Master Thesis

im Rahmen des
Universitätslehrganges „Geographical Information Science & Systems“
(UNIGIS MSc) am Interfakultären Fachbereich für GeoInformatik (Z_GIS)
der Paris Lodron-Universität Salzburg

zum Thema

„Mit genetischem Algorithmus parametrisierte grossflächige Einzelbaumdetektion aus LiDAR Daten im Kanton Luzern“

vorgelegt von

Dipl. Inform. Adrian Simon Kuhn
105136, UNIGIS MSc Jahrgang 2018

Betreuer:
Dr. Christian Neuwirth

Zur Erlangung des Grades
„Master of Science (Geographical Information Science & Systems) – MSc(GIS)”

Luzern, 18. März 2020

Eidesstattliche Erklärung

Ich erkläre hiermit eidesstattlich durch meine eigenhändige Unterschrift, respektive durch die digitale Signatur im Falle einer elektronischen Fassung, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Alle Stellen, die wörtlich oder inhaltlich den angegebenen Quellen entnommen wurden, sind als solche kenntlich gemacht.

Die vorliegende Arbeit wurde bisher in gleicher oder ähnlicher Form noch nie als Bachelor-, Master-, Diplomarbeit oder Dissertation eingereicht.

Adrian Simon Kuhn

Digitally signed by Adrian Simon Kuhn
Date: 2020.03.23 13:08:26 +01'00'

Luzern, 18. März 2020 Adrian Simon Kuhn

Zusammenfassung

Bäume und Wälder sind aus ökologischer und ökonomischer Sicht ein wertvolles Gut. Als symbiotische Partner zur Fauna sind sie ein wesentlicher Bestandteil der Lebensgrundlage auf der Erde. Darüber hinaus nutzen die Menschen sie in vielerlei Form als Nahrungsquelle, als Schutzvorrichtung vor Naturgewalten, als Erholungsgebiete und als Rohstofflieferanten für eine Vielzahl diverser Produkte.
Mit den jüngst beobachteten klimatischen Veränderungen und den einhergehenden extremen Wetterereignissen sind Bäume zunehmendem Stress ausgesetzt. In der Schweiz müssen Bäume vermehrt mit heissen und trockenen Sommern klarkommen. Mit der Veränderung steigt die alpine Waldgrenze und Bäume können in neuen Gebieten Fuss fassen. Um die Veränderungen der Baumbestände sichtbar zu machen und über die Zeit zu dokumentieren, sollten Einzelbäume in Geodatensätzen erfasst werden. Dabei sind effiziente und automatisierte Verfahren zur grossflächigen Erfassung und Nachführung erforderlich.
Die vorliegende Arbeit befasst sich aus diesem Grund mit den möglichen Verfahren, wie aus LiDAR Daten Einzelbäume extrahiert werden können. Die methodische Fragestellung der Arbeit ist, ob es aus der Vielzahl verschiedener Verfahren zur Einzelbaumextraktion aus LiDAR Daten eines gefunden und adaptiert werden kann, sodass es erfolgreich auf grosser Fläche anwendbar ist. Das Ziel dabei ist, dies mit bereits vorhandenen zweckneutralen Daten und mit gängiger GIS-Infrastruktur zu realisieren und auf dem Gebiet des Kantons Luzern mit 1'493.51 Quadratkilometer in Form einer Ersterfassung praktisch anzuwenden.

Als Datengrundlage stand eine LiDAR Punktwolke aus dem Jahr 2018 mit einer mittleren Punktdichte von 16 Punkten pro Quadratmeter zur Verfügung. Zudem konnte auf ein davon abgeleitetes digitales Oberflächen- und Terrainmodell mit einer Auflösung von 0.25 Meter pro Pixel zurückgegriffen werden. Alle Produkte wurden in gekachelter Form mit einer Fläche von je einem Quadratkilometer aufbereitet.

Zur Evaluation der möglichen Verfahren wurden empirisch quadratische Testgebiete mit 100 Meter Kantenlänge und möglichst heterogenen Baumbeständen definiert. Die Baumkronenspitzen der Testgebiete wurden in manueller Arbeit digitalisiert und dienten als Referenzdatensätze. Für die optimale Parametrisierung der Verfahren wurde auf einen genetischen Algorithmus zurückgegriffen. Dabei wurden mittels evolutionären Prozesses nur diejenigen Lösungen weiterverfolgt, welche die Einzelbaumdetektion am besten lösen konnten.

Komplexe Verfahren, welche Einzelbäume direkt in der LiDAR Punktwolke extrahieren, waren nicht geeignet, grossflächige Gebiete zu prozessieren. Obwohl die Resultate solcher Verfahren besser waren, konnten sie aufgrund des enormen Rechenaufwands nicht auf die grosse Fläche hochskaliert werden. Der beste Kompromiss aus Rechenaufwand und Qualität lieferten Verfahren, welche das dreidimensionale Problem auf die Ebene projizierten und die Einzelbäume anhand Bildsegmentierungsverfahren extrahierten. Der dabei entwickelte *region-based*

marker-controlled watershed Algorithmus kann für geschlossenen Wald, respektive Gebiete über 1'000 Meter über Meer und einzelnstehenden Bäumen mit unterschiedlichen Parametern bestückt werden. Die optimalen Parameter wurden evolutionär nach 1'000 Generationen gefunden und der Algorithmus war in der Lage, die Bäume bei einer Lage- und Höhentoleranz von einem Meter mit einem durchschnittlichen F_1-Score von 0.72 zu erkennen. Damit schneidet er weniger gut ab als vergleichbare Verfahren von anderen Forschungsarbeiten, was mit der Heterogenität der Bäume begründbar ist. Am besten schnitt der Algorithmus in Gebieten ab, wo die Bäume vereinzelt stehen und belaubt sind. Gute Resultate liefert der Algorithmus auch für geschlossene Nadelwälder. Probleme bekundete das Verfahren insbesondere im Laubwald, wo in der geschlossenen Laubdecke keine Kronenspitzen auszumachen sind. Damit erhebt das Verfahren nicht den Anspruch, alle Bäume zu detektieren. Das Resultat ist vielmehr ein Punktdatensatz mit Einzelbäumen und markanten Bäumen in geschlossenen Wäldern.

Bei der praktischen Anwendung des Algorithmus auf das gesamte Kantonsgebiet wurden rund 11.5 Millionen Einzelbäume gefunden. Dank dem Fokus auf laufzeitoptimierten Code mit *numpy* Arrays und die Verwendung von parallelen Prozessen konnte der Einzelbaumdatensatz nach 37 Stunden berechnet werden. Damit wurde der Beweis erbracht, dass es ohne spezielle Hardware möglich ist, auf grossflächigem Gebiet Einzelbäume zu extrahieren. Der erstellte Punktdatensatz wird Einzug halten in die Sammlung der Geodatensätze des Kantons Luzern und wird in diversen Produkten zur Anwendung kommen.

Ein Anknüpfungspunkt zu der vorliegenden Arbeit ist die Ableitung von weiteren Baumparametern. So wurde darauf verzichtet, weitere Daten wie beispielsweise multispektrale Orthofotos hinzuzuziehen, um eine Unterscheidung zwischen Nadel- und Laubbäumen zu machen oder gar die Baumart zu bestimmen. Die Forschung diesbezüglich liefert vielversprechende Ansätze und es wäre spannend zu ermitteln, ob dies auch auf grossflächigem Gebiet in vernünftiger Rechenzeit möglich wäre.

Abstract

From an ecological and economic point of view trees and forests are valuable assets. As symbiotic partners to the fauna they are an essential part of earths ecosystem. Furthermore, people use trees in many different ways as a source of food, as protection against the forces of nature, as recreational areas and as a raw-material used in wide range of products.

With the recently observed climatic changes and the accompanying extreme weather events, trees are being exposed to more stress. In Switzerland, trees increasingly have to endure hot and dry summers. With the climate change, the alpine timberline is rising and trees can capture new areas. In order to make these changes in tree populations visible and documented over time, individual trees should be recorded in geodata. Efficient and automated procedures for large-scale recording and updating of such data are therefore required.

For this reason, this thesis outlines the possible methods of extracting individual trees from LiDAR data. The methodological question of this work is whether one of the many different procedures for extracting individual trees from LiDAR data can be found and adapted to processing large areas. The goal is to achieve this with already existing purpose-neutral data and within a common GIS infrastructure. Finally, it should be applied as a practiced case study in the area of the canton of Lucerne with 1'493.51 square kilometers.

The data set used in this study was a LiDAR point cloud from 2018 with an average point density of 16 points per square meter. Derived from this, a digital surface and terrain model with a resolution of 0.25 meter per pixel would be used. All products were rasterized with an area of one square kilometer per tile.

For the evaluation of possible tree extraction methods, squared test areas with an edge length of 100 meter with heterogeneous tree populations were empirically defined. The crown tops in these areas were digitized manually and served as reference data. A genetic algorithm was used for the optimal parameterization of the solution. By means of an evolutionary process, only the best performing individuals were selected for reproduction.

Complex procedures that extract single trees directly in the LiDAR point cloud were not applicable for processing large areas. Although the results of such methods would have been better at a smaller scale, they could not be scaled up to larger areas due to enormous computing effort. The best compromise between computational effort and quality was provided by methods simplifying the problem to two dimensions and extracting individual trees by image segmentation. A region-based marker-controlled watershed algorithm was developed which can be applied with different parameters for closed forests, areas higher than 1'000 meter above sea level and single isolated trees. The optimal parameters were found evolutionarily after 1'000 generations and the algorithm was able to identify the trees with an average F_1-

score of 0.72 at a position and height tolerance of one meter. It is not as good as comparable methods from other research, which can be explained by the heterogeneity of the trees in the given area. The algorithm is most accurate in areas with isolated leaf-on trees. The algorithm produces good results for coniferous forests as well. The algorithm showed problems especially in deciduous forests with no distinctive crown tops in the canopy height model. As a consequence, the method does not claim to detect all trees. The result is rather a point data set with single trees and prominent trees in closed forests.

11.5 million trees were found in the practical application of the algorithm on the entire cantonal territory. Thanks to the focus on runtime-optimized code with *numpy* arrays and the use of parallel processes, the single tree dataset could be calculated after 37 hours. This is the proof of concept for extracting single trees over a large area without special hardware. The created point data set will be included in the geodata collection of the Canton of Lucerne and is going to be used in various products.

A starting point for future work is the derivation of further tree parameters. It was decided not to use further data sources such as multispectral orthophotos to distinguish coniferous and deciduous trees or even to determine the tree species. Research in this area showing promising results and it would be exciting to find out whether this could be done on larger scales in reasonable computing time.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Studien zur Entwicklung des Waldbestandes in der Schweiz zeigen, dass die Gesamtfläche des Waldes zwischen 1880 und 2000 um 22% zugenommen hat (Ginzler et al. 2011). Aktuellere Zahlen belegen, dass die Ausdehnung der Schweizer Wälder bis heute anhält und sich in jüngster Zeit insbesondere in den Höhenlagen weiter beschleunigt hat (Abegg et al. 2014). Mögliche Gründe deckt eine Studie der Eidgenössischen Forschungsanstalt für Wald, Schnee und Landschaft WSL auf. Mit dem fortschreitenden Klimawandel und den höheren Durchschnittstemperaturen steigt die alpine Waldgrenze. Daneben müssen Wälder in tiefer gelegenen Gebieten mit ausgeprägteren Trockenperioden und extremeren Wetterlagen auskommen (Pluess et al. 2016).

Vegetation und insbesondere Bäume haben einen entscheidenden Einfluss auf lokale klimatische Bedingungen. Australische Wissenschaftler belegen, dass urbane Vegetation im Grossraum Perth die Oberflächentemperaturen insbesondere im Sommer zu senken vermag. Dabei haben Büsche und Bäume einen grösseren Kühleffekt als Gras (Duncan et al. 2019).

Bäume haben nicht nur den erwähnten ökologischen Nutzen, sondern erfüllen auch ökonomische Funktionen. Die WSL der Schweiz hat mit einer Umfrage bei Revierförstern ermittelt, dass über die Hälfte aller Wälder der Holzproduktion und etwa 43% der Wälder dem Schutz vor Naturgefahren (z.B. Bannwälder) dienen (Brändli und Denzler 2011). Zudem tragen insbesondere früchtetragende Obstbäume zur Existenz- und Ernährungssicherheit der landwirtschaftlichen Betriebe, respektive der Bevölkerung bei. Hochstammobstbäume werden gestützt auf das Landwirtschaftsgesetz und die Artikel 63 und 64 der Direktzahlungsverordnung im Kanton Luzern mit Qualitätsbeiträgen finanziell gefördert (lawa 2019).

1.1 Motivation

Die Wichtigkeit von Wäldern und Bäumen für ein gesundes Ökosystem, als Rohstoff- und Nahrungslieferanten ist folglich unbestritten. Mit der zunehmenden dynamischen Veränderung der Bestände ist es wichtig, dass Einzelbäume Einzug in Geodatensätze finden. Wenn es einen Geodatensatz gäbe, in welchem jeder einzelne Baum enthalten wäre, würde dies die Raum- und Städteplanung erleichtern und der Forst- und Landwirtschaft wichtige Informationen zu Holz-, respektive Ernteerträge liefern. Idealerweise wären die Bäume eines solchen Datensatzes mit Attributen wie Stammumfang, Baumhöhe, Kronenweite oder sogar der Baumart versehen.

Darüber hinaus spielen Einzelbäume bei der 3D Visualisierung eine wichtige Rolle. Schattenwürfe und Sichtlinien werden exakter und Landschaften wirken authentischer, wenn die Vegetation örtlich und botanisch korrekt abgebildet ist.

1.2 Ausgangslage

Die Abteilung Geoinformation der Dienststelle Raum und Wirtschaft (rawi) fungiert als Dienstleisterin sowohl für andere Dienststellen und Abteilungen des Kantons Luzern als auch für die Bevölkerung. Als GIS-Kompetenzzentrum werden Geodaten erhoben, zentral verwaltet und in eigenen Anwendungen und Produkten in Wert gesetzt. Zu den Themengebieten Bodenbedeckung, Landwirtschaft, Waldbestand, Waldfunktionen und Waldsoziologie existieren bereits umfassende und aktuelle Datenbestände (rawi Kanton Luzern 2020). Was zurzeit fehlt, ist ein Datensatz mit allen Einzelbäumen. Das Bundesamt für Landestopografie swisstopo stellt ein schweizweites topografisches Landschaftsmodell (TLM) zur Verfügung, welches einen Einzelbaumdatensatz enthält. Jener hat jedoch auf dem Kantonsgebiet von Luzern einen Nachführungsstand aus den Jahren 2012 und 2013 (Bundesamt für Landestopografie 2017).

Im Kanton Luzern arbeitet die Abteilung Geoinformation zurzeit an einer 3D-Strategie, wobei die Visualisierung der Einzelbäume eines von vielen Themen ist. Wenn man nebst den Bäumen in geschlossenen Wäldern auch Einzelbäume in urbanen Gebieten, Parkanlagen und Gärten dazu nimmt, werden manuelle Arbeiten zur Ersterhebung und periodischen Nachführung unrealistisch. Aus all diesen genannten Gründen besteht Bedarf nach einem automatisierten Verfahren zur Herstellung eines aktuellen Einzelbaumdatensatzes im Kanton Luzern.

1.3 Forschungsfrage und Ziele

In der Literatur gibt es ein breites Spektrum von Verfahren zur Einzelbaumdetektion aus LiDAR Daten. Die Herangehensweise, die Qualität der Resultate und die Laufzeiten dieser Verfahren unterscheiden sich dabei sehr stark. Oftmals sind die Verfahren nicht darauf ausgelegt, grosse Gebiete zu prozessieren und müssen mit viel Hintergrundwissen auf einen meist homogenen Baumbestand parametrisiert werden (Franceschi et al. 2018). Im Rahmen dieser Arbeit wird untersucht, ob bestehende automatisierte Methoden zur Einzelbaumdetektion auf ein grosses Gebiet mit heterogenem Baumbestand adaptiert und mit möglichst kurzer Laufzeit auf gängiger Bürohardware ausgeführt werden können.

Das Ziel ist, im ersten Schritt einen Überblick über die bewährten Verfahren und Methoden zu geben. Danach werden aus den möglichen Verfahren diejenigen herausgefiltert, deren Resultate und Laufzeiten geeignet sind, das Gebiet des Kantons Luzern zu prozessieren. Dabei wird auf genetische Programmierung aus der künstlichen Intelligenz zurückgegriffen, um mittels repräsentativen Testgebieten den geeigneten Algorithmus zu finden und mit optimalen Parametern zu bestücken. Der so ermittelte Algorithmus wird im letzten Schritt auf dem Kantonsgebiet angewendet, um als Resultat einen produktiv einsetzbaren Einzelbaumdatensatz des Kantons Luzern zu erhalten.

1.4 Gliederung der Arbeit

Der erste Teil der Arbeit befasst sich mit den methodischen Grundlagen. Dabei werden die in der Literatur erwähnten Verfahren zur Einzelbaumdetektion analysiert und gängige Bewertungsmethoden aufgeführt. Zudem wird in das Thema von evolutionären Algorithmen eingeführt.

Im zweiten Teil ist das Testverfahren zur Findung des am besten geeigneten Algorithmus dokumentiert. Dazu werden zuerst die zur Verfügung stehenden Daten präsentiert. Zentraler Part ist die Beschreibung des Testaufbaus inklusive Testframework und des entwickelten Workflows. Mit der Bewertung und Diskussion der Resultate der getesteten Verfahren wird dieser Teil der Arbeit abgeschlossen.

Mit der praktischen Anwendung des im Testprozedere ermittelten Algorithmus beschäftigt sich der dritte Teil. Dabei wird auf die Vorgehensweise eingegangen, wie der im vorherigen Kapitel evaluierte Algorithmus optimiert wird, so dass er auf das Gebiet des Kantons Luzern angewendet werden kann. Auch dieser Teil wird mit einer Ergebnisdokumentation und -diskussion abgeschlossen.

Der letzte Teil beinhaltet eine Diskussion der Resultate mit einer Zusammenfassung der wesentlichen Erkenntnisse und Ansätze für mögliche Verbesserungen. Dabei wird die Forschungsfrage beantwortet, ob mit dem gewählten Vorgehen ein adäquates Verfahren zur Einzelbaumdetektion auf grossem Gebiet gefunden werden konnte.

1.5 Der Kanton Luzern

Der Kanton Luzern ist einer von 26 Kantonen der Schweiz und liegt als Binnenkanton in der Zentralschweiz umgeben von den Kantonen Bern, Aargau, Zug, Schwyz, Ob- und Nidwalden. Der Norden des Kantons gehört geografisch zum Mittelland und ist geprägt von eiszeitlich geformten Hügellandschaften und Seen. Im Süden geht der Kanton in voralpine Regionen über und mit dem höchsten Punkt auf dem Brienzer Rothorn auf 2'349.7 Meter über Meer hat der Kanton einen Alpengipfel auf seinem Gebiet (Kiener 2018). Das Gebiet des Kantons Luzern mit einer Fläche von **1'493.51 Quadratkilometer** wird zu über 50% landwirtschaftlich genutzt, während Wald und bestockte Flächen rund 30% der Fläche ausmachen (Bundesamt für Statistik 2016). Die Waldgesellschaften des Kantons werden im Mittelland dominiert vom Waldmeister- und Waldhirsen-Buchenwäldern. Im Berggebiet nehmen Tannen-Buchenwälder, subalpine Fichten- und Tannenwälder und (Ahorn-)Eschenwälder überhand (UTAS AG et al. 2014).

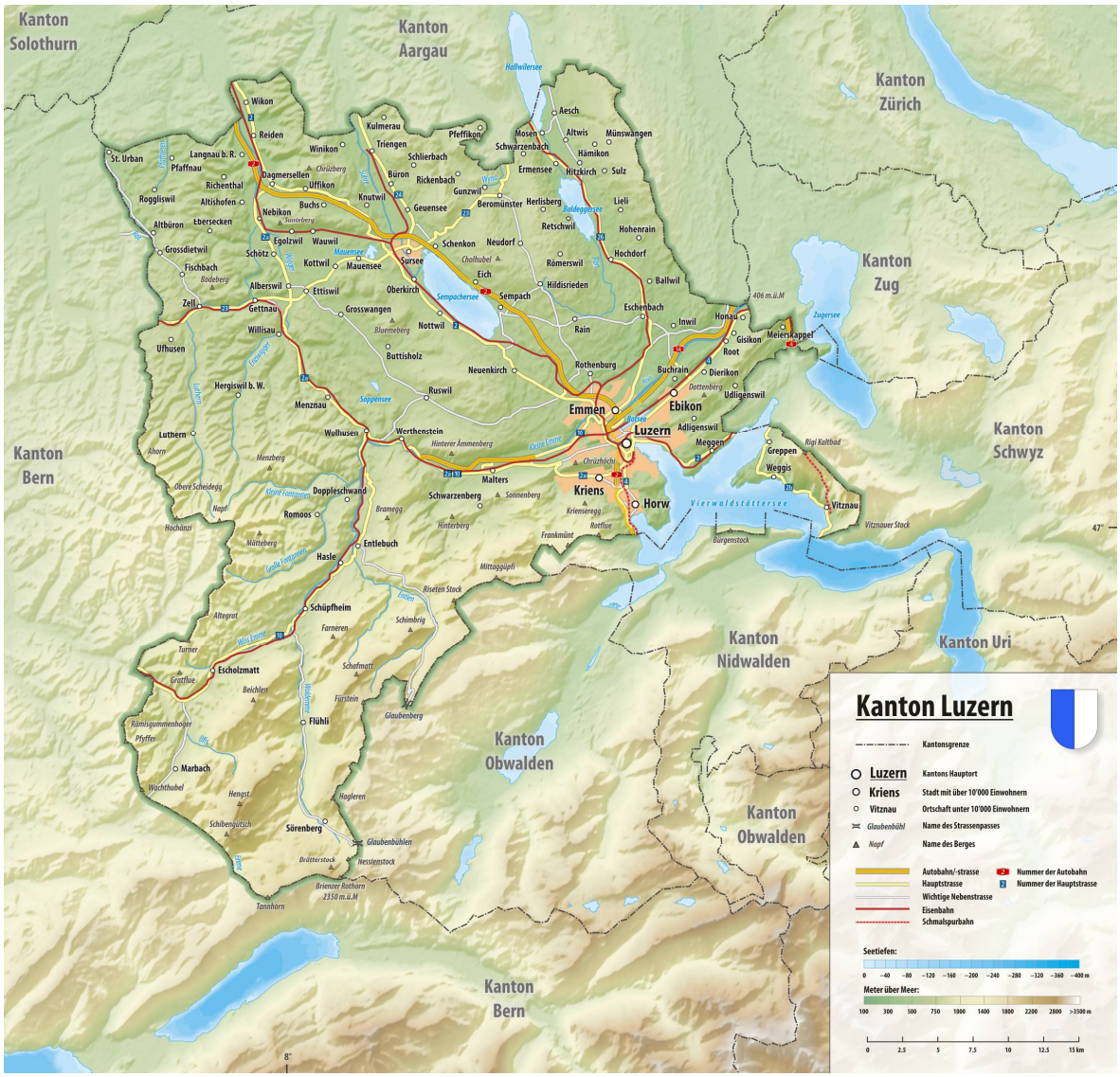


Abb. 1: Reliefkarte des Kantons Luzern

(Tschubby 2013)

2 Methodische Grundlagen

2.1 LiDAR (Light Detection And Ranging)

In der Fernerkundung wird LiDAR *(**Li**ght **D**etection **A**nd **R**anging)* als aktives Verfahren zur Messung von präzisen Oberflächendaten eingesetzt (NOAA 2012). Im Gegensatz zu Radar *(**Ra**dio **D**etection **A**nd **R**anging)*, wo langwellige elektromagnetische Wellen zum Einsatz kommen, setzt LiDAR auf kurzwellige Laserimpulse. Somit kann man räumlich kleinere Objekte auflösen, dies jedoch im Gegensatz zu Radar nur auf kurze Distanzen und bei klarer Sichtlinie (z.B. wolkenlose Bedingungen).

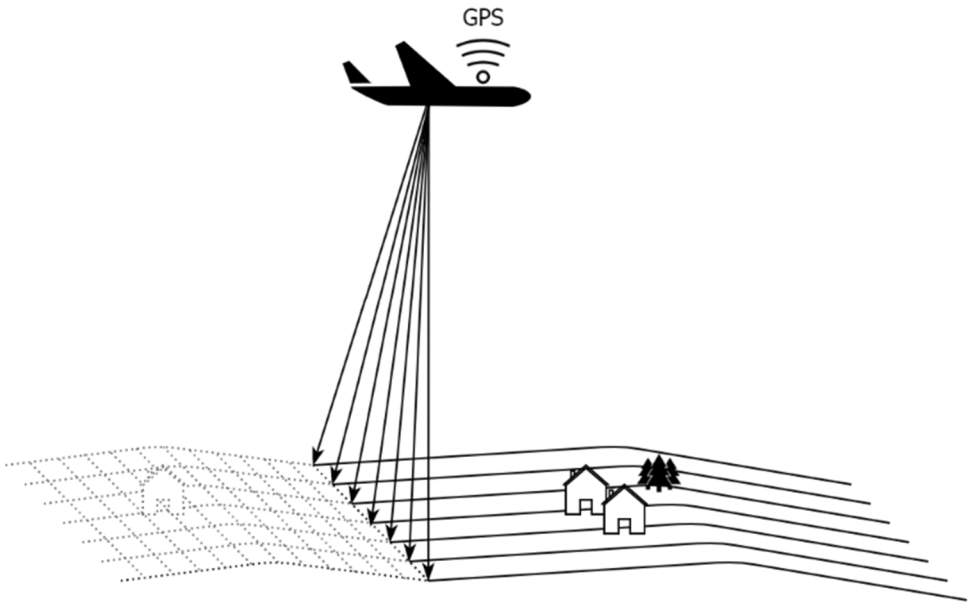


Abb. 2: Schematische Darstellung der Funktionsweise von LiDAR

Die schematische Darstellung in Abb. 2 zeigt, wie bei ALS *(Airborne Laser Scanning)* die Erdoberfläche aus einem Flugzeug mit mehreren Laserimpulsen abgetastet wird. Durch die Messung der Zeitspanne zwischen Emittierung eines Laserimpulses und der Detektion des reflektierten Signals kann die Distanz berechnet werden. Dies setzt voraus, dass man beispielsweise dank GPS *(Global Positioning System)* zu jedem Zeitpunkt weiss, wo sich der Sensor in der Luft befindet. Wie schwierig dies bei LiDAR Sensoren in Flugzeugen ist, erklärt die US-amerikanische National Oceanic and Atmospheric Administration in ihrer Einführung zu LiDAR sehr eindrücklich:

> «[…], to achieve a high level of accuracy, this process is a bit more complicated since it is important to know, within a centimeter or so, where the plane is as it flies at 100 to 200 miles per hour, bumping up and down, while keeping track of hundreds of thousands of lidar pulses per second.» (NOAA 2012: 7)

Das Resultat einer LiDAR Befliegung ist eine georeferenzierte Punktwolke, welche einem dreidimensionalen Abbild der beobachteten Erdoberfläche entspricht. Heutige LiDAR Plattformen sind in der Lage, mehrere Reflektionen eines einzelnen Laserimpulses zu messen und können damit durch Vegetationsoberflächen hindurchblicken.

2.2 Einzelbaumdetektion aus LiDAR Daten

Wenn man Einzelbäume aus LiDAR Daten extrahieren möchte, haben sich die folgenden grundlegenden drei Schritte bewährt (Wang et al. 2018):

1. Einteilung der Punkte in Baum und Nicht-Baum Punkte
2. Identifikation und Segmentierung von Einzelbäumen
3. Extraktion von Baumparametern aus den segmentierten Einzelbäumen

2.2.1 Klassierung der LiDAR Punkte

Wang et al. (2018) fassen die Forschungsergebnisse für den ersten Schritt zusammen. Sie nennen dabei konventionelle Methoden, welche auf dem digitalen Oberflächenmodell (DOM) und dem digitalen Terrainmodell (DTM) beruhen. Durch die Subtraktion des DTM vom DOM erhält man nur noch Punkte, welche über der Oberfläche liegen. Bäume haben in der resultierenden Punktwolke eine charakteristische Verteilung und können so klassiert werden. Andere Verfahren machen sich das volle Spektrum von LiDAR Laserimpulsen zu Nutze, um Vegetation zu erkennen. Weiter verweisen Wang et al. (2018) auf Verfahren, welche auf zusätzliche Sensoren beruhen, die multispektrale Aufnahmen der Oberfläche liefern.

2.2.2 Einzelbaumsegmentierung

Zur Realisierung des zweiten Schritts, der Identifikation und Separierung der Einzelbäume, gibt es eine Fülle von Forschungsarbeiten. Jakubowski et al. (2013) erwähnen in ihrer Arbeit 39 Forschungsartikel zur Einzelbaumsegmentierung. Sie teilen die Resultate in bildbasierte und LiDAR basierte Verfahren ein, wobei auch Kombinationen beider Verfahren zur Anwendung kommen. Der Vergleich bei den LiDAR basierten Verfahren geht zurück bis auf die Pionierarbeiten zur Einzelbaumsegmentierung aus LiDAR Daten von Juha Hyyppä (1999). Bei den LiDAR basierten Verfahren erkennen Jakubowski et al. (2013) Verfahren, welche zwar LiDAR Daten als Ausgangsdaten haben, jene aber als **Baumkronenoberfläche *(canopy height model)*** auf die Ebene projiziert werden, damit wieder bildbasierte Verfahren zum Einsatz kommen können. Diese Verfahren grenzen sie von Verfahren ab, welche direkt in der **Punktwolke *(point-cloud based)*** operieren.

Seit dem Jahr 2013 haben sich die bewährten Verfahren über die Baumkronenoberfläche weiterentwickelt und verfeinert. Duncanson et al. (2014) schlugen ein Verfahren vor, welches mittels mehreren Iterationen durch unterschiedliche Vegetationshöhen die Bäume zuverlässig von tiefer Vegetation abgrenzen konnte. Silva et al. (2016) hatten die Idee, die lokalen Maxima einer Baumkronenoberfläche mittels *Voronoi Tessellation* in Einzelbäume zu segmentieren.

Der Top-Down Ansatz von Dalponte und Coomes (2016) weitet lokale Maxima der Baumkronenoberfläche mittels Entscheidungsbaum so lange aus *(region growing)*, wie die benachbarten Punkte noch zum individuellen Baum gehören. Ein ähnliches Verfahren, jedoch mittels Konturen rund um lokale Maxima, schlugen Wu et al. (2016) vor.

Mit zunehmender Rechenleistung haben sich auch diejenigen Algorithmen weiterentwickelt, welche Einzelbäume direkt in der Punktwolke extrahieren. Pollock (1996) schlug früh ein Verfahren vor, welches von einem synthetischen Baummodell ausgehend die Einzelbäume detektieren konnte. Dieses Verfahren nutzten Xiao et al. (2016) in dreidimensionalen LiDAR Punktwolken und entwickelten einen adaptiven *mean shift* Algorithmus, wobei sie das Baummodell je nach Höhe des Baumes anders parametrisiert haben.

Einige Verfahren basieren auf der Auswertung des gesamten Laserspektrums von LiDAR Daten (Reitberger et al. 2009, Dai et al. 2018, Dalponte et al. 2019). Einen komplett neuen Ansatz verfolgten Wang et al. (2018). Sie fassen mehrere LiDAR Punkte in würfelförmige Voxel zusammen und klassieren die Voxel zu einzelnen Bäumen. Ein Voxel steht dabei für die dreidimensionale Repräsentation eines Pixels. Zudem entstanden Verfahren, welche sich sowohl auf eine Baumoberfläche als auch auf eine Berechnung direkt in der Punktwolke abstützen (Ferraz et al. 2016, Jaafar et al. 2018).

Das Segmentierungsverfahren mittels ***watershed*** Algorithmus zieht sich wie ein roter Faden durch die Forschungsarbeiten. Erstmals wurde das Verfahren von Beucher und Lantuéjoul (1979) zur Kantendetektion von Strukturen in Bildern vorgeschlagen. Zur Separierung von Einzelbäumen wurde dieser Algorithmus erstmals von Hyypää (1999) angewendet. Danach wurde es von einer Reihe von Forschern übernommen und adaptiert (Schardt et al. 2002, Wang et al. 2004, Chen et al. 2006, Mohd Zaki et al. 2015, Barnes et al. 2017).

Die Idee dieses aus der Hydrologie stammenden Verfahrens ist, die um 180 Grad gedrehte Baumkronenoberfläche mit Wasser zu fluten. Vertiefungen entsprechen Baumkronen, während sich an den Rändern der gefluteten Becken die Grenzen zu den benachbarten Bäumen und somit die Baumkronen ausbilden.

2.2.3 Extraktion von Baumparametern

Der **Standort** und die **Höhe** eines Baumes ergeben sich implizit aus der Segmentierung des vorherigen Schrittes. Der höchste LiDAR Punkt entspricht der Baumkronenspitze. Die Höhe des Baumes ergibt sich durch die Subtraktion jenes Baumkronenpunktes vom zu Grunde liegenden Terrainmodell an derselben Stelle.

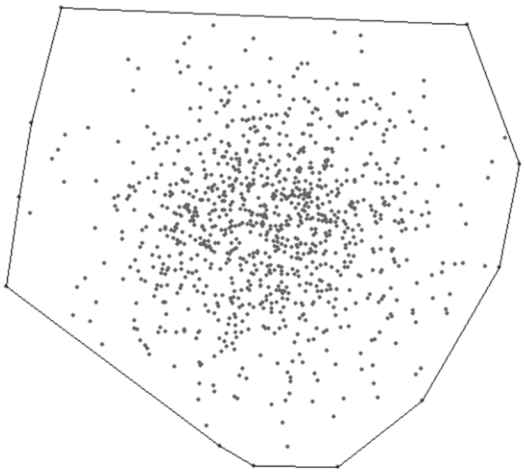


Abb. 3: Konvexe Hülle einer Punktwolke im zweidimensionalen Raum

Die **Baumgeometrie** oder der **Kronendurchmesser** ergibt sich aus der konvexen Hülle, welche alle Punkte eines individuellen Baumes einschliesst (Sikchi 2017). Dabei spielt es keine Rolle, ob die Einzelbäume mittels Baumkronenoberfläche gefolgt von einer Bildsegmentierung oder direkt in der Punktwolke berechnet werden. Ersteres ergibt eine konvexe Hülle um alle Pixel im $\mathbb{R}^2$, was einem Polygon entspricht. Letzteres entspricht einem Polyeder, der dreidimensionalen Entsprechung der konvexen Hülle im $\mathbb{R}^3$, welcher alle LiDAR Punkte eines individuellen Baumes umschiesst.

Die Berechnung des **Vegetationsvolumens** und der **Biomasse** über Grund wird von Giannico et al. (2016) beschrieben. Sie sind dabei nicht von Einzelbäumen ausgegangen, sondern haben über die mittlere Höhe eines geschlossenen Baumbestandes die Biomasse modelliert.

Die automatisierte Extraktion der **Baumart** geht über die Einzelbaumselektion hinaus. Vielversprechende Ansätze bei dieser Fragestellung zeigt Lars Waser (2012) in seiner Doktorarbeit. Für Schweizer Wälder erreichte er eine Klassifikationsgenauigkeit von 70 – 85% bei der Erkennung von 4 bis 7 Baumarten. Dabei kamen sowohl LiDAR Daten als auch optische Sensoren inklusive nahes Infrarotspektrum zum Einsatz.

2.3 Bewertungsmethoden für Klassierungsverfahren

Zwei wichtige Kriterien zur Bewertung von Klassierungsalgorithmen sind die Präzision *(accuracy)* und die Laufzeit *(performance)*. Zaforemska et al. (2019) habe in ihrer Arbeit fünf Algorithmen zur Einzelbaumsegmentierung aus LiDAR Daten verglichen und gezeigt, dass diese beiden Kriterien antagonistisch sind. Es kamen sowohl drei Verfahren mit Baumkronenoberfläche (Chen et al. 2006, Dalponte und Coomes 2016, Silva et al. 2016) als auch zwei Verfahren direkt in der Punktwolke zum Einsatz (Li et al. 2012, Xiao et al. 2016). Dabei haben sie gezeigt, dass das *point-cloud* basierte Verfahren von Xiao et al. (2016) die besten Resultate erzielte, jedoch die schlechteste Laufzeit aufwies. Sie fassen zusammen, dass die Verfahren mit einer Baumkronenoberfläche, über alle Baumarten gesehen, fast gleich gut

abgeschnitten haben. Diese populären Verfahren seien einfach und schnell und liefern für konisch geformte Bäume durchaus gute Resultate. Generell betonen sie jedoch, dass diese Verfahren das Problem zu stark vereinfache und Information verloren ginge. Insbesondere bei grossen Laubbäumen beobachteten sie ein *Overfitting* der Algorithmen, wobei einzelne Äste als separate Bäume klassiert wurden (Zaforemska et al. 2019).

2.3.1 Präzision

Zur Messung der Präzision von Klassierungsalgorithmen hat sich das *Accuracy Assessment* etabliert. Es entspricht dem Standardwerkzeug bei der Klassierung von Fernerkundungsdaten (Congalton und Green 2019). Dabei wird die von einem Algorithmus ermittelte Klassierung mit einem Referenzdatensatz verglichen, wobei von letzterem angenommen wird, dass er den realen Gegebenheiten entspricht.

Die Klassierung in Baum oder Nicht-Baum ist binärer Natur, was die Messung der Präzision vereinfacht. Bei einer binären Klassierung werden die tatsächlichen (Referenz) mit den vorhergesagten Klassen verglichen und die Resultate in der **Konfusionsmatrix** in *True Positives (TP), True Negatives (TN), False Positives (FP)* und *False Negatives (FN)* eingeteilt.

		Referenz	
		Baum	Nicht-Baum
Klassierung	Baum	TP	FP
	Nicht-Baum	FN	TN

Tab. 1: Konfusionsmatrix einer binären Klassierung in Baum und Nicht-Baum

Die Tab. 1 zeigt die Konfusionsmatrix für die Einteilung der Klassen in Baum und Nicht-Baum. Über das Verhältnis aus der Anzahl von TP und TN zu FP und FN lassen sich objektive Messgrössen des Klassierungsalgorithmus ableiten. Beispielsweise lässt sich die *Overall Accuracy* (1), die *User's Accuracy* (2), die *Producer's Accuracy* (3) oder der F_1*-Score* (4) berechnen.

$$OA = \frac{TP+TN}{TP+TN+FN+FP} \tag{1}$$

$$UA = \frac{TP}{TP+FP} \tag{2}$$

$$PA = \frac{TP}{TP+FN} \tag{3}$$

$$F_1 = 2\,\frac{(UA*PA)}{(UA+PA)} \tag{4}$$

Die *Producer's Accuracy* wird oft auch als *Recall Value* oder kontextbasiert als Baumdetektierungsrate bezeichnet, während die *User's Accuracy* als *Precision Value* die Korrektheit der

detektierten Bäume widerspiegelt (Li et al. 2012). In diversen Forschungsarbeiten rund um die Einzelbaumdetektion wird der F_1-Score zur Messung der Präzision eines Algorithmus verwendet (Li et al. 2012, Silva et al. 2016, Franceschi et al. 2018). Der F_1-Score als harmonisches Mittel berücksichtigt beide Werte gleichermassen. Je näher der F_1-Score an 1 ist, desto höher sind die PA und die UA und desto akkurater ist das Verfahren (Li et al. 2012).

Eine weitere statistische Kennzahl ist der Kappa-Index nach Cohen (1960). Zu dessen Berechnung wird neben der *Overall Accuracy* (1) eine *Expected Accuracy* (5) benötigt. Letztere beschreibt die Güte einer zufälligen Klassifikation aufgrund der vorliegenden Konfusionsmatrix.

$$EA = \frac{(TN+FP)*(TN+FN)+(FN+TP)*(FP+TP)}{n^2} \tag{5}$$

$$\kappa = \frac{OA-EA}{1-EA} \tag{6}$$

Der Kappa-Index (6) vergleicht die erfolgte Klassierung mit einer Klassifikation per Zufallsprinzip. Je grösser der Wert, desto eher kann von einer signifikant akkuraten Klassierung ausgegangen werden. Wang et al. (2018) verwenden für die Messung der Präzision ihres Voxel basierten Algorithmus den Kappa-Index.

Es zeigt sich, dass in den bisher getätigten Forschungsarbeiten rund um die Einzelbaumdetektion kein Konsens über das geeignete Verfahren herrscht. Das klassische *Accuracy Assessment* hat sich zwar etabliert, der zur Anwendung kommende Index jedoch nicht. Li et al (2012) bevorzugen den F_1-Score mit der Begründung, dass jener beide Fehlertypen berücksichtige und somit die Über- und Untersegmentierung minimiere. Sie formulieren es explizit mit *«[…], traditional accuracy indices such as total accuracy and kappa coefficient are not applicable»* (Li et al. 2012: 83). Sie erkennen in ihrem Fazit ein Potenzial für zukünftige Forschungsarbeiten, welche einen Standard ausarbeiten und etablieren sollten.

2.3.2 Laufzeitkomplexität

Zur Beurteilung der Laufzeit eines Algorithmus greift man in der Informatik auf die O-Notation zurück. Diese Notation stammt von frühen Arbeiten Paul Bachmanns (1894) zur Zahlentheorie. Dabei wird eine Funktion angegeben, welche die Wachstumsrate bei zunehmender Anzahl zu prozessierenden Objekten repräsentiert.

Bezeichnung	O-Notation	Beschreibung
konstant	$O(1)$	Die Laufzeit ist unabhängig von der Datenmenge.
linear	$O(n)$	Die Laufzeit wächst proportional mit der Datenmenge.
quadratisch	$O(n^2)$	Eine Verdoppelung der Datenmenge bewirkt eine Vervierfachung der Laufzeit.
logarithmisch	$O(\log n)$	Bei einer verdoppelten Datenmenge wächst die Laufzeit linear.

Bezeichnung	O-Notation	Beschreibung
exponentiell	$O(2^n)$	Die Laufzeit wird doppelt so hoch, wenn eine einzelne Einheit der Datenmenge dazu kommt.
faktoriell	$O(n!)$	Mit zunehmender Datenmenge verhält sich die Laufzeit faktoriell.

Tab. 2: Typische Laufzeitkomplexitäten von Algorithmen

Die gängigsten Laufzeiten sind in der Tab. 2 aufgeführt, wobei die Laufzeit von oben nach unten zeitlich länger und damit schlechter wird. Die Notation ist hilfreich bei der groben Einschätzung der erwarteten Laufzeit eines Algorithmus.

Die meisten Forschungsarbeiten rund um Einzelbaumdetektion machen keine Aussagen zu den Laufzeitkomplexitäten ihrer Algorithmen. Eine Ausnahme bilden Wang et al. (2018) mit dem Voxel basierten Ansatz. Sie bemerken, dass sich die Laufzeitkomplexität nicht so einfach bestimmen lässt. Während das Clustering der Punkte in Voxel noch eine lineare Laufzeit $O(n)$ habe, sei die Laufzeit der anschliessenden Klassierung der Voxel zu Einzelbäumen stark abhängig von der Anzahl benachbarter Zellen und somit direkt Abhängig von der Grössen- und Formvariabilität der zu detektierenden Bäume (Wang et al. 2018).

Die Komplexität des punktbasierten Verfahrens von Li et al. (2012) lässt sich dank der exakten Beschreibung ihres Algorithmus ableiten. Berücksichtigt man die Anzahl Bäume eines Gebietes hat der Algorithmus eine quadratische Laufzeit. Bei diesem Verfahren wird mit jeder Iteration nur ein einzelner Baum vom Rest der Punktwolke abgetrennt. Die Punkte des ersten gefundenen Baumes werden dabei nur einmal prozessiert. Die Punkte des letzten Baumes wurden jedoch bereits bei jedem anderen zuvor gefundenen Baum durchwandert. Mit n als der Anzahl Bäume in einem abgeschlossenen Gebiet ergibt sich folgende quadratische Laufzeit in O-Notation (7):

$$O\left(\frac{n(n+1)}{2}\right) \Rightarrow O(n^2) \tag{7}$$

Die Laufzeitkomplexität von Verfahren mit Baumkronenoberfläche ist direkt abhängig vom verwendeten Bildsegmentierungsverfahren. Die Bildsegmentierung mittels *watershed* Algorithmus beispielsweise hat eine lineare Laufzeitkomplexität $O(n)$, da jeder Bildpunkt nur einmal prozessiert werden muss (Kornilov und Safonov 2018).

2.4 Evolutionäre Algorithmen

Den für das Zielgebiet adäquaten Algorithmus mit den besten Parametern zu finden, entspricht einem Optimierungsproblem, dessen Lösung nicht berechnet werden kann. Für solche Probleme gibt es aus dem Bereich der künstlichen Intelligenz unter anderem das Verfahren der evolutionären oder genetischen Algorithmen respektive Programmierung. Die Stärke dieser Algorithmen kommt bei heuristischen oder nicht berechenbaren Optimierungsproblemen zum

Tragen. Die Grundidee besteht darin, dass Algorithmen (Individuen, Kreaturen) sich mittels unterschiedlicher Parameter bei der Lösung eines konkreten Problems konkurrieren. Nur die besten überleben und dürfen sich zur neuen Generation weiterentwickeln. Viele Generationen später nähern sich die Algorithmen der optimalen Lösung des Problems an (Gad 2018b).

Die Idee, Maschinen oder Software mittels evolutionärer Prozesse weiter zu entwickeln, geht zurück auf Alan Turing. Er gilt als Vater der künstlichen Intelligenz und der theoretischen Computerwissenschaften. Obwohl die technischen Möglichkeiten noch nicht vorhanden waren, beschreibt er bereits sehr früh das theoretische Konzept von lernenden Maschinen, indem er den Lernprozess von Maschinen dem Lernen eines Schülers von seinem Lehrer gleichstellt. Er macht bereits die Analogie zur Evolutionsbiologie und drückt seine Hoffnung aus, dass die Selektion der guten Schüler um ein Vielfaches schneller und zielgerichteter sein möge (Turing 1950).

25 Jahre später wurden diese Ideen von John Holland (1975) aufgegriffen. Er gilt heute als Pionier in der Arbeit rund um genetische Algorithmen. Es entstanden auch erste praktische Anwendungen, wie beispielsweise die Arbeiten von Richard Forsyth. Er entwarf ein statistisches Programm in PASCAL, welches aus einem Set von medizinischen Parametern eine einfache Formel herleiten sollte, ob ein Herzpatient überleben oder sterben wird. Mittels Trainingsdatensätzen und evolutionärer Weiterentwicklung mit 500 Generationen ermittelte sein Programm eine Formel, welche nur noch vom Blutdruck und der ausgeschiedenen Urinmenge pro Stunde abhängig ist. Mit dieser Formel erreichte er eine Trefferquote von 81% (Forsyth 1981).

2.4.1 Terminologie

Die Begriffe bei evolutionären Algorithmen bedienen sich der darwinistischen Evolutionstheorie, wobei es zum biologischen immer ein softwaretechnisches Äquivalent gibt. Dabei entsprechen Chromosomen und Gene den Parametern eines Algorithmus, welche als Strings in Listen gespeichert werden. Individuen und Generationen sind das biologische Äquivalent zu einzelnen oder einer Sammlung von Algorithmen (Bodenhofer 1999).

Von zentraler Bedeutung sind die Begriffe Fitness, Selektion und Mutation. Zur Berechnung der **Fitness** eines Algorithmus benötigt man eine Fitnessfunktion. Dies entspricht einem objektiven Mass, wie gut ein Algorithmus das ihm gestellte Problem lösen konnte. Bei Klassierungsproblemen können die Resultate der bewährten *Accuracy Assessments* direkt als Fitnessfunktion angewendet werden (siehe Kapitel 2.3.1).

Gemäss Bodenhofer (1999) ist die **Selektion** durch Präferenz von fitteren Algorithmen der zentrale Prozess, dass man sich einer optimalen Lösung annähern kann. Mittels *Crossover* Mechanismus werden die Gene (Parameter) des einen Individuums (Algorithmus) mit den Genen eines anderen fitten Individuums vermischt.

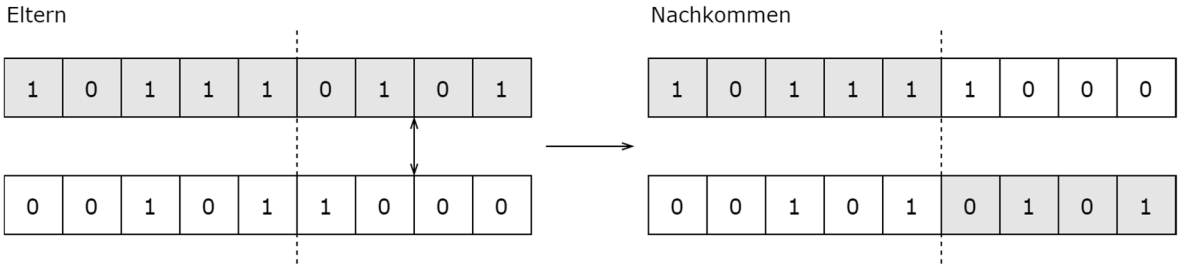


Abb. 4: Crossover eines binären Chromosoms an zufälliger Stelle

Die dabei einfachste Methode ist die Zerschneidung des Chromosoms an einer oder mehreren zufälligen Stellen gemäss Beispiel in Abb. 4. Der anschliessende paarweise Austausch erzeugt das Chromosom zweier Nachkommen.
Üblicherweise zerlegt man die Parameter in binäre Strings, um ein einfaches Chromosom zu erhalten. Chee Chun Gan (2016) schlägt ein indexiertes Chromosom vor, falls man die Parameter nicht in binäre Strings zerlegen möchte. Die Gene entsprechen dann nicht mehr den realen Parameterwerten, sondern einem Index, welcher auf den realen Wert aus einem möglichen Genpool zeigt.

Der letzte Schritt ist die **Mutation**. Genauso wie in der Natur die Reproduktion nicht immer perfekt abläuft oder ein Gen durch externe Einflüsse (z.B. Strahlung) mutiert werden kann, werden die Parameter eines Algorithmus zufällig verändert. Nur so erhält man die Chance, dass die Suche nach guten Lösungen neue Wege einschlagen kann.

3 Methodenevaluierung

Einen optimalen Algorithmus zur Einzelbaumdetektion zu finden und bestmöglich zu parametrisieren, basiert meist auf Hintergrundwissen und Erfahrung über das prozessierte Gebiet und dessen Baumbestand. Da dieses Wissen für das vorliegende Gebiet des Kantons Luzern nicht vorhanden ist und von einem heterogenen Baumbestand ausgegangen werden muss, soll das beste Verfahren und dessen Parameter mittels genetischen Algorithmus hergeleitet werden. Dazu wird ein entsprechendes Testframework entwickelt und zur Evaluierung der verschiedenen Methoden eingesetzt.

Die Idee, optimale Parameter zur Einzelbaumdetektion mit einem selbstlernenden Algorithmus zu finden, haben Franceschi et al. als erste verfolgt. Sie zeigen, dass mittels Partikelschwarmoptimierung hergeleitete Parameter bessere Resultate liefern als Algorithmen, welche mit Expertenwissen kalibriert werden. Im Gegensatz zu genetischen Algorithmen wissen die Individuen eines Partikelschwarms voneinander und können gute Lösungen eines Problems untereinander austauschen (Franceschi et al. 2018).

3.1 Datengrundlage

Für die vorliegende Arbeit mussten keine Primärdaten erfasst werden, da auf einen umfassenden Datenbestand zurückgegriffen werden konnte, welcher in den folgenden Kapiteln beschrieben wird.

3.1.1 LiDAR Punktwolke

Die Firma Flotron AG mit Sitz in Thun (Schweiz) hat im Auftrag des Kantons Luzern im Jahr 2018 das Kantonsgebiet beflogen und einen LiDAR Datensatz der Oberfläche generiert.

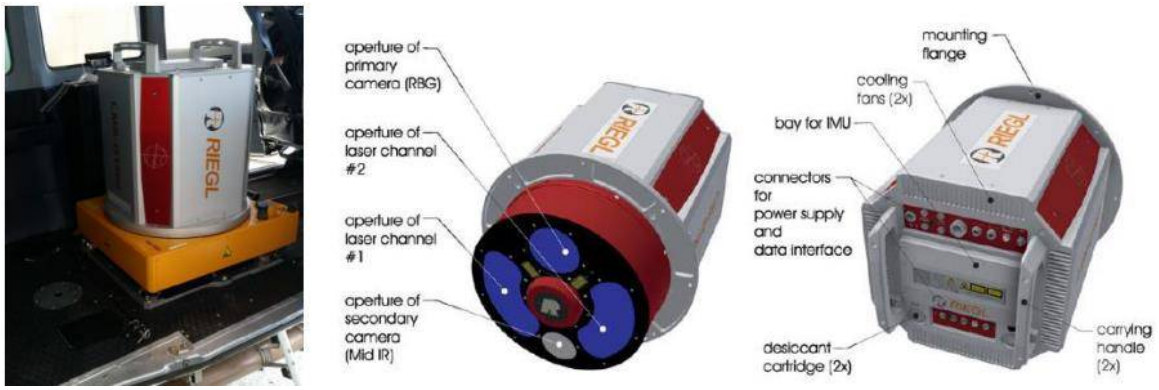


Abb. 5: Laserscanner Riegl LMS-Q1560

Montiert im Flugzeug und als Schemazeichnung (Quelle: Flotron AG und Riegl)

Gemäss technischem Bericht kam der Laserscanner LMS-Q1560 der Firma RIEGL Laser Measurement Systems GmbH zum Einsatz, welcher zwei parallel betriebene Laserquellen beinhaltet (Abb. 5). Damit wurde die Oberfläche mit bis zu 530'000 Laserimpulsen pro Sekunde abgetastet.

Parameter	Wert
Punktdichte	Im Mittel 16 Punkte pro m^2
Höhengenauigkeit	$< \pm 0.1$m
Lagegenauigkeit	$< \pm 0.2$m
Bezugsrahmen	LV95 / LN02
Klassierung	6 Klassen (Boden, Vegetation tief, Vegetation hoch, Gebäude, Brücken und Überführungen, Sonstiges)

Tab. 3: Parameter der LiDAR Daten

Die Tab. 3 zeigt die Präzision des erzeugten Datensatzes. Mit 16 Punkten pro Quadratmeter resultiert ein mittlerer Punktabstand von 0.25 Meter. Die Punktwolke wurde in dem in der Schweiz gebräuchlichen Koordinatensystem CH1903+ / LV95[1] gespeichert. Der Tabelle ist weiter zu entnehmen, dass die LiDAR Punktwolke bereits klassiert wurde. Diese Arbeit wurde durch Subunternehmer in halbautomatischem Verfahren durchgeführt und von der Firma Flotron AG stichprobenweise überprüft. Für die Extraktion der Einzelbäume ist die Klasse **Vegetation hoch** von Interesse. Diese Klasse beinhaltet alle Vegetationspunkte **ab 3 Meter über Boden**. Diese Vorgabe wird für die Abgrenzung der Bäume von der restlichen Vegetation übernommen, respektive die LiDAR Punktwolke wird auf diese Klasse gefiltert, bevor Einzelbäume extrahiert werden. Im Endresultat erscheinen deshalb nur Bäume, welche grösser als 3 Meter sind.

[1] https://spatialreference.org/ref/epsg/ch1903-lv95/

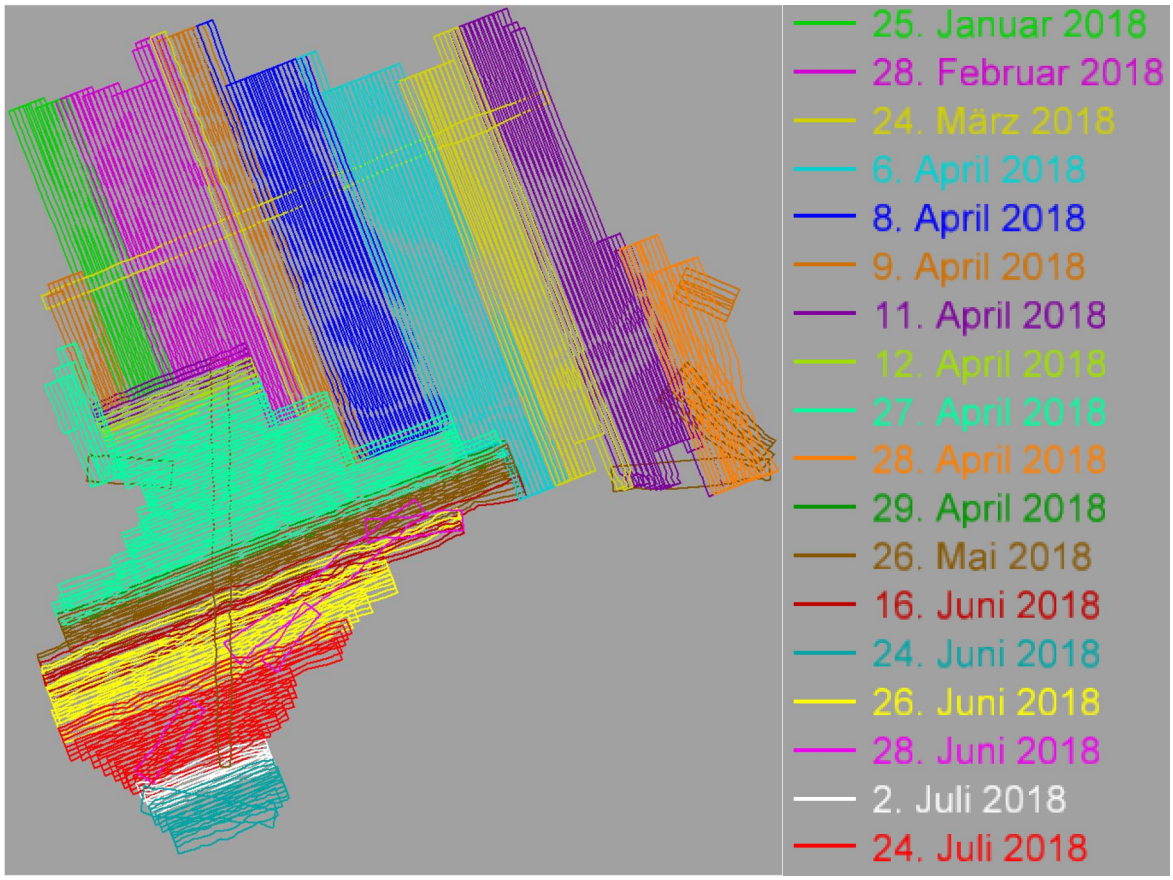


Abb. 6: Fluglinien mit Flugdaten
(Quelle: Technischer Bericht Flotron AG)

Der LiDAR Datensatz entstand nach 18 Flugtagen zwischen dem 25. Januar 2018 und 24. Juli 2018 (Abb. 6). Das zeitlich grosse Befliegungsfenster hat einen direkten Einfluss auf die Extraktion der Einzelbäume. Es muss damit gerechnet werden, dass Laubbäume sowohl mit als auch ohne Blätter vorkommen. Ein Algorithmus, welcher das ganze Kantonsgebiet prozessiert, muss damit umgehen können.
Der Hersteller des Datensatzes weist auf Datenlücken hin, welche jedoch keinen Einfluss auf die Extraktion der Einzelbäume haben. Es handelt sich dabei entweder um Wasserflächen oder nichtreflektierende Oberflächen wie beispielsweise Glas- oder Metalldächer, frischer Asphalt oder generell feuchte Oberflächen.

3.1.2 Digitales Oberflächen- und Terrainmodell (DOM und DTM)

Über das gesamte Kantonsgebiet stehen ein digitales Oberflächen- und Terrainmodell zur Verfügung. Diese beiden Produkte wurden aus den LiDAR Daten abgeleitet und weisen eine räumliche Auflösung von 0.25 Meter pro Pixel auf. Das Oberflächenmodell besteht aus der gerasterten und interpolierten dreidimensionalen Fläche aller erstreflektierten LiDAR Punkte *(first returns)*. Das Terrainmodell wurde analog aus den als Boden klassierten LiDAR Punkten hergestellt.

Die Auflösung leitet sich direkt aus der mittleren Punktwolkendichte von 16 Punkten pro Quadratmeter ab. Die mittlere Lagegenauigkeit eines Baumes liegt somit bei einem Wert von 0.25 Meter. Eine Auflösung von 0.25 Meter pro Pixel liefert genug Informationen, um Einzelbäume zu erkennen, ohne in Übersegmentierung auszuarten (Barnes et al. 2017).

3.1.3 Kachelschema

Sowohl die LiDAR Punktwolke als auch das Oberflächen- und Terrainmodell liegen in gekachelten Datensätzen vor. Dabei hat eine einzelne Kachel eine Ausdehnung von 1'250 mal 750 Meter, was einer Fläche von rund einem Quadratkilometer entspricht. Eine Kachel des Oberflächen- und Terrainmodells hat bei der räumlichen Auflösung von 0.25 Meter pro Rasterzelle eine Ausdehnung von 5'000 mal 3'000 Pixel.

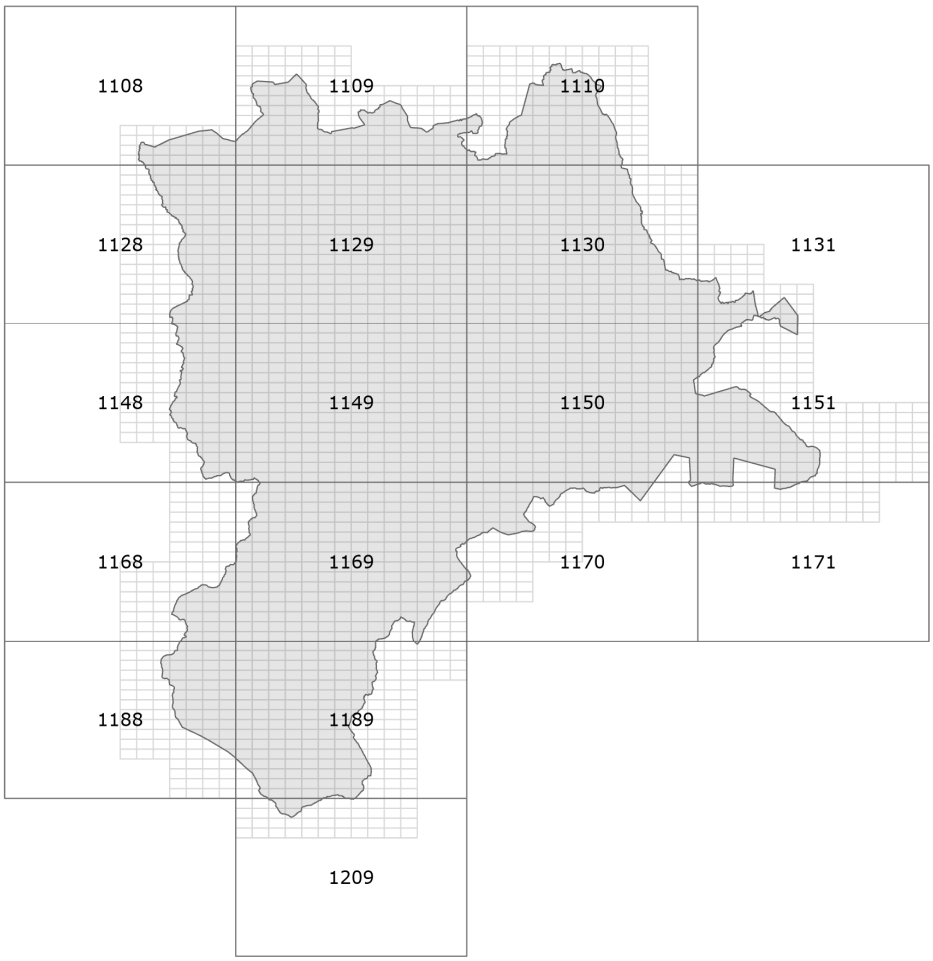


Abb. 7: Kartenblätter der Landeskarten 1:25'000 im Kantonsgebiet von Luzern

Die Abb. 7 zeigt die Kacheln über dem Kantonsgebiet von Luzern überlagert mit den grösseren Kartenblättern im Massstab 1:25'000 des Bundesamtes für Landestopografie swisstopo. Es gibt insgesamt 1'760 Kacheln, welche innerhalb oder teilweise innerhalb des Kantons Luzern liegen. Das identische Kachelschema der LiDAR Punktwolke und des Oberflächen- und Höhenmodells erleichtert die Prozessierung, da alle Kacheln gleich gross und mit derselben siebenstelligen Nummer abgespeichert sind.

3.2 Testaufbau

Das Testverfahren sollte einerseits dazu dienen, einen Algorithmus für die Einzelbaumdetektion zu finden, welcher bezüglich Präzision und Laufzeit grossflächig angewendet werden kann. Andererseits sollten die getesteten Algorithmen mittels evolutionären Prozesses optimal parametrisiert werden können.

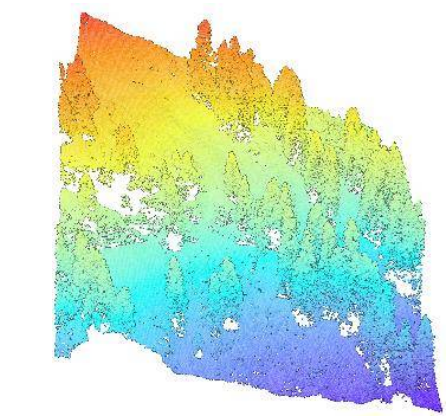
3.2.1 Definition der Testgebiete

Die Testgebiete wurden so ausgewählt, dass sie ein möglichst repräsentatives Bild der Baumsituationen abgeben. So musste der Algorithmus in der Lage sein, mit unterschiedlichen Baumarten (Laub- und Nadelbäume), Baumgrössen, unterschiedlichem Gelände (flaches mittelländisches Gebiet und steiles Berggelände) und unterschiedlichen Strukturen (landwirtschaftliche Gebiete, kleinstrukturierte Siedlungen und urbane Gebiete) umgehen zu können. Darüber hinaus gab es aufgrund der unterschiedlichen Befliegungszeiten Laubbäume mit Blättern *(leaf-on)* und Laubbäume ohne Blätter *(leaf-off)*.

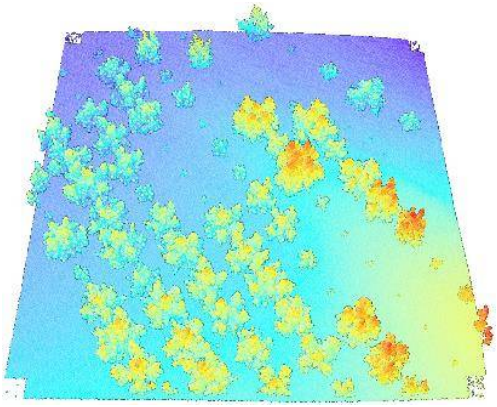
Testgebiet	Kachel-Nr.	Punkte	Bäume	Beschreibung
Bergwald	1189174	298'270	120	In Gruppen vorkommende hohe Nadelbäume knapp unterhalb der Baumgrenze in steilem Gelände.
Obstgarten	1129476	312'857	71	Landwirtschaftsgebiet mit Hochstammobstgarten in mittelländischem Gebiet.
Laubwald	1169177	421'826	212	Geschlossener Wald mit mehrheitlich Laubbäumen *(leaf-on)* durchzogen von einem Graben mit steilen Flanken.
Nadelwald	1129374	350'072	353	Geschlossener Nadelwald mit Jungbestand in mittelländischem Gebiet in leicht abfallendem Gelände.
Siedlung	1150166	255'676	67	Stark räumlich strukturierte Gartensiedlung mit Reiheneinfamilienhäusern.
Stadtpark	1150422	241'607	35	Stadtpark mit grossen und kleinen Laubbäumen *(leaf-off)* in der Stadt Luzern.
Total		**1'880'308**	**858**	

Tab. 4: Charakteristik der Testgebiete

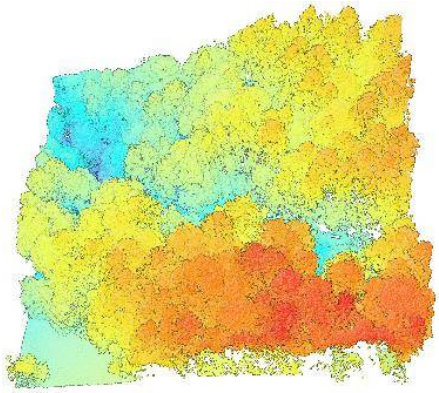
Die Grösse eines einzelnen Testgebietes wurde mit einem Quadrat von 100 Meter Kantenlänge so gewählt, dass der Testlauf eines Algorithmus nicht viel Zeit in Anspruch nahm. Ein Testgebiet enthielt jedoch genug Bäume, dass die Qualität eines Algorithmus gemessen werden konnte. Die Tab. 4 listet die Parameter der sechs Testgebiete. Insgesamt enthalten die Testgebiete rund 1.88 Millionen LiDAR Punkte. Darin gibt es 858 Bäume, deren Lage der Kronenspitzen direkt in der LiDAR Punktwolke von Hand digitalisiert wurden. Der so hergestellte Punktdatensatz diente als Referenz zur Evaluation der Algorithmen.



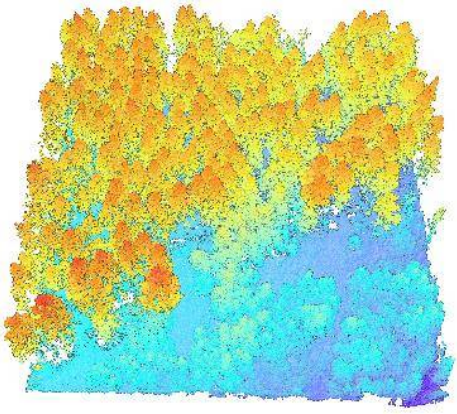

Bergwald




Obstgarten




Laubwald




Nadelwald

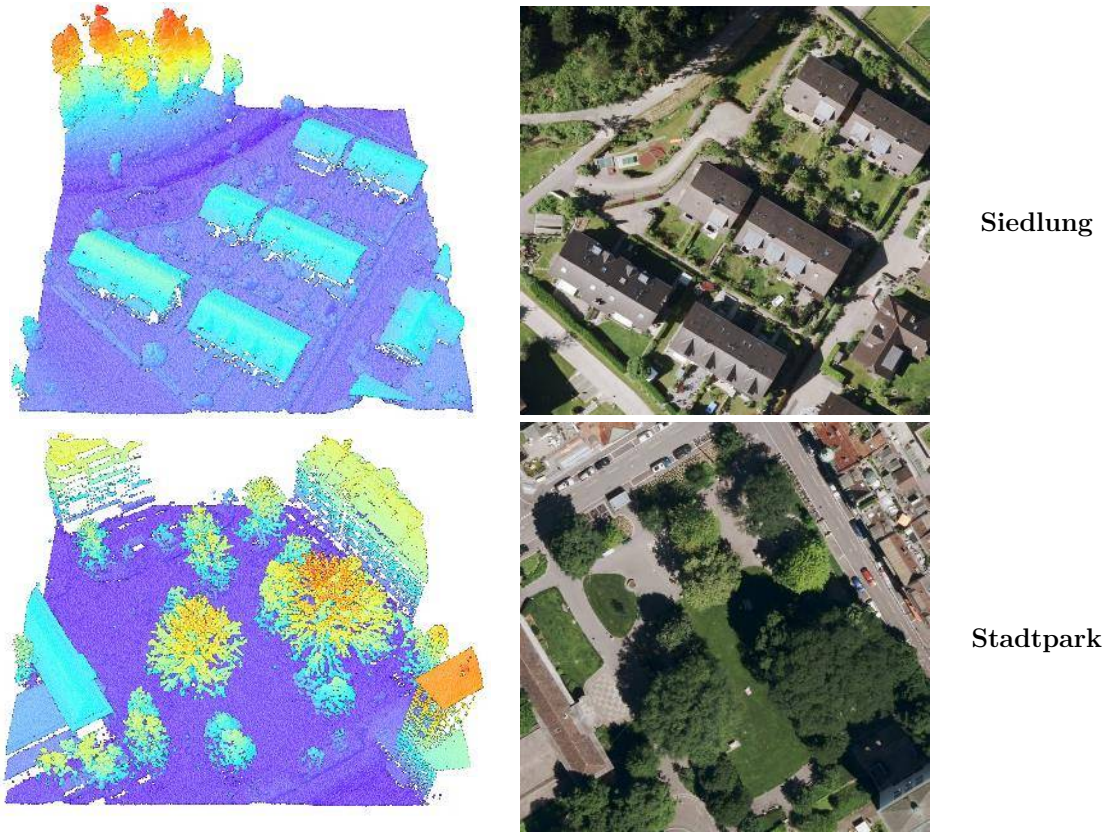


Abb. 8: LiDAR Punktwolke (links) und Orthofoto (rechts) der Testgebiete

Die Abb. 8 gibt einen Einblick in die sechs Testgebiete. Man erkennt, wie sich die Gebiete bezüglich Baumarten, Baumgrössen, Gelände und Strukturierung unterscheiden. Beispielsweise hat die markante Bergulme *(Ulmus glabra)* im Stadtpark einen Kronendurchmesser von rund 30 Meter, während die Kronen der kleinen Zierbäume zwischen den Reiheneinfamilienhäuser der Siedlung nur etwa 2 Meter breit sind. An den Bäumen im Stadtpark ist in der LiDAR Punktwolke gut zu erkennen, dass sie kein Laub tragen *(leaf-off)*. Im Vergleich zu den Bäumen im geschlossenen Laubwald sind im Stadtpark die einzelnen Äste der Bäume sichtbar. Der Unterschied zwischen Laub- und Nadelwälder ist daran erkennbar, dass die Nadelbäume erstens dichter stehen und zweitens eine deutliche Kegelform aufweisen. Eine besondere Herausforderung bildet der Obstgarten, da jene Bäume teilweise bis zu fünf in etwa gleichhohe Kronenspitzen aufweisen.

3.2.2 Entwicklung des Testframeworks

Zur Evaluation und kontinuierlichen Verbesserung der getesteten Algorithmen wurde ein Framework in Python (Python Software Foundation 2019) entwickelt. Als Vorlage diente der Onlineartikel von Ahmed Gad (2018a). Das Framework ermöglicht es, einen Algorithmus zur Einzelbaumextraktion zu prüfen und dessen Parameter mittels evolutionären Prozesses zu optimieren.

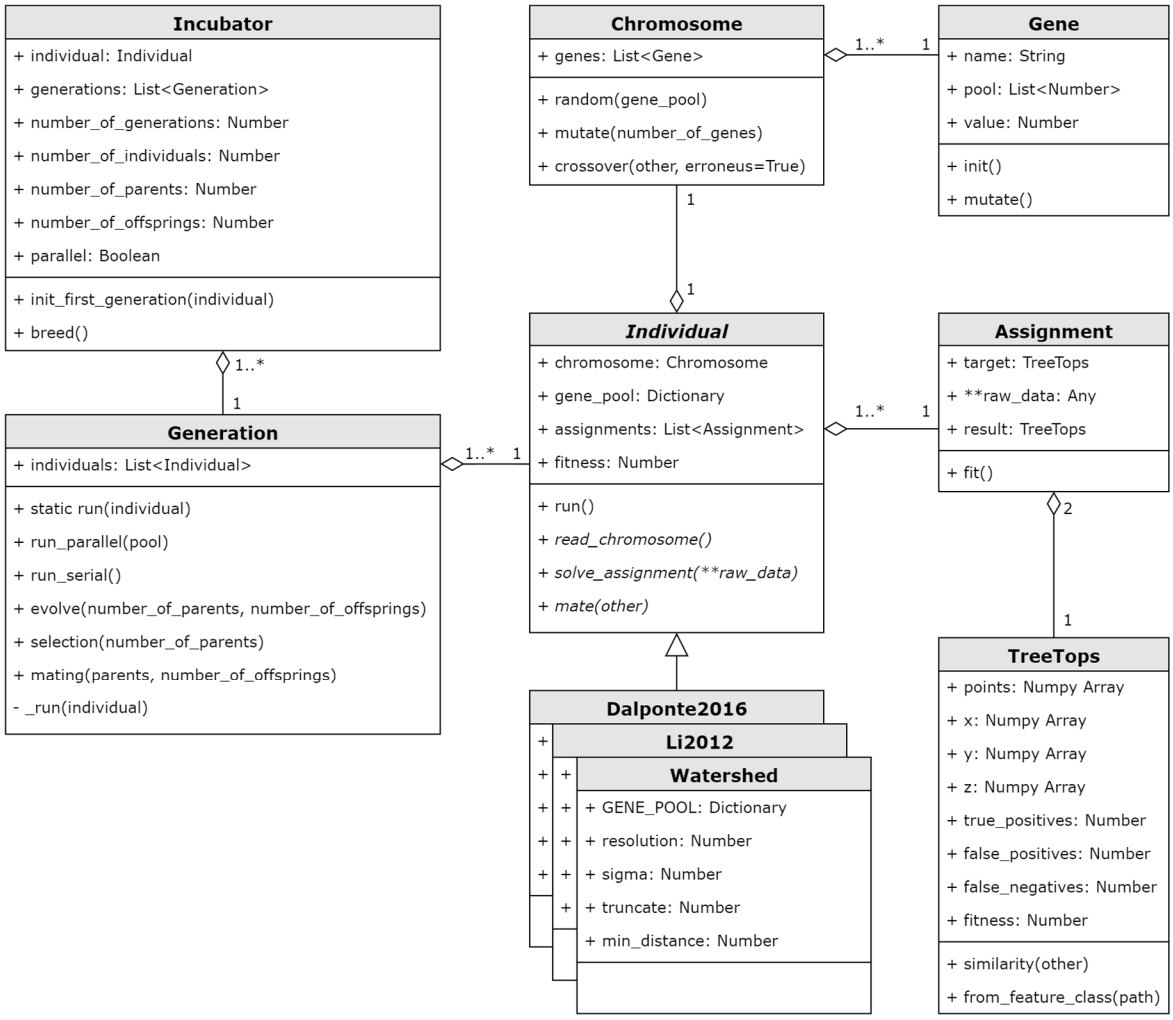


Abb. 9: Klassen des Testframeworks für den evolutionären Prozess

Die Abb. 9 zeigt das Klassenmodell des Testframeworks. Es besteht aus Klassen in Python der Version 3.6 (Python Software Foundation 2019) und kann zur evolutionären Entwicklung eines beliebigen Algorithmus verwendet werden. Startpunkt ist der Inkubator *(Incubator)*, welcher mit einer konkreten Instanz eines Individuums *(z.B. Watershed)*, einer beliebigen Anzahl Aufgaben *(Assignments)* und diversen Prozessparametern gestartet *(breed)* werden kann. Die Aufgaben entsprechen im vorliegenden Fall der Einzelbaumextraktion in allen sechs Testgebieten. Ein Algorithmus ist jeweils abgeleitet von der Basisklasse Individuum *(Individual)* und enthält damit nebst dem statischen individuellen Genpool *(GENE_POOL)* eine Reihe von Parametern *(Gene)* verpackt in einem Chromosom *(Chromosome)*. Der Inkubator initialisiert die erste Generation von Individuen und lässt, wahlweise in serieller oder paralleler Prozessierung, alle Individuen die gestellte Aufgabe lösen. Sowohl die Vorgabe *(target)* als auch das Resultat *(result)* einer Aufgabe besteht im konkreten Fall aus einem Punktdatensatz *(TreeTops)*, welcher mittels Ähnlichkeitsfunktion *(similarity)* die Treffergenauigkeit berechnen kann. Aus dieser Metrik entsteht die messbare Fitness eines Algorithmus. Wie in anderen Forschungsarbeiten zur Einzelbaumextraktion kommt hier der F_1-Score zum Einsatz, um falsch positive und falsch negative Treffer gleichermassen zu minimieren (Li et al. 2012). Je grösser der F_1-Score eines Resultats, desto exakter hat ein Algorithmus das Problem gelöst.

Der Inkubator sorgt dafür, dass sich die Individuen innerhalb einer Generation mittels evolutionären Schritts *(evolve)* bestehend aus Selektion *(selection)* und Paarung *(mating)* weiterentwickeln.

Jeder getestete Algorithmus hat einen hart codierten individuellen Genpool, welcher aus allen möglichen Parametern und dessen Intervallen besteht. Als Werte in den Chromosomen werden damit keine realen Werte gespeichert, sondern lediglich der Index der Liste aus dem Genpool (Chee Chun Gan 2016):

```
GENE_POOL = {
   "parameter_1": [i for i in range(1, 20)],
   "parameter_2": [i for i in numpy.arange(0.0, 100.0, 0.5)],
   "parameter_3": [i for i in numpy.arange(0.0, 20.0, 0.1)]
}
```

Die wohl zentralste Funktion ist die Ähnlichkeitsmessung zweier Punktdatensätze in der Klasse *TreeTops*, um die Fitnessparameter zu bestimmen. Die Punkte sind als dreidimensionales *numpy-Array* gespeichert (Oliphant 2006). Mit der Distanzfunktion *cdist* aus dem Python Packet *scypi.spatial* (Virtanen et al. 2019) wird die minimale euklidische Distanz vom Referenzdatensatz zu den berechneten Kronenspitzen ermittelt. Damit erhält man für jeden Punkt die minimale Distanz zum nächsten Punkt aus dem zweiten Punktdatensatz. Berechnet man dies in beide Richtungen, bekommt man die Distanzen für falsch positive und falsch negative Punkte:

```
d_fp = numpy.min(spatial.distance.cdist(self.points, other.points), axis=1)
d_fn = numpy.min(spatial.distance.cdist(other.points, self.points), axis=1)
```

Danach können mittels Über- oder Unterschreitung einer maximal erlaubten Abweichung in Metern *(threshold)* die Anzahl der richtig positiven, falsch positiven und falsch negativen Resultate gezählt werden. Dabei spielt es für die Berechnung der richtig positiven Treffer keine Rolle, ob man die Distanzmatrix der falsch positiven oder falsch negativen Punkte berücksichtigt:

```
tp = len(d_fp[numpy.where(d_fp <= threshold)])
fp = len(d_fp[numpy.where(d_fp > threshold)])
fn = len(d_fn[numpy.where(d_fn > threshold)])
```

Der letzte Schritt besteht darin, die *User's Accuracy*, die *Producer's Accuracy* und den *F_1-Score* zu berechnen. Dabei muss berücksichtig werden, dass der Nenner Null sein kann. In diesem Fall wird der F_1-Score auf 0 gesetzt:

```
ua = tp / (tp + fp)
```

```
pa = tp / (tp + fn)
f_score = 0 if ua + pa == 0 else 2 * ((ua * pa) / (ua + pa))
```

3.2.3 Setup des evolutionären Algorithmus

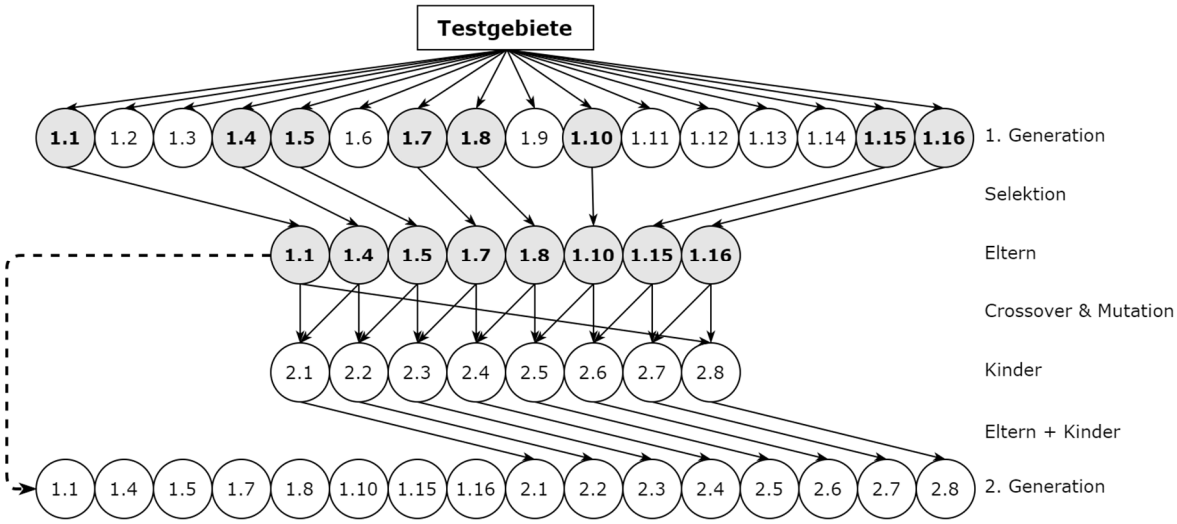


Abb. 10: Evolutionsschritt für eine Population mit 16 Individuen

Die Abb. 10 zeigt einen einzelnen Evolutionsschritt eines Algorithmus unter Test. Vom Algorithmus werden jeweils 16 Individuen erstellt, wobei jedes Individuum ein zufälliges Genom und damit zufällige Parameter bekommt. Die ersten Individuen sind durchnummeriert von 1.1 bis 1.16 und bilden die erste Generation, welche die Einzelbäume in allen sechs Testgebieten extrahieren müssen. Die Resultate der Algorithmen werden auf ihre Güte geprüft, damit die acht besten selektiert werden können. Sie bilden die Eltern und dürfen sich vermehren. Die Vermehrung geschieht über eine Crossover Funktion, wobei die Hälfte des Genoms des einen Individuums mit der Hälfte des Genoms eines anderen Individuums gekreuzt wird. Jeder Elternteil steuert somit die Hälfte seines Genoms für ein erstes und die andere Hälfte für ein zweites Kind bei. Per Zufall werden Gene mutiert, so dass der Evolutionsvorgang komplett neue Individuen generieren und damit neue Lösungen kreieren kann. Die Eltern der ersten Generation bilden gemeinsam mit ihren acht Nachkommen die zweite Generation. Dies ist sinnvoll, damit gute Elterngene und damit gute Lösungen nicht aussterben und die Individuen sich zu einer optimalen Lösung hin entwickeln. Ab der zweiten Generation müssen nur noch die Kinder Einzelbäume extrahieren, damit deren Fitness ermittelt werden kann. Ab dann beginnt der nächste Evolutionsschritt für die Individuen der zweiten Generation.

3.3 Evaluierung der Algorithmen

Algorithmen zur Einzelbaumextraktion, welche direkt in der dreidimensionalen Punktwolke operieren, liefern tendenziell bessere Resultate, haben jedoch längere Laufzeiten (Zaforemska et al. 2019). Um zu prüfen, ob dies für die gegebenen Daten zutrifft, wurde aus den punktbasierten Verfahren ein einzelner Algorithmus näher untersucht.

Bei den Verfahren mit Baumkronenoberfläche ist der eingesetzte Algorithmus nicht entscheidend. Die Resultate bei Zaforemska et al. (2019) sind vergleichbar. Barnes et al. (2017) zeigten, dass ein *marker-controlled watershed* Algorithmus im Vergleich zu einem *region growing* Ansatz leicht besser abschneidet. Daraus lässt sich vermuten, dass die Parametrisierung des Algorithmus einen grösseren Einfluss auf das Endresultat hat als das zu Grunde liegende Verfahren. Aus diesen Gründen wurde aus den Verfahren mit Baumkronenoberfläche nur ein Ansatz entwickelt und für das zu prozessierende Gebiet optimal parametrisiert.

3.3.1 Point-cloud region growing

In der Software *R* (R Core Team 2013) gibt es das Package *lidR* (Roussel et al. 2020), welches unter anderem das Einzelbaumextraktionsverfahren von Li et al. (2012) beinhaltet. Das Verfahren erwartet fixe Parameter beispielsweise für die minimale Baumhöhe, einen Suchradius oder zwei Schwellwerte. Zusammen mit der Python Bibliothek *rpy2* (Gautier 2016), welche es ermöglicht, *R* Code im Python-Kontext auszuführen, konnte der Algorithmus in das entwickelte Testframework eingebunden und geprüft werden.

Resultate

Von diesem Algorithmus wurden 100 Generationen gerechnet. Es war nicht zu erwarten, dass sich das Resultat mit mehr Generationen markant verbessert hätte.

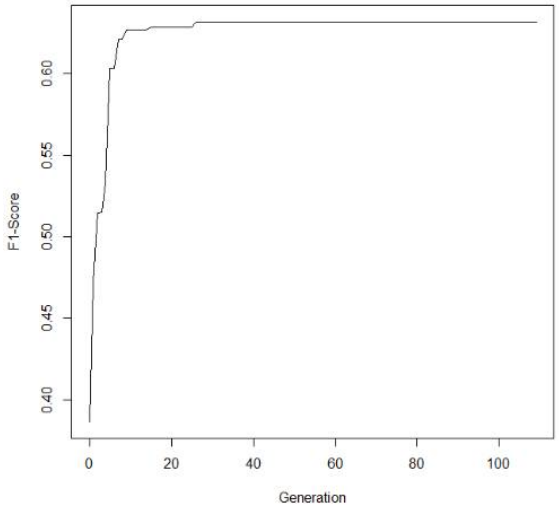


Abb. 11: Entwicklung des F_1-Scores für *point-cloud region growing*

Das Resultat in Abb. 11 zeigt, wie sich der durchschnittliche F_1-Score je Testgebiet durch den evolutionären Prozess und der Anpassung der Parameter stetig verbesserte. Nach 100 Generationen erreichte der Algorithmus einen maximalen Wert von 0.63 mit 420 korrekt detektierten Bäumen, 119 falsch positiven und 444 falsch negativen Treffern. Diese Werte wurden bereits mit der 26. Generation erreicht und danach nicht mehr übertroffen.
Die tolerierte Abweichung gegenüber dem Referenzdatensatz bezüglich Lage und Höhe betrug 0.25 Meter, da dies der mittleren Distanz zweier LiDAR Punkte entsprach. Faktisch musste

der Algorithmus mit dieser Vorgabe exakt den gleichen LiDAR Punkt als Kronenspitze wie im Referenzdatensatz markieren, um einen Baum korrekt zu klassieren.

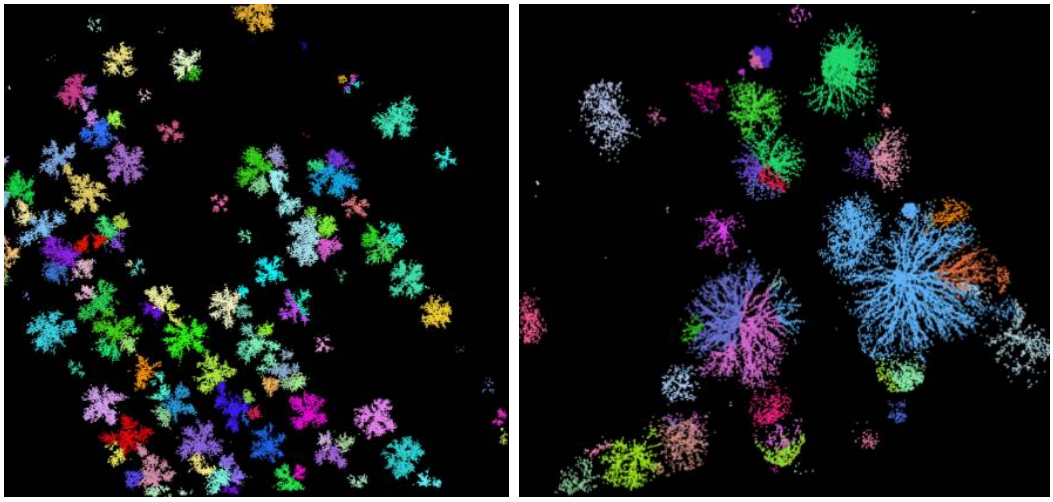


Abb. 12: Obstgarten und Stadtpark mit *point-cloud region growing*

Da der Algorithmus die einzelnen LiDAR Punkte zu individuellen Bäumen zuordnet, geht keine Information verloren. Etwaige Parameter wie Durchmesser oder Höhe müssten in weiteren Schritten berechnet werden. Die Abb. 12 zeigt das Resultat des Algorithmus für den Obstgarten und den Stadtpark. Man erkennt, dass im Obstgarten einige Bäume übersegmentiert und die multiplen Kronenspitzen als individuelle Bäume klassiert wurden. Im Stadtpark wurden weniger Bäume übersegmentiert. Der Algorithmus hat jedoch nicht erkannt, dass sich nordwestlich der grossen blauen Bergulme noch ein kleinerer Baum verbirgt und diesen untersegmentiert.

Laufzeit

Obwohl die acht neuen Individuen einer jeden zusätzlichen Generation in acht Prozessorkernen parallel gerechnet haben, benötigten die 100 Generationen eine Rechenzeit von 43 Stunden. Ein Testlauf des Algorithmus mit einer originalgrossen Kachel von rund einem Quadratkilometer musste nach 3 Stunden bei einem Fortschritt von 2% abgebrochen werden. Die eingesetzte Hard- und Software scheiterte an der quadratischen Laufzeitkomplexität dieses Algorithmus (siehe Kapitel 2.3.2). Aus diesem Grund wurden keine weiteren Schritte unternommen, diesen Algorithmus besser zu parametrisieren. Da anzunehmen ist, dass andere Verfahren direkt in der Punktwolke laufzeittechnisch nicht besser abschneiden würden, wurden keine weiteren dieser Verfahren evaluiert.

3.3.2 Region-based marker-controlled watershed

Aus den bildbasierten Verfahren scheint bezüglich Laufzeit und Resultat ein *watershed* Ansatz am vielversprechendsten. Der entscheidende Faktor bei der *watershed* Segmentierung sind adäquat platzierte Marker, welche den Baumkronen entsprechen müssen (Chen et al. 2006). Ausgehend von diesen Markern werden die Gebiete virtuell geflutet. Am Ende resultieren

genauso viele Bildsegmente (Bäume) wie Marker. Als gute Marker haben sich lokale Maxima erwiesen (Schardt et al. 2002, Chen et al. 2006, Mohd Zaki et al. 2015).

Watershed Algorithmen neigen zu Übersegmentierung, wenn die Baumkronenoberfläche nicht geglättet wird. Das Problem verschärft sich bei laublosen Bedingungen, da einzelne grosse Äste als separate Bäume klassiert werden. Die Glättung der Oberfläche mit einem Gaussfilter hat sich als geeignete Gegenmassnahme etabliert (Chen et al. 2006).

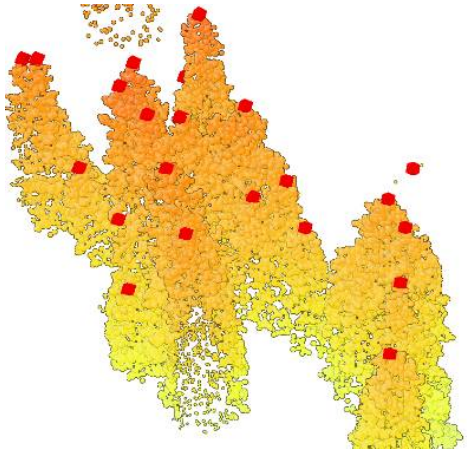


Abb. 13: Dicht stehende Nadelbäume im Bergwald

Erste Tests mit dem *watershed* Algorithmus haben gezeigt, dass die evolutionäre Entwicklung der Parameter dazu führte, dass ein Gleichgewicht zwischen Über- und Untersegmentierung hinsichtlich aller sechs Testgebiete entstand. Um bessere Resultate zu erhalten, war es besser, die Parameter regionsbasiert zu variieren. Empirisch hat sich gezeigt, dass die Trennung in geschlossenen Wald *(forest)* und einzelne Bäume, respektive kleine Baumgruppen *(field)* entscheidend ist. Ein weiteres Kriterium ist die Unterscheidung in Nadel- und Laubbäume, da erstere tendenziell näher beieinanderstehen können. Teilweise stehen über 20 Meter hohe Nadelbäume mit einem engen Kronenabstand in lockeren Gruppen (Abb. 13). Aus diesem Grund wurden die Gebiete vor der Segmentierung aufgrund ihrer Fläche in zwei Regionen eingeteilt und separat prozessiert. Da in den nördlichen Voralpen ab mittelmontaner Stufe zunehmend Nadelbäume vorkommen, wurden alle Gebiete höher als 1'000 Meter über Meer auch zur Waldregion gezählt (Ellenberg und Leuschner 2010). Die optimalen Schwellwerte für das vorliegende Gebiet lernte der Algorithmus durch den evolutionären Prozess selbständig. Die anschliessende Rundung hatte keine Auswirkung auf das Resultat.

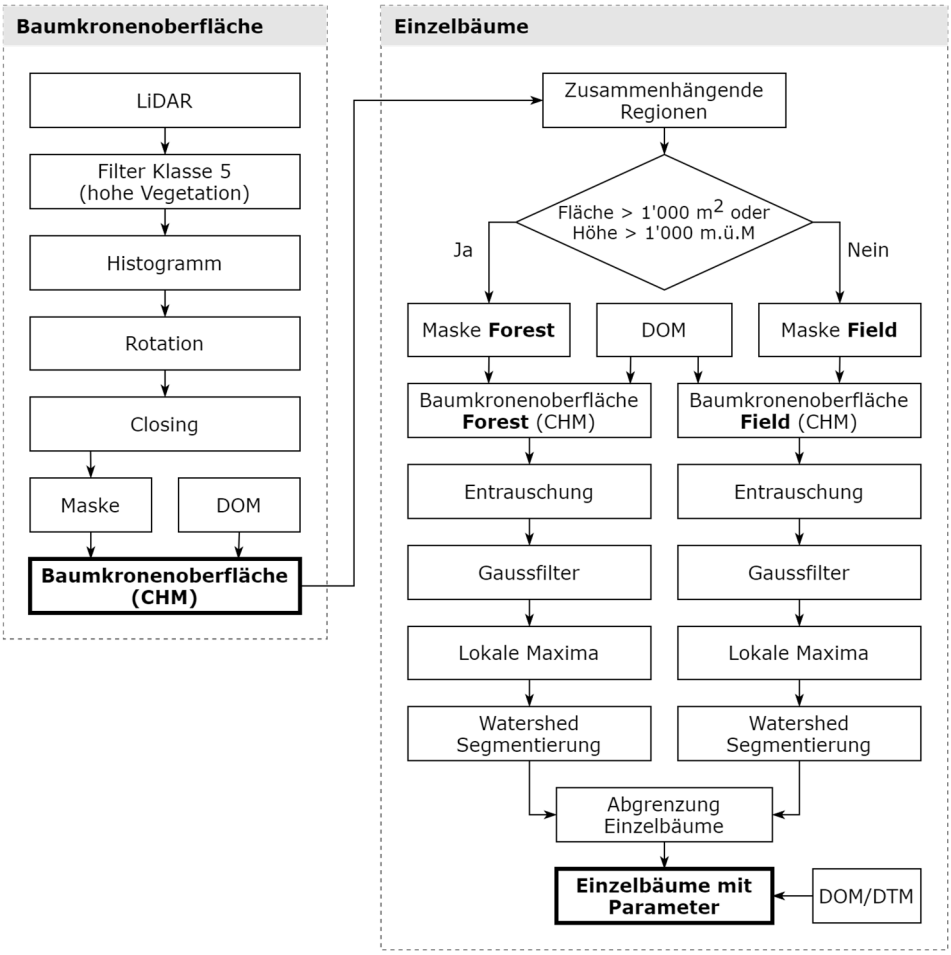


Abb. 14: Flussdiagramm des *watershed* Algorithmus

Die Abb. 14 zeigt den Vorgang zur Extraktion der Einzelbäume, wobei die Arbeiten von Chen et al. (2006) als Grundlage dienten. Die Neuerung besteht aus den regionsbasiert adaptierten Parametern und zusätzlichen Bildaufbereitungsschritten, wobei bei letzteren die Bibliothek *scikit-image* (van der Walt et al. 2014) zum Einsatz kommt. Der Prozess ist komplett in Python (Python Software Foundation 2019) realisiert und konnte somit mit dem bestehenden Testframework evaluiert werden.

Im ersten Schritt wird aus der LiDAR Punktwolke und dem digitalen Oberflächenmodell (DOM) die Baumkronenoberfläche (CHM) erzeugt. Durch filtern der Punktwolke auf die Klasse 5 (hohe Vegetation), die Projektion der Punkte auf die Ebene (Histogramm), eine Rotation um -90 Grad und das morphologische *closing* entsteht eine binäre Maske. Weisse Regionen repräsentieren Gebiete mit Bäumen. Das DOM wird mit der Maske überlagert, um die Baumkronenoberfläche *(canopy height model)* zu erhalten. Danach wird im zweiten Schritt zur Einzelbaumextraktion die Baumkronenoberfläche in Gebiete unterteilt, welche entweder grösser als 1'000 Quadratmeter oder höher als 1'000 Meter über Meer liegen. Die zwei resultierenden Masken werden wiederum auf das DOM angewendet, um die *watershed* Segmentierung regionsbasiert auszuführen. Vor der eigentlichen Segmentierung werden die Baumkronenoberflächen entrauscht, mit einer Gaussfunktion geglättet und mittels lokaler Maxima Marker gesetzt. Die *watershed* Segmentierung wird ausgehend von diesen Startpunkten

vollzogen, wobei die Höheninformation verloren geht. Wenn im letzten Schritt die abgegrenzten Einzelbäume wieder mit dem DOM und dem DTM überlagert werden, können nebst der Position und des Durchmessers die relativen Baumhöhen wieder berechnet werden.

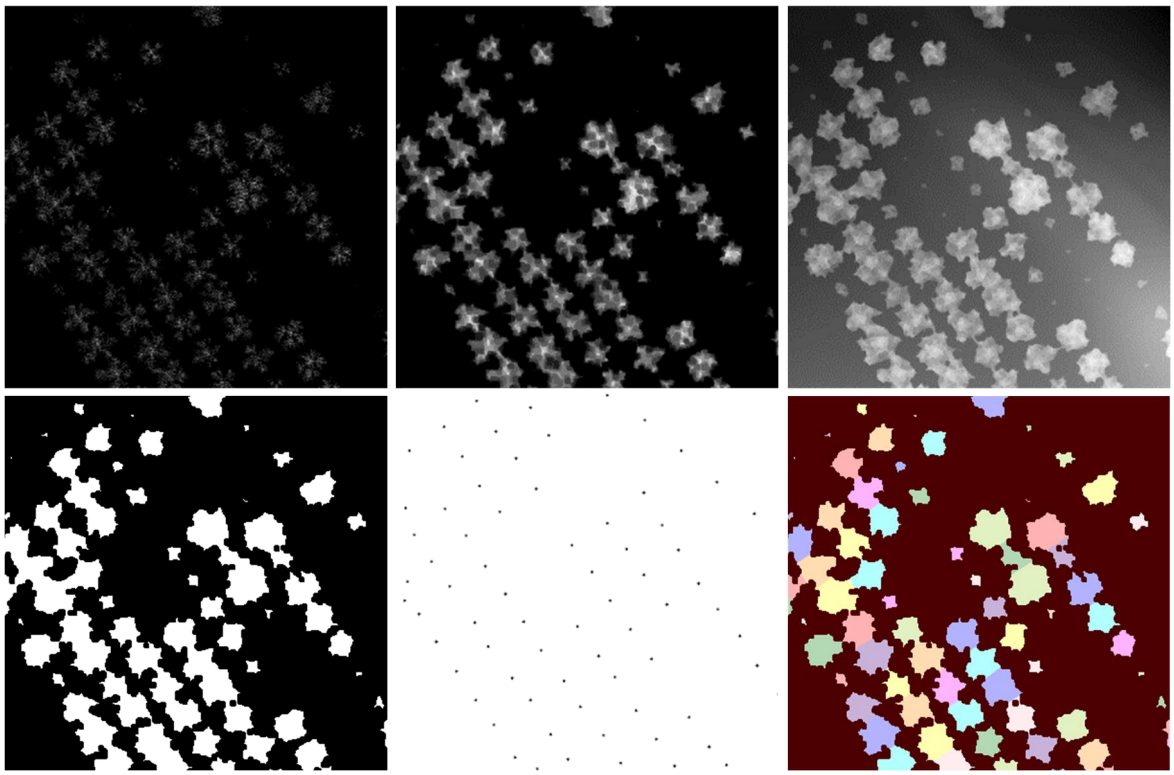


Abb. 15: Zwischenresultate des *watershed* Algorithmus für den Obstgarten

Der Workflow zeigt, dass die LiDAR Punktwolke einerseits dazu gebraucht wird, um das Oberflächenmodell auf diejenigen Bereiche zu filtern, welche Bäume enthalten und andererseits um am Ende die exakten Koordinaten der Bäume zu bestimmen. Am Beispiel des Obstgartens erkennt man in Abb. 15 die einzelnen Zwischenresultate. Oben links sind die auf die Bildebene projizierten LiDAR Punkte der Klasse 5 (hohe Vegetation) erkennbar (Histogramm). Direkt daneben ist das Resultat des morphologischen *closing*, welches dazu gebraucht wird, um Lücken zwischen den einzelnen Punkten zu schliessen. Ansonsten gäbe es Löcher in der Maske für das digitale Oberflächenmodell, welches oben rechts abgebildet ist. In der unteren Reihe ist links die binäre Maske zur Filterung des Oberflächenmodells auf Bereiche mit Bäumen zu erkennen. Im mittleren Bild sind die Marker der lokalen Maxima erkennbar, welche als Startpunkte für die finale *watershed* Segmentierung dienen. Das Endresultat ist im Bild unten rechts zu sehen, wobei jeder individuelle Baum mit einer zufälligen Farbe eingefärbt wurde.

Resultate

Als erlaubte Toleranz zur evolutionären Berechnung der optimalen Parameter wurde ein Minimalabstand von 0.5 Meter zum Referenzdatensatz eingestellt. Ein Optimum stellt sich jedoch immer unabhängig der gewählten Toleranz ein.

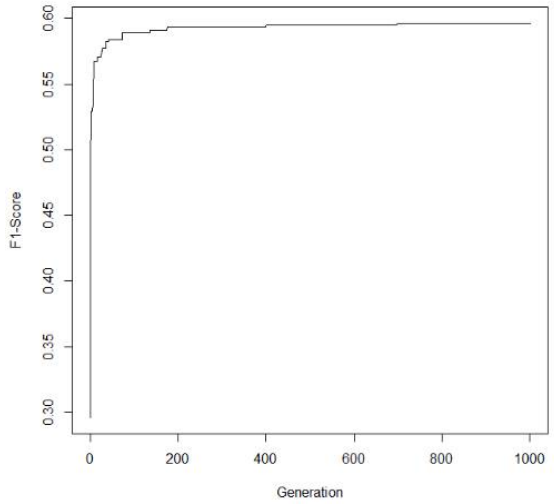


Abb. 16: Entwicklung des F_1-Scores für *watershed*

Die Parameter für den *watershed* Algorithmus haben sich so entwickelt, dass sich der F_1-Score asymptotisch gegenüber dem Wert 0.6 verhielt (Abb. 16). Berechnet wurden insgesamt 1'000 Generationen, wobei ein F_1-Score von 0.59 erreicht wurde. Dieser Wert wurde näherungsweise sehr schnell erreicht und verbesserte sich nur noch in der 400. und der 700. Generation.

Parameter	Beschreibung	Wert	*forest*	*field*
forest threshold	Minimale Fläche in Quadratmetern, ab der eine Region mit den Parametern für die Region *forest* prozessiert wird.	1'000		
height threshold	Minimale Höhe in Metern über Meer, ab der eine Region mit den Parametern für die Region *forest* prozessiert wird.	1'000		
histogram closing radius	Radius des Kreises in Anzahl Pixel, mit der die morphologische Bildoperation *closing* für die auf die Ebene projizierte Punktwolke aller hohen Vegetationspunkte ausgeführt wird.	3.3		
denoising weight	Gewichtung des Entrauschens der Baumkronenoberfläche. Mit höherem Wert werden gröbere Pixelfehler korrigiert.		0	75
sigma	Standardabweichung der Gaussschen Verteilung zur Glättung der Baumkronenoberfläche. Je höher der Wert, desto mehr wird die Oberfläche geglättet.		0.7	2.3
truncate	Bestimmt zusammen mit *sigma* die Grösse der Maske, die bei der Glättung der Baumkronenoberfläche mit der Gaussschen Funktion verwendet wird.		25	4.0
minimum distance	Definiert den minimalen Abstand zweier lokaler Maxima (zweier Baumkronen) in Anzahl Pixel.		5	6
compactness	Definiert, wie kompakt die Gebiete nach der *watershed* Segmentierung sein dürfen. Je grösser der Wert, desto gleichförmigere Gebiete (Baumkronen) entstehen.		110	120

Tab. 5: Angelernte Parameter des Algorithmus

Die nach 1'000 Generationen evolutionär gelernten Parameter sind in der Tab. 5 dokumentiert. Man erkennt, dass es offenbar für Waldgebiete nicht notwendig war, die Baumkronenoberfläche zu entrauschen *(denoising)*. Individuen mit einem Entrauschungsfaktor grösser als Null waren nach einer gewissen Zeit ausgestorben. Zudem musste die Oberfläche für Wälder weniger stark geglättet werden, was am kleineren Sigma erkennbar ist. Offenbar würden zu viele Baumkronen nicht mehr erkannt werden, wenn die Glättung zu gross wäre. Die Individuen lernten, dass Bäume in geschlossenem Wald und Nadelbäume in Hochlagen näher beisammenstehen. Der minimale Baumabstand war deshalb für diese Gebiete kleiner als bei Bäumen in offenem Gelände.

Durch die Rasterung des Oberflächenmodells und des Histogramms der LiDAR Punktwolke mit einer Auflösung von 0.25 Meter pro Pixel litt die Lage- und Höhengenauigkeit der ermittelten Baumkronen. Aus diesem Grund war der F_1-Score bei einer Toleranz von 0.25 Meter im Vergleich zum Verfahren direkt in der Punktwolke um einiges tiefer.

Toleranz	0.25m				0.5m				1m			
Testgebiet	TP	FP	FN	F_1	TP	FP	FN	F_1	TP	FP	FN	F_1
Bergwald	33	55	87	**0.32**	69	19	51	**0.66**	79	9	41	**0.76**
Obstgarten	20	52	51	**0.28**	41	31	30	**0.57**	60	12	11	**0.84**
Laubwald	36	156	176	**0.18**	89	103	121	**0.44**	103	89	108	**0.51**
Nadelwald	87	248	266	**0.25**	206	129	147	**0.60**	274	61	78	**0.80**
Siedlung	22	36	45	**0.35**	42	16	25	**0.67**	50	8	17	**0.80**
Stadtpark	8	27	27	**0.23**	22	13	13	**0.62**	25	10	10	**0.71**
Total	**206**	**574**	**652**	**0.25**	**469**	**311**	**387**	**0.59**	**591**	**189**	**265**	**0.72**

Tab. 6: Präzision des *watershed* Algorithmus je Testgebiet

Die Tab. 6 dokumentiert, wie sich die Werte für *true positives (TP), false positives (FP), false negatives (FN)* und der F_1-Score mit grösserer Toleranz verbesserten. Über alle Testgebiete hinweg erreichte der Algorithmus bei einer Toleranz von einem Meter einen Wert von 0.72, was im Vergleich zu anderen Studien tiefer war (Li et al. 2012, Silva et al. 2016). Die Variabilität bezüglich Baumarten und -grössen war in den sechs Testgebiete jedoch grösser als in den Gebieten der verglichenen Studien.

Am meisten Mühe bekundete der Algorithmus im Laubwald. Bei einer geschlossenen Laubdecke wurde es praktisch unmöglich, mit diesem Verfahren einzelne Bäume voneinander zu trennen. Dies war absehbar, da es bei der manuellen Erstellung des Referenzdatensatzes bereits unmöglich war, einzelne Bäume voneinander zu trennen. In absoluten Zahlen wies der Algorithmus im Laub- und Bergwald im Vergleich zum Referenzdatensatz zu wenig Bäume aus (viele falsch negative Treffer). Im Obstgarten, im Nadelwald und in der Siedlung stimmte die Anzahl der Bäume sehr gut und der Algorithmus erreichte bei einem Meter Toleranz einen F_1-Score über 0.8. Im Stadtpark stimmte zwar die Anzahl der Bäume, der F_1-Score war jedoch aufgrund von Übersegmentierung der grossen Bäume tiefer.

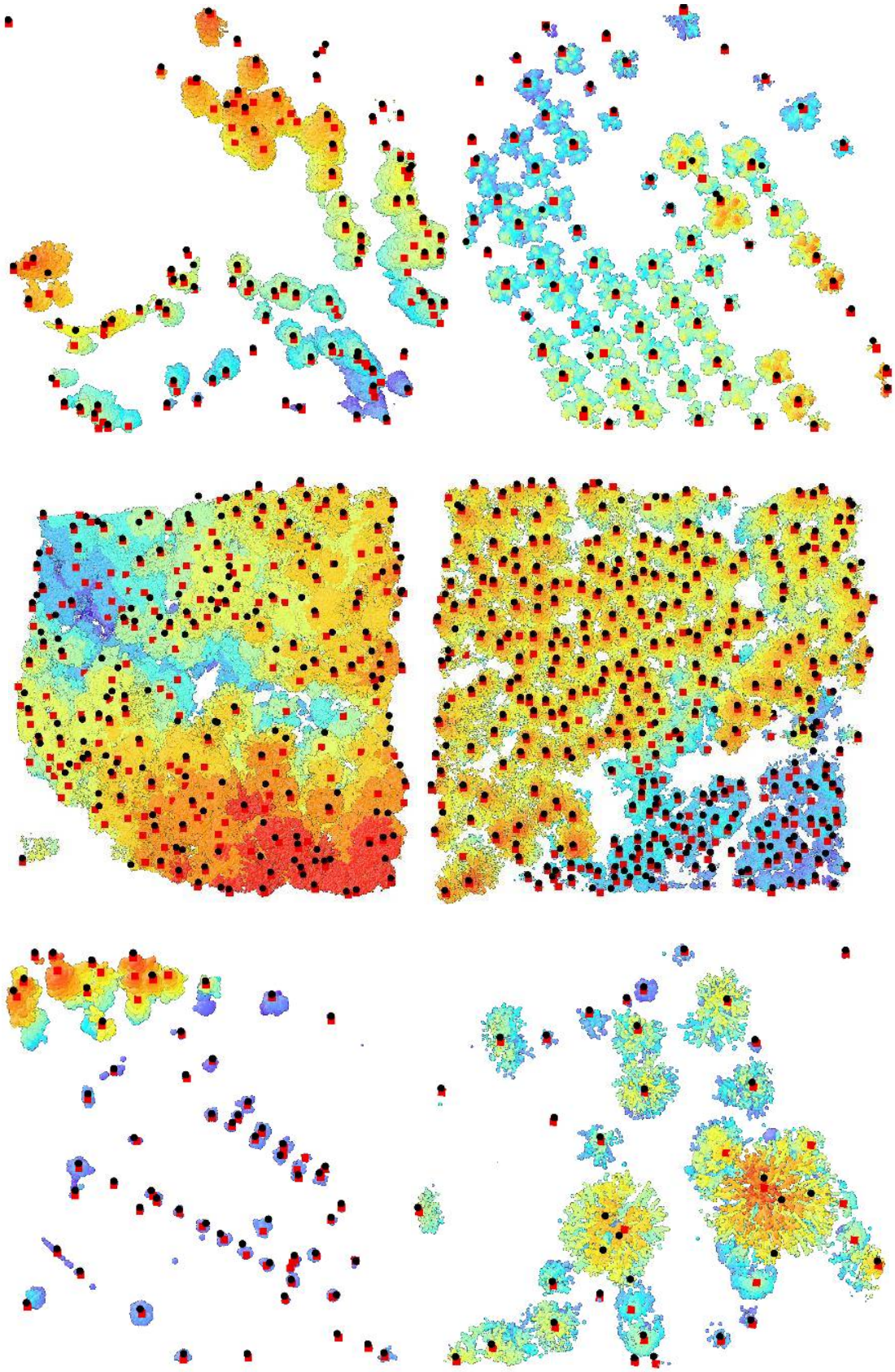


Abb. 17: Resultate des *watershed* Algorithmus je Testgebiet
Referenzdatensatz (rot) im Vergleich zu den ermittelten Baumkronen (schwarz) für die Gebiete Bergwald, Obstgarten, Laubwald, Nadelwald, Siedlung, Stadtpark

Die Abb. 17 zeigt die Resultate für alle Testgebiete im direkten Vergleich. Es lässt sich erkennen, dass bei regelmässiger Anordnung und Grösse der Bäume das Resultat besser war. Am besten wurden die Bäume im Obstgarten erkannt, gefolgt von der Siedlung und dem Nadelwald. Schon fast chaotisch erscheint das Bild beim Laubwald, auf dem die Kronenspitzen scheinbar willkürlich über die geschlossene Laubdecke verteilt sind.

Fehlerdokumentation

Der Algorithmus neigte zu Fehlern, welche entweder aufgrund der Datengrundlage zu Stande kommen oder sich inhärent zum Algorithmus ausbildeten.

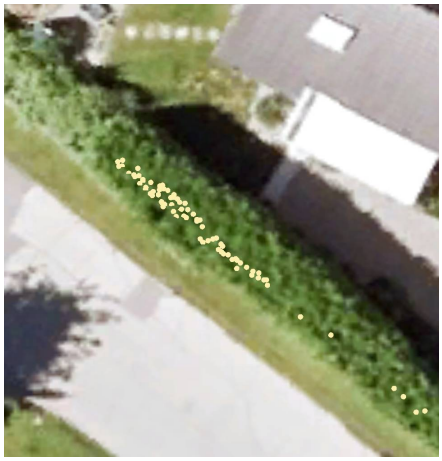


Abb. 18: Als hohe Vegetation falsch klassierte LiDAR Punkte

Das Verfahren verlässt sich darauf, dass die LiDAR Punktwolke korrekt klassiert ist. In den Testgebieten konnten Fälle beobachtet werden, bei denen beispielsweise tiefe Hecken als hohe Vegetation klassiert wurden (Abb. 18). Im resultierenden Einzelbaumdatensatz werden an solchen Stellen Bäume zu stehen kommen. Je nachdem wie häufig die Falschklassierung vorkommen sollte, müssten geeignete Gegenmassnahmen getroffen werden.

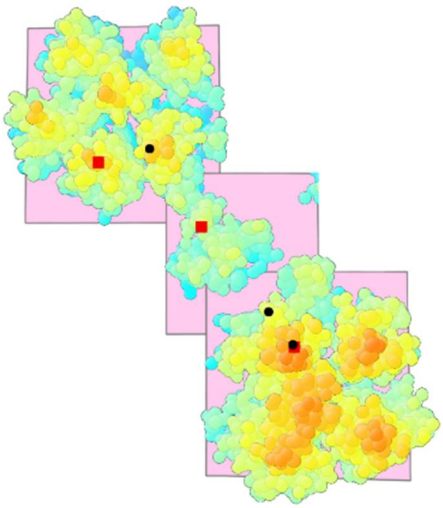


Abb. 19: Kronenspitzen und Begrenzung dreier Bäume im Obstgarten
Referenzdatensatz (rot), berechnete Baumkronen (schwarz)

Die Ermittlung der Lage und Höhe der Baumkrone kann auf unterschiedliche Art und Weise erfolgen und hat einen unmittelbaren Einfluss auf das Resultat. Die Abb. 19 zeigt anhand der Begrenzungsboxen, wie der Algorithmus die drei Bäume korrekt erkannt hat. Die roten Punkte sind die Kronenspitzen aus dem Referenzdatensatz, während die schwarzen Punkte die vom Algorithmus ermittelten Baumkronen markieren. Die Kronenspitzen wurden bei den beiden nördlichen Bäumen nicht korrekt gesetzt und haben trotz korrekt erkannten Bäumen den F_1-Score negativ beeinflusst. Die Spitzen wurden am höchsten Punkt des digitalen Oberflächenmodells innerhalb des begrenzenden Polygons gesetzt. Beim nördlichen Baum stimmt dieser Punkt offenbar nicht mit der tatsächlichen Punktwolke überein, was durch Informationsverlust bei der Rasterung begründbar ist. Beim mittleren Baum wurde der höchste Punkt aufgrund von Überlappung in den Nachbarbaum gesetzt. Das Resultat sähe anders aus, wenn man die Lage der Kronenspitze in den Schwerpunkt der Begrenzungsbox setzen würde. Damit würde man eine bessere Lagegenauigkeit auf Kosten der Höhengenauigkeit erzielen.

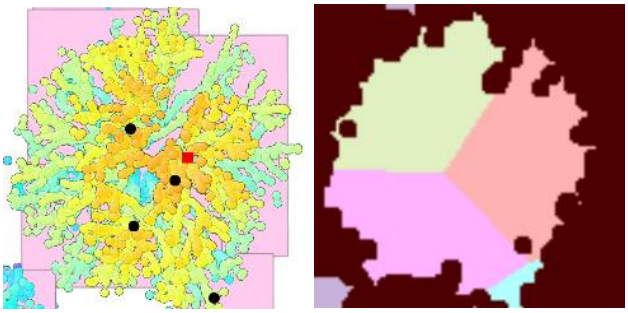


Abb. 20: Übersegmentierung von grossen laublosen Bäumen

Eine weitere beobachtete Fehlerquelle ist die Übersegmentierung insbesondere von grossen laublosen Bäumen (Abb. 20). Der Algorithmus war nicht immer in der Lage, die Oberfläche so weit zu glätten, dass grosse Bäume als einzelne Objekte erkannt wurden. Respektive die Glättung hätte so stark sein müssen, dass in anderen Gebieten zu viele Bäume untersegmentiert worden wären. Da die nördlichen Gebiete des Kantons im Spätwinter und Anfang Frühling beflogen wurden, wird die Übersegmentierung in diesen Gebieten tendenziell grösser sein.

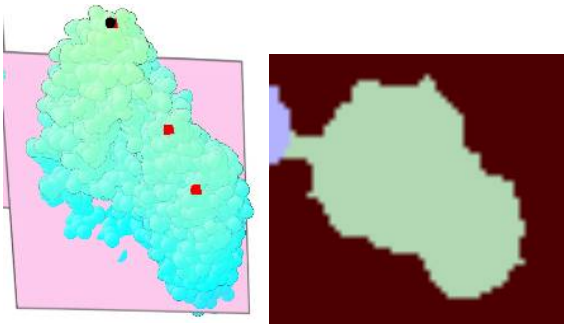


Abb. 21: Untersegmentierung von grossen dicht stehenden Nadelbäumen

Dicht stehende Nadelbäume wurden durch den Algorithmus teilweise untersegmentiert und damit nicht voneinander getrennt. Die Abb. 21 zeigt eine Baumgruppe im Bergwald bestehend

aus drei Fichten, welche vom Algorithmus als einzelner Baum klassiert wurde. In steilem Gelände kommt erschwerend hinzu, dass sich die Baumkronen treppenartig aneinanderreihen und sich kaum lokale Maxima ausbilden.

Die Über- und Untersegmentierung liess sich folglich nicht komplett verhindern. Durch die Wahl des F_1-Scores als Fitnessparameter wurden diese beiden Fehler jedoch über alle Testgebiete hinweg gleichermassen minimiert.

Laufzeitanalyse

Die Berechnung der 1'000 Generationen nahm rund 5 Stunden in Anspruch. Bei acht Kindern und je sechs Testgebieten wurden Total 48'000 Mal Einzelbäume detektiert, was einem Durchschnitt von rund 0.4 Sekunden pro prozessiertem Testgebiet entspricht. Dieser Wert konnte nur dank der parallelen Berechnung mit acht Prozessorkernen erreicht werden.
Zur Hochrechnung der Laufzeit auf das gesamte Kantonsgebiet wurde aus den originalgrossen Kacheln diejenige ausgewählt, welche am meisten Speicherplatz benötigte (1.9 Gigabyte) und somit eine hohe Punktdichte enthielt. Die Extraktion der Einzelbäume, inklusive Export von einigen Baumparametern wie der Höhe oder des Durchmessers, dauerte für dieses Gebiet rund 3 Minuten. Hochgerechnet auf das gesamte Kantonsgebiet mit 1'760 Kacheln sollte dieser Algorithmus nach rund 88 Stunden terminieren. Wenn die Kacheln parallel prozessiert werden, kann die Laufzeit noch um ein Vielfaches reduziert werden. Bei dieser Hochrechnung nicht berücksichtigt sind allfällige Vor- und Nachbereitungsschritte.

3.4 Ergebnisse und Diskussion

Verfahren, welche direkt in der LiDAR Punktwolke operieren, liefern exaktere Resultate, wären jedoch zu komplex, als dass sie auf das gesamte Kantonsgebiet angewendet werden können. Mit dem *region-based marker-controlled watershed* Algorithmus, welcher mit einer Baumkronenoberfläche arbeitet, konnte ein adäquates Verfahren gefunden werden, welches einem guten Kompromiss zwischen Präzision und Laufzeit entspricht. Über alle Testgebiete hinweg weist der Algorithmus in absoluten Zahlen zu wenig Bäume aus. Wenn man bedenkt, dass unter einer geschlossenen Baumkronenoberfläche noch kleinere Bäume vorkommen, wird am Ende die Zahl der Bäume zu klein sein. Der Algorithmus wird demnach nicht Anzahl und Position aller Bäume im beobachteten Gebiet liefern, sondern nur die Teilmenge von eindeutig bestimmbaren markanten Bäumen. Als markant gelten diejenigen Bäume, welche isoliert stehen oder in Gruppen oder geschlossenem Wald eindeutige Kronenspitzen aufweisen.

Bei einem mittleren F_1-Score von 0.72 bei einem Meter Lage- und Höhentoleranz, werden Über- und Untersegmentierung bestmöglich minimiert. Eine objektive Bewertung dieses Resultats anhand des F_1-Scores lässt sich nur im Vergleich zu anderen Arbeiten machen. Dabei hat sich gezeigt, dass der Wert tiefer ausfällt (Li et al. 2012, Silva et al. 2016), was mit der grossen Variabilität der Bäume im prozessierten Gebiet zu begründen ist. Wenn man beispielsweise den finalen Einzelbaumdatensatz in einer Hintergrundkarte über das gesamte Kantonsgebiet einbauen möchte, so spielt die Höhengenauigkeit keine Rolle und die

Lagegenauigkeit von einem Meter ist durchaus genügend. Auch falsch positive oder falsch negative Bäume sollten das gesamte Kartenbild nicht allzu stark beeinträchtigen. Wenn der Einzelbaumdatensatz hingegen in einer kleinräumigen 3D Szene für ein zukünftiges Bauprojekt zum Einsatz kommen sollte, sind die Anforderungen an die Präzision der zu Grunde liegenden Daten höher. Die automatisiert erstellten Einzelbäume müssten manuell überprüft und gegebenenfalls korrigiert werden, um einem solchen Anwendungsszenario gerecht zu werden.

Die Hochrechnung der Laufzeit hat gezeigt, dass der Algorithmus auf einem gängigen Bürocomputer nach maximal 3.5 Tagen terminieren sollte. Die Laufzeit ist kurz, da der Algorithmus eine lineare Laufzeitkomplexität aufweist. Mit paralleler Prozessierung wird die Laufzeit voraussichtlich halbiert werden können. Aus diesen Gründen wird das gesamte Kantonsgebiet mit dem entwickelten *region-based marker-controlled watershed* Algorithmus prozessiert werden.

4 Anwendung auf dem Gebiet des Kantons Luzern

Eine typische Prozesskette bei der Verarbeitung von grossen wissenschaftlichen Datenbeständen besteht aus dem Laden, der Verarbeitung und der Speicherung der Daten. Dabei eignen sich insbesondere rechenintensive Schritte zur parallelen Prozessierung (Ghosh 2019). Ein Problem lässt sich nur dann parallel verarbeiten, wenn es in Teilschritte zerlegt werden kann. Rasterdaten sind a priori gebietsweise zerlegt *(domain decomposition)* und eigenen sich besonders gut zur parallelen Prozessierung. Dazu braucht es einen Hauptprozess, welcher die Subprozesse steuert und die Resultate zusammenfügt (Ding und Densham 1996). Nach diesem Paradigma richtet sich die Entwicklung des finalen Workflows zur Einzelbaumdetektion auf dem gesamten Kantonsgebiet.

4.1 Adaption des Algorithmus

Der im vorherigen Kapitel evaluierte *region-based marker-controlled watershed* Algorithmus konnte nicht ohne Anpassungen auf das gesamte Kantonsgebiet angewendet werden. Einerseits musste dafür gesorgt werden, dass Einzelbäume an Kachelrändern nicht doppelt gezählt werden. Andererseits musste der Algorithmus durch Parallelisierung laufzeittechnisch optimiert und in einen Gesamtablauf mit Vor- und Nachbearbeitung eingebettet werden. Darüber hinaus gab es korrigierende Massnahmen, welche zur besseren Qualität des finalen Datensatzes getroffen werden mussten.

4.1.1 Randproblematik

Die zur Verfügung stehenden LiDAR Daten, das Oberflächen- und Terrainmodell lagen in gekachelter Form vor, ohne dass sich die Gebiete überlappten. Um Randartefakte zu vermeiden, war es notwendig, die Kacheln vor der Prozessierung zu puffern (Isenburg 2015). Bei der Einzelbaumsegmentierung hat Bischoping (2007) das Problem so gelöst, dass nach dem Puffern der Kacheln und der Extraktion der Einzelbäume der Schwerpunkt eines Baumes bestimmt, zu welcher Kachel er gehört. Respektive wenn der Schwerpunkt eines Baumes innerhalb des Puffers zu liegen kommt, wird er bei der soeben prozessierten Kachel eliminiert. Er wird bei der Verarbeitung der Nachbarkachel nochmals gefunden und zu jener zugeordnet werden. Diese Vorgehensweise wurde für die vorliegende Arbeit übernommen.

Da die Laufzeit mit zunehmender Puffergrösse leidet, konnte dieser nicht beliebig gross gewählt werden. Als Puffergebiet wurde der Algorithmus mit einem Streifen von 20 Meter Breite parametrisiert. Mit dieser Breite war sichergestellt, dass Bäume mit einem Kronenradius bis 20 Meter (Kronendurchmesser bis 40 Meter) noch als Ganzes zu einer Kachel zugeordnet werden. Es ist unwahrscheinlich, dass es im Kantonsgebiet Bäume mit grösserem Kronenradius gibt.

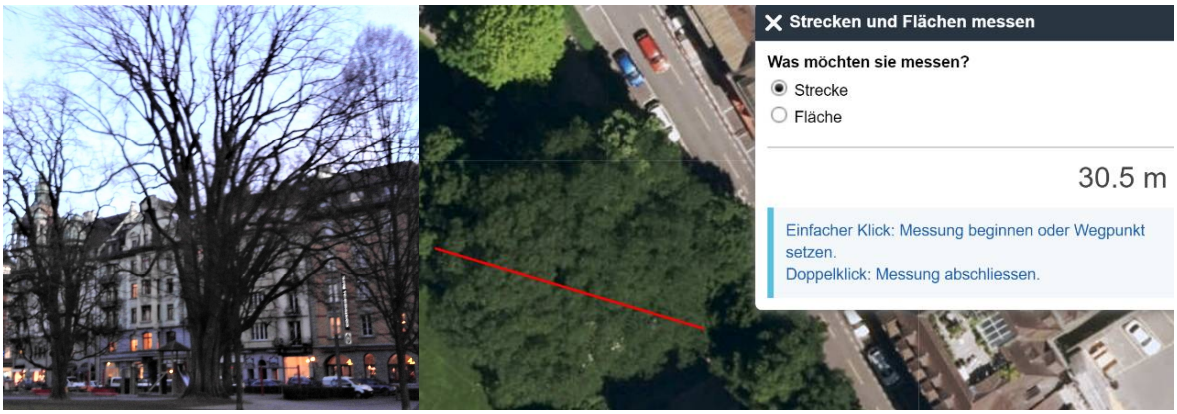


Abb. 22: Bergulme im Stadtpark mit 30 Meter Kronendurchmesser

Auf der kollaborativen Webseite *monumentaltrees.com* (2020) ist beispielsweise die Bergulme *(Ulmus glabra)* verzeichnet, welche im Testgebiet des Stadtparks bereits dokumentiert wurde (siehe Kapitel 3.2.1). Der prächtige Baum hat einen Stammumfang von rund 5.75 Meter und einen Kronendurchmesser von rund 30 Meter. Es ist davon auszugehen, dass grössere Bäume eher unwahrscheinlich sind. Daraus folgt, dass mit einem Randgebiet von 20 Meter gewährleistet ist, dass keine Übersegmentierung durch Randartefakte entstehen werden.

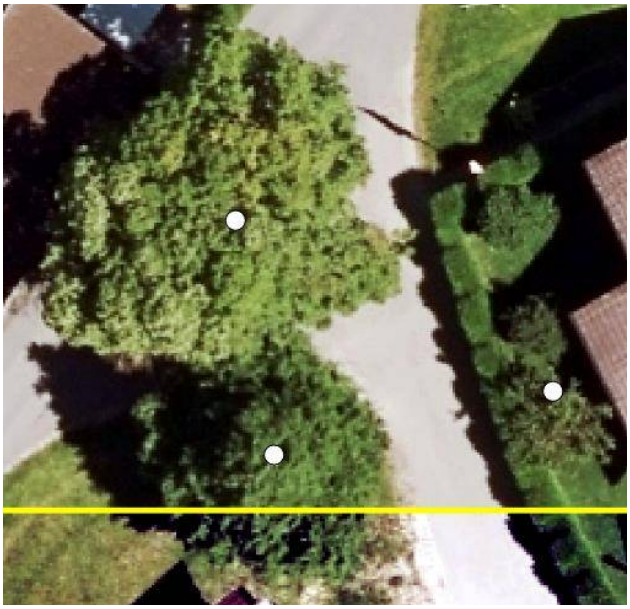


Abb. 23: Baumgruppe an Kachelgrenze

Um die Annahme zu testen, wurden exemplarische Kacheln prozessiert, welche Bäume an Kachelrändern haben. Die Abb. 23 zeigt, dass die getroffene Massnahme zur Vermeidung von Randartefakten gut funktionierte. Der südliche Baum geht über die gelbe Kachelgrenze hinaus, wurde jedoch nur einmal detektiert.

4.1.2 Behandlung von Datenfehlern

Die ersten Testläufe mit mehreren Kacheln offenbarten einige unrealistische Resultate, welche auf Fehler in den LiDAR Daten zurückzuführen waren. So gab es Bäume mit unrealistischen relativen Höhen (teilweise über 800 Meter), unrealistischen Durchmessern (bis zu 80 Meter) und unmöglichen Lagen mit über 1'900 Meter über Meer.

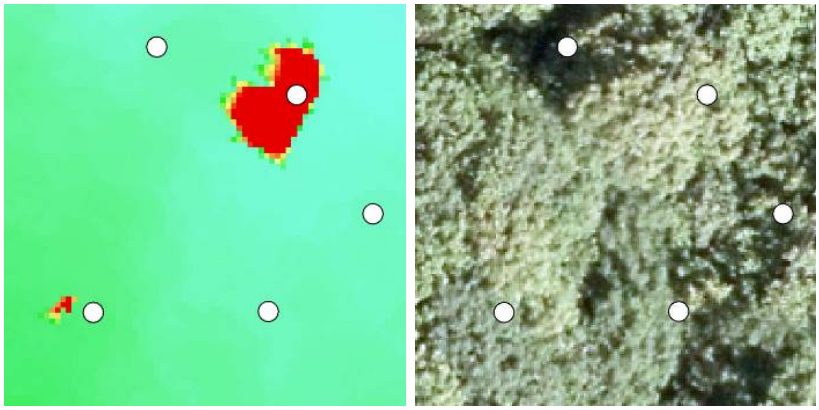


Abb. 24: Artefakte im digitalen Oberflächenmodell
Einzelbäume (weisse Punkte) auf DOM (links) und Orthofoto (rechts)

Die zu hohen Bäume liessen sich durch Artefakte in den LiDAR Daten erklären, welche sich ins digitale Oberflächenmodell durchgeschlagen haben. Die roten Bereiche im Oberflächenmodell der Abb. 24 liegen auf rund 1'430 Meter über Meer, während die umliegenden grünlichen Gebiete auf rund 1'350 Meter über Meer liegen. Das ergibt einen Höhenunterschied von 80 Meter mit senkrecht abfallenden Wänden. Das Orthofoto beweist, dass hier keine Struktur vorhanden ist (zum Beispiel ein Aussichtsturm oder ein Felsen), welche diese Oberfläche erklären könnte und es muss sich deshalb um einen Fehler in den LiDAR Daten handeln. Denkbar sind unbereinigte Reflektionen von Vögeln oder Gleitschirmfliegern, welche sich zum Zeitpunkt der Befliegung zwischen der Erdoberfläche und des Flugzeugs befanden. Bei der Berechnung der relativen Baumhöhe ergeben sich an Stellen mit solchen Artefakten unrealistisch hohe Werte. Die Website *monumentaltrees.ch* (Monumental Trees 2020) listet eine Weisstanne *(Abies alba)* als höchsten bekannten Baum der Schweiz mit einer Höhe von 58.10 Meter. Aus diesem Grund wurden alle Höhenresultate ab 60 Meter ausgenullt, da man von falschen Werten ausgehen muss. Die relative Höhe jener Bäume bleibt somit unbekannt.

Die zu grossen Durchmesser waren erklärbar durch untersegmentierte Bäume. Wenn zwei oder mehr Bäume als Einzelbaum zusammengefasst werden, entstehen unrealistische Kronendurchmesser. Wie bereits bei der Behandlung der Randproblematik vermerkt, ist es unrealistisch, dass es Bäume mit mehr als 40 Meter Kronendurchmesser gibt (siehe Kapitel 4.1.1). Deshalb wurden in der Attributtabelle des Einzelbaumdatensatzes alle Durchmesserwerte grösser als 40 Meter ausgenullt und damit auf unbekannt gesetzt.

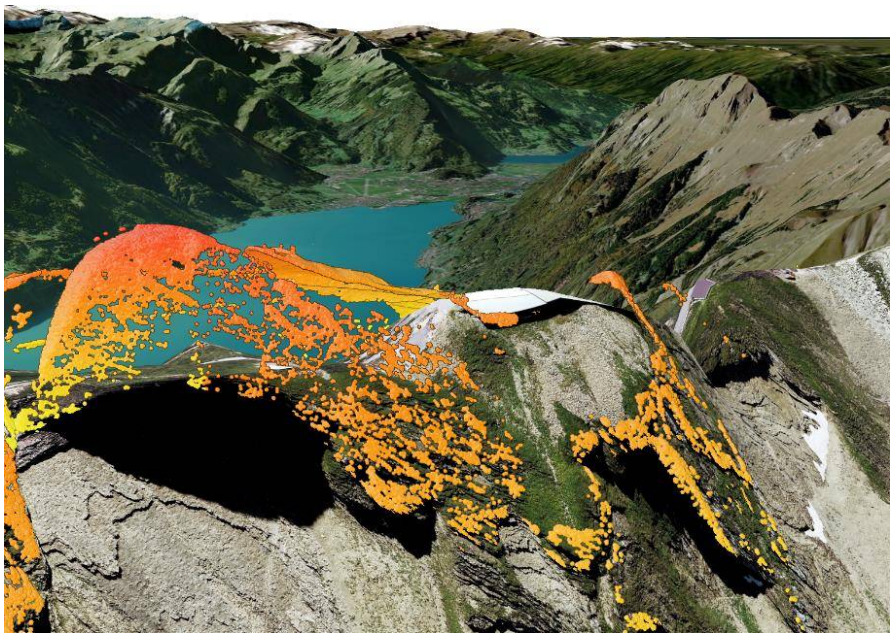


Abb. 25: Hohe Vegetation auf rund 2'200 Meter über Meer
Bergstation Brienzer Rothorn

Bereits bei der Methodenevaluierung hat sich gezeigt, dass es falsch klassierte LiDAR Punkte gibt. Der Algorithmus verlässt sich darauf, dass alle Punkte der Klasse 5 zur hohen Vegetation gehören und damit Bäumen entsprechen. Das Problem zeigte sich vor allem in Berggebieten, wo viele Steine und markante Geländekämme als hohe Vegetation klassiert wurden. Die Abb. 25 zeigt, wie beispielsweise rund um die Bergstation auf dem Brienzer Rothorn auf rund 2'200 Meter über Meer viele LiDAR Punkte als hohe Vegetation klassiert wurden. Ohne Korrektur dieses Fehlers würden erstens viel zu viele Bäume detektiert und zweitens Bäume bis auf über 2'000 Meter über Meer ausgewiesen werden. Der Fehler musste folglich in einem nachgelagerten Plausibilisierungsschritt korrigiert werden.

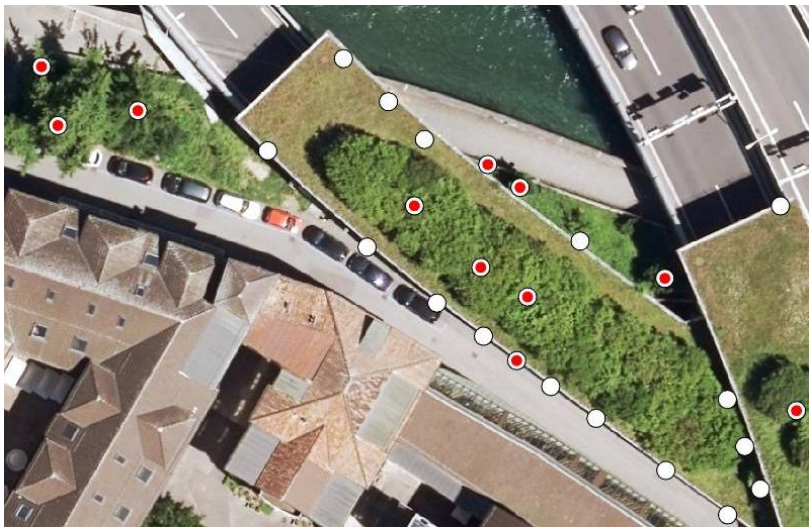


Abb. 26: Einzelbäume entlang von Mauern und Kunstbauten
Einzelbäume ohne Nachbehandlung (weiss), mit Nachbehandlung (rot)

Auch in städtischen Gebieten gab es viele falsch klassierte LiDAR Punkte, so dass eine Nachbehandlung zwingend notwendig wurde. Die Abb. 26 zeigt, wie beispielsweise entlang von Kunstbauten (Brücken, Tunneleinfahrten, etc.) systematisch falsche hohe Vegetationspunkte vorkamen und deshalb viele falsch positive Einzelbäume detektiert wurden. Auch Sonnenschirme oder einzelne LiDAR Punkte auf Hausdächern entsprachen Artefakten, welche eliminiert werden mussten.

Zur Minimierung dieser Fehler wurden erstens LiDAR Punkte ab 1'900 Meter über Meer eliminiert. Der Wert leitete sich aus der Baumgrenze in voralpinen Gebiet ab (Ellenberg und Leuschner 2010) und deckt sich mit dem empirischen Studium des Orthofotos für die relevanten Gebiete im Kanton Luzern. Um weitere Artefakte in der LiDAR Punktwolke zu eliminieren, wurde zweitens eine Formanalyse der mutmasslichen Einzelbäume gemacht. So konnte bei unrealistischen Grössen oder Formen mit grosser Wahrscheinlichkeit davon ausgegangen werden, dass es sich um falsch positive Resultate aufgrund falscher Klassierung der LiDAR Punktwolke handelte.

Parameter	Wert
Standort	$<$ 1'900 m ü.M.
Baumhöhe	$\geq$ 3 m
Kronendurchmesser breiteste Stelle (Hauptachse der Ellipse)	$>$ 0.5 m
Kronendurchmesser schmalste Stelle (Nebenachse der Ellipse)	$>$ 0.5 m
Exzentrizität der Baumkrone (Verhältnis von Neben- zu Hauptachse)	$>$ 0.25

Tab. 7: Randbedingungen für gültige Bäume

Die Tab. 7 listet alle Bedingungen, welche erfüllt sein mussten, damit ein Resultat als Einzelbaum behandelt wurde und bei der Nachbearbeitung im finalen Datensatz verblieb. Alle Werte wurden empirisch aufgrund zahlreicher Beispiele ermittelt und decken sich mit der Vorstellung der Form eines Baumes. Insbesondere mit der Exzentrizität der Baumkrone liess sich sehr einfach ermitteln, ob es sich um ein längliches Objekt wie beispielsweise ein Felsband, ein Dachvorsprung oder eine Mauerkante handelte. Da die Exzentrizität dem Verhältnis zwischen der Neben- und Hauptachse der Baumkronenellipse entspricht, ist bei kleinen Werten die Form viel länger als breit. Bei einem Wert von 1 würde es sich um einen perfekten Kreis und damit um eine perfekt runde Baumkrone handeln. Die Abb. 26 visualisiert mit roten Punkten, welche Bäume dank der Formanalyse im Resultat verblieben. Es ist gut erkennbar, dass das Problem entschärft, jedoch nicht vollständig gelöst wurde und es manuelle Schritte brauchen wird, um den finalen Datensatz von offensichtlich falschen Bäumen zu bereinigen. Die manuelle Arbeit konnte jedoch bereits ein Stückweit minimiert werden. Nicht auszuschliessen ist, dass damit auch korrekt detektierte Bäume eliminiert wurden, da sie entweder zu klein waren oder eine unförmige Kronenform besassen.

4.1.3 Laufzeitoptimierung und Multiprocessing

Zahlreiche Anpassungen sorgten dafür, dass der Algorithmus speicher- und prozesseffizient wurde. Beispielsweise wurden alle Klassen mit *slots* versehen, damit Python zur Kompilierungszeit wusste, wie viele Attribute eine Klasse haben wird. Damit können im besten Fall bis zu 50% des Arbeitsspeichers eingespart werden (Khalid 2017). Nicht mehr verwendete Datensätze wurden konsequent mit Löschanweisungen entfernt, damit die automatische Speicherverwaltung von Python *(garbage collector)* den Platz im Arbeitsspeicher schnellstmöglich wieder freigeben konnte.

Zur Effizienzsteigerung des Algorithmus wurden alle Schleifenanweisungen dahingehend geprüft, ob sie mit *numpy* Funktionen (Oliphant 2006) ersetzt werden konnten. Als Beispiel dient folgendes Codefragment, welches ein zusammenhängendes Gebiet in der binären Bildmaske für hohe Vegetation aufgrund der Fläche und der Höhe über Meer in Gebiete für Wald und offenes Gelände aufteilt. Die Bedingung ist im Codefragment zur besseren Lesbarkeit mit der booleschen Variable *cond_forest* abstrahiert.

```
field = skimage.morphology.label(high_vegetation_mask)
forest = numpy.copy(field)
for region in skimage.measure.regionprops(field):
   if region.cond_forest:
      field[field == region.label] = 0
   else:
      forest[forest == region.label] = 0
```

Wenn ein Waldgebiet vorliegt, wird dieses in der Maske für offenes Gelände schwarz gefärbt (auf 0 gesetzt) und vice versa. Die Lösung mit *for*-Schleife dauert pro Kachel rund 20 Sekunden, da die Regionen mit einzelnen Anweisungen zugewiesen werden.

Das folgende optimierte Codefragment ergibt funktional das gleiche Resultat, hat jedoch eine Laufzeit von weniger als eine Sekunde.

```
field = skimage.morphology.label(high_vegetation_mask)
forest = numpy.copy(field)
region_props = skimage.measure.regionprops(field)
forest_regions = [r.label for r in region_props if r.cond_forest = True]
forest[numpy.isin(forest, forest_regions, invert=True)] = 0
field[numpy.isin(field, forest_regions)] = 0
```

In der Schleife werden nur noch die Labels für Waldgebiete ausgelesen, um danach die Gebiete mit je einer einzigen Anweisung zur Maske für Wald oder offenes Gelände zuteilen zu können. Das Bespiel zeigt, wie mit kleinen Optimierungen viel bewirkt werden kann. Da das Code-

fragment rund 20 Mal schneller wurde, konnte bei 1'760 Kacheln mit einer absoluten Laufzeitreduktion von fast 10 Stunden gerechnet werden.

Die Einzelbaumdetektion konnte ohne grosse Vorkehrungen in mehrere Prozesse ausgelagert werden, da der Prozess dank den Kacheln räumlich zerlegt werden konnte und nur lesend auf die Daten zugegriffen werden musste. Python bietet mit der internen *multiprocessing* Bibliothek von Haus aus ein mächtiges Werkzeug, um dies zu bewerkstelligen. Folgende Codezeilen veranschaulichen, wie ein Pool von parallelen Prozessen angelegt wurde.

```
p = multiprocessing.Pool(settings.get("processing.number_of_processes"))
logging_settings = itertools.repeat(settings.get("logging"))
self.tiles = p.starmap(self._run, zip(self.tiles, logging_settings))
```

Mittels *starmap* wird dem Pool mitgeteilt, welche Funktion *(self._run)* ausgeführt werden muss. Der zweite Parameter enthält die zu verarbeitenden Kacheln *(self.tiles)* und Logging-Einstellungen. Die Protokollierung *(logging)* der Arbeitsschritte ist ein wichtiger Bestandteil zur Nachvollziehbarkeit des Algorithmus. Das Codefragment zeigt, dass jedem Prozess ein Logging-Setup mitgegeben werden muss, damit jener den internen Logger selbständig initialisieren kann. Der interne Logger wird in Windows Umgebungen nicht über mehrere Prozesse hinweg geteilt (Sajip 2020).

Verteilte oder parallel arbeitende Software braucht besondere Aufmerksamkeit, wenn ein gemeinsamer Speicher schreibend geteilt wird. Wenn zwei Prozesse gleichzeitig denselben Datensatz manipulieren und in Konflikt geraten, spricht man von *race conditions*. Ein anderes Phänomen sind sich gegenseitig blockierende Prozesse *(locks)*, wenn beide auf die Freigabe einer Ressource warten (DeNero 2014). Um dies zu vermeiden, wurde dafür gesorgt, dass jeder Prozess in eine eigene Datei protokolliert. Protokollierung von mehreren Prozessen in eine einzelne Datei wäre grundsätzlich möglich, hätte jedoch einen weiteren losgelösten Prozess inklusive Warteschlange *(queue)* benötigt, was wiederum die Laufzeit verschlechtert hätte.

Laufzeithochrechnung

Um den finalen Algorithmus auf dessen Laufzeit und Qualität zu prüfen, wurden zufällig 17 Kacheln ausgewählt und vollständig prozessiert. Dabei liess sich ermitteln, wo die Grenzen der parallelen Prozessierung lagen.

Anzahl Prozesse	17 Kacheln	Ø pro Einzelkachel	Hochrechnung für 1'760 Kacheln
1 (seriell)	26 min 08 s	92 s	45 Std. 00 min
2	17 min 12 s	61 s	29 Std. 49 min
3	14 min 54 s	53 s	25 Std. 54 min
4	14 min 26 s	51 s	24 Std. 56 min
5	12 min 09 s	43 s	21 Std. 01 min
6	12 min 29 s	44 s	21 Std. 30 min

Anzahl Prozesse	17 Kacheln	Ø pro Einzelkachel	Hochrechnung für 1'760 Kacheln
7	15 min 02 s	53 s	25 Std. 54 min
8	13 min 30 s	48 s	23 Std. 28 min

Tab. 8: Laufzeiten je Anzahl Prozesse

In der Tab. 8 ist zu erkennen, dass die Laufzeit ab 5 parallelen Prozessen plafonierte und sich nicht mehr verbesserte. Im Gegenteil wurde die Laufzeit bei noch mehr parallelen Prozessen teilweise wieder schlechter. Der Grund dafür war, dass aufgrund der Datenmenge pro Kachel nicht nur die Anzahl parallel rechnender Prozessoren, sondern auch die Zugriffsgeschwindigkeit auf die Festplatte eine entscheidende Rolle spielte. Pro verarbeitete Kachel wurden, inklusive aller Nachbarkacheln zur Verhinderung von Randartefakten, bis zu 10 Gigabyte Daten von der Festplatte in den Arbeitsspeicher geladen. Dies konnte aufgrund der Bauform der im Einsatz stehenden Festplatte nur in serieller Art und Weise geschehen. Das bedeutet, dass sich alle parallel arbeitenden Prozesse den lesenden Zugriff auf die Daten teilten. Im ersten Schritt des Algorithmus, wenn alle Daten geladen wurden, war demnach die Lesegeschwindigkeit der Festplatte der limitierende Faktor. Um diese Limitation ein Stückweit aufzuheben, wurden die LiDAR Daten per Zufallsprinzip von der internen und von einer extern angeschlossenen Festplatte geladen. Damit konnte auch der Zugriff auf die Daten parallelisiert werden. Beim zweiten Schritt kamen die Prozessoren an ihre Grenzen und entschieden über den maximalen Datendurchsatz pro Zeit. Die Laufzeit war mit steigender Anzahl parallelen Prozessen zunehmend dem Zufall unterworfen, ob und wie viele Prozesse sich im gleichen Schritt des Algorithmus befanden (laden oder prozessieren). Optimal war, wenn etwa gleich viele Prozesse Daten luden wie prozessierten und somit die Festplatten und die Prozessoren immer gleichzeitig und maximal ausgelastet waren.

Hardware

Die Extraktion der Einzelbäume auf dem gesamten Kantonsgebiet soll auf einem durchschnittlichen Bürocomputer ausgeführt werden können, so dass das Verfahren bei neuen Datengrundlagen schnell und unkompliziert wiederholt werden kann. Zur Einordung des späteren Resultates ist es wichtig, die Leistungsmerkmale der eingesetzten Hardware zu kennen.

Parameter	Wert
Hersteller	Acer
Modell	Aspire V17 Nitro
Prozessor	Intel® Core™ i7-6700HQ mit 2.60GHz CPU (8 Kerne)
Arbeitsspeicher	32 GB
Betriebssystem	Microsoft® Windows® 10 Home
Systemarchitektur	64-Bit
Interne Festplatte	Seagate 2 TB HDD SATA III 2,5 Zoll mit 5'400 U/min
Externe Festplatte	Intenso 6 TB 3,5 Zoll mit 5'400 U/min, angeschlossen an USB 3.0

Tab. 9: Hardwareparameter

Die Parameter des eingesetzten Notebooks gemäss Tab. 9 entsprechen nicht unbedingt gängiger Bürohardware und die spätere Bewertung der Laufzeit muss diesen Umstand berücksichtigen. Minimalvoraussetzung ist ein grosser interner Speicher, da die LiDAR Punktwolke 1.28 Terrabyte Speicherplatz benötigt. Das digitale Oberflächen- und Höhenmodell benötigen nochmals eine Kapazität von 210 Gigabyte. Wenn ein Verfahren Zwischenresultate in Form von Punktwolken ablegen möchte, wäre nochmals zusätzlicher Speicherplatz notwendig.

4.1.4 Workflow

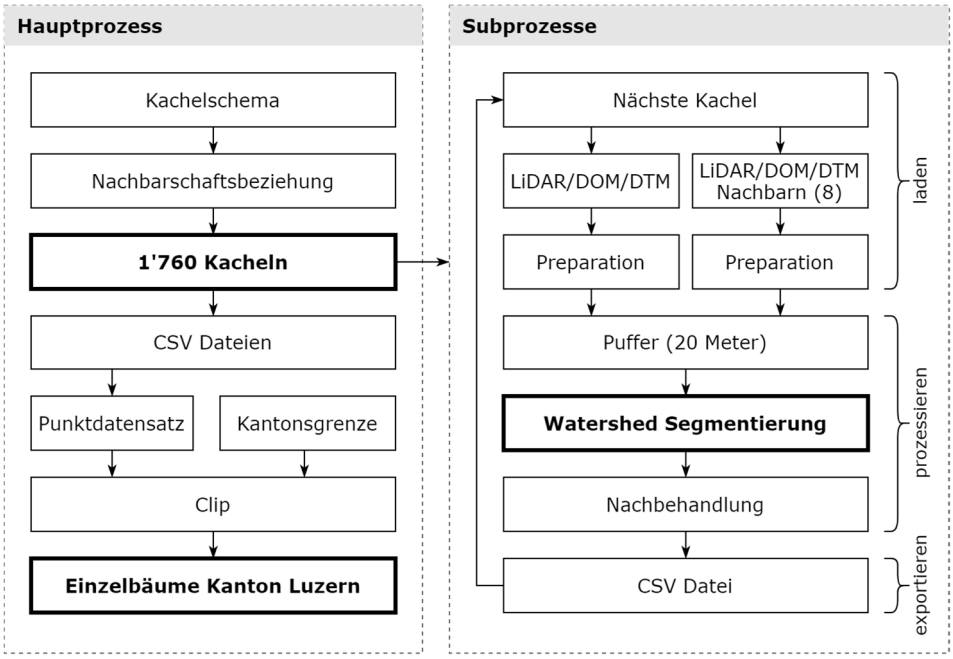


Abb. 27: Workflow zur Einzelbaumdetektion

Der in Python (Python Software Foundation 2019) entwickelte Workflow gemäss Abb. 27 ist in einen Hauptprozess und mehrere Subprozesse aufgeteilt. Der Hauptprozess kümmert sich um die Vorbereitung der zu prozessierenden Kacheln, indem er zu jeder Kachel die acht Nachbarschaftskacheln ableitet. Er lädt zuvor alle Parameter aus der Konfiguration und steuert die Subprozesse an. In den parallel arbeitenden Subprozessen werden im ersten Schritt die notwendigen Daten geladen und präpariert. Dazu gehört die Adaption der Ausdehnung aller Kacheln, da insbesondere in Randgebieten nicht alle Kacheln gleich gross sind. Nebst der Zentrumskachel müssen alle acht Nachbarkacheln geladen und präpariert werden, um einen 20 Meter breiten Puffer dazuzurechnen. Der Puffer wird dabei ressourceneffizient mittels *numpy* Arithmetik berechnet (Oliphant 2006). Dies wird sowohl für die LiDAR Punktwolke als auch für das digitale Oberflächen- und Terrainmodell gemacht. Im zweiten Schritt werden die Daten prozessiert und die Einzelbäume extrahiert, wobei der im vorherigen Kapitel evaluierte *region-based marker-controlled watershed* Algorithmus zum Einsatz kommt (siehe Kapitel 3.3.2). Im Nachbehandlungsschritt werden Bäume im Pufferbereich zu den Nachbarkacheln eliminiert. Zudem werden Bäume mit unrealistischen Standorten oder Formen gelöscht (siehe Kapitel 4.1.2). Um Konflikte durch die parallele Prozessierung zu vermeiden, schreiben

alle Prozesse im dritten Schritt ihre Resultate in eine eigene kommaseparierte Textdatei *(CSV)*. Erst wenn alle Kacheln abgearbeitet sind, greift der Hauptprozess die Textdateien auf und exportiert einen Punktdatensatz in eine *ESRI File Geodatabase*. Als letzter Nachbearbeitungsschritt werden die Bäume mit der Kantonsgrenze verschnitten, damit im Endresultat nur noch Bäume auf dem Kantonsgebiet verbleiben.

4.2 Ergebnisse und Diskussion

4.2.1 Finaler Punktdatensatz

Der final produzierte Punktdatensatz enthält rund **11.5 Millionen Einzelbäume**. Da das Problem mit den falsch klassierten LiDAR Punkten nicht vollständig eliminiert werden konnte, ist davon auszugehen, dass die Anzahl zu hoch ist. In Anbetracht der Erkenntnis aus der Methodenevaluierung (siehe Kapitel 3.4), dass der Algorithmus nur einzelnstehende Bäume und markante Bäume in Wäldern erkennen kann, ist der Wert hingegen wieder zu tief. Da die tatsächliche Anzahl der Bäume unbekannt ist, bleibt es offen, ob sich die beiden Effekte ausgleichen und wie akkurat der erzielte Wert tatsächlich ist. Fest steht, dass der Datensatz vor einer produktiven Verwendung manuell bereinigt werden muss. Dazu kann das aktuelle Orthofoto mit dem Einzelbaumdatensatz überlagert werden, damit offensichtlich falsch positive Bäume gelöscht werden können. Zusätzliche Bäume müssen keine erfasst werden.

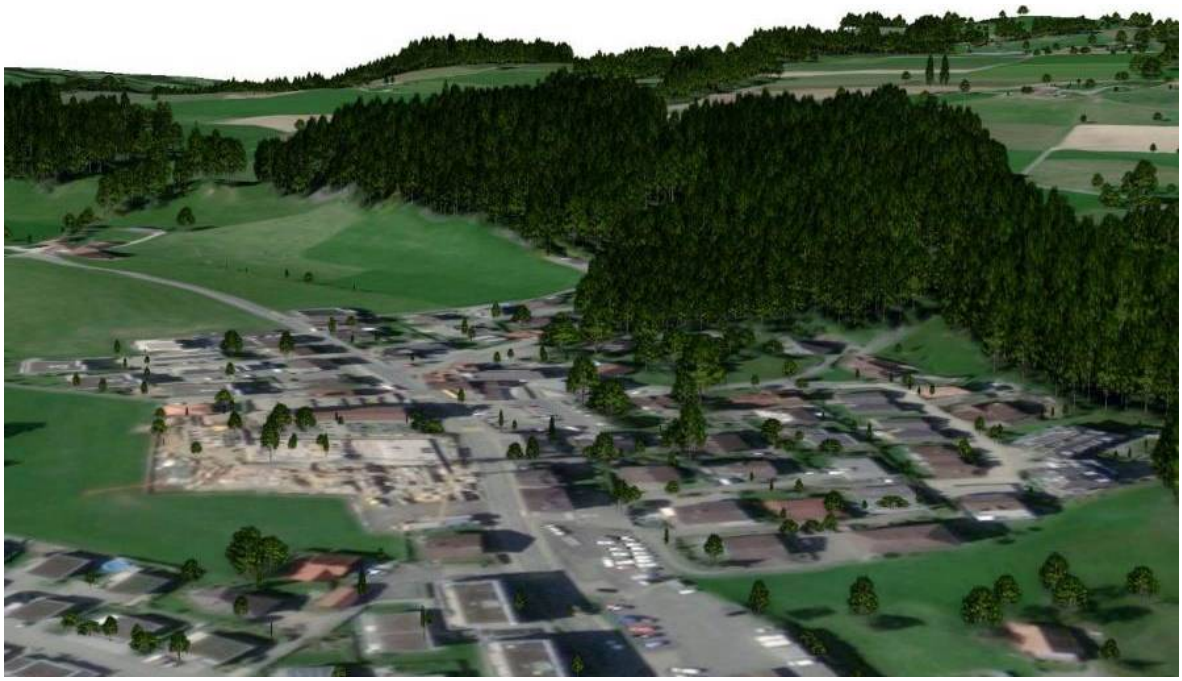


Abb. 28: Fotorealistische Darstellung der Einzelbäume in ArcGIS Pro

Die Abb. 28 zeigt ein Anwendungsbeispiel für den finalen Datensatz. Er wurde für diesen Zweck in ArcGIS Pro geladen und mittels fotorealistischer Bäume symbolisiert. Dabei werden die Bäume nicht nur lagegenau, sondern auch aufgrund ihrer relativen Höhe und ihres

Kronendurchmessers korrekt dargestellt. Es ist denkbar, dass bei digitalen Ausschreibungsverfahren von geplanten Bauprojekten der Datensatz zum Einsatz kommen könnte, um realistische Szenerien darzustellen. Aufgrund der ermittelten Qualität ist es bei einem solchen Szenario notwendig, den Datensatz manuell zu überarbeiten.

Die Attribute des Datensatzes wurden gemeinsam mit Fachpersonen bestimmt und das abgeleitete Datenmodell hält sich an die Richtlinien für Geodatensätze im Kanton Luzern. Es handelt sich um einen technischen Datensatz mit möglichst vielen Rohdaten, welcher für einzelne Anwendungen vereinfacht werden soll.

Feldname	Typ	Beschreibung
ObjectID	UUID	ESRI spezifische eindeutige Objektnummer.
Shape	Geometry	Geometrie des Punktes in drei Dimensionen (North, East, Z) im Bezugsrahmen CH1903+ / LV95.
baumnummer	Integer	Eindeutige zufällige Nummer des Baumes.
flaeche	Double	Fläche in Quadratmetern, welche das Gebiet der auf die Ebene projizierte Baumkrone einnimmt.
rel_baumhoehe	Double	Relative Baumhöhe in Metern bestehend aus der Differenz von *dom_max* und DTM an der Stelle von *dom_max*.
major_axis_length	Double	Kronendurchmesser des Baumes an der breitesten Stelle in Metern. Besteht aus der Länge der Hauptachse einer hypothetischen Ellipse, welche das Gebiet der auf die Ebene projizierte Baumkrone einnimmt.
minor_axis_length	Double	Kronendurchmesser des Baumes an der schmalsten Stelle in Metern. Besteht aus der Länge der Nebenachse einer hypothetischen Ellipse, welche das Gebiet der auf die Ebene projizierte Baumkrone einnimmt.
dtm_min	Double	Minimum des digitalen Terrainmodells in Metern über Meer auf der Fläche des Baumes.
dtm_mean	Double	Mittelwert des digitalen Terrainmodells in Metern über Meer auf der Fläche des Baumes.
dtm_median	Double	Median des digitalen Terrainmodells in Metern über Meer auf der Fläche des Baumes.
dtm_max	Double	Maximum des digitalen Terrainmodells in Metern über Meer auf der Fläche des Baumes.
dom_min	Double	Minimum des digitalen Oberflächenmodells in Metern über Meer auf der Fläche des Baumes.
dom_mean	Double	Mittelwert des digitalen Oberflächenmodells in Metern über Meer auf der Fläche des Baumes.
dom_median	Double	Median des digitalen Oberflächenmodells in Metern über Meer auf der Fläche des Baumes.

Feldname	Typ	Beschreibung
dom_max	Double	Maximum des digitalen Oberflächenmodells in Metern über Meer auf der Fläche des Baumes.

Tab. 10: Datenmodell des Punktdatensatzes für Einzelbäume

Die Tab. 10 beschreibt das Datenmodell des erzeugten Punktdatensatzes. Die Lage eines Einzelbaumes ergab sich aus dem Schwerpunkt der auf die Ebene projizierten Baumkrone. Damit wurde für den finalen Datensatz die Lagegenauigkeit als wichtiger eingestuft als die Höhengenauigkeit. Die Punkte sind als dreidimensionaler Datensatz und damit inklusive Höhenangabe abgespeichert. Die Höhe des Punktes ergab sich aus dem Maximalwert des digitalen Oberflächenmodells im Gebiet der Baumfläche und entspricht damit näherungsweise der Position der Baumkrone. Die relative Höhe eines Baumes wurde mit der Differenz zwischen digitalem Oberflächen- und Terrainmodell an der Stelle des höchsten Punktes im Oberflächenmodell berechnet. Andere Höhenberechnungen aus den Rohdaten wären jederzeit möglich.

4.2.2 Statistische Auswertung der Einzelbäume

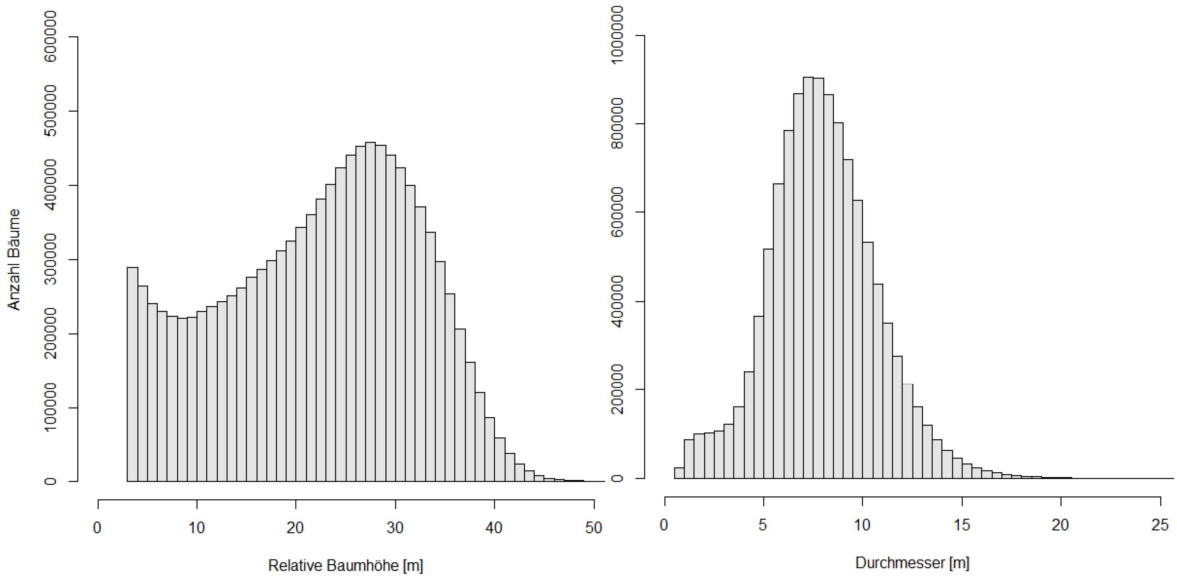


Abb. 29: Verteilung der relativen Baumhöhen und Kronendurchmesser

Die Abb. 29 zeigt die Verteilung der Bäume je Höhe und Durchmesser. Bei der relativen Baumhöhe ist gut zu erkennen, dass es eine markante Abweichung zur Normalverteilung mit einem Anstieg bei kleineren Höhen zwischen 3 und 7 Meter gibt. Resultate kleiner als 3 Meter wurden als falsch positiv bewertet und eliminiert. Es bleibt Spekulation, ob dieser Anstieg ein Hinweis auf die Existenz von weiteren falsch positiven Bäumen ist oder ob diese Verteilung der Tatsache entspricht. Insgesamt bleibt die Verteilung mit einer *Skewness* von -0.22 linksschief mit einem Maximum bei 28 Meter.
Die Verteilung der Kronendurchmesser ist mit einer *Skewness* von 0.35 rechtsschief. Die *Kurtosis* von 4.14 zeigt, dass die Verteilung steiler gewölbt ist als die Verteilung der Baumhöhe

(*Kurtosis* von 2.14). Statistisch gesehen sind Bäume mit einer relativen Höhe von 28 Meter und einem Kronendurchmesser von rund 8 Meter am häufigsten anzutreffen.

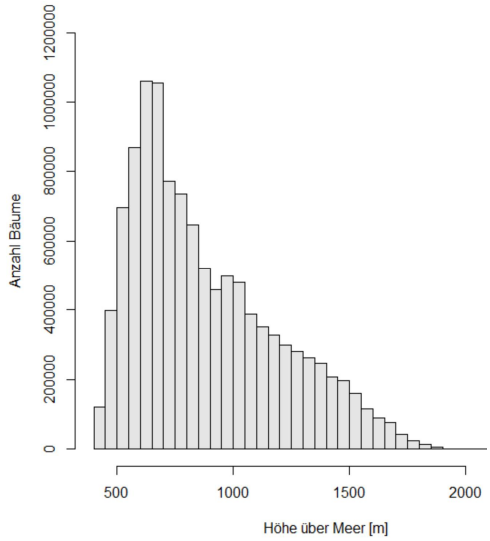


Abb. 30: Verteilung der Baumstandorte je Höhenlage in Metern über Meer

Eine weitere interessante Verteilung zeigt die Abb. 30 mit den Baumstandorten je Höhenlage in Metern über Meer. Erwartungsgemäss nimmt die Anzahl mit zunehmender Höhe ab, was in einer rechtsschiefen Verteilung resultiert (*Skewness* von 0.79). Am meisten Bäume befinden sich auf rund 800 bis 900 Meter über Meer. Das Resultat wird vermutlich mehr von der Topografie des Kantonsgebietes als von der Verteilung der Bäume an und für sich bestimmt. Beim Übergang von 900 auf 1'000 Meter über Meer ist eine leichte Zunahme zu erkennen. Dies spiegelt die unterschiedliche Behandlung von Gebieten unter oder über 1'000 Meter über Meer wider. Für Gebiete über 1'000 Meter hat der Algorithmus einen kleineren minimalen Baumabstand angelernt, was in einer minimalen Zunahme der Anzahl Bäume beim Übergang zu Höhenlagen über 1'000 Meter über Meer resultiert.

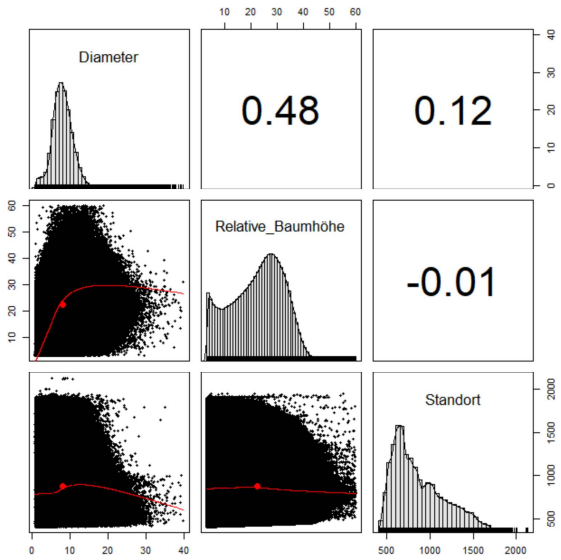


Abb. 31: Korrelation von Durchmesser, relative Baumhöhe und Standort

Wenn man die Korrelation von Durchmesser, relative Höhe und Standort überprüft, zeigt sich das Bild gemäss Abb. 31. Berechnet wurde der Korrelationskoeffizient nach *Spearman*, dessen Wert pro Parameterkombination in den Kästchen oben rechts ersichtlich ist. Die einzige jedoch moderat positive Korrelation besteht zwischen der Baumhöhe und des Kronendurchmessers. Bei einem Wert von 0.48 kann man die statistische Aussage machen, dass höhere Bäume eher grössere Kronendurchmesser haben. Es ist zu vermuten, dass diese beiden Variablen kausal voneinander abhängig sind. Wegen zunehmender relativer Höhe eines Baumes nimmt dessen Kronendurchmesser zu.
Andere Korrelationen sind nicht erkennbar. So gibt es keinen Zusammenhang von Durchmesser oder der relativen Baumhöhe zum Wuchsstandort. Bäume in Höhenlagen sind statistisch gesehen nicht kleiner oder grösser als Bäume in tiefer gelegenen Gebieten. Aufgrund von nicht erhobenen Parametern bleibt es offen, ob das Resultat beispielsweise für unterschiedliche Baumarten anders aussehen würde.

4.2.3 Erzielte Laufzeit

Der final erzeugte Datensatz wurde mit **6 parallelen Prozessen** hergestellt und endete nach rund **37 Stunden Rechenzeit**. Pro Quadratkilometer benötigte der Algorithmus demnach durchschnittlich 1.5 Minuten. Erzeugt wurde ein Punktdatensatz in einer *ESRI File Geodatabase* mit einer Grösse von 1.27 Gigabyte. Es ist davon auszugehen, dass bei gängiger Bürohardware oder nur einer Festplatte die Laufzeit zwischen 40 und 50 Stunden betragen würde.

Die Abweichung zur hochgerechneten Laufzeit ist erklärbar durch Kacheln mit grosser Datenmenge und entsprechend vielen Bäumen. Es gibt beispielsweise eine Kachel mit rund 27'000 Bäumen pro Quadratkilometer, da sie flächendeckend aus geschlossenem Wald besteht. Die Berechnung solcher Kacheln dauerte teilweise bis zu 3 Minuten. Ausserdem war nicht zuverlässig sicherzustellen, dass bei der gesamten Rechenzeit die Hardwareressourcen exklusiv zur Verfügung standen. Mit jedem Hintergrundtask wie beispielsweise der Aktualisierung des Betriebssystems oder eines anderen Programms wurde die Laufzeit negativ beeinflusst.

Die Parallelisierung des Datenleseschrittes mit zwei Festplatten hat sich bewährt. Es konnte beobachtet werden, dass die Geschwindigkeit immer dann hoch war, wenn die Zugriffe auf die interne und externe Festplatte in etwa gleichverteilt waren. Die Laufzeit könnte verbessert werden, wenn sich die parallelen Prozesse untereinander abstimmen würden. Eine steuernde Instanz müsste koordinieren, dass immer gleichviele Prozesse Daten laden wie prozessieren. Weitere Verbesserungen der Laufzeit gäbe es, wenn man es nicht dem Zufall überliesse, ob Daten von der internen oder externen Festplatte geladen werden. Auch hier müsste gesteuert werden, dass immer alle Festplatten gleichzeitig belastet wären. Der Algorithmus würde noch schneller, wenn man auf schnellere Solid-State-Festplatten (SSD) zurückgriffe. Der Einfluss auf die Laufzeit durch die Verwendung einer anderen Programmiersprache wurde nicht untersucht und kann deshalb nicht abgeschätzt werden. Es ist denkbar, dass hardwarenähere Sprachen wie beispielsweise C/C++ einen positiven Einfluss auf die Laufzeit haben könnten.

Aus dem Gebiet der Einzelbaumsegmentierung aus LiDAR Daten oder anderen Fernerkundungsdaten konnten keine Forschungsarbeiten gefunden werden, aus denen man aufgrund der Grösse des Studiengebiets, des angewendeten Verfahrens, der verwendeten Hardware und der resultierenden Laufzeit vergleichbare Orientierungswerte herleiten könnte. Während das Studiengebiet und dessen Grösse praktisch immer dokumentiert wird, fehlen absolute Angaben zur Prozessierungsart und -zeit durchgehend. Aus diesem Grund lässt sich die hier erreichte Laufzeit nicht einordnen und muss als Momentaufnahme für sich alleine stehen.

5 Schlussfolgerung und Ausblick

Die vorliegende Arbeit hat die Fülle an Verfahren und die Historie zur Einzelbaumextraktion aus Fernerkundungsdaten beleuchtet. Es konnte exemplarisch gezeigt werden, dass komplexe Verfahren zur Einzelbaumextraktion aus LiDAR Daten nicht geeignet sind, auf ein grosses Gebiet angewendet zu werden. Verfahren, welche direkt in der LiDAR Punktwolke operieren, würden zwar bessere Resultate liefern, sind aber derart rechenintensiv, dass sie für Gebiete grösser als einen Quadratkilometer nicht auf gängiger Bürohardware ausgeführt werden können. Ein guter Kompromiss zwischen Qualität und Laufzeit liefern Verfahren, welche das Problem auf zwei Dimensionen reduzieren und die Einzelbäume mittels Bildsegmentierung detektieren. Dabei hat die Parametrisierung des Algorithmus einen grösseren Einfluss auf die Qualität des Resultats als das grundlegende Segmentierungsverfahren. Durch die Vereinfachung des Problems nimmt man in Kauf, dass nicht alle, sondern nur markante und insbesondere alleinstehende Bäume erkannt werden. Diese Resultate decken sich mit den Erkenntnissen anderer Forschungsarbeiten (Zaforemska et al. 2019).

Der entwickelte *region-based marker-controlled wathershed* Algorithmus konnte dank den Testgebieten und der evolutionären Weiterentwicklung optimal trainiert und auf die Fläche des Kantons Luzern von 1'493.51 Quadratkilometer angewendet werden. Der erstellte Einzelbaumdatensatz enthielt rund 11.5 Millionen Bäume und wurde mittels parallelen Prozessen nach rund 37 Stunden berechnet. Es hat sich gezeigt, dass die Testgebiete bezüglich Variabilität der Baumgrössen und -arten gut gewählt waren. Der Algorithmus hat einen Kompromiss zwischen Über- und Untersegmentierung der Einzelbäume gefunden. Zusätzlich hat er selbständig gelernt, dass Bäume in geschlossenen Wäldern und ab 1'000 Meter über Meer tendenziell näher beisammenstehen und deshalb mit anderen Parametern prozessiert werden müssen. Die Testgebiete haben jedoch nicht offengelegt, wie gravierend das Problem von falsch klassierten LiDAR Punkten war. Mittels Nachbearbeitung und empirisch festgelegten Parametern mussten sehr viele falsch positive Resultate herausgefiltert werden. Es wäre denkbar, diese Parameter in einer erneuten Iteration mit noch mehr Testgebieten auch mittels genetischen Algorithmus anzulernen. Eventuell könnte das Endresultat verbessert werden, so dass kein manueller Aufwand zur Bereinigung mehr anfiele. Noch besser wäre, wenn die LiDAR Punktwolke vor der Einzelbaumdetektion korrekt klassiert wird und damit das Problem an der Quelle gelöst würde. Andere Verfahren, welche auch auf die korrekte Klassierung angewiesen wären, würden ebenso von dieser Korrektur profitieren.

Das Resultat der Einzelbäume könnte noch weiter optimiert werden, wenn man zusätzliche Daten hinzuzöge. Die Forschung bei der Erkennung von Nadel- und Laubbäumen oder gar Baumarten mittels multispektraler Luftbilder zeigt vielversprechende Ansätze (Waser 2012). Mit dieser Information könnte der Algorithmus weiter verbessert werden, da die Regionen mit unterschiedlicher Parametrisierung pro Baumart noch feiner definiert und zusätzliche Attribute aufgenommen werden könnten. Im Grunde genommen ist die Unterscheidung der Seg-

mentierungsregionen in geschlossenen Wald und offenes Feld eine Kompromisslösung, ein Zwischenschritt hin zur unterschiedlichen Prozessierung je Baumart. Es wäre zu erforschen, wie sich die Hinzunahme weiterer Datenquellen auf die Laufzeit auswirken würde und ob der Algorithmus immer noch mit gängiger Bürohardware auf grossem Gebiet lauffähig bliebe.

Die erzielte Laufzeit liesse sich mit ausgeklügeltem Management der Prozesse und besserer Lastverteilung noch verbessern. Generell schnellere Komponenten wie beispielsweise Solid-State Festplatten (SSD) würden sich auch positiv auf die Laufzeit auswirken. Zudem ist denkbar, dass mit hardwarenäheren Programmiersprachen wie beispielsweise C/C++ die LiDAR Daten schneller prozessiert werden könnten. Die dokumentierte Laufzeit stellt eine Momentaufnahme in der sich schnell entwickelnden digitalen Welt dar und die erzielte Leistung wird mit anderer Hardware kaum je exakt gleich reproduziert werden können.

6 Literaturverzeichnis

Abegg, M., *et al.*, 2014. *Viertes Schweizerisches Landesforstinventar - Ergebnistabellen und Karten im Internet zum LFI 2009-2013 (LFI4b)* [online]. Birmensdorf: Eidg. Forschungsanstalt WSL. Verfügbar unter: http://www.lfi.ch/resultate/ [Zuletzt geprüft am 06.01.2019].

Bachmann, P. G. H., 1894. *Die analytische Zahlentheorie. Dargestellt von Paul Bachmann.* Leipzig B.G. Teubner.

Barnes, C., *et al.* 2017. Individual Tree Crown Delineation from Airborne Laser Scanning for Diseased Larch Forest Stands. *Remote Sensing,* 9(3), 231.

Beucher, S. und Lantuéjoul, C., 1979. *Use of Watersheds in Contour Detection.* Rennes, France: Centre de Géostatistique et de Morphologie Mathématique.

Bischoping, H., 2007. *Massenprozessierung von Laserscan-Daten, Erhebung des Streuobstbaumbestandes in Baden Württemberg* Master thesis (Master of Science (Geographical Information Science & Systems)). Paris Lodron-Universität Salzburg.

Bodenhofer, U., 1999. *Genetic Algorithms: Theory and Applications.* Software Competence Center Hagenberg.

Brändli, U.-B. und Denzler, L., 2011. *Ergebnisse aus dem dritten Landesforstinventar 2004-06. Posterserie in 15 Teilen* [online]. Birmensdorf: Eidg. Forschungsanstalt WSL. Verfügbar unter: http://www.lfi.ch/publikationen/publ/posterLFI3.php.

Bundesamt für Landestopografie, 2017. *swissTLM 3D Version 1.5* Schweizerische Eidgenossenschaft, Bundesamt für Landestopografie swisstopo.

Bundesamt für Statistik, 2016. Arealstatistik 2013/18. [online], (31.01.2020). Verfügbar unter: https://www.bfs.admin.ch/bfs/de/home/statistiken/raum-umwelt/erhebungen/area.html.

Chee Chun Gan, G. L., 2016. An improved chromosome formulation for genetic algorithms applied to variable selection with the inclusion of interaction terms. [online]. Verfügbar unter: https://arxiv.org/abs/1604.06727 [Zuletzt geprüft am 30.10.2019].

Chen, Q., *et al.* 2006. Isolating individual trees in a savanna woodland using small footprint lidar data. *Photogrammetric Engineering and Remote Sensing,* 72(8), 923-932.

Cohen, J. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement,* 20(1), 37-46.

Congalton, R. und Green, K., 2019. *Assessing the Accuracy of Remotely Sensed Data: Principles and Practices, Third Edition.* Boca Raton, Florida, USA: CRC Press, Taylor & Francies Group.

Dai, W. X.*, et al.* 2018. A new method for 3D individual tree extraction using multispectral airborne LiDAR point clouds. *Isprs Journal of Photogrammetry and Remote Sensing,* 144, 400-411.

Dalponte, M. und Coomes, D. A. 2016. Tree-centric mapping of forest carbon density from airborne laser scanning and hyperspectral data. *Methods in Ecology and Evolution,* 7(10), 1236-1245.

Dalponte, M., Frizzera, L. und Gianelle, D. 2019. Individual tree crown delineation and tree species classification with hyperspectral and LiDAR data. *Peerj,* 6.

DeNero, J., 2014. *Parallel Computing* [online]. Composing Programs. Verfügbar unter: https://composingprograms.com/pages/48-parallel-computing.html [Zuletzt geprüft am 31.01.2020].

Ding, Y. und Densham, P. J. 1996. Spatial strategies for parallel spatial modelling. *International Journal of Geographical Information Systems,* 10(6), 669-698.

Duncan, J. M. A.*, et al.* 2019. Turning down the heat: An enhanced understanding of the relationship between urban vegetation and surface temperature at the city scale. *Science of the Total Environment,* 656, 118-128.

Duncanson, L. I.*, et al.* 2014. An efficient, multi-layered crown delineation algorithm for mapping individual tree structure across multiple ecosystems. *Remote Sensing of Environment,* 154, 378-386.

Ellenberg, H. und Leuschner, C., 2010. *Vegetation Mitteleuropas mit den Alpen: in ökologischer, dynamischer und historischer Sicht.* UTB GmbH.

Ferraz, A.*, et al.* 2016. Lidar detection of individual tree size in tropical forests. *Remote Sensing of Environment,* 183, 318-333.

Forsyth, R. 1981. Beagle — A Darwinian approach to pattern recognition. *Kybernetes,* 10(3), 159-166.

Franceschi, S.*, et al.* 2018. Identifying treetops from aerial laser scanning data with particle swarming optimization. *European Journal of Remote Sensing,* 51(1), 945-964.

Gad, A., 2018a. Genetic Algorithm Implementation in Python. [online]. Verfügbar unter: https://towardsdatascience.com/genetic-algorithm-implementation-in-python-5ab67bb124a6 [Zuletzt geprüft am 11.01.2020].

Gad, A., 2018b. Introduction to Optimization with Genetic Algorithm. [online], 2019. Verfügbar unter: https://www.linkedin.com/pulse/introduction-optimization-genetic-algorithm-ahmed-gad [Zuletzt geprüft am 24.11.2019].

Gautier, L., 2016. rpy2: An interface to R running embedded in a Python process. [online]. Verfügbar unter: https://rpy2.readthedocs.io/ [Zuletzt geprüft am 17.01.2020].

Ghosh, S., 2019. Multiprocessing vs. Threading in Python: What Every Data Scientist Needs to Know. [online]. Verfügbar unter: https://blog.floydhub.com/multiprocessing-vs-threading-in-python-what-every-data-scientist-needs-to-know/ [Zuletzt geprüft am 29.01.2020].

Giannico, V.*, et al.* 2016. Estimating Stand Volume and Above-Ground Biomass of Urban Forests Using LiDAR. *Remote Sensing,* 8(4), 339.

Ginzler, C., Brändli, U.-B. und Hägeli, M. 2011. Waldflächenentwicklung der letzten 120 Jahre in der Schweiz. *Schweizerische Zeitschrift für Forstwesen,* 162(9), 337-343.

Holland, J. H., 1975. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence.* MIT Press.

Hyyppä, J. 1999. Detecting and estimating attributes for single trees using laser scanner. *Photogramm J Finland,* 16, 27-42.

Isenburg, M., 2015. Use Buffers when Processing LiDAR in Tiles !!! [online], 2019. Verfügbar unter: https://rapidlasso.com/2015/08/07/use-buffers-when-processing-lidar-in-tiles/ [Zuletzt geprüft am 30.10.2019].

Jaafar, W.*, et al.* 2018. Improving Individual Tree Crown Delineation and Attributes Estimation of Tropical Forests Using Airborne LiDAR Data. *Forests,* 9(12).

Jakubowski, M. K.*, et al.* 2013. Delineating Individual Trees from Lidar Data: A Comparison of Vector- and Raster-based Segmentation Approaches. *Remote Sensing,* 5(9), 4163-4186.

Khalid, M. Y. U., 2017. *Python Tips* [online]. pythontips.com. Verfügbar unter: http://book.pythontips.com/ [Zuletzt geprüft am 31.01.2020].

Kiener, F., 2018. *Luzern (Kanton)* [online]. Historisches Lexikon der Schweiz HLS. Verfügbar unter: https://hls-dhs-dss.ch/de/articles/007382/2018-02-07/ [Zuletzt geprüft am 31.01.2020].

Kornilov, A. S. und Safonov, I. V. 2018. An Overview of Watershed Algorithm Implementations in Open Source Libraries. *Journal of Imaging,* 4(10).

lawa, 2019. *Landschaftsqualitätsbeiträge Luzern.* Kanton Luzern: Bau-, Umwelt- und Wirtschaftsdepartement, Landwirtschaft und Wald (lawa).

Li, W. K., *et al.* 2012. A New Method for Segmenting Individual Trees from the Lidar Point Cloud. *Photogrammetric Engineering and Remote Sensing,* 78(1), 75-84.

Mohd Zaki, N., *et al.*, 2015. Individual Tree Crown (ITC) Delineation Using Watershed Transformation Algorithm For Tropical Lowland Dipterocarp. *International Conference on Space Science and Communication (IconSpace).* Langkawi, Malaysia.

Monumental Trees, 2020. *Kollaborative Webseite zur Erfassung von monumentalen Bäumen* [online]. Verfügbar unter: https://www.monumentaltrees.com/ [Zuletzt geprüft am 29.01.2020].

NOAA, 2012. Lidar 101: An Introduction to Lidar Technology, Data, and Applications. [online]. Verfügbar unter: https://coast.noaa.gov/data/digitalcoast/pdf/lidar-101.pdf [Zuletzt geprüft am 30.11.2019].

Oliphant, T., 2006. Guide to NumPy. [online]. Verfügbar unter: http://www.numpy.org/.

Pluess, A. R., Augustin, S. und Brang, P., 2016. *Wald im Klimawandel. Grundlagen für Adaptationsstrategien.* Bern; Birmensdorf; Bern: Bundesamt für Umwelt BAFU; Eidg. Forschungsanstalt WSL; Haupt.

Pollock, J. R., 1996. *The automatic recognition of individual trees in aerial images of forests based on a synthetic tree crown image model.* The University of British Columbia, Vancouver, Canada.

Python Software Foundation, 2019. Python 3.6. [online]. Verfügbar unter: https://www.python.org/ [Zuletzt geprüft am 18.01.2020].

R Core Team, 2013. R: A Language and Environment for Statistical Computing. [online]. Verfügbar unter: http://www.R-project.org/ [Zuletzt geprüft am 17.01.2020].

rawi Kanton Luzern, 2020. *Geoportal Kanton Luzern* [online]. Bau-, Umwelt- und Wirtschaftsdepartement, Raum und Wirtschaft, Geoinformation, Kanton Luzern. Verfügbar unter: https://geoportal.lu.ch/ [Zuletzt geprüft am 31.01.2020].

Reitberger, J., *et al.* 2009. 3D segmentation of single trees exploiting full waveform LIDAR data. *Isprs Journal of Photogrammetry and Remote Sensing,* 64(6), 561-574.

Roussel, J.-R., *et al.*, 2020. lidR: Airborne LiDAR Data Manipulation and Visualization for Forestry Applications. [online]. Verfügbar unter: https://CRAN.R-project.org/package=lidR [Zuletzt geprüft am 17.01.2020].

Sajip, V., 2020. Python HOWTO, Logging Cookbook. [online]. Verfügbar unter: https://docs.python.org/3/howto/logging-cookbook.html [Zuletzt geprüft am 15.02.2020].

Schardt, M., *et al.* 2002. Assessment of Forest Parameters by Means of Laser Scanning. *International Archives of Photogrammetry and Remote Sensing,* 34.

Sikchi, H., 2017. Convex Hulls: Explained, Convex Hull Computation. [online]. Verfügbar unter: https://medium.com/@harshitsikchi/convex-hulls-explained-baab662c4e94 [Zuletzt geprüft am 30.11.2019].

Silva, C. A.*, et al.* 2016. Imputation of Individual Longleaf Pine (Pinus palustris Mill.) Tree Attributes from Field and LiDAR Data. *Canadian Journal of Remote Sensing,* 42(5), 554-573.

Tschubby, 2013. Reliefkarte des Kantons Luzern, Eigenes Werk, CC BY-SA 3.0. https://commons.wikimedia.org/w/index.php?curid=27315427.

Turing, A. M. 1950. I. — Computing Machinery And Intelligence. *Mind,* LIX(236), 433-460.

UTAS AG, Baggenstos, M. und Häfliger, P., 2014. *Kommentar Waldbau - Arbeitsbuch für die waldbauliche Praxis.* Giswil und Kanton Luzern: Bau-, Umwelt- und Wirtschaftsdepartement, Landwirtschaft und Wald (lawa).

van der Walt, S.*, et al.* 2014. scikit-image: image processing in Python. *Peerj,* 2, e453.

Virtanen, P.*, et al.*, 2019. SciPy 1.0 - Fundamental Algorithms for Scientific Computing in Python. [online]. Verfügbar unter: https://www.scipy.org/ [Zuletzt geprüft am 17.01.2020].

Wang, J. H., Lindenbergh, R. und Menenti, M. 2018. Scalable individual tree delineation in 3D point clouds. *Photogrammetric Record,* 33(163), 315-340.

Wang, L., Gong, P. und Biging, G. S. 2004. Individual Tree-Crown Delineation and Treetop Detection in High-Spatial-Resolution Aerial Imagery. *Photogrammetric Engineering & Remote Sensing,* 70(3), 351-357.

Waser, L. T., 2012. *Airborne remote sensing data for semi-automated extraction of tree area and classification of tree species.* Doctoral Thesis. ETH Zürich.

Wu, B.*, et al.* 2016. Individual tree crown delineation using localized contour tree method and airborne LiDAR data in coniferous forests. *International Journal of Applied Earth Observation and Geoinformation,* 52, 82-94.

Xiao, W.*, et al.* 2016. Individual Tree Crown Modeling and Change Detection From Airborne Lidar Data. *Ieee Journal of Selected Topics in Applied Earth Observations and Remote Sensing,* 9(8), 3467-3477.

Zaforemska, A., Xiao, W. und Gaulton, R. 2019. Individual tree detection from UAV LiDAR data in a mixed species woodland. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* XLII-2/W13, 657-663.

7 Anhang

7.1 Python Sourcecode

Im Rahmen dieser Arbeit sind zwei Python Code Projekte entstanden, welche unter folgenden GitHub Repositories öffentlich eingesehen werden können.

URL	Beschreibung
https://github.com/adrian-kuhn/incubator/	Python command line application (CLI) containing a genetic algorithm to find best parameters for arbitrary assignments.
https://github.com/adrian-kuhn/tree-detection/	Python command line application (CLI) to delineate single trees from LiDAR data, DOM and DTM.

Tab. 11: Öffentliche Code Repositories auf GitHub