auch die Zellen größer werden und mehr Punkte zusammengefasst werden.

6.2. Auswahl der Detailstufen?

6.2.1. Abstand zur Kamera

- Schwellwert
- Abstand zur kleinsten Kugel, die den Voxel inkludiert
- Abstand mit Größe des Voxels dividieren
- Wenn Abstand größer als Schwellwert
  - Knoten rendern
- sonst
  - Kinderknoten überprüfen

6.2.2. Auto

- wie Abstand zur Kamera
- messen wie lang rendern dauert
- Dauer kleiner als Mindestdauer
  - Schwellwert erhöhen
- Dauer kleiner als Maximaldauer
  - Schwellwert verringern

6.2.3. Gleichmäßig

- gleich für alle Knoten
- auswahl der Stufe

7. Kamera/Projektion

7.1. Kontroller

- bewegt Kamera
- kann gewechselt werden, ohne die Kameraposition zu ändern

7.1.1. Orbital

- rotieren um einem Punkt im Raum

- Kamera fokussiert zum Punkt
- Entfernung der Kamera zum Punkt variabel
- Punkt entlang der horizontalen Ebene bewegbar
- To-do: Oben-Unten Bewegung

7.1.2. First person

- rotieren um die Kamera Position
- Bewegung zur momentanen Blickrichtung
- Bewegungsgeschwindigkeit variabel
- To-do: Oben-Unten Bewegung

7.2. Projektion

7.2.1. Perspektive

- Projektion mit Field of View Kegel

7.2.2. Orthogonal?

Todo: Orthogonal?

7.2.3. Side-by-Side 3D?

8. Bedienung/Interface

Todo: Bedienung/Interface

Todo: Referenzen

Bibliographie

[1] H. Weiser *u. a.*, „Terrestrial, UAV-borne, and airborne laser scanning point clouds of central European forest plots, Germany, with extracted individual trees and manual forest inventory measurements“. [Online]. Verfügbar unter: https://doi.org/10.1594/PANGAEA.942856

[2] J. J. Donager, A. J. Sánchez Meador, und R. C. Blackburn, „Adjudicating perspectives on forest structure: how do airborne, terrestrial, and mobile lidar-derived estimates compare?“, *Remote Sensing*, Nr. 12, S. 2297, 2021.

[3] M. Disney, „Terrestrial LiDAR: a three-dimensional revolution in how we look at trees“, *New Phytologist*, Nr. 4, S. 1736–1741, 2019, doi: https://doi.org/10.1111/nph.15517.

[4] T. Suzuki, S. Shiozawa, A. Yamaba, und Y. Amano, „Forest Data Collection by UAV Lidar-Based 3D Mapping: Segmentation of Individual Tree Information from 3D Point Clouds“, *International Journal of Automation Technology*, S. 313–323, 2021, doi: 10.20965/ijat.2021.p0313.

[5] A. Burt, M. Disney, und K. Calders, „Extracting individual trees from lidar point clouds using treeseg“, *Methods in Ecology and Evolution*, Nr. 3, S. 438–445, 2019, doi: https://doi.org/10.1111/2041-210X.13121.

[6] L. Graham, „The LAS 1.4 specification“, *Photogrammetric engineering and remote sensing*, Nr. 2, S. 93–102, 2012.

[7] M. Isenburg, „LASzip: lossless compression of LiDAR data“, *Photogrammetric engineering and remote sensing*, Nr. 2, S. 209–217, 2013.

[8] C. Hug, P. Krzystek, und W. Fuchs, „Advanced Lidar Data Processing with Lastools“, 2004. Verfügbar unter: https://api.semanticscholar.org/CorpusID:14167994

[9] M. Schütz, S. Ohrhallinger, und M. Wimmer, „Fast Out-of-Core Octree Generation for Massive Point Clouds“, *Computer Graphics Forum*, Nr. 7, S. 155–167, 2020, doi: https://doi.org/10.1111/cgf.14134.