Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

Laserchicken—A tool for distributed feature calculation from massive LiDAR point cloud datasets

C. Meijer [a,*], M.W. Grootes [a], Z. Koma [b], Y. Dzigan [a], R. Gonçalves [a], B. Andela [a], G. van den Oord [a], E. Ranguelova [a], N. Renaud [a], W.D. Kissling [b]

[a] Netherlands eScience Center, Science Park 140, 1098 XG Amsterdam, The Netherlands
[b] Institute for Biodiversity and Ecosystem Dynamics (IBED), University of Amsterdam, P.O. Box 94240, 1090 GE Amsterdam, The Netherlands

A R T I C L E   I N F O

Article history:
Received 27 November 2019
Received in revised form 12 June 2020
Accepted 6 November 2020

Keywords:
LiDAR
Point cloud
Feature extraction
Airborne Laser Scanning (ALS)
Biodiversity
Ecology
3D vegetation structure

A B S T R A C T

Point cloud datasets provided by LiDAR have become an integral part in many research fields including archaeology, forestry, and ecology. Facilitated by technological advances, the volume of these datasets has steadily increased, with modern airborne laser scanning surveys now providing high-resolution, (super-)national scale, multi-terabyte point clouds. However, their wider scientific exploitation is hindered by the scarcity of open source software tools capable of handling the challenges of accessing, processing, and extracting meaningful information from massive datasets, as well as by the domain-specificity of existing tools. Here we present Laserchicken, a user-extendable, cross-platform Python tool for extracting statistical properties of flexibly defined subsets of point cloud data, aimed at enabling efficient, scalable, distributed processing of multi-terabyte datasets. We demonstrate Laserchicken's ability to unlock these transformative new resources, e.g. in macroecology and species distribution modelling, where it is used to characterize the 3D vegetation structure at high resolution (<10 m) across whole countries or regions. We further discuss its potential as a domain agnostic, flexible tool that can also facilitate novel applications in other research fields.

Code metadata

Current code version	0.4.2
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2019_325
Legal Code License	Apache 2.0
Code versioning system used	git
Software code languages, tools, and services used	python (≥3.6)
Compilation requirements, operating environments & dependencies	laspy, scikit-learn, scipy, pytest, mock, plyfile, python-dateutil, shapely, PySh, pandas, click, colorama, psutil, numpy
If available Link to developer documentation/manual	
Support email for questions	c.meijer@esciencecenter.nl

Software metadata

Current software version	0.2.0
Permanent link to executables of this version	https://github.com/eEcoLiDAR/laserchicken/releases/tag/0.2.0
Legal Software License	Apache 2.0
Computing platforms/Operating Systems	Linux, OS X, Microsoft Windows
Installation requirements & dependencies	laspy, scikit-learn, scipy, pytest, mock, plyfile, python-dateutil, shapely, PyShp, pandas, click, colorama, psutil, numpy
If available, link to user manual - if formally published include a reference to the publication in the reference list	https://laserchicken.readthedocs.io
Support email for questions	c.meijer@esciencecenter.nl

* Corresponding author.
E-mail address: c.meijer@esciencecenter.nl (C. Meijer).

https://doi.org/10.1016/j.softx.2020.100626

1. Motivation and significance

LiDAR, enabling the detailed 3D characterization of an area, has been transformative in many fields [1–5]. For instance, (land-)surveying, archaeology, forestry, and ecology/biology have extensively adopted LiDAR systems operating from a variety of scanning platforms, with especially airborne laser scanning (ALS) being a flexibly deployable technique capable of covering large spatial extents.

For most scientific domains, the desired information is encoded in the properties of ensembles of points (3-dimensionally localized returns from the emitted scanning pulse), either co-located in a predefined spatial extent, or associated with a certain object. The exploitation of LiDAR datasets thus requires the definition of suitable metrics, referred to as features, enabling the extraction of relevant information from the point cloud. Accordingly, various domains have invested significant effort into identifying features that enable the estimation of properties such as biomass and tree diameter (forestry), habitat structures (animal ecology [6,7]), or single trees (urban planning [8]), with the use of LiDAR in any new context potentially requiring new metrics (and effort).

Concurrent with these efforts, advances in LiDAR and data-storage technology, leading to LiDAR's availability at relatively low-cost, have facilitated the generation of massive datasets. For example, modern ALS datasets on (super-)national scales encompass hundreds-of-billions of data points and tens-of-terabytes of data (e.g. AHN3,[1] LiDAR aérien 2015,[2] National LiDAR Dataset[3]). Providing high spatial resolution data covering geographic extents of tens to hundreds-of-thousands of square kilometers, such datasets represent a transformative new resource in research fields where detailed local environmental information (on scales of meters) is known to be important, but which simultaneously span large geographic extents. However, their scientific exploitation, e.g. in macroecology and global change biology, has faced a number of challenges [9].

Firstly, the data volume represents a major challenge. Extracting features from point clouds requires performing calculations over their constituent points. With these numbering in the hundreds-of-billions, even the simplest calculations pose a challenge in terms of required CPU time and data access. However, the calculations for one subset of points are usually independent of further distant points in the dataset, making feature extraction amenable to distribution and parallelization, i.e tractable in principle. An additional problem in scientifically exploiting ALS/LiDAR data is the domain-specificity of features. Extendability and flexibility, i.e. enabling the user to define appropriate features and subsets of data points in line with their requirements, are therefore key requirements of any software to be used in this context. Exactly this aspect, however, is severely limited in non-open source software packages, which may furthermore cause interoperability problems in the context of open, accessible, and reproducible scientific research [10,11].

The current landscape of software for processing LiDAR data fails to adequately address these combined challenges. For instance, amongst the commonly used software and tools, some are not open source (e.g. LAStools [12,13] and OPALS [14,15]), some are aimed at smaller volumes of data (e.g. FUSION[4]), some are fully tailored to their field and hence lack flexibility (e.g. lidR[5] is focused on forestry and limited to an area-based approach[6]), and, crucially, no available free open source software (FOSS) tools currently support full horizontal scalability via workload distribution.

Here, we present Laserchicken, an open source, user-extendable, Python package facilitating the extraction of features from point cloud datasets in a simultaneously flexible and efficient manner, amenable to scaleable distributed processing of massive data volumes, and employable across a range of computational environments.

2. Software description

Laserchicken is a Python library comprised of several modules, each performing a single task (detailed below). In the interest of cross-platform employability and scaleable distributed processing the library has been kept light-weight and solely focused on feature extraction, providing functions optimized for execution in a single process.

In the following, we highlight Laserchicken's characteristics, describe its architecture (Section 2.1) and workflow (Section 2.2), and briefly discuss its performance (Section 2.3).

In Laserchicken, the LiDAR dataset is referred to as the environment point cloud (EPC), and the subsets of points over which a metric is to be calculated are referred to as neighborhoods. Each neighborhood is defined by a target volume and a target point (e.g. a cube of a certain size and its centroid, respectively), with all points enclosed in the volume constituting the neighborhood (Fig. 1). Four volume definitions are implemented: an infinite square cell, an infinite cylinder, a cube and a sphere (Fig. 1). All target points together form the target point cloud (TPC), exemplified for a regular grid in Fig. 2. The TPC can be freely defined by the user. Features are calculated over the list of neighborhoods, with the feature values being associated with each neighborhood's defining target point, thus forming the enriched target point cloud (eTPC). This novel concept forms the core of Laserchicken's flexible and domain agnostic functionality, for example enabling Laserchicken to be seamlessly employed for extracting features in a classic area-based approach (see Fig. 2) as commonly used in a macroecological context [7] and Section 3.1, as well for characterizing the local geometry of a point cloud around a given point (e.g. within a sphere) to explore and identify structures within a dataset at high resolution (see Section 3.2).

2.1. Software architecture

Laserchicken consists of four core modules (`load`, `compute neighbors`, `features`, `export`), and two optional modules (`filter`, `normalize`), which provide a processing workflow for feature extraction (Fig. 3), with file based input (EPC and TPC) and output (eTPC).

Laserchicken relies on standard Python libraries (numpy, scipy), with additional usage of laspy, pylas, and plyfile libraries for input and output, and of the shapely library to support (geo-)spatial filtering of points.

2.2. Workflow and software functionality

Fig. 3 depicts Laserchicken's workflow for feature extraction. The rest of this section provides descriptions of Laserchicken's modules accompanied with code examples.

[1] https://www.pdok.nl/introductie/-/article/actueel-hoogtebestand-nederland-ahn3-.
[2] http://donnees.ville.montreal.qc.ca/dataset/lidar-aerien-2015.
[3] https://environment.data.gov.uk.
[4] http://forsys.cfr.washington.edu/fusion/fusion_overview.html.
[5] https://github.com/Jean-Romain/lidR.
[6] See Section 3.1.

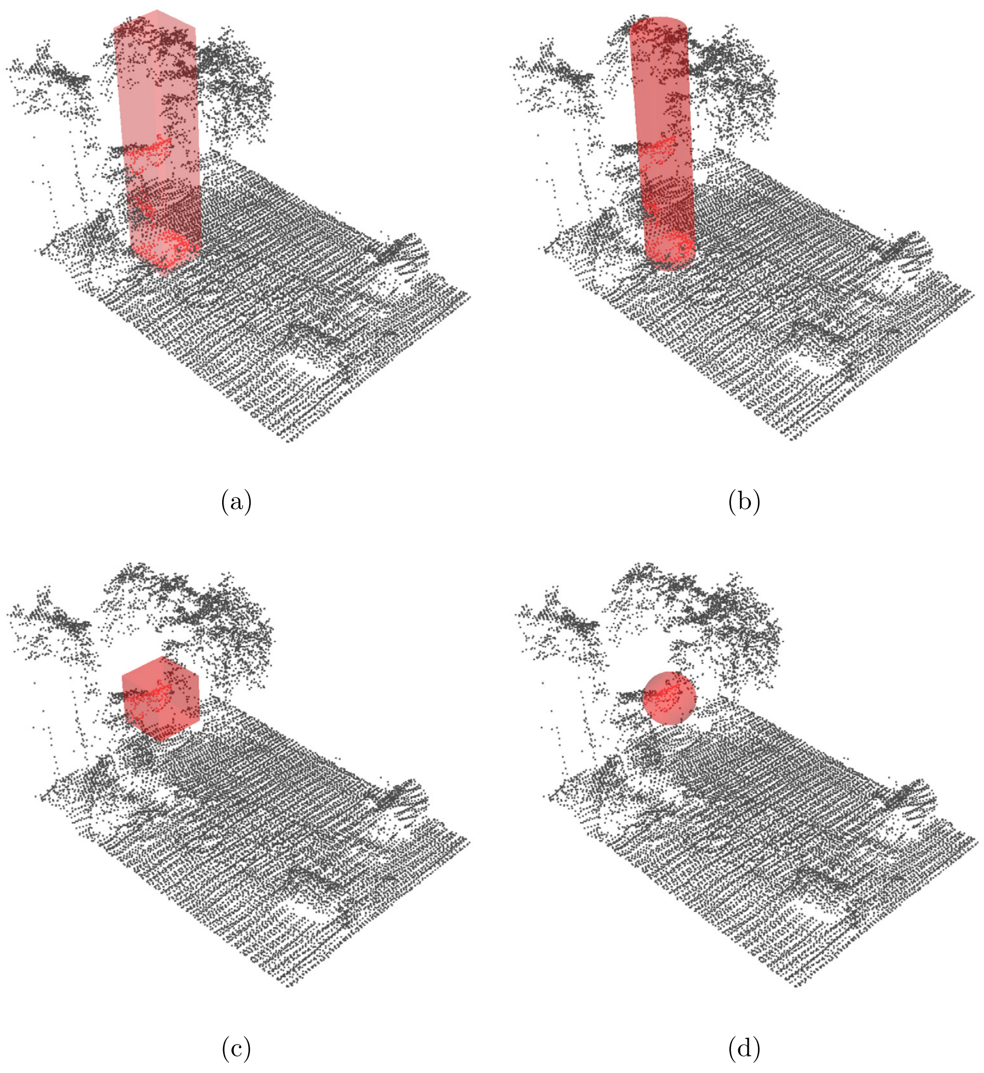

Fig. 1. Examples of volume geometries (red) available to define neighborhoods (shown as red points; enclosed): an infinite square cell (a), an infinite cylinder (b), a cube (c), and a sphere (d). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

2.2.1. Module: `load`

The `load` module provides functionality to load point cloud datasets provided in ASPRS LAS/LAZ, or in PLY format, and is used for both input point clouds. In conjunction with the PDAL library [16], this provides access to a comprehensive range of point cloud data formats.

```python
# Example code for loading data
from laserchicken import load
point_cloud = load('AHN3.las')
```

2.2.2. Module: `Compute Neighbors`

The `Compute neighbors` module constructs the neighborhoods as defined by the TPC and target volume by identifying the points in the EPC which reside in the specified volume centered on the target points, returning each as a list of indices to the EPC. This essential step of computing neighboring points for large samples of points is computationally expensive. Laserchicken uses the optimized ckDtree class (kdTrees are a space-partitioning data structure) provided by the scipy library to organize both the EPC and TPC in kdTrees in an initial step prior to the computation of neighbors, subsequently accelerating the process of computing neighbors by using the indices of the points with respect to the kDtrees. All volumes are defined using the cylinder volume type as the base volume class, enabling Laserchicken to make use of efficient $k = 2$ dimensional kDtrees.

```python
# Example code for computing neighbors
from laserchicken import compute_neighborhoods
from laserchicken import build_volume
targets = point_cloud
volume = build_volume('sphere', radius=5)
neighborhoods = compute_neighborhoods(point_cloud,
     targets, volume)
```

2.2.3. Module: `Features`

Feature extraction is performed by the `Features` module, which requires the EPC, the TPC, the computed list of neighborhoods, and a list of requested features as input. For each target point it selects the points of the associated neighborhood and calculates a vector of the requested features over these. This feature vector is appended to the target point, thus defining the eTPC.

```python
# Example code for computing features
from laserchicken import compute_features
compute_features(point_cloud, neighborhoods,
    targets, ['std_z','mean_z','slope'], volume)
```

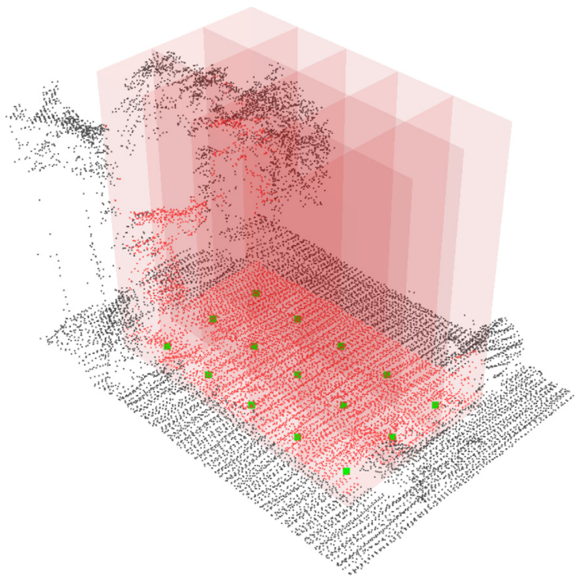
Fig. 2. Illustration of a target point cloud (TPC, green points) representing the centroids of a regular grid cell, and the neighborhoods (red points) defined by a square infinite cell target volume (red columns). Features are calculated over the neighborhood of each target point and then associated with the target point, thus forming the enriched target point cloud (eTPC). Points that are not included in the neighborhoods are shown in black. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

```
# Example code for computing parameterized
    features
from laserchicken import
    register_new_feature_extractor
from laserchicken.feature_extractor.
    band_ratio_feature_extractor import
    BandRatioFeatureExtractor
register_new_feature_extractor(
    BandRatioFeatureExtractor(None, 1, data_key='
    normalized_height'))
register_new_feature_extractor(
    BandRatioFeatureExtractor(1, 2, data_key='
    normalized_height'))
register_new_feature_extractor(
    BandRatioFeatureExtractor(2, None, data_key='
    normalized_height'))
register_new_feature_extractor(
    BandRatioFeatureExtractor(None, 0, data_key='z
    '))
```

Currently, a number of features are implemented, including percentiles of the height distribution and eigenvectors (Table 1). Computationally expensive calculations requiring multi-dimensional linear algebraic operations (e.g. eigenvectors and eigenvalues) have been vectorized using the einsum function of the numpy library to optimize performance. Their implementation can serve as a template for new features requiring similar operations.

2.2.4. Module: `Export`

The Export module serializes the eTPC in PLY, CSV, or LAS/LAZ format for further analysis with the user's choice of software. The PLY format is preferred, as it is flexibly extendable, and provides economical memory usage as well as the option of recording provenance data. Depending on the use case and domain, conversion to another format (e.g. GeoTiff) may be desirable. We note that the PDAL [16] and GDAL [26] libraries can be interfaced with Laserchicken's output for such a purpose.

```
# Example code for exporting data
from laserchicken import export
export(point_cloud, 'my_output.ply')
```

2.2.5. Module: Filter (optional)

Laserchicken provides the option of filtering the EPC prior to extracting features. Points may be filtered on the value of a single attribute relative to a specified threshold (e.g. above a certain normalized height above ground), or on specific values of their attributes (e.g. LAS standard classification). It is also possible to filter with (geo-)spatial layers such as polygons (e.g. regions of interest, land cover types), i.e. selectively including or excluding points.

```
# Example code for filtering with a polygon in wkt
    format
from laserchicken.filter import select_polygon
polygon = "POLYGON((" + \
          " 131963.984125 549718.375000," + \
          " 132000.000125 549718.375000," + \
          " 132000.000125 549797.063000," + \
          " 131963.984125 549797.063000," + \
          " 131963.984125 549718.375000))"
point_cloud = select_polygon(point_cloud, polygon)
```

```
# Example code for attribute filtering
from laserchicken.filter import select_above,
    select_below
points_below_1m = select_below(point_cloud, '
    normalized_height', 1)
points_above_1m = select_above(point_cloud, '
    normalized_height', 1)
```

2.2.6. Module: Normalize (optional)

A number of features (Table 1) require the normalized height above ground as input. Laserchicken provides the option of internally constructing a digital terrain model (DTM) and deriving this quantity.

To this end, the EPC is divided into small cells (e.g. 1 m or 2.5 m squared). The lowest point in each cell is taken as the height of the DTM. Each point in the cell is then assigned a normalized height with respect to the derived DTM height. This results in strictly positive heights and smooths variations in elevation on scales larger than the cell size. The normalized EPC can be used directly in further analysis, or serialized to disk.

```
# Example code for normalizing height
from laserchicken.normalize import normalize
normalize(point_cloud)
```

2.3. Distribution and performance

Laserchicken provides a library of (vectorized) single process operations on point cloud data sets. As such, although the actual focus of Laserchicken lies on enabling the processing of massive data sets as discussed below, we briefly benchmark the performance of the library against the widely used FOSS LiDAR processing tool lidR.[7] We note, however, that unlike Laserchicken, which allows the user to freely specify any set of targets, lidR requires (and is optimized for) the extraction of features on regular grids. The results of using both libraries to extract a set of 3 features (90th percentile normalized height, median normalized height, entropy normalized height in vertical bins of 0.5 m; see

[7] https://github.com/Jean-Romain/lidR.

Table 1
Features currently implemented in Laserchicken..

Feature name	Formal description	Example of use	References
Point density	$\frac{N}{V}$ where V is the target volume or area	Point cloud spatial distribution	
Pulse penetration ratio	$\frac{N_{\text{ground}}}{N_{\text{total}}}$	Tree species classification	[17]
Echo ratio	$100 \cdot \frac{N_{\text{3D}}}{N_{\text{2D}}}$	Roof detection	[18]
Skewness	$\frac{1}{\sigma^3} \cdot \sum \frac{(Z_i - \bar{Z})^3}{N}$	Vegetation, ground, and roof classification and detection	[19]
Kurtosis	$\frac{1}{\sigma^4} \cdot \sum \frac{(Z_i - \bar{Z})^4}{N}$	Vegetation, ground, and roof classification and detection	[19]
Standard deviation	$\sqrt{\sum \frac{(Z_i - \bar{Z})^2}{N-1}}$	Classification of reed within wetland	[20]
Variance	$\sum \frac{(Z_i - \bar{Z})^2}{N-1}$	Classification of reed within wetland	[20]
Sigma Z	$\sqrt{\sum \frac{(R_i - \bar{R})^2}{N-1}}$ where R_i is the residual after plane fitting		Adapted from [20]
Minimum Z	Z_{min}	Simple digital terrain model in wetlands	[20]
Maximum Z	Z_{max}	Height and structure of forests	[21]
Mean Z	$\frac{1}{N} \cdot \sum Z_i$	Height and structure of forests	[21]
Median Z	Z_{median}	Height and structure of forests	[21]
Range Z	$\lvert Z_{\text{max}} - Z_{\text{min}} \rvert$	Height and structure of forests	[21]
Percentiles Z	Height of every 10th percentile.	Height and structure of forests	[21]
Eigenvalues	$\lambda_1, \lambda_2, \lambda_3$, with $\lvert\lambda_1\rvert \geq \lvert\lambda_2\rvert \geq \lvert\lambda_3\rvert$	Classification of urban objects	[22]
Normal vector	eigen vector $\vec{v}_3$	Roof detection	[23]
Slope	$\tan(\arccos(\vec{v}_3 \cdot \vec{k}))$, where $\vec{k} = [0, 0, 1]^T$	Planar surface detection	[24]
Entropy Z	$-\sum_i P_i \cdot \log_2 P_i$, with $P_i = \frac{N_i}{\sum_j N_j}$ and N_i points in bin i	Foliage height diversity	[25]
Coefficient variance Z	$\frac{1}{\bar{Z}} \cdot \sqrt{\sum \frac{(Z_i - \bar{Z})^2}{N-1}}$	Urban tree species classification	[8]
Non-ground density absolute mean	$\frac{100}{N_{\text{non-ground}}} \cdot \sum_{i \in \text{non-ground}} [Z_i > \bar{Z}_{\text{non-ground}}]$	Urban tree species classification	[8]
Band ratio	$\frac{N_{Z_i < Z < Z_j}}{N_{\text{tot}}}$ with Z_i and Z_j provided by user	Height and vertical structure of vegetation	

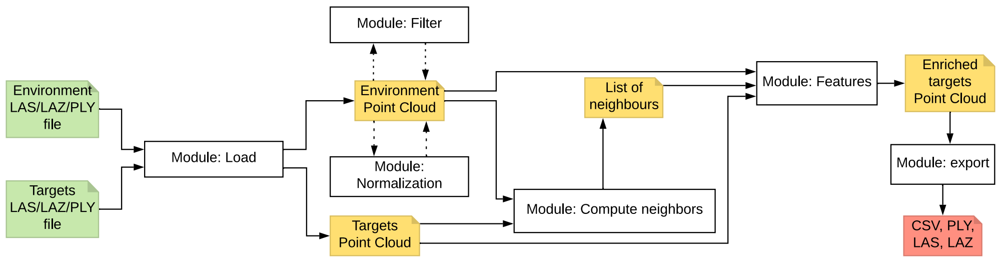

Fig. 3. Laserchicken workflow. A LiDAR dataset and a set of target points – the environment point cloud (EPC) and the target point cloud (TPC), respectively – are loaded from file, followed by the construction of neighborhoods for each target point. Features are subsequently extracted over each neighborhood and appended to the associated target points, forming the enriched target point cloud (eTPC), which can be exported in several formats. Optionally, the EPC can be filtered or normalized prior to further processing.

Table 1) for a regular grid of 200 × 200 cells with 10 m × 10 m resolution covering a total spatial extent of 2 km × 2 km with varying volumes of LiDAR point cloud data are shown in Table 2. Laserchicken requires between 80% and 50% longer to extract these features, with the difference decreasing with data volume, likely as the result of its general purpose target definitions compared to lidR. However, the actual perceivable difference in use is much smaller, as for the test data of sizes typical of real applications ($\gtrsim$ 5 GB) the process is fully I/O[8] dominated, with comparable data loading times of $\gtrsim$ 80 s in both libraries making the difference $\sim$ 10%.

This I/O dominance and the accompanying memory requirements also highlight the major challenges involved in processing massive LiDAR datasets Laserchicken has been designed to address. Even when running a currently available FOSS tool such as lidR with parallel processes enabled, the achievable performance is limited by the host system's available memory, as these tools run on a (single) local system – a system with 32 GB will only be able to support $\sim$4 processes at a given time. The only solution to this dilemma is horizontal scalability, i.e. the ability to distribute the needed computations over many nodes/machines, such as cloud or cluster infrastructures. To the extent of our knowledge Laserchicken is, at the time of writing, the only FOSS that inherently supports such arbitrary horizontal scaling, as demonstrated by the examples of features and workflows provided in Section 3, providing it with the ability to easily and rapidly outscale existing LiDAR processing FOSS tools. Combined with the freely definable targets and user-definable features this makes Laserchicken a powerful tool for researchers in any domain to process LiDAR data – and in particular massive datasets – also including those making use of features extracted on regular grids.

8 Input/Output dominated, i.e. the loading and serializing of data constitutes the dominant activity in execution time.

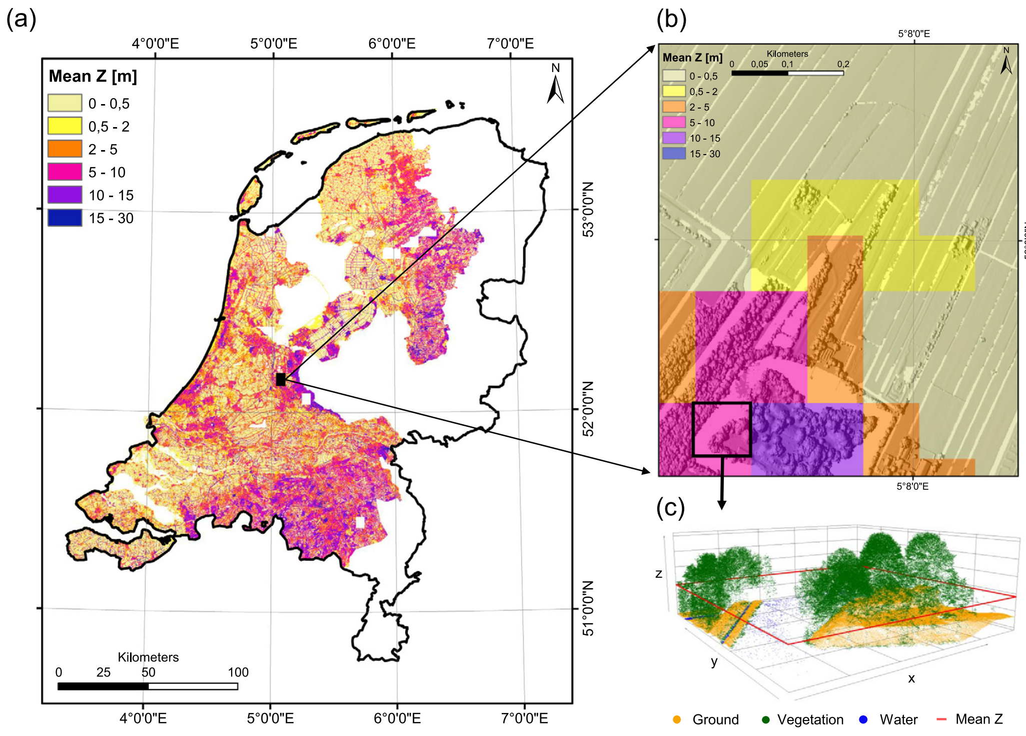

Fig. 4. An example of a LiDAR feature (Mean Z) calculated from the AHN3 dataset of the Netherlands using a grid with 100 m ×100 m resolution (for visualization purposes). (a) Mean Z calculated with an area-based approach over an area spanning roughly two thirds of the Netherlands (i.e. reflecting current data availability, with empty regions representing the lack of data coverage of the AHN3 until July 2019). (b) A landscape representation of the calculated Mean Z for an area in the centre of the Netherlands. (c) Representation of the local point cloud within one grid cell showing the pre-classified points (ground, vegetation, water) and the derived Mean Z.

Table 2
Mean execution time and standard deviation in seconds (10 samples) for the extraction of 90th percentile normalized height, median normalized height, and entropy normalized height in vertical bins of 0.5 m features on a regular grid of 200 × 200 10 m × 10 m infinite vertical cells covering a 2 km × 2 km spatial extent with varying volumes of LiDAR point cloud data executed as a single process (single-threaded) on a MacBook Pro with 2,3 GHz Intel Core i5 processor and 16 GB RAM.

Data volume	163 MB	1.4 GB	5.1 GB	7.4 GB
lidR	0.41 s ± 0.01 s	4.4 s ± 0.2 s	14.1 s ± 0.5 s	17.4 s ± 0.6 s
Laserchicken	0.73 s ± 0.08 s	7.63 s ± 0.2 s	21.4 s ± 0.3 s	26.0 s ± 0.8 s

3. Illustrative examples

Below, we showcase the use of Laserchicken for ecological research, highlighting the flexibility of the package in providing a tool for a broad range of applications to ALS/LiDAR data, as well as its ability to support distributed processing of massive datasets.

3.1. Local habitat characterization at national scales

Laserchicken is currently employed in the eEcoLiDAR project [9] at the University of Amsterdam in cooperation with the Netherlands eScience Center, with the scientific goal to characterize the vertical and horizontal complexity of vegetation and landscapes at high resolution across regional to national/-continental scales on the basis of ALS data. In particular, the project uses Laserchicken to calculate features as listed in Table 1 from the AHN3 dataset[9] (currently ∼640 billion points covering ∼30 000 km^2) in 10 m × 10 m cells covering the whole Netherlands, i.e. in a classical area/grid based approach (see Fig. 4). The processing is carried out an a cluster of 6 virtual machines (VM), each with 6 cores, 32 GB RAM, and 256 GB local HDD.

The AHN3 ALS dataset is first re-gridded to a regular grid of 2 km × 2 km tiles (i.e a full multiple of the desired resolution of 10m on a side) covering the full geographical extent of the data using the MassivePotreeConverter [27] tool, with the points associated with each tile, i.e the EPC, being serialized in LAZ format. For each tile a LAZ TPC file defining the center points of the 10 m × 10 m cells it contains is created. These input data (EPC and associated TPC LAZ files) are subsequently distributed across the cluster's virtual machines.

For feature extraction multiple processes of Laserchicken are executed in parallel across the cluster, with the eTPCs for each tile being serialized locally in PLY format. Processing the AHN3 dataset at 10 m × 10 m resolution with 12 parallel processes (2 per VM due to RAM requirements) and extracting the full list of features specified in Table 1 completes in 96 h of wall time, with the required time scaling inversely with the number of processes.

The eTPC files are merged and rasterized (using a bespoke python tool) into (several) multi-band GeoTIFF files for further processing. An example of the GeoTiff output is shown in Fig. 4.

3.2. Point cloud structure: towards classification and object identification

The classification of ALS returns by the specific object/surface from which they return (e.g. ground, vegetation, structures of human origin) is non-trivial, in particular for echos *not* associated with the earth's surface. Accordingly, the scientific exploitation of ALS datasets often requires significant classification efforts or further preparatory processing, e.g. to quantify local (geometric) structures in the point cloud data. For example, the identification of planar surfaces can be very effectively employed in defining an input point cloud for the detailed characterization of vegetation

9 https://www.pdok.nl/introductie/-/article/actueel-hoogtebestand-nederland-ahn3-.

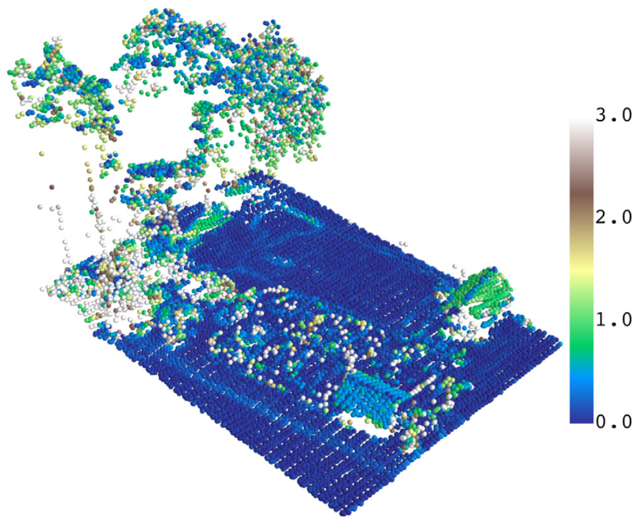

Fig. 5. A point cloud colored-coded by the local value (0.5 m radius) of the slope feature (see Table 1), which can help in a segmentation approach to identify planar surfaces. Contiguous planar surfaces with relatively homogenous slope values are visible, including rooftops (green and sky blue), ground points (dark blue), and the slopes of embankments (blue). In contrast, trees and other vegetation (e.g. left upper part) have diverse slope values. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

structures (which requires the removal of returns from structures of human origin as well as ground points), and has been shown to strongly improve the computational efficiency of LiDAR processing when identifying linear vegetation elements (e.g. hedges) in agricultural landscapes [28].

By employing the EPC as the TPC, Laserchicken trivially facilitates the calculation of local geometric properties of the point cloud, e.g. normal vectors, eigenvectors, or the local slope, required as input for such classification schemes. In combination with its vectorized calculation of (such) properties, and the ability to distribute the required processing, Laserchicken thus facilitates such computationally intensive (pre-)processing and provides a framework to expand the spatial extent over which it can be applied.

More precisely, using Laserchicken, the extraction of the local eigenvectors, normal vectors, and slope for a 1 m diameter spherical target geometry around each point for a point cloud dataset with a density of $\sim$20 pts m^{-2}, executed using 6 distributed processes on the cluster of virtual machines described in Section 3.1, requires ca. 13 min for an area of 1 km^2 making such processing of datasets covering tens to hundreds of square kilometers a tractable proposition, albeit still computationally expensive and accordingly constrained in spatial extent in practice. A sub-region processed in this manner is shown in Fig. 5, and highlights the potential for detection of planar surfaces.

4. Impact

4.1. Ecology and biogeography

Although LiDAR/ALS technology already represents a key resource in animal-habitat studies at local scales [6], its use at regional/continental scales still remains limited [7]. With the availability and accessibility of high-resolution, nationwide ALS datasets steadily increasing, the key remaining bottle-neck is the lack of open source, flexible, user-extendable tools capable of utilizing modern distributed computing infrastructures to efficiently extract information from multi-terabyte LiDAR datasets. The Laserchicken package presented here provides such a tool, expanding beyond the capabilities of existing FOSS tools by leveraging its horizontal scalability to handle the challenge these data volumes pose.

Within the eEcoLiDAR project [9], Laserchicken has already been instrumental in defining new classification methods for wetland habitats [29]. Laserchicken is currently being used to extract comprehensive 3D habitat measurements across the entire Netherlands, forming the basis for nationwide studies that model species distributions of birds and insect pollinators at high resolution, taking the detailed 3D habitat structure into account. Furthermore, Laserchicken provides a framework to quantitatively investigate the technical challenges for upscaling local habitat measurements to regional and continental scales, and for assessing the robustness and transferability of LiDAR metrics for monitoring habitat structure across space and time. This will provide unprecedented insights into animal-habitat relationships and the impact of land use change. It will further provide a baseline for monitoring habitat changes within and across diverse types of ecosystems and thus foster the use of LiDAR-derived indicators in the context of Essential Biodiversity Variables (EBVs) of habitat structure [30,31].

4.2. Other domains

Although developed in the context of macroecological research, the Laserchicken toolkit is domain agnostic, and can be used in service of any research objective which requires the calculation of characteristics for subsets of a (massive) point cloud dataset. For instance, Laserchicken provides the capabilities to address problems in archaeology, such as the detection of archaeological remnants, and especially standing remains, in vegetated settings [32–34]. In contrast to constructing a high-resolution terrain model which often fails to detect these important types of remnants, a segmentation of the point cloud data on the basis of the local geometry of the neighborhood of each point has been shown to constitute a promising approach [33]. However, the computational load combined with a lack of suitable software has been identified as significant impediment to adopting such an approach. Laserchicken provides all the required capabilities, making it ideally suited for identifying archaeological remnants following this approach, or any other local structures that require the calculation of features from point cloud datasets.

5. Conclusions

Laserchicken constitutes a powerful software for extracting features from point cloud datasets, employable on a wide range of workflows across a broad spectrum of computing infrastructures. The concepts of a flexibly definable target point cloud and target volumes, and the implementation of a standard architecture, facilitating the extension by user-defined features, make Laserchicken a domain agnostic tool. Crucially, Laserchicken also facilitates distribution of the computationally expensive operation of feature extraction from massive, multi-terabyte datasets, making the scientific exploitation of such increasingly common resources a feasible proposition and unlocking their potential for a wide range of scientific domains. Laserchicken is fully functional in its current form as described here. Nevertheless, further development of its functionality is on-going, and additions from interested communities, in particular to the set of extractable features beyond those listed in Table 1, are welcome.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Smith DE, Zuber MT, Frey HV, Garvin JB, Head JW, Muhleman DO, et al. Topography of the northern hemisphere of mars from the mars orbiter laser altimeter. Science 1998;279(5357):1686–92. http://dx.doi.org/10.1126/science.279.5357.1686, URL http://science.sciencemag.org/content/279/5357/1686.

[2] Martinez-Rubi O, Kleijn Md, Verhoeven S, Drost N, Attema J, Meersbergen Mv, et al. Using modular 3D digital earth applications based on point clouds for the study of complex sites. Int J Digit Earth 2016;9(12):1135–52. http://dx.doi.org/10.1080/17538947.2016.1205673.

[3] Hu S, Li Z, Zhang Z, He D, Wimmer M. Efficient tree modeling from airborne lidar point clouds. Comput Graph 2017;67:1–13. http://dx.doi.org/10.1016/j.cag.2017.04.004, URL http://www.sciencedirect.com/science/article/pii/S0097849317300420.

[4] LiDAR Drives forwards. Nat Photonics 2018;12(8):441. http://dx.doi.org/10.1038/s41566-018-0235-z, URL https://www.nature.com/articles/s41566-018-0235-z.

[5] Colaço AF, Molin JP, Rosell-Polo JR, Escolà A. Application of light detection and ranging and ultrasonic sensors to high-throughput phenotyping and precision horticulture: current status and challenges. Horticult Res 2018;5(1):35. http://dx.doi.org/10.1038/s41438-018-0043-0, URL https://www.nature.com/articles/s41438-018-0043-0.

[6] Davies AB, Asner GP. Advances in animal ecology from 3D-LiDAR ecosystem mapping. Trends Ecol Evol 2014;29(12):681–91. http://dx.doi.org/10.1016/j.tree.2014.10.005, URL https://www.cell.com/trends/ecology-evolution/abstract/S0169-5347(14)00224-9.

[7] Bakx TRM, Koma Z, Seijmonsbergen AC, Kissling WD. Use and categorization of light detection and ranging vegetation metrics in avian diversity and species distribution research. Divers Distrib 2019;25(7):1045–59, URL https://www.jstor.org/stable/26662604.

[8] Koma Z, Koenig K, Höfle B. Urban tree classification using full-waveform airborne laser scanning. ISPRS Ann Photogramm Remote Sensing Spat Inf Sci 2016;III-3:185–92. http://dx.doi.org/10.5194/isprs-annals-III-3-185-2016.

[9] Kissling WD, Seijmonsbergen AC, Foppen RPB, Bouten W. eEcoLiDAR, eScience infrastructure for ecological applications of LiDAR point clouds: reconstructing the 3D ecosystem structure for animals at regional to continental scales. Res Ideas Outcomes 2017;3:e14939. http://dx.doi.org/10.3897/rio.3.e14939.

[10] Kissling WD, Hardisty A, García EA, Santamaria M, Leo FD, Pesole G, et al. Towards global interoperability for supporting biodiversity research on essential biodiversity variables (EBVs). Biodiversity 2015;16(2–3):99–107. http://dx.doi.org/10.1080/14888386.2015.1068709.

[11] Wilkinson M, Dumontier M, Aalbersberg I, Appleton G, Axton M, Baak A, et al. The FAIR guiding principles for scientific data management and stewardship. Sci Data 2016;3. http://dx.doi.org/10.1038/sdata.2016.18.

[12] Isenburg M, Liu Y, Shewchuk J, Snoeyink J. Streaming computation of delaunay triangulations. ACM Trans Graph 2006;25(3):1049–56. http://dx.doi.org/10.1145/1141911.1141992, http://doi.acm.org/10.1145/1141911.1141992.

[13] Isenburg M, Liu Y, Shewchuk J, Snoeyink J, Thirion T. Generating raster DEM from mass points via TIN streaming. In: Raubal M, Miller HJ, Frank AU, Goodchild MF, editors. Geographic information science. Berlin, Heidelberg: Springer Berlin Heidelberg; 2006, p. 186–98.

[14] Otepka J, Mandlburger G, Karel W. The opals data manager – efficient data management for processing large airborne laser scanning projects. In: ISPRS annals of photogrammetry, remote sensing and spatial information sciences. vol. I-3, Copernicus GmbH; 2012, p. 153–9. http://dx.doi.org/10.5194/isprsannals-I-3-153-2012, https://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/I-3/153/2012/.

[15] Pfeifer N, Mandlburger G, Otepka J, Karel W. OPALS – A framework for airborne laser scanning data analysis. Comput Environ Urban Syst 2014;45:125–36. http://dx.doi.org/10.1016/j.compenvurbsys.2013.11.002, URL http://www.sciencedirect.com/science/article/pii/S0198971513001051.

[16] PDAL Contributors. PDAL Point Data Abstraction Library. Nov. 2018, http://dx.doi.org/10.5281/zenodo.2556738.

[17] Yu X, Litkey P, Hyyppä J, Holopainen M, Vastaranta M. Assessment of low density full-waveform airborne laser scanning for individual tree detection and tree species classification. Forests 2014;5(5):1011–31. http://dx.doi.org/10.3390/f5051011, URL https://www.mdpi.com/1999-4907/5/5/1011.

[18] Höfle B, Mücke W, Dutter M, Rutzinger M, Dorninger P. Detection of building regions using airborne lidar : a new combination of raster and point cloud based GIS methods. In: Car A, Griesebner G, Strobl J, editors. Geospatial crossroads GI_Forum '09 : Proceedings of the geoinformatics forum salzburg, : Geoinformatics on stage, July 7–10, 2009. Germany: Wichmann Verlag; 2009, p. 66–75.

[19] Crosilla F, Macorig D, Scaioni M, Sebastianutti I, Visintini D. Lidar data filtering and classification by skewness and kurtosis iterative analysis of multiple point cloud data categories. Appl Geomatics 2013;5(3):225–40. http://dx.doi.org/10.1007/s12518-013-0113-9.

[20] Zlinszky A, Mücke W, Lehner H, Briese C, Pfeifer N. Categorizing wetland vegetation by airborne laser scanning on lake balaton and kis-balaton, Hungary. Remote Sens 2012;4(6):1617–50. http://dx.doi.org/10.3390/rs4061617, URL https://www.mdpi.com/2072-4292/4/6/1617.

[21] Næsset E. Predicting forest stand characteristics with airborne scanning laser using a practical two-stage procedure and field data. Remote Sens Environ 2002;80(1):88–99. http://dx.doi.org/10.1016/S0034-4257(01)00290-5, URL http://www.sciencedirect.com/science/article/pii/S0034425701002905.

[22] Weinmann M, Weinmann M, Mallet C, Brédif M. A classification-segmentation framework for the detection of individual trees in dense MMS point cloud data acquired in urban areas. Remote Sens 2017;9(3):277. http://dx.doi.org/10.3390/rs9030277.

[23] Dorninger P, Pfeifer N. A comprehensive automated 3D approach for building extraction, reconstruction, and regularization from airborne laser scanning point clouds. Sensors 2008;8(11):7323–43. http://dx.doi.org/10.3390/s8117323, URL https://www.mdpi.com/1424-8220/8/11/7323.

[24] Székely B, Koma Z, Karátson D, Dorninger P, Wörner G, Brandmeier M, et al. Automated recognition of quasi-planar ignimbrite sheets as paleosurfaces via robust segmentation of digital elevation models: an example from the central andes. Earth Surf Process Landf 2014;39(10):1386–99. http://dx.doi.org/10.1002/esp.3606, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/esp.3606.

[25] Bae S, Reineking B, Ewald M, Mueller J. Comparison of airborne lidar, aerial photography, and field surveys to model the habitat suitability of a cryptic forest species – the hazel grouse. Int J Remote Sens 2014;35(17):6469–89. http://dx.doi.org/10.1080/01431161.2014.955145.

[26] GDAL/OGR contributors. GDAL/OGR Geospatial Data Abstraction software Library. Open Source Geospatial Foundation; 2019, URL https://gdal.org.

[27] Rubi OM, Verhoeven S, Dzigan Y, van Oord G, Goncalves R. Nlesc/massive-potreeconverter: 1.1.0. 2017, http://dx.doi.org/10.5281/zenodo.910906.

[28] Lucas C, Bouten W, Koma Z, Kissling W, Seijmonsbergen A. Identification of linear vegetation elements in a rural landscape using LiDAR point clouds. Remote Sens 2019;11(3):1–16. http://dx.doi.org/10.3390/rs11030292.

[29] Koma Z, Seijmonsbergen AC, Meijer C, Bouten W, Kissling WD. Object-based habitat mapping of reedbeds using country-wide airborne laser scanning point clouds. In: GEOBIA 2018 - From pixels to ecosystems and global sustainability. Montpellier, France: Centre d'Etudes Spatiales de la BIOsphère (CESBIO) and Office national d'études et de recherches aérospatiales (ONERA) and Espace pour le développement (ESPACE DEV) and Société T.E.T.I.S; 2018, URL http://hal.univ-reunion.fr/hal-01960403.

[30] Pereira HM, Ferrier S, Walters M, Geller GN, Jongman RHG, Scholes RJ, et al. Essential biodiversity variables. Science 2013;339(6117):277–8. http://dx.doi.org/10.1126/science.1229931, URL https://science.sciencemag.org/content/339/6117/277.

[31] Valbuena R, O'Connor B, Zellweger F, Simonson W, Vihervaara P, Maltamo M, et al. Standardizing ecosystem morphological traits from 3D information sources. Trends Ecol Evol 2020. http://dx.doi.org/10.1016/j.tree.2020.03.006, URL http://www.sciencedirect.com/science/article/pii/S0169534720300811.

[32] Opitz RS, Ryzewski K, Cherry JF, Moloney B. Using airborne LiDAR survey to explore historic-era archaeological landscapes of montserrat in the eastern caribbean. J Field Archaeol 2015;40(5):523–41. http://dx.doi.org/10.1179/2042458215Y.0000000016.

[33] Opitz R, Nuninger L. Point clouds segmentation of mixed scenes with archeological standing remains: A multi-criteria and multi-scale iterative approach. Int J Herit Digit Era 2014;3(2):287–304. http://dx.doi.org/10.1260/2047-4970.3.2.287.

[34] Inomata T, Triadan D, Vázquez López VA, Fernandez-Diaz JC, Omori T, et al. Monumental architecture at Aguada Fénix and the rise of Maya civilization. Nature 2020. http://dx.doi.org/10.1038/s41586-020-2343-4.