Erstellung eines kantonalen Baumkatasters auf Basis von grossflächigen Geodaten und LiDAR – Technischer Bericht

Natalia Kolecka *

Bronwyn Price

Christian Ginzler

12.12.2025

Eidgenössische Forschungsanstalt WSL

Forschungseinheit Landschaftsdynamik

Gruppe Fernerkundung

Zürcherstrasse 111

8903 Birmensdorf

*Kontakt: natalia.kolecka@wsl.ch

Inhaltsverzeichnis

1. Ausgangslage und Ziel

Mit dem Klimawandel wird die Hitzebelastung in Siedlungsgebieten weiter zunehmen. Insbesondere grosskronige Bäume reduzieren die Wärmebelastung. Daneben bilden sie wichtige ökologische Vernetzungskorridore.

Mit der Erstellung eines Baumkatasters wird eine wichtige Grundlage für die Anpassung an den Klimawandel geschaffen: Der aktuelle Baumbestand wird kartiert, Veränderungen über die Zeit werden sichtbar und wertvolle Bäume können identifiziert werden.

Leider sind Einzelbaumdaten oft nur in grösseren Städten, auf öffentlichem Grund und nicht aktuell vorhanden. Der freie Zugang zu Geodaten und die wiederholte Erfassung von Fernerkundungsdaten mit sehr hoher Auflösung (räumlich wie zeitlich) eröffnet aber heute neue Möglichkeiten, die notwendigen Information flächendeckend zu erfassen. Der Aufwand, die Zugänglichkeit zu Daten, das Potential und die Grenzen der Ergebnisse sollen im Projekt zeitnah untersucht werden. Im Speziellen werden für die Bauzonen des Kanton Zürichs die Informationen zu Einzelbäumen aus dem Topographischen Landschaftsmodell (TLM) von swisstopo mit den hochaufgelösten Punktwolken LiDAR verknüpft. Daneben werden auch die kantonalen LiDAR Daten verwendet. Die entwickelten Methoden werden für unterschiedliche Inputdaten getestet. Die gemeinsame Verwendung von swisstopo Daten und kantonalen Daten wird eine sehr hohe Aktualität der Baumcharakteristika erlauben. Als Resultat wird ein Baumkataster für das Siedlungsgebiet des Kantons Zürich erstellt und die Grundlage gelegt für einen schweizweiten Baumkataster.

Ziele:

Entwicklung eines umfassenden Baumkatasters für die städtischen Gebiete des Kantons Zürich durch Verknüpfung von LiDAR-Daten mit Geodaten aus dem topografischen Landschaftsmodell (TLM) von swisstopo.

Entwicklung eines halbautomatisierten Workflows zur Verarbeitung gross angelegter Daten, der regelmässige Aktualisierungen auf Basis frei verfügbarer Geodaten und Werkzeuge (Software) ermöglicht.

2. Verwendete Daten

2.1 Kantonale Airborne Laser Scanning (ALS) Punktwolken, 2022

Quelle: Kanton Zürich, Amt für Geoinformation

Verwendung: Ableitung der maximalen Vegetationshöhe zur Baumdetektion, Trennung von Laub- und Nadelbäumen, Ableitung der Voxel-basiertes Vegetationsvolumina und Berechnung des Vegetationsvolumens pro Baumkrone

Referenzen:

- Aktuelle Höhendaten: https://www.zh.ch/de/news-uebersicht/mitteilungen/2023/planen-bauen/geoinformation/aktuelle-hoehendaten-fuer-den-kanton-zuerich.html
- LiDAR-Daten: https://maps.zh.ch/download/hoehen/2022/lidar/

2.2 Kantonale Waldmaske

Quelle: Kanton Zürich

Verwendung: Ausschluss bewaldeter Flächen aus der Analyse.

Referenz: Geoportal Kanton Zürich, Waldareal: https://geo.zh.ch/data/datasets/09475d43-41ed-f925-bd51-5963897ab6fb

2.3 TLM Einzelbaum

Quelle: swisstopo, swissTLM3D – Bodenbedeckung, TLM_EINZELBAUM_GEBUESCH

Verwendung: Validierung und Verknüpfung der detektierten Baumkronen mit Einzelbaumstandorten.

Referenz: swissTLM3D: www.swisstopo.admin.ch/de/landschaftsmodell-swisstlm3d

2.4 Baumkataster der Grün Stadt Zürich (BK ZH)

Quelle: Grün Stadt Zürich, Baumkataster (Geodatenportal Stadt Zürich)

Verwendung: Validierung der automatisiert detektierten Baumkronen im Siedlungsgebiet der Stadt Zürich; Abgleich mit offiziell erfassten, durch die Stadt bewirtschafteten Bäumen.

Hinweise: Hohe Lagegenauigkeit bei tachymetrisch vermessenen Strassenbäumen; geringere Genauigkeit bei Bäumen in Grünflächen (typisch 0.5–2 m). Datensatz enthält alle städtischen Bäume im Strassenraum, welche von Grün Stadt Zürich verwaltet oder gepflegt werden. Ergänzt wird das Baumkataster durch das Obstbauminventar sowie Bäume ausgewählter öffentlicher Grünanlagen und private Bäume.

Referenz: Geodaten Stadt Zürich – Baumkataster: https://www.stadt-zuerich.ch/geodaten/download/Baumkataster

3. Workflow

Der Arbeitsablauf bestand aus drei Hauptphasen: (1) Vorverarbeitung der ALS-Daten und Vorbereitung der Datensätze, (2) Entwicklung und Implementierung der Segmentierungs- und Klassifizierungsmethode und (3) gezielte manuelle Verfeinerung ausgewählter Ergebnisse. Die Methode wurde zunächst in Trimble eCognition 10.5 (https://geospatial.trimble.com/en/products/software/trimble-ecognition) entwickelt und dann in R übersetzt, gefolgt von einer selektiven Nachbearbeitung in ArcGIS Pro 3.4.2 (https://www.esri.com/en-us/arcgis/products/arcgis-pro/overview).

Annahmen:

- Im normalisierten Raum wurde ein Schwellenwert für die Baumhöhe von 3 m angewendet, um die Analyse auf baumrelevante Vegetation zu beschränken (Mindestbaumhöhe = 3 m).
- Ein Schwellenwert für die Baumkronengrösse von 6 m² wurde für die Erkennung von Bäumen angewendet, die nicht im TLM-Datensatz kartiert sind, um den Einfluss fehlerhafter Eingangsdaten zu begrenzen (Mindestkronenfläche = 6 m²).
- Es sollten nur Bäume ausserhalb von Wäldern kartiert werden – bewaldete Flächen wurden mithilfe der kantonalen Waldmaske ausgeschlossen.

3.1 Vorverarbeitung der Eingangsdaten

Für die Baumkartierung wurden die höhen-normalisierten ALS-Punktwolken verwendet. Es wurden nur Punkte verwendet, die als Vegetation klassifiziert wurden (ASPRS Klassen 3, 4, 5). Ausserdem wurden Punkte unterhalb von 3 m über dem Boden ausgeschlossen, um niedrige Vegetation und Sträucher zu entfernen. Die Verarbeitung erfolgte mit LAStools (https://lastools.github.io/).

3.1.1 Ableitung des Rasters der maximalen Vegetationshöhe

Zur Erstellung des Rasters der maximalen Vegetationshöhe (1 m Auflösung) wurden die normalisierten und gefilterten ALS-Punktwolken unter Verwendung des höchsten Punktes innerhalb jeder Rasterzelle auf eine räumliche Auflösung von 1 m berechnet.

Code:

```
set INPUT=c:\LAZ

set OUTPUT=c:\MAX_VH

lasgrid64 -i %INPUT%*.laz -odir %OUTPUT% -step 1 -highest -keep_class 3 4 5 -otif
-cores 12
```

3.1.2 Ableitung des voxelbasierten Vegetationsvolumenrasters (VOL_ABV3M)

Um ein voxelbasiertes Vegetationsvolumen abzuleiten, wurde die normalisierte und gefilterte ALS-Punktwolke in ein regelmässiges 1 × 1 × 1 m Voxelraster unterteilt. Anschliessend wurde für jede 1 m-Rasterzelle die Gesamtzahl der vertikal belegten Voxel gezählt und als Pixelwert gespeichert.

Code:

```
set INPUT=c:\LAZ

set OUTPUT=c:\VOXEL

lasvoxel64 -v -i %INPUT%*.laz -keep_class 3 4 5 -odir %OUTPUT% -step 1 -olaz

set INPUT=c:\VOXEL
```

```
set OUTPUT=c:\VOXEL_TIFF

lasgrid64  -i %INPUT%*.laz -counter  -step 1   -o %OUTPUT%\VOXEL_100_3m.tif  - -
keep_z_above 2.99 -merged
```

3.2 Methodenentwicklung in eCognition

Für die Entwicklung der Methode zur Abgrenzung von Baumkronen wurden das normalisierte Raster der maximalen Vegetationshöhe, die kantonale Waldmaske und die TLM-Einzelbaum-Punkte in Trimble eCognition 10.5 verarbeitet. Das Ziel dieser Phase war es, das Segmentierungsverhalten zu testen und geeignete Parameterwerte zu definieren.

Bewaldete Gebiete wurden ausmaskiert. Das Raster der maximalen Vegetationshöhe wurde mit einem 3×3-Mittelwertfilter geglättet und anschliessend mit einem Wasserscheidenalgorithmus segmentiert (Abbildung 1), um vorläufige Kronenobjekte in nicht bewaldeten Gebieten abzuleiten. Pixel unterhalb einer Höhe von 3 m an den Segmentgrenzen wurden entfernt, um niedrige Vegetation zu unterdrücken. Jedes Segment wurde dann mit TLM-Einzelbaum-Punkten verschnitten, um die Anzahl der Baumstandorte pro Kroneneinheit zu bestimmen. Segmente, die mehrere Punkte enthielten, wurden anhand der TLM-Einzelbaum-Punkte mit Thiessen-Polygonen weiter unterteilt, und die daraus resultierenden sehr kleinen Artefakte (1 m²) wurden mit benachbarten Kronen ähnlicher Höhe zusammengeführt.

Segmente wurden als Bäume klassifiziert, wenn sie einen TLM-Einzelbaum-Punkt überlappten (es wurde davon ausgegangen, dass die in TLM markierten Bäume tatsächlich existieren und ihre Höhe oder Grösssse wurde nicht zusätzlich überprüft) oder, wenn kein solcher Punkt vorhanden war, wenn sie eine Mindestkronenfläche von 6 m² überschritten. Die resultierenden Baumobjekte wurden als Shapefile exportiert.

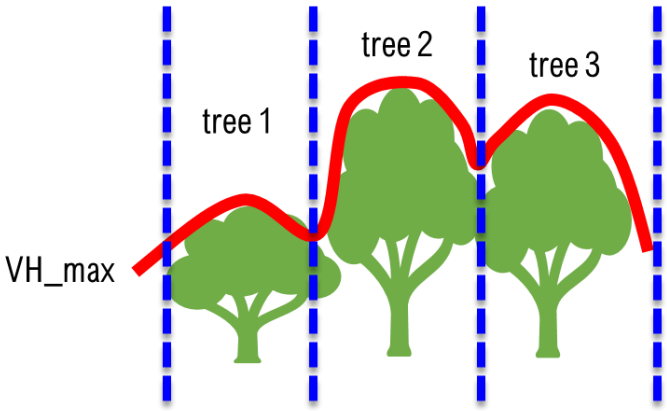


Abbildung 1. Prinzip der Wasserscheidensegmentierung. Die rote Linie stellt das Vegetationshöhenmodell dar, und die blauen Linien zeigen, wie die Bäume in den lokalen Minima voneinander getrennt sind.

3.3 Automatisierter Workflow in R für die Baumsegmentierung und -klassifizierung

Der eCognition-Regelsatz wurde in R übersetzt und an die verfügbaren Algorithmen angepasst. Die Kernschritte des Workflows wurden beibehalten, darunter die Glättung des

Vegetationshöhenmodells, die wassergebietsbasierte Kronensegmentierung, die Maskierung von Waldflächen, der Ausschluss von niedriger Vegetation (<3 m) und die Verknüpfung von Segmenten mit TLM-Einzelbaum-Punkten.

Im R-Workflow wurden mehrere Verbesserungen vorgenommen: Kleine Lücken in der Vegetationsmaske wurden automatisch mit morphologischen Filtern geschlossen, und innerhalb des Skripts wurden Thiessen-Polygone generiert, um Polygone mit mehreren Bäumen zu unterteilen. Diese Verbesserungen ermöglichten eine vollständig automatisierte, reproduzierbare Ableitung einzelner Baumkronen unter Beibehaltung der Logik und der Klassifizierungskriterien des ursprünglichen eCognition-Workflows.

Eine detaillierte Beschreibung des Codes findet sich unter „5. R-Verarbeitungsskript“

3.4 Klassifizierung von Laubbäumen

Laubbäume und Nadelbäume wurden anhand der in den kantonalen ALS-Daten von 2022 unterschieden. Der Ansatz folgt dem von Liang et al. (2007) beschriebenen Prinzip, bei dem der Abstand zwischen der ersten und den nachfolgenden Returns jedes Laserpulses als Indikator für die Durchlässigkeit des Baumkronendachs verwendet wird. Laubabwerfende (Laubbäume) weisen in der Regel grösssere Abstände zwischen dem ersten und dem folgenden Returns auf als immergrüne Bäume, deren dichtes Laub zu einer geringeren Eindringtiefe führt.

Die ALS-Punktwolken wurden zunächst nach GPS-Zeit sortiert, um die Integrität der Impulssequenz zu erhalten. Für jeden Impuls wurde die Rücklaufentfernung (Differenz zwischen der Höhe des ersten und des letzten Rücklaufs) berechnet und zu einem 1 × 1 m-Raster aggregiert, das die mittlere Rücklaufentfernung aller Impulse innerhalb jeder Zelle darstellt. Für die Verarbeitung wurde LAStools (https://lastools.github.io/) verwendet.

Baumkronenpolygone wurden auf dieses Raster gelegt und zonale Statistiken berechnet, um die mittlere Rücklaufentfernung pro Krone abzuleiten. Um Unterschiede in der Gesamthöhe der Bäume zu berücksichtigen, wurde die mittlere Rücklaufentfernung durch die maximale Kronenhöhe normalisiert. Bäume mit einem normalisierten Wert von mehr als 0,5 wurden als Laubbäume klassifiziert, während Werte von 0,5 oder weniger als Nadelbäume klassifiziert wurden

Code:

```
:: RASTER LaubNadel

set INPUT=c:\LAZ

set OUTPUT=c:\LAZ_SORT_GPS

lassort64 -i %INPUT%*.laz -odir %OUTPUT% -olaz -gps_time -cores 12


set INPUT=c:\LAZ_SORT_GPS

set OUTPUT=c:\LAZ_RETURN

:: LASRETURN64 has a bug. The old 32bit version was used

lasreturn -i %INPUT%*.laz -odir %OUTPUT% -olaz -compute_gap_to_next_return -
classify_missing_as 8 -classify_duplicate_as 12 -classify_violation_as 8 -cores 12


set INPUT=c:\LAZ_RETURN

set OUTPUT=c:\LAZ_LAUBNADEL
```

```
lasgrid64 -i %INPUT%*.laz -odir %OUTPUT% -otif -step 1 -avg -attribute 0 -keep_z_above 3 -
keep_class 3 4 5 -keep_number_of_returns 2 -drop_attribute_above 0 25 -keep_scan_angle -20
20 -cores 12
```

Referencen:

Liang, Xinlian & Hyyppä, Juha & Matikainen, Leena. (2007). Deciduous-coniferous tree classification using difference between first and last pulse laser signatures. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences. 36. 253-257

3.5 Ableitung der Baumattribute

Für jede abgegrenzte Baumkrone wurde eine Reihe von geometrischen und aus LiDAR-Daten abgeleiteten Attributen berechnet, um ihre Position, Grösse, Struktur und Beziehung zu externen Referenzdatensätzen zu beschreiben. Die Höhe jedes Baumes basiert auf dem in dieser Studie verwendeten Raster-VHM und kann von der in den TLM-EZB-Daten angegebenen Höhe abweichen. Tabelle 1 fasst alle Attribute, ihre Definitionen und die zu ihrer Berechnung verwendeten Werkzeuge zusammen.

Tabelle 1. Übersicht über die Baumkronenattribute, einschliesslich ihrer Definitionen und der zu ihrer Ableitung verwendeten Verarbeitungswerkzeuge

Attribut	**Beschreibung**	**Tool**
CENTER_X, CENTER_Y	Koordinaten des Baumkronenmittelpunkts (Markierungspunkt innerhalb des Polygons)	Calculate Geometry in ArcGIS
TLM_EZB_X, TLM_EZB_Y	Koordinaten des überlappenden TLM-Einzelbaum-Punkts; [0,0], wenn keine Überlappung vorliegt	Spatial Join in ArcGIS
CROWN_AREA	Fläche der Baumkrone (m²)	Calculate Geometry in ArcGIS
MAX_HGHT	Maximale Vegetationshöhe innerhalb der Krone (m)	Zonal Statistics as Table in ArcGIS
MEAN_HGHT	Mittlere Vegetationshöhe innerhalb der Krone (m)	Zonal Statistics as Table in ArcGIS
VOL_ABV3M	Vegetationsvolumen (Anzahl der 1×1×1 m-Voxel über 3 m)	Zonal Statistics as Table in ArcGIS
DCDS_PROB	Wahrscheinlichkeit (0-1), dass der Baum laubabwerfend ist (die mittlere Rücklaufentfernung, normiert durch die maximale Kronenhöhe)	LAStools / Zonal Statistics as Table in ArcGIS
DECIDUOUS	Binärer Indikator für den Baumtyp (1 = laubabwerfend, 0 = nadelbaumartig)	Reclassify von DCDS_PROB
IN_TLMEZB	Binärer Indikator (1 = überlappt TLM-Baumpunkt, 0 = keine Überlappung)	Spatial Join in ArcGIS

4. Bewertung

Um die Vollständigkeit und Konsistenz der automatisierten Baumerkennung zu bewerten, wurden zwei unabhängige Referenzdatensätze verwendet: der TLM-Einzelbaum-Datensatz, der offizielle Punktpositionen einzelner Bäume im Kanton Zürich enthält, und das Baumkataster der Grün Stadt Zürich (BK ZH), das öffentlich verwaltete Bäume innerhalb der Stadt Zürich dokumentiert. Alle erkannten Baumkronen wurden durch räumliche Überlagerungsanalyse mit Referenzpunkten verglichen. Eine Krone wurde als Übereinstimmung gezählt, wenn ihr Polygon einen Referenzbaum-Punkt schnitt. Für jeden Bewertungsbereich wurden zwei komplementäre Masse berechnet: (1) der

Anteil der erkannten Bäume, die sich mit Referenzbäumen überschneiden (Abdeckung des erkannten Datensatzes), und (2) der Anteil der Referenzbäume, die von einer erkannten Baumkrone geschnitten werden (Abdeckung der offiziellen Datensätze). Dieser bidirektionale Vergleich liefert eine ausgewogene Bewertung der Erkennungsleistung, einschliesslich der Übereinstimmung mit offiziellen Bestandsaufnahmen und dem Umfang der zusätzlichen Bäume, die durch den automatisierten Workflow erfasst wurden

5. Ergebnisse

5.1 Verteilung der Baumkronen

Im Kanton Zürich wurden ausserhalb der Waldgebiete 74,7 Millionen m² Baumkronenfläche kartiert (Tabelle 1). Bezogen auf die gesamte Kantonsfläche ohne Wälder entspricht dies einer Nichtwald-Baumkronenbedeckung von 6,1 %. In Siedlungen mit mehr als 100 Einwohnern steigt dieser Anteil auf 13,4 %, was die höhere Baumdichte widerspiegelt, die typischerweise in besiedelten Gebieten anzutreffen ist. Innerhalb der Stadt Zürich sind 15,0 % der Gemeindefläche ausserhalb von Waldgebieten von Baumkronen bedeckt.

Diese Werte liefern den quantitativen Kontext für das Kartodiagramm in Abbildung 2, das die räumliche Verteilung der nicht bewaldeten Baumkronenbedeckung in den Gemeinden veranschaulicht. Die angegebenen Anteile stellen die Baumkronenfläche ausserhalb von Wäldern im Verhältnis zur gesamten Gemeindefläche ohne Wald dar.

Zusätzlich zur Kronenfläche ermöglicht die extrahierte Baumschicht auch eine Bewertung der Zusammensetzung der Baumarten (Laub- vs. Nadelbäume). Wie in Tabelle 2 zusammengefasst, dominieren Laubbäume in allen räumlichen Massstäben und machen in der Regel 84–87 % der ausserhalb von Wäldern erfassten Bäume aus. Dieses Muster steht im Einklang mit dem überwiegend laubreichen Charakter der städtischen und stadtnahen Vegetation im Kanton Zürich.

Abbildung 3 zeigt drei detaillierte Beispiele (P1, P2, P3) mit den erfassten Kronenpolygonen und den wichtigsten Merkmalen, die typische Muster in dichten städtischen, ländlichen und Seeufergebieten veranschaulichen.

Tabelle 1: Baumkronenfläche ausserhalb von Waldgebieten und deren Anteil sowohl an der Nichtwaldfläche als auch an der Gesamtfläche (einschliesslich Wald) für den Kanton Zürich und ausgewählte städtische Regionen.

Region	Baumkronenfläche (ausserhalb von Waldgebieten) [ha]	Anteil der nicht bewaldeten Fläche, die von Baumkronen bedeckt ist [%]	Anteil der Gesamtfläche (einschliesslich Waldgebiete), die von Baumkronen bedeckt ist [%]
Kanton Zurich (ausserhalb von Waldgebieten)	7,469	6.1%	4.3%
Siedlungen mit >100 Einwohnern	4,392	13.4%	13.3%
Stadt Zurich	1,044	15.0%	11.4%

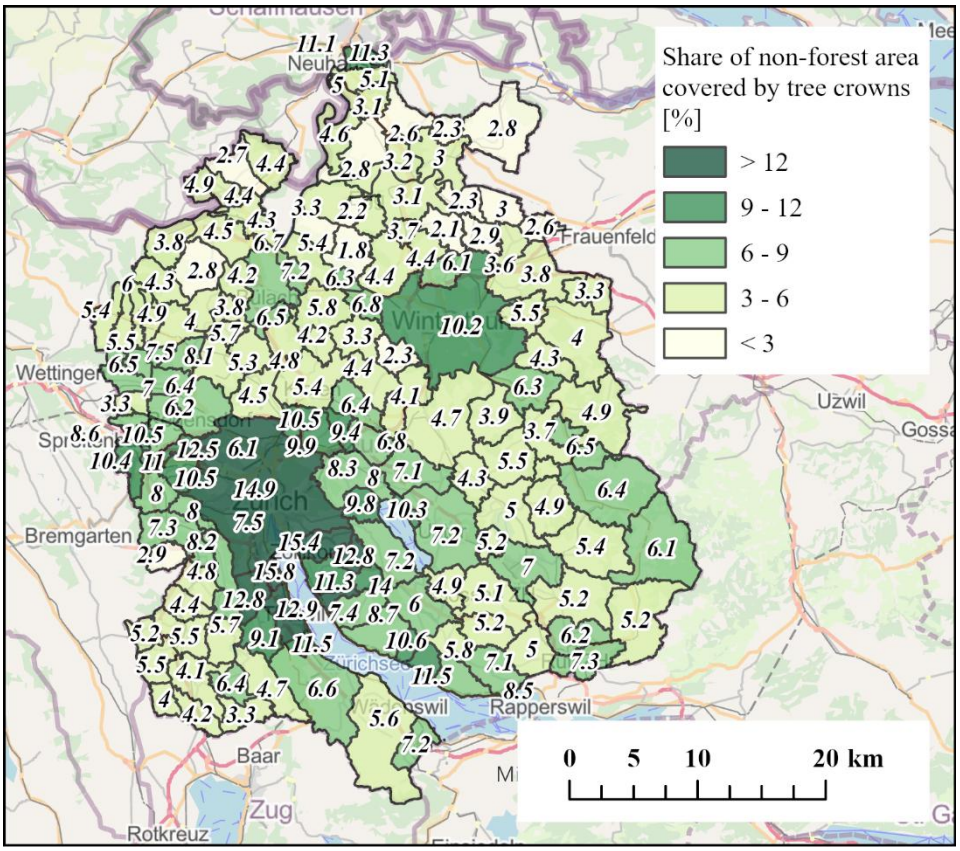


Abbildung 2: Anteil der Baumkronenfläche von Nicht-Waldbäumen in den Gemeinden. Die angegebenen Anteile beziehen sich auf die Baumkronenfläche ausserhalb von Wäldern im Verhältnis zur Gesamtfläche der Gemeinde ohne Wald [Grundkarte: OpenStreetMap contributors]

Tabelle 2: Anzahl und Anteil der erfassten Laub- und Nadelbäume (ausserhalb von Wäldern) für den Kanton Zürich und ausgewählte städtische Regionen.

Region	Alle Bäume	Laubbäume	Nadelbäume
Kanton Zurich	1,162,646	1,009,956 (86.8%)	152,960 (13.2%)
Siedlungen mit >100 Einwohnern	750,169	629,340 (83.9%)	120,829 (16.1%)
City of Zurich	157,504	133,705 (84.9%)	23,799 (15.1%)

Abbildung 3. Beispiele für erkannte Baumkronen und abgeleitete Attribute in drei Gebieten (P1 – Stadt, P2 – Seeufer, P3 – ländliches Gebiet) des Kantons Zürich. Jede Zeile bezieht sich auf das jeweilige Beispielgebiet und zeigt das Orthobild sowie die entsprechenden Karten der Kronenfläche, der maximalen Höhe und des Vegetationsvolumens über 3 m.

5.2 Bewertung der Genauigkeit und Vollständigkeit

5.2.1 Vergleich mit TLM Einzelbaum

Um die Vollständigkeit und Übereinstimmung der automatisierten Baumerkennung zu bewerten, wurden die resultierenden Baumkronen mit dem offiziellen TLM Einzelbaum-Datensatz verglichen (Tabelle 3). Für jeden Bewertungsbereich geben wir an, (1) wie viele erkannte Kronen sich mit TLM-Baumpunkten überschneiden und (2) wie viele TLM-Bäume durch die Methode korrekt erkannt werden. Dies liefert zwei sich ergänzende Perspektiven: die Abdeckung der erkannten Kronen durch TLM und die Abdeckung der TLM-Bäume durch die automatische Erkennung.

Tabelle 3. Vergleich zwischen automatisch erkannten Bäumen und TLM-Einzelbaum-Bäumen in verschiedenen Bewertungsbereichen

Fläche	Erkannte Bäume			TLM EZB Bäume		
	Alle	In TLM EZB	Nicht in TLM EZB	Alle	Erkannte	Nicht Erkannte
Kanton Zurich (ausserhalb von Waldgebieten)	1,162,646	645,762	516,884	769,380	645,683	123,697
Siedlungen mit >100 Einwohnern	750,169	451,080	299,089	542,865	451,046	91,819
Stadt Zurich	157,504	97,348	60,156	111,279	97,335	13,944

Wichtige Beobachtungen:

- Insgesamt wurden im Kanton Zürich ausserhalb von Waldgebieten 1,16 Millionen Bäume automatisch erkannt, was eine umfassende Erfassung der städtischen und stadtnahen Vegetation darstellt.
- Etwa 55 % der erkannten Baumkronen überschneiden sich mit TLM-Einzelbaum-Punkten, was auf eine erhebliche Übereinstimmung zwischen der automatisierten Methode und dem offiziellen Datensatz hinweist.
- Die restlichen 45 % sind Bäume, die in TLM nicht erfasst sind, was den Mehrwert des LiDAR-basierten Ansatzes für die Identifizierung nicht kartierter oder neu gepflanzter Bäume unterstreicht.
- Die Übereinstimmungsraten steigen mit der Siedlungsdichte: In der Stadt Zürich stimmen mehr als 87 % der TLM-Bäume mit den erkannten Baumkronen überein, was eine hohe Vollständigkeit und Positionskonsistenz in dichten städtischen Gebieten belegt.
- Diskrepanzen treten in verschiedenen Bereichen auf, darunter Strasssenbäume, Bäume am Rande der Siedlungen, neu gepflanzte oder gefällte Bäume sowie Baumgruppen, die nicht in TLM kartiert sind.
- Insgesamt zeigt der Vergleich eine hohe Übereinstimmung mit den offiziellen Daten, während gleichzeitig deutlich wird, dass die LiDAR-basierte Methode eine grosse Anzahl zusätzlicher Bäume erfasst, die für die Aktualisierung oder Ergänzung bestehender Baumkataster relevant sind.

5.2.2 Vergleich mit dem Baumkataster der Grün Stadt Zürich

Um die Vollständigkeit der automatisierten Baumerkennung weiter zu bewerten, wurde die resultierende Baumschicht mit dem Baumkataster der Grün Stadt Zürich (BK ZH) verglichen, in dem die auf öffentlichen Flächen (Strassen, Parks, kommunale Grundstücke) innerhalb der Stadt Zürich gepflegten Bäume dokumentiert sind. Während der BK ZH nur öffentliche Bäume enthält, erfasst die automatisch abgeleitete Ebene alle Bäume (öffentliche und private) innerhalb des Stadtgebiets. Somit liefert der Vergleich sowohl ein Mass für die Übereinstimmung auf öffentlichem Grund als auch einen Hinweis auf zusätzliche Bäume, die ausserhalb des offiziellen Bestands erfasst wurden.

In Tabelle 4 stellen die beiden Zeilen zwei unabhängige Bestände dar, die jeweils gegeneinander bewertet wurden. Dies ermöglicht einen direkten Vergleich der Vollständigkeit (wie viele BK ZH-Bäume wurden gefunden) und des Mehrwerts (wie viele zusätzliche Bäume wurden ausserhalb von BK ZH erfasst).

Tabelle 4. Räumliche Übereinstimmung zwischen den erfassten Bäumen und dem Baumkataster der Grün Stadt Zürich (BK ZH) innerhalb der Stadt Zürich. Der zweiseitige Vergleich zeigt übereinstimmende und nicht übereinstimmende Bäume in jedem Datensatz

Datensatz	Alle Bäume	In dem anderen Datensatz erfasst	Nicht erfasst
Erkannte Bäume	157,504	102,162	55,342
BK ZH	150,129	118,707	31,422

Wichtige Beobachtungen:

- In der Stadt Zürich (ausserhalb des Waldes) wurden durch den automatisierten Workflow 157.504 Bäume erfasst.
- Die BK ZH enthält 150.129 offiziell registrierte Bäume.
- Der räumliche Abgleich zeigt, dass ≈65 % der erfassten Bäume (102.162) mit Einträgen in der BK ZH übereinstimmen.
- Aus Sicht des offiziellen Datensatzes werden ≈79 % der BK ZH-Bäume (118.707) durch die automatische Erkennung erfasst.
- Erkannte Bäume ohne BK ZH-Übereinstimmung (55.343) befinden sich häufig auf privaten Grundstücken oder in öffentlichen Grünflächen, die nicht im BK ZH-Bestand enthalten sind. Darüber hinaus entstehen einige Diskrepanzen durch komplexe oder unregelmässige Kronen, die automatisch in mehrere Baum-Polygone segmentiert werden, was dazu führt, dass mehrere erkannte Kronen einem einzigen BK ZH-Baum entsprechen (Abbildung 4).
- BK ZH-Bäume ohne erkannte Entsprechung (31.422) umfassen in der Regel kleine, schmale oder verdeckte Bäume (z. B. durch Gebäude oder andere Vegetation beschattet) sowie Fälle von ALS-Okklusion. Weitere nicht zugeordnete Fälle treten auf, wenn mehrere BK ZH-Bäume in sehr kurzen Abständen (sogar < 1 m) erfasst werden, was wahrscheinlich Unterholz oder mehrschichtige Bepflanzungen darstellt, die mit ALS-basierten Methoden nicht zuverlässig getrennt werden können (Abbildung 5).

Abbildung 4. Beispiele für erkannte Baumkronen ohne BK ZH-Übereinstimmung: auf privaten Grundstücken (links) und komplexe oder unregelmässige Kronen (rechts). Die Baumkronen ohne entsprechenden BK ZH-Punkt sind blau hervorgehoben. Die roten Kreuze symbolisieren BK ZH-Bäume.

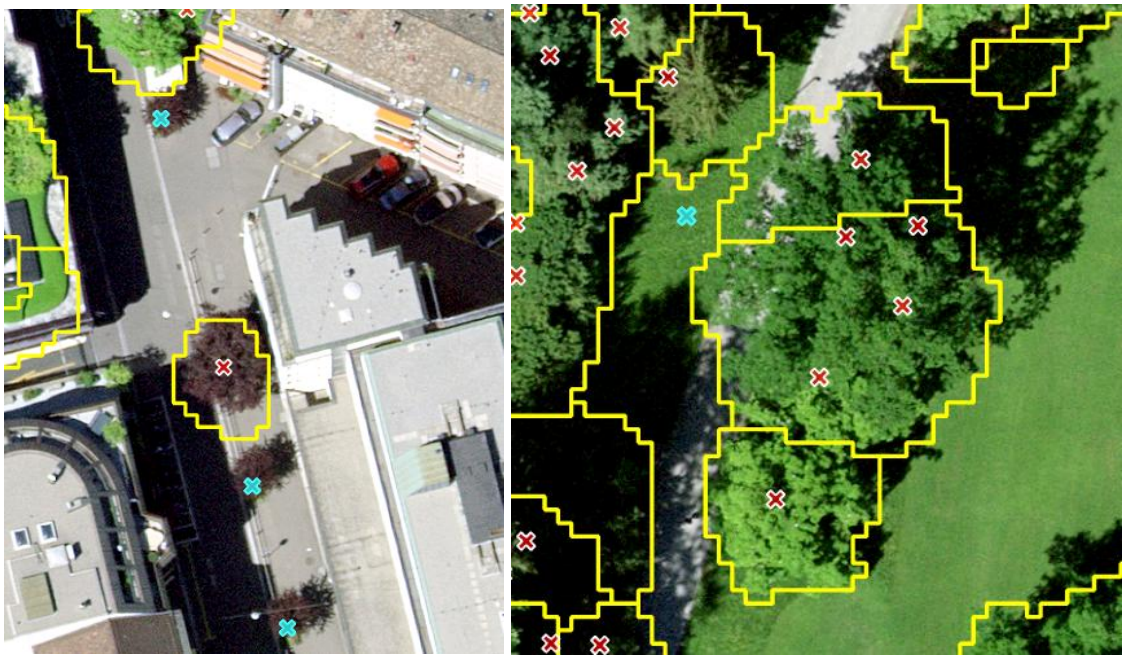


Abbildung 5. Beispiele für BK-ZH-Bäume ohne erkennbare Entsprechung: kleine, schmale Kronen und verdeckte Bäume (links) sowie mehrere BK-ZH-Bäume, die in sehr kurzen Abständen erfasst wurden und mit ALS-basierten Methoden nicht zuverlässig voneinander getrennt werden können (rechts). Die BK-ZH-Punkte ohne entsprechende Baumkrone sind blau hervorgehoben. Die roten Kreuze symbolisieren BK-ZH-Bäume.

5.3 Gelieferte Daten

1) **Geodatensatz – Baumschicht für den gesamten Kanton (ausserhalb von Waldgebieten)**
   - Vektorpolygonschicht, die die erfassten Baumkronen in allen Nichtwaldgebieten des Kantons Zürich darstellt.
   - CRS: EPSG:2056 (CH1903+/LV95).
   - Qualitätshinweise: Geometrien validiert; sehr kleine Polygone (<4 m²) entfernt; Waldgebiete ausgeblendet.
2) **Attributtabelle (jedem Polygon beigefügt)**
   - Felder (Bezeichnungen und Kurzbeschreibung):
     - ID – eindeutige Polygon-ID (Ganzzahl)
     - CENTER_X, CENTER_Y – Koordinaten des Kronenzentrums (Beschriftungspunkt)
     - TLM_EZB_X, TLM_EZB_Y – Koordinaten des überlappenden TLM-Einzelbaum-Punkts (oder 0,0, wenn keiner vorhanden)
     - IN_TLMEZB – Binärflag (1 = überlappt TLM-Einzelbaum, 0 = keine Überlappung)
     - CROWN_AREA – Kronenfläche in m²
     - MAX_HGHT – maximale Vegetationshöhe in m (aus VHM)
     - MEAN_HGHT – mittlere Vegetationshöhe in m (aus VHM)
     - VOL_ABV3M – voxelbasiertes Vegetationsvolumen (Anzahl der vollständigen 1×1×1 m-Voxel über 3 m)
     - DCDS_PROB – Wahrscheinlichkeit (0-1), dass der Baum laubabwerfend ist (die mittlere Rücklaufentfernung, normiert durch die maximale Kronenhöhe)
     - DECIDUOUS – binärer Indikator für den Baumtyp (1 = Laubbaum, 0 = Nadelbaum)
3) **R-Verarbeitungsskript**
   - ein vollständig automatisierter Workflow zur Erstellung einer Baumschicht durch Verarbeitung von LiDAR- und TLM-Daten
   - Datei: *BK_ZH_v1_20250917.R*

6. R-Verarbeitungsskript

6.1 Struktur des R-Verarbeitungsworkflows

1) Initialisierung
   a) Löscht den gesamten R-Arbeitsbereich, um mit einer sauberen Umgebung zu starten (`rm(list = ls())`).
   b) Lädt die erforderlichen R-Bibliotheken:
      i) tidyverse für Datenmanipulation und Pipelines
      ii) terra für Rasterverarbeitung
      iii) sf für Vektordaten (Polygone und Punkte)
      iv) lidR für LiDAR-basierte Analyse und Wasserscheiden-Segmentierung
      v) mapview für interaktive Kartenvisualisierung
      vi) EBImage für Rasterbildverarbeitung (z. B. Füllen kleiner Löcher)
      vii) FNN für nearest-neighbor Berechnungen
   c) Legt Arbeits- und Ausgabeverzeichnisse fest; erstellt den Ausgabeordner, falls er nicht existiert
   d) Definiert Dateipfade und Parameter für:
      i) Vegetationshöhenmodell (VHM) Raster, abgeleitet aus LiDAR-Daten
      ii) Waldmasken-Shapefile (auszuschließende Flächen)
      iii) TLM Einzelbaum-Punktdaten (EZB)
      iv) Weitere Parameter: Mindestflächen, Höhenschwellen, Größe von Löchern im Raster, Subset-Optionen.
2) Daten laden und vorverarbeiten
   a) Lädt das VHM-Raster und überprüft die Projektion (EPSG:2056).
   b) Lädt die Waldmaske, transformiert sie in das Schweizer Koordinatensystem und filtert nur Waldflächen heraus.
   c) Lädt TLM Einzelbaumdaten, weist eindeutige IDs zu, stellt gültige Geometrie und CRS sicher.
   d) Schneidet Daten optional auf einen kleinen Testbereich zu (wenn `process_full_raster = FALSE`), um die Verarbeitung zu beschleunigen.
   e) Maskiert Waldflächen aus beiden Datensätzen:
      i) VHM-Raster: behält nur Nicht-Wald-Pixel
      ii) TLM-Baumpunkte: behält nur Punkte außerhalb von Waldflächen
3) Baumsegmentierung – Segmentierung des Vegetationshöhenmodells in einzelne Baumkronen
   a) Wasserscheidensegmentierung
      i) Glättet das Vegetationshöhenraster mit einem fokalen Mittelwertfilter (3×3 Pixel) zur Reduktion kleiner Höhenabweichungen.
      ii) Wendet den lidR-Wasserscheidenalgorithmus an, um potenzielle Baumkronen zu segmentieren.
      iii) Konvertiert Rastersegmente in Vektorpolygone (sf-Objekte) und entfernt Polygone unterhalb der Mindestfläche.
   b) Anwendung des Vegetationshöhen-Schwellenwerts
      i) Erstellt eine Binärmaske für Vegetation ≥ 3 m.
      ii) Wandelt Raster in EBImage-Objekte um, um kleine Löcher (≤ 4 Pixel) zu füllen.
      iii) Konvertiert die gefüllte Maske zurück in ein Raster und stellt sicher, dass sie mit der ursprünglichen Ausdehnung übereinstimmt.

c) Kleine Artefakte entfernen
   i) Wandelt die Maske ≥ 3 m in Polygone um.
   ii) Entfernt Polygone kleiner als ein definierter Mindestwert (1 m²).
   iii) Bereinigt die Geometrien, um gültige, einfache Polygone zu erhalten.

4) Klassifizierung von Baum-Polygonen
   a) Identifizieren von Polygonen mit TLM-Baumpunkten
      i) Zählt die Anzahl der TLM-Punkte, die mit jedem Polygon überlappen.
      ii) Extrahiert die maximale Vegetationshöhe pro Polygon aus dem Raster.
      iii) Weist initial eine Baumklasse zu:
         (1) "tree": genau ein TLM-Baumpunkt
         (2) "has_tree_many": mehrere überlappende TLM-Punkte
         (3) "small_no_ezb": keine TLM-Punkte, Vegetation ≥ 3 m, Fläche < 6 m²
         (4) "other": alle anderen
   b) Aufteilen von Mehrbaum-Polygonen
      i) Erzeugt Voronoi/Thiessen-Polygone um jeden TLM-Baumpunkt.
      ii) Schneidet Mehrbaum-Polygone anhand der Voronoi-Zellen auf, um einzelne Kronen zu isolieren.
      iii) Identifiziert Fragmente ohne TLM-Punkte und führt sie mit dem nächstgelegenen Baumpolygon zusammen, das ähnliche mittlere VHM-Höhe hat.
      iv) Ergebnis: bereinigter Satz von Polygonen, die alle als "tree" klassifiziert sind.
   c) Verarbeitung von Polygone hoher Vegetation ohne TLM-Punkte
      i) Wählt Polygone aus, die als "small_no_ezb" klassifiziert wurden.
      ii) Große Polygone (> 6 m²) werden als echte Bäume klassifiziert.
      iii) Kleine Polygone werden potentiell mit benachbarten Baum-Polygonen zusammengeführt.

5) Zusammenstellung des endgültigen Baumdatensatzes
   a) Kombiniert alle relevanten Polygone:
      i) Einzelbaum-Polygone ("tree")
      ii) Geteilte Mehrbaum-Polygone ("has_tree_many")
      iii) Polygone mit hoher Vegetation ("small_no_ezb")
   b) Bereinigt und validiert Geometrien.
   c) Berechnet Flächen, weist eindeutige IDs zu und berechnet mittlere VHM-Höhe pro Polygon.
   d) Beinhaltet die relevanten Spalten: ID, Koordinaten (Zentrum und TLM), Anzahl TLM-Punkte, Kronenfläche, maximale und mittlere Höhe.

6) Visualisierung
   a) Optional: Visualisiert die endgültigen Baum-Polygone in mapview, farblich nach Klassifikation oder maximaler Höhe.
   b) Überlagert TLM-Baumpunkte zur visuellen Validierung.

7) Export
   a) Exportiert den endgültigen Polygon-Datensatz als Shapefile (final_tree_layer_YYYYMMDD_HHMMSS.shp) in das Ausgabeverzeichnis.

6.2 Code

```
# ========================================================== #
# Tree Segmentation and Classification Workflow
# ========================================================== #
# Author: Natalia Kolecka
# Date: 2025-12-12
# Description: End-to-end workflow to segment trees from ALS data,
#              classify polygons based on TLM tree points,
#              clean geometries, merge small polygons, and export results.
# ========================================================== #

# ------------------------- #
# INITIAL SETUP ----
# ------------------------- #

rm(list = ls())  # Clear all objects from the workspace to start fresh

# Load required libraries
library(tidyverse)  # Data manipulation and piping
library(terra)       # Raster processing
library(sf)          # Vector (polygon/point) processing
library(lidR)        # LiDAR data analysis & watershed segmentation
library(mapview)     # Interactive maps for visualization
library(EBImage)     # Raster image processing (used for hole-filling)
library(FNN)         # Nearest neighbor calculations (optional)

# ------------------------- #
# PARAMETERS ----
# ------------------------- #

params <- list(
  project_dir = "../R_BKZH_Workspace",
  out_dir            = "outputs",         # Directory for outputs
  vhm_file          = "data/max_als_ktzh_T1091_aoi_workshop.tif",  # Vegetation Height Model
  forest_file        = "data/forest_mask.shp",  # Forest mask polygons
  ezb_file           = "data/ezb.shp",         # Single-tree points (TLM)
  crs_epsg           = 2056,                   # Coordinate reference system
  process_full_raster= TRUE,                   # Process entire raster or subset
  subset_size        = 500,                   # Size of subset if not processing full raster
  min_poly_area      = 1,                      # Minimum polygon area to keep (m²)
  height_threshold   = 3,                     # Minimum height for vegetation to be considered
tree (m)
  hole_size          = 4                       # Maximum size of small holes in VHM to fill
(pixels)
)

# ------------------------- #
# FUNCTIONS ----
# ------------------------- #

# Function to ensure polygons are valid, single-part, and distinct
clean_polygons <- function(polys){
  polys %>%
    st_make_valid() %>%                    # Fix invalid geometries
    st_collection_extract("POLYGON") %>%  # Keep only POLYGON geometries
    st_cast("POLYGON")                    # Convert multipolygons to single-part polygons
  # distinct(geometry, .keep_all = TRUE) # Optional: remove exact duplicates
}

# Function to compute polygon centroids
get_centroids <- function(polys){
  coords <- st_centroid(polys) %>% st_coordinates()  # Compute centroid coordinates
  colnames(coords) <- c("CENTER_X", "CENTER_Y")
  coords
}

# Function to overlay TLM points and get first matching coordinates
get_tlm_coords <- function(polys, tlm_points){
  ov <- st_intersects(polys, tlm_points)   # Find intersections between polygons and TLM
points
  coords <- t(sapply(ov, function(ids){
    if(length(ids) == 0) return(c(0,0))   # If no points intersect, assign (0,0)
    st_coordinates(tlm_points[ids[1], ])  # Take first intersecting TLM point
  }))
  colnames(coords) <- c("TLM_EZB_X", "TLM_EZB_Y")
  coords
```

```
}

# ------------------------- #
# PATHS ----
# ------------------------- #

setwd(params$project_dir)                    # Set working directory
if(!dir.exists(params$out_dir)) dir.create(params$out_dir)   # Create output folder if it
doesn't exist

vhm_file    <- file.path(params$project_dir, params$vhm_file)
forest_file <- file.path(params$project_dir, params$forest_file)
ezb_file    <- file.path(params$project_dir, params$ezb_file)

# ------------------------- #
# LOAD DATA ----
# ------------------------- #

# Load vegetation height raster
vhm <- rast(vhm_file)
crs(vhm) <- paste0("EPSG:", params$crs_epsg)

# Load forest mask polygons
forest <- st_read(forest_file) %>%
  select(fmask, geometry) %>%  # Only keep 'fmask' attribute
  st_zm(drop = TRUE) %>%       # Drop Z/M dimensions if present
  st_transform(params$crs_epsg) # Transform to target CRS

forestmask <- forest %>% filter(fmask == "forest_mask")  # Keep only forest polygons

# Load TLM single-tree points
ezb <- st_read(ezb_file) %>%
  select(geometry) %>%             # Only geometry needed
  rowid_to_column("pid") %>%       # Add unique ID column
  st_transform(params$crs_epsg) %>%
  st_zm(drop = TRUE)               # Drop Z/M dimensions if present

# ------------------------- #
# OPTIONAL SUBSETTING ----
# ------------------------- #

if(!params$process_full_raster){
  # Crop raster and vector data to a smaller area for faster processing
  ext_cut <- ext(c(ext(vhm)[1], ext(vhm)[1]+params$subset_size,
                   ext(vhm)[3], ext(vhm)[3]+params$subset_size))
  vhm <- crop(vhm, ext_cut)
  forestmask <- st_crop(forestmask, ext_cut)
  ezb <- st_crop(ezb, ext_cut)
}

# ------------------------- #
# MASK FOREST AREAS ----
# ------------------------- #

# Keep only non-forest areas
ezb_nf <- st_difference(ezb, st_union(forestmask))  # Remove TLM points inside forest
vhm_nf <- terra::mask(vhm, vect(forestmask), inverse = TRUE)  # Mask forest areas in raster

# ------------------------- #
# VHM SMOOTHING AND WATERSHED SEGMENTATION ----
# ------------------------- #

# Apply 3x3 mean filter to smooth vegetation height model
vhm_sm <- terra::focal(vhm_nf, w = 3, fun = mean, na.rm = TRUE)
crs(vhm_sm) <- paste0("EPSG:", params$crs_epsg)

# Perform watershed segmentation on smoothed VHM
wtr <- lidR::watershed(vhm_sm)()
names(wtr) <- "segm"

# Convert segmented raster to polygons
wtr_poly <- as.polygons(wtr, names = "segm") %>%
  st_as_sf() %>%
  mutate(area_m2 = as.numeric(st_area(.))) %>%   # Calculate area of each polygon
  filter(area_m2 > params$min_poly_area) %>%    # Keep polygons above minimum area
  select(-area_m2)

wtr_poly <- clean_polygons(wtr_poly)  %>%        # Ensure valid geometries
```

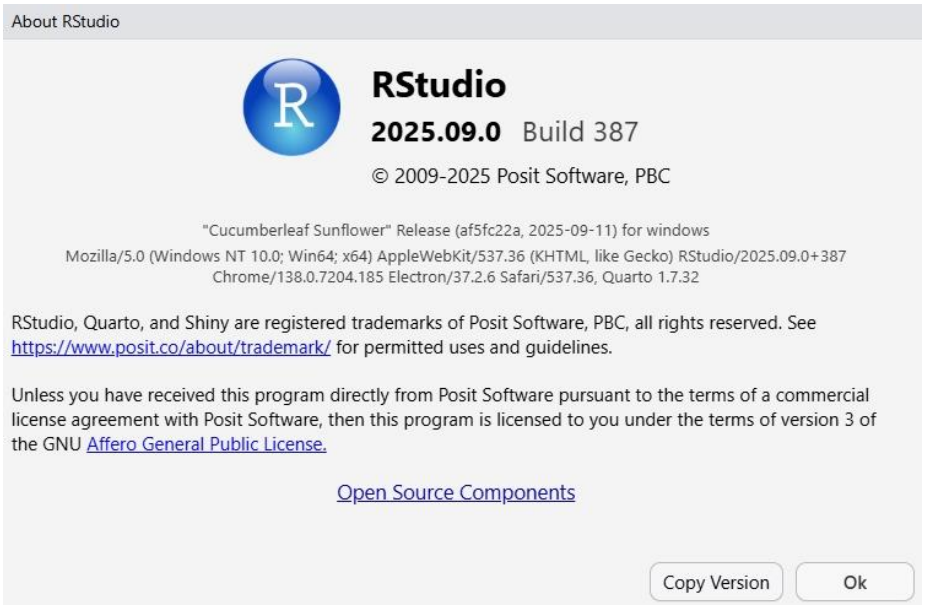
```
  mutate(poly_id = row_number())                # Add unique polygon ID

# ------------------------- #
# MASK LOW VEGETATION (< height_threshold) ----
# ------------------------- #

# Create binary mask: 1 if height >= threshold, 0 otherwise
vhm_mask <- (vhm_sm >= params$height_threshold)*1

# Convert raster mask to EBImage object for morphological operations
vhm_img <- Image(as.matrix(vhm_mask, wide = TRUE))

# Fill small holes in binary raster
vhm_filled <- vhm_img %>%
  `!`() %>%                    # Invert image for hole detection
  bwlabel() %>%                 # Label connected components
  {
    lbl <- .
    tab <- table(lbl)
    small_holes <- as.numeric(names(tab[tab <= params$hole_size])) # Identify small holes
    lbl[lbl %in% small_holes] <- 0                                 # Remove small holes
    lbl
  } %>%
  `>`(0) %>% `!`()             # Convert back to binary mask

# Convert filled binary mask back to SpatRaster
vhm_mask_filled <- rast(matrix(as.numeric(vhm_filled), nrow = nrow(vhm_mask)),
                        crs = crs(vhm_mask)) %>%
  setNames("vhm_high")
ext(vhm_mask_filled) <- ext(vhm)

# Convert filled raster to polygons
vhm_mask_poly <- as.polygons(vhm_mask_filled, dissolve = TRUE) %>%
  st_as_sf() %>%
  st_make_valid() %>%
  filter(vhm_high == 1) %>%           # Keep only high vegetation
  clean_polygons() %>%
  mutate(area_m2 = as.numeric(st_area(.))) %>%
  filter(area_m2 > params$min_poly_area) %>%
  select(-area_m2, -vhm_high)

# Intersect high vegetation polygons with watershed polygons to shrink segments
wtr_poly_shrink <- st_intersection(wtr_poly, vhm_mask_poly) %>%
  clean_polygons() %>%
  mutate(area_m2 = as.numeric(st_area(.)))

# ------------------------- #
# CLASSIFICATION BASED ON TLM POINTS AND MAX HEIGHT ----
# ------------------------- #

# Count number of TLM points per polygon
wtr_poly_class <- wtr_poly_shrink %>%
  mutate(n_points = lengths(st_intersects(., ezb_nf)),
         ID = row_number())  # Unique ID

# Extract max vegetation height per polygon
wtr_poly_class$max_height <- terra::extract(vhm_nf, vect(wtr_poly_class),
                                            fun = max, na.rm = TRUE)[[2]]

# Classify polygons
wtr_poly_class <- wtr_poly_class %>%
  mutate(tree_class = case_when(
    n_points == 1 ~ "tree",                                 # Single tree
    n_points > 1  ~ "has_tree_many",                       # Multi-tree
    n_points == 0 & max_height >= params$height_threshold & area_m2 >= 6 ~ "tree",  # High
vegetation, sufficient area
    n_points == 0 & max_height >= params$height_threshold & area_m2 < 6 ~ "small_no_ezb", #
Small patch
    TRUE ~ "other"
  ))

# Separate polygons by class and remove duplicate geometries
trees_wtr <- wtr_poly_class %>% filter(tree_class == "tree") %>%
  distinct(geometry, .keep_all = TRUE)
multitrees <- wtr_poly_class %>% filter(tree_class == "has_tree_many") %>%
  distinct(geometry, .keep_all = TRUE)
small_wtr <- wtr_poly_class %>% filter(tree_class == "small_no_ezb") %>%
  distinct(geometry, .keep_all = TRUE)
```

```
# ------------------------ #
# SPLIT MULTI-TREE POLYGONS USING VORONOI/THIESSEN ----
# ------------------------ #

## Generate Thiessen / Voronoi polygons around TLM points
thiessen_poly <- ezb_nf %>%
  st_geometry() %>%                # Extract tree point geometries
  do.call(c, .) %>%               # Combine into a single geometry list
  st_voronoi() %>%                # Compute Voronoi polygons
  clean_polygons() %>%
  st_set_crs(2056) %>%            # Assign CRS
  st_crop(ext(vhm)) %>%           # Crop to VHM extent
  st_difference(st_union(forestmask)) %>%  # Remove forested areas
  clean_polygons() %>%
  st_sf(geometry = .)

# Split polygons with multiple trees using Thiessen polygons
multitrees_thiessen <- multitrees %>%
  st_intersection(., thiessen_poly) %>%
  clean_polygons() %>%
  mutate(n_points = lengths(st_intersects(., ezb_nf)),
         tree_class = case_when(
           n_points == 1 ~ "tree",
           n_points == 0 ~ "small_no_ezb",
           TRUE ~ "other"
         ))

# Separate split polygons
trees_thiessen <- multitrees_thiessen %>% filter(tree_class == "tree")
small_thiessen <- multitrees_thiessen %>% filter(tree_class == "small_no_ezb")

# ------------------------ #
# COMBINE TREES AND SMALL POLYGONS ----
# ------------------------ #

# Combine original trees and Thiessen trees
trees <- bind_rows(trees_wtr, trees_thiessen) %>%
  mutate(area_m2 = as.numeric(st_area(.)),
         ID = row_number(),
         mean_vhm = terra::extract(vhm_nf, vect(.), fun = mean, na.rm = TRUE)[[2]])

# Combine small polygons (non-TLM) from both sources
small_polys <- bind_rows(small_wtr, small_thiessen) %>%
  mutate(area_m2 = as.numeric(st_area(.)),
         ID = row_number(),
         mean_vhm = terra::extract(vhm_nf, vect(.), fun = mean, na.rm = TRUE)[[2]])

# ------------------------ #
# MERGE SMALL POLYGONS TO NEIGHBORS ----
# ------------------------ #

# Assign small polygons to nearest tree neighbor
assigned_list <- list()      # Small polygons assigned to trees
independent_list <- list()   # Small polygons becoming independent trees

for(i in seq_len(nrow(small_polys))){
  poly_i <- small_polys[i, ]
  rel <- st_relate(poly_i, trees)[1, ]           # Compute spatial relationships
  nb <- which(substr(rel, 5, 5) == "1")         # Identify neighbors touching polygon

  if(length(nb) > 0){
    diffs <- abs(trees$mean_vhm[nb] - poly_i$mean_vhm)
    best_nb <- nb[which.min(diffs)]             # Choose neighbor with closest mean height
    poly_i$assign_id <- trees$ID[best_nb]
    assigned_list[[length(assigned_list)+1]] <- poly_i
  } else if(poly_i$area_m2 >= 6){               # If no neighbor but large enough, keep as
independent tree
    poly_i$tree_class <- "tree"
    independent_list[[length(independent_list)+1]] <- poly_i
  }
}

assigned_polys <- if(length(assigned_list)>0) bind_rows(assigned_list) %>% st_make_valid()
else NULL
independent_polys   <-   if(length(independent_list)>0)   bind_rows(independent_list)   %>%
st_make_valid() else NULL
```

```
# Merge assigned fragments into original trees
trees_expanded <- trees
if(!is.null(assigned_polys)){
  for(id in unique(assigned_polys$assign_id)){
    base_poly <- trees_expanded %>% filter(ID == id)
    frags <- assigned_polys %>% filter(assign_id == id)
    merged_geom <- st_union(c(base_poly$geometry, frags$geometry)) %>%
      clean_polygons() %>%
      st_union() %>%               # Dissolve edges
      st_combine() %>%             # Flatten structure
      clean_polygons()
    trees_expanded$geometry[trees_expanded$ID == id] <- merged_geom
  }
}
table(st_geometry_type(trees_expanded))

# Recalculate area and mean VHM after merging
trees_expanded <- trees_expanded %>%
  mutate(area_m2 = as.numeric(st_area(.)),
         mean_vhm = terra::extract(vhm_nf, vect(.), fun = mean, na.rm = TRUE)[[2]])

# ------------------------- #
# FINAL DATASET ----
# ------------------------- #

# Compute centroids
centroids_xy <- get_centroids(trees_expanded)

# Extract first matching TLM point coordinates
tlm_ezb_xy <- get_tlm_coords(trees_expanded, ezb_nf)

# Prepare final dataset with required fields
final_dataset <- trees_expanded %>%
  rename(MAX_HGHT = max_height,
         MEAN_HGHT = mean_vhm,
         CROWN_AREA = area_m2,
         IN_TLMEZB = n_points) %>%
  mutate(CENTER_X = centroids_xy[, "CENTER_X"],
         CENTER_Y = centroids_xy[, "CENTER_Y"],
         TLM_EZB_X = tlm_ezb_xy[, "TLM_EZB_X"],
         TLM_EZB_Y = tlm_ezb_xy[, "TLM_EZB_Y"]) %>%
  select(ID, CENTER_X, CENTER_Y, TLM_EZB_X, TLM_EZB_Y, IN_TLMEZB,
         CROWN_AREA, MAX_HGHT, MEAN_HGHT)

# ------------------------- #
# EXPORT ----
# ------------------------- #

out_file <- file.path(params$out_dir,
                      paste0("final_tree_layer_",    format(Sys.time(),    "%Y%m%d_%H%M%S"),
".shp"))

st_write(final_dataset, out_file, delete_dsn = TRUE)  # Export as Shapefile

# ==================== END OF SCRIPT ==================== #
```

6.3 Details zur R-Version

Developed in: R version 4.5.1 (2025-06-13 ucrt)



6.4 Details zur R-Session

> sessionInfo()

R version 4.5.1 (2025-06-13 ucrt)

Platform: x86_64-w64-mingw32/x64

Running under: Windows 11 x64 (build 26100)

Matrix products: default

LAPACK version 3.12.1

locale:

[1] LC_COLLATE=English_United Kingdom.utf8 LC_CTYPE=English_United Kingdom.utf8

[3] LC_MONETARY=English_United Kingdom.utf8 LC_NUMERIC=C

[5] LC_TIME=English_United Kingdom.utf8

time zone: Europe/Zurich

tzcode source: internal

attached base packages:

[1] stats graphics grDevices datasets utils methods base

other attached packages:

[1] FNN_1.1.4.1 EBImage_4.50.0 units_0.8-7 mapview_2.11.4 lidR_4.2.1
[6] sf_1.0-21 terra_1.8-60 lubridate_1.9.4 forcats_1.0.0 stringr_1.5.2
[11] dplyr_1.1.4 purrr_1.1.0 readr_2.1.5 tidyr_1.3.1 tibble_3.3.0
[16] ggplot2_4.0.0 tidyverse_2.0.0

loaded via a namespace (and not attached):
[1] gtable_0.3.6 raster_3.6-32 htmlwidgets_1.6.4
[4] lattice_0.22-7 tzdb_0.5.0 bitops_1.0-9
[7] leaflet.providers_2.0.0 vctrs_0.6.5 tools_4.5.1
[10] crosstalk_1.2.2 generics_0.1.4 stats4_4.5.1
[13] parallel_4.5.1 proxy_0.4-27 pkgconfig_2.0.3
[16] KernSmooth_2.23-26 satellite_1.0.6 data.table_1.17.8
[19] RColorBrewer_1.1-3 S7_0.2.0 leaflet_2.2.3
[22] lifecycle_1.0.4 compiler_4.5.1 farver_2.1.2
[25] tiff_0.1-12 codetools_0.2-20 stars_0.6-8
[28] htmltools_0.5.8.1 class_7.3-23 RCurl_1.98-1.17
[31] lazyeval_0.2.2 crayon_1.5.3 pillar_1.11.0
[34] jquerylib_0.1.4 classInt_0.4-11 abind_1.4-8
[37] locfit_1.5-9.12 tidyselect_1.2.1 digest_0.6.37
[40] stringi_1.8.7 fastmap_1.2.0 grid_4.5.1
[43] cli_3.6.5 magrittr_2.0.4 base64enc_0.1-3
[46] utf8_1.2.6 leafem_0.2.5 e1071_1.7-16
[49] withr_3.0.2 scales_1.4.0 sp_2.2-0
[52] timechange_0.3.0 fftwtools_0.9-11 jpeg_0.1-11
[55] png_0.1-8 hms_1.1.3 rlang_1.1.6
[58] Rcpp_1.1.0 glue_1.8.0 DBI_1.2.3
[61] renv_1.1.5 BiocGenerics_0.54.0 rstudioapi_0.17.1
[64] jsonlite_2.0.0 R6_2.6.1

Referencen auf R-Pakete

1) **tidyverse** : Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R, Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen TL, Miller E, Bache SM, Müller K, Ooms J, Robinson D, Seidel DP, Spinu V, Takahashi K, Vaughan D, Wilke C, Woo K, Yutani H (2019). “Welcome to the tidyverse.” _Journal of Open Source Software_, *4*(43), 1686. doi:10.21105/joss.01686
2) **terra**: Hijmans R (2025). _terra: Spatial Data Analysis_. R package version 1.8-60, https://rspatial.org/

3) **sf**: Pebesma, E., 2018. Simple Features for R: Standardized Support for Spatial Vector Data. The R Journal 10 (1), 439-446, https://doi.org/10.32614/RJ-2018-009
4) **lidR**:
Roussel, J.R., Auty, D., Coops, N. C., Tompalski, P., Goodbody, T. R. H., Sánchez Meador, A., Bourdon, J.F., De Boissieu, F., Achim, A. (2020). lidR : An R package for analysis of Airborne Laser Scanning (ALS) data. Remote Sensing of Environment, 251 (August), 112061. doi:10.1016/j.rse.2020.112061
Jean-Romain Roussel and David Auty (2025). Airborne LiDAR Data Manipulation and Visualization for Forestry Applications. R package version 4.2.1. https://cran.r-project.org/package=lidR
5) **mapview**: Appelhans T, Detsch F, Reudenbach C, Woellauer S (2025). _mapview: Interactive Viewing of Spatial Data in R_. R package version 2.11.4, https://github.com/r-spatial/mapview
6) **units**: Pebesma E, Mailund T, Hiebert J (2016). "Measurement Units in R." _R Journal_, *8*(2), 486-494. https://doi.org/10.32614/RJ-2016-061
7) **FNN**: Beygelzimer A, Kakadet S, Langford J, Arya S, Mount D, Li S (2024). _FNN: Fast Nearest Neighbor Search Algorithms and Applications_. R package version 1.1.4.1.
8) **EBImage**: Gregoire Pau, Florian Fuchs, Oleg Sklyar, Michael Boutros, and Wolfgang Huber (2010): EBImage – an R package for image processing with applications to cellular phenotypes. Bioinformatics, 26(7), pp. 979-981, 10.1093/bioinformatics/btq046