PyTLidar: A Python Package for Tree QSM Modeling from Terrestrial Lidar Data

John Hagood[1¶], Fan Yang[1¶], Shruti Motiwale[3], Breanna Shi[1], Jeffery B. Cannon[2¶], and James T. Stroud[4¶]

1 College of Computing, Georgia Institute of Technology, United States
2 The Jones Center at Ichauway, United States
3 Pasteur Labs, Inc.
4 School of Biological Sciences, Georgia Institute of Technology, United States
¶ Corresponding author

Corresponding Author Emails: jhagood7@gatech.edu, fyang352@gatech.edu, Jeffery.Cannon@jonesctr.org, stroud@gatech.edu

Summary

PyTLidar is an open-source Python package that reconstructs 3D tree Quantitative Structure Models (QSM) from Terrestrial Lidar Scanning (TLS) data, providing a user-friendly tool that improves and expands upon the MATLAB-based TreeQSM method (Raumonen et al., 2013). QSMs are used to automate detection and calculation of various topological and volumetric measurements that would normally take great effort to gather in the field. PyTLidar provides an accessible, extensible, and GUI-driven workflow for researchers and practitioners in forestry, ecology, and 3D vegetation modeling to create QSMs. The package also integrates interactive visualization tools for inspecting model quality and viewing calculated tree properties. The ease of use and installation of PyTLidar provides ecologists an option to gather forest measurements that does not rely on proprietary tools.

The key features of PyTLidar are a reproduction of the TreeQSM core functionality, and enhancing the experience of setting up experiments and viewing results. It provides functionality for loading and extracting point cloud data from .las and .laz files as well as automatic calculation of a range of initial parameters for the QSM model based on point cloud structure. The QSM creation methods include generation of a Voronoi partition of the point cloud, segment detection, detection of parent-child relationships of branches, and cylinder fitting. PyTLidar also calculates various tree metrics such as branch length and volume and provides these results in text format as well as visual graphics. QSMs are also output in an interactive format which can be viewed directly in the GUI as well as saved and shared as an html file. All of this is packaged within a user-friendly GUI while also providing support for command line and direct Python interfacing.

Statement of Need

Terrestrial Laser Scanning (TLS) is an active remote sensing technology which uses infrared laser pulses to collect millions of three-dimensional coordinate points on the surface of objects, preserving spatial information and providing unprecedented detail on structural information. The technology is rapidly being adopted for diverse uses in forestry and ecology, as it is useful for estimating forest structure

(Donager et al., 2021), aboveground biomass (AGB) (Atkins et al., 2025), canopy gap fraction and forest fuels (Loudermilk et al., 2023), crown shape (Zhu et al., 2020), disturbance patterns (Cannon et al., 2024), tree competition (Metz et al., 2013), physiology (Hakala et al., 2015), and other ecological properties. To realize the potential of TLS for use in forestry and ecological applications, accurate and efficient reconstruction of QSMs from TLS point cloud data is essential (Hackenberg et al., 2015).

The use of QSM software on point cloud data permits estimation of detailed components of 42 branch architecture such as branch diameter, volume, and stem taper (Lau et al., 2018), providing detailed information for fine-scale estimates of AGB, canopy architecture, and more. TreeQSM is a software that has been widely used in forestry and ecology for modeling tree structures from TLS point clouds (Terryn et al., 2020). Comparing to other similar softwares, TreeQSM stands out for speed, reliability, and ease of use, while SimpleForest (Jan et al., 2021) (available within Computree) seems to be similarly capable to TreeQSM, but is only available through Computree, which has been undergoing an extended upgrade process and lacks up-to-date documentation. AdQSM (Fan G, 2020) is extremely fast and simple but lacks many of the statistics and visualizations other tools have and has not been officially released by the authors. aRchi (Martin-Ducup M., 2022) provides various functions but takes significantly longer to create a QSM than comparable packages. 3dForest (Bubnik V., 2025) has a promising GUI but was unstable during testing, crashing when loading data. There is also a lack of viable options within Python specifically. While TreeQSM is used in many applications, its reliance on MATLAB makes it less accessible for users, and its lack of graphical interface makes the tool less user-friendly and its parameter tuning less efficient. Thus we aimed to port and improve TreeQSM.

PyTLidar addresses these issues by providing a native Python implementation of TreeQSM's core algorithms, wrapped in a streamlined graphical user interface that allows researchers to visualize and evaluate models. It promotes reproducible and exploratory research by offering transparent parameter control, open-source licensing, and seamless integration into Python-based analysis workflows. This work lowers the barrier for adoption of QSM modeling by removing the MATLAB dependency, enhancing accessibility for the broader open-source geospatial and ecological modeling community. PyTLidar is currently being used for ongoing projects in ecological monitoring and evolutionary field research.

Method

TreeQSM models individual trees from terrestrial lidar scans by first creating a Voronoi partition of the point cloud. This assigns the points within the cloud to an initial cluster based on its proximity to an initial point. The size of these regions, referred to as cover sets, are determined by the input patch diameters. Since the cover sets form the building blocks for reconstructing the tree's global shape, the selection of the patch diameter can have a major impact on the quality of the resulting QSM. Larger patch diameters will connect points further away, counteracting occlusion, with a corresponding loss in detail. Conversely, smaller patch diameters will increase detail, but be more susceptible to occlusion. As part of the construction of the initial partition, the algorithm also determines the neighboring cover sets. Based on topological investigation of neighboring cover sets, the point cloud is segmented into individual branches, with parent-children relationships of branches identified. This process is repeated, re-creating the cover sets using a range of patch diameters based on location along the tree. Points determined to be along the trunk use size values closer to the max patch diameter, while points further up the tree use values closer to min patch diameter. After the second segment detection, each branch is approximated as a collection of connected cylinders of varying radius, length, and orientation. The cylinders are fit using standard root-finding methods to minimize distance between points and approximated cylinder surface (Raumonen et al., 2013). This cylinder-based representation offers a simple yet effective regularization of the complex tree structure, supporting downstream analyses such as stem volume estimation or structural trait extraction (Markku et al., 2015).

Software Description

PyTLidar is organized into several key modules: core QSM algorithms (treeqsm.py), batch processing utilities (treeqsm_batch.py), GUI components built with PyQt6 (Python bindings for the Qt 6 framework), and visualization tools using Plotly. The software follows a modular design that allows researchers to either use the complete GUI application or integrate individual components into their own Python workflows.

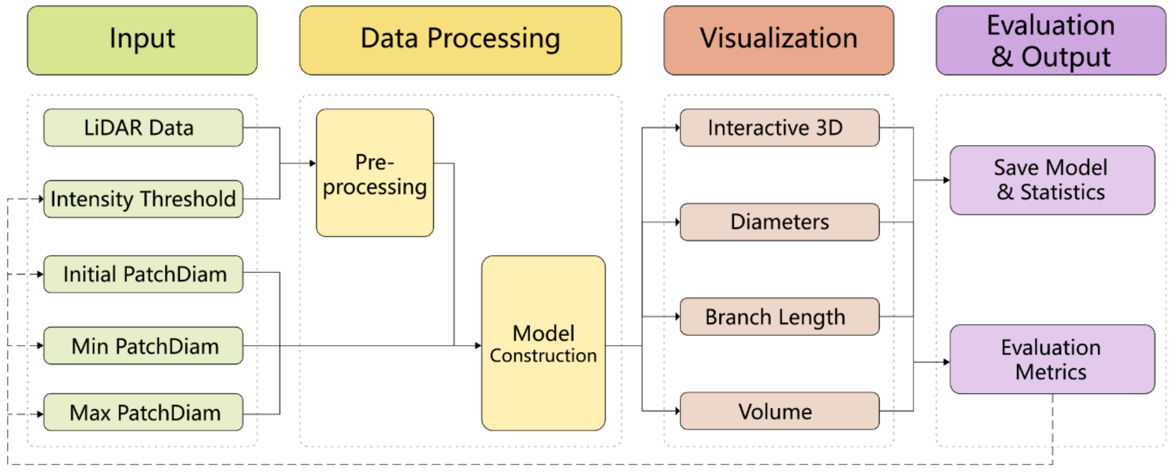
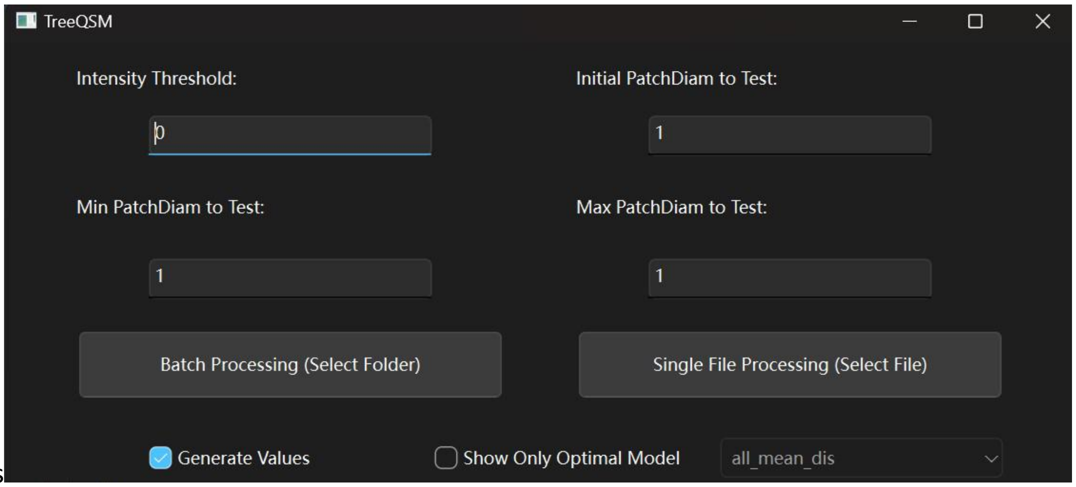


Figure 1: PyTLidar creates a QSM from an input point cloud and a set of parameters representing the size of the initial building blocks of the model. Structural measurements derived from the model and the model itself can then be viewed and evaluated within the tool.

When using the GUI, users can input or automatically generate values for key modeling parameters, including the minimum, and maximum patch diameters within a user-defined parameter (Figure 2). Also, an intensity threshold can be set to filter the point cloud data, helping to remove lidar returns due to noise or vegetation prior to modeling. Users may choose between batch processing of an entire directory of point cloud files or processing a single file. The GUI also includes options for displaying only the optimal model, based on performance metrics such as mean point distance to constructed surface.


TreeQSM

Intensity Threshold: 0

Initial PatchDiam to Test: 1

Min PatchDiam to Test: 1

Max PatchDiam to Test: 1

Batch Processing (Select Folder)
Single File Processing (Select File)

[✓ Generate Values]

[ ] Show Only Optimal Model

all_mean_dis

s

Figure 2: Graphical User Interface for parameter input and data selection for PyTLidar

After parameter and file selection, the software opens a new interface displaying data processing progress. Once the QSM reconstruction process is complete, PyTLidar provides interactive 3D visualization of the generated QSM using plotly (Figure 3). Users can inspect the structural fidelity of the reconstructed model, including trunk and branch geometry, and compare different parameter configurations for best fit. This combination of visual feedback and customizable processing offers an efficient path toward accurate and transparent tree structure analysis. If running in batch mode, users may also set the number of parallel cores to utilize to run simultaneous processes.

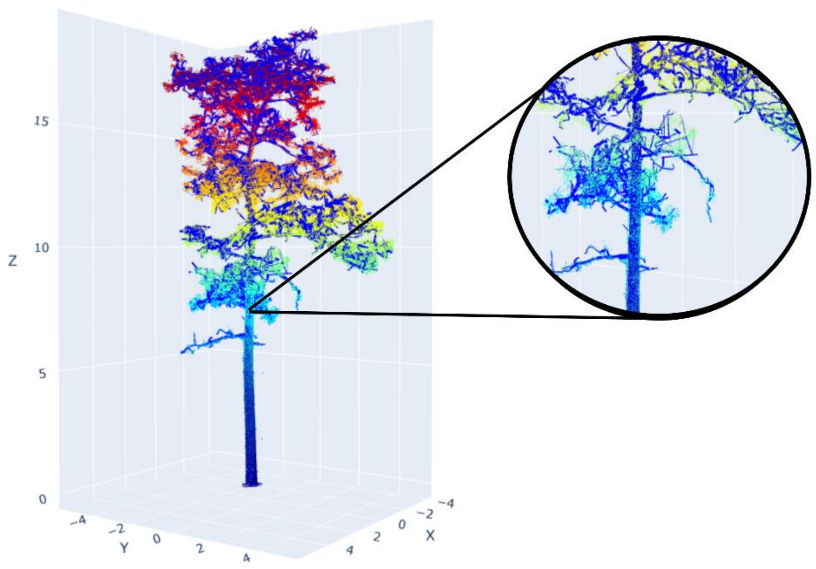


Figure 3: Point cloud and fitted cylinder of a sample pine. The graphical interface allows a user to compare the produced model (blue lines) directly to the point cloud (points colored by height) to visually evaluate fit.

Users can also review the morphological summaries of the QSM, including distribution of branch diameters, branch volume, surface area, and length with regard to diameter or order from stem, as with the original TreeQSM implementation (Figure 4). All of the produced figures are saved for later viewing and reference.

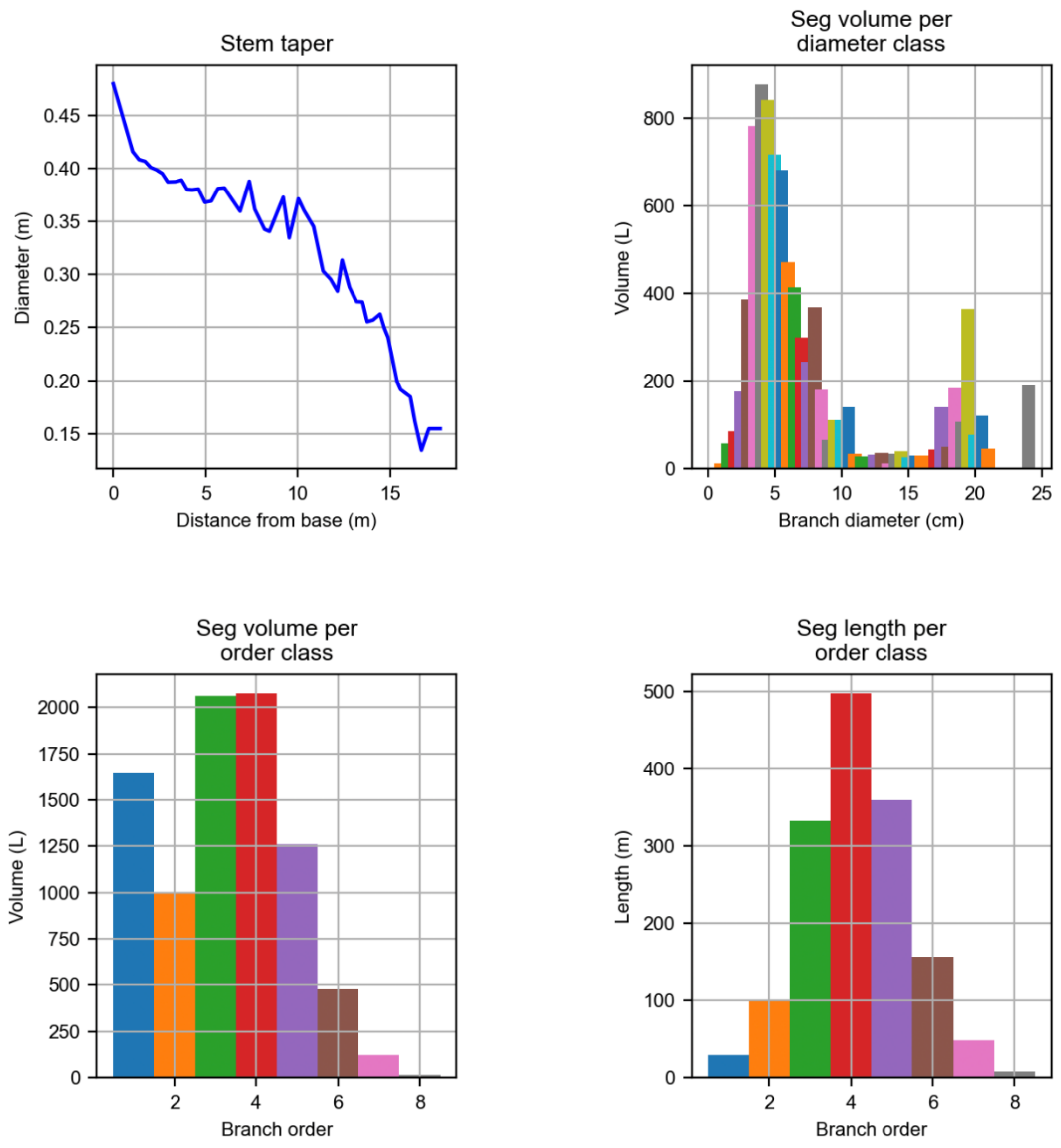


Figure 4: Example output data from sample pine. The output plots allow the user to derive summaries of various aspects of the input tree viewed within different categories including segment measurements by angle, direction and diameter class, as well as overall stem taper.

Both treeqsm.py and treeqsm_batch.py may be run directly from the command line. This allows users to integrate the same functionality provided in the GUI into their own scripts with ease, whether those scripts are in python or not. Python users can use the package directly and get the full functionality by importing treeqsm.

Availability and Installation

he latest development version of PyTLidar as well as usage instructions are available at this GitHub repository. The package requires Python 3.8+ and a few key dependencies listed in the requirements. Installation instructions and example datasets are provided in the repository documentation. The latest release version is available on PyPi and can be installed using “pip install PyTLidar”.

Future Additions

While the initial release is focused on porting only TreeQSM, several future additions to PyTLidar are planned. The first planned enhancement is to provide a novel pipeline for analyzing lidar scans of entire forest ecosystems to quantify vegetation structure at particular locations. This would allow users to load a series of lidar scan tiles and GPS observations of fauna and directly measure the environments, providing greater insights on components of habitat structural complexity.

Other planned enhancements include functions provided to users for processing lidar point clouds, including but not limited to various methods to perform ground filtering, tree segmentation and leaf/wood separation. The intended goal for this package is to provide a single source for any user processing terrestrial lidar to perform every step of their analysis.

Acknowledgements

We acknowledge contributions and guidance during the development of the package from Dori Peters, Amir Hossein Alikhah Mishamandani and other staff from the Human-Augmented Analytics Group to make this happen.

References

[1] P. Raumonen, M. Åkerblom, M. Kaasalainen, and Others, ‘TreeQSM: Quantitative Structure Models of Trees from Terrestrial Laser Scanning Point Clouds’, 2013. [Online]. Available: https://github.com/InverseTampere/TreeQSM. [Accessed: 08-June-2025].

[2] P. Raumonen *et al.*, ‘Fast Automatic Precision Tree Models from Terrestrial Laser Scanner Data’, *Remote Sensing*, vol. 5, no. 2, pp. 491–520, 2013.

[3] J. Hackenberg, H. Spiecker, K. Calders, M. Disney, and P. Raumonen, ‘SimpleTree —An Efficient Open Source Tool to Build Tree Models from TLS Clouds’, *Forests*, vol. 6, no. 11, pp. 4245–4294, 2015.

[4] J. Hackenberg, C. Morhart, J. Sheppard, H. Spiecker, and M. Disney, ‘Highly Accurate Tree Models Derived from Terrestrial Laser Scan Data: A Method Description’, *Forests*, vol. 5, no. 5, pp. 1069–1105, 2014.

[5] Å. Markku, P. Raumonen, M. Kaasalainen, and E. Casella, ‘Analysis of Geometric Primitives in Quantitative Structure Models of Tree Stems’, *Remote Sensing*, vol. 7, no. 4, pp. 4581–4603, 2015.

[6] L. Terryn *et al.*, ‘Tree species classification using structural features derived from terrestrial laser scanning’, *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 168, pp. 170–181, 2020.

[7] J. J. Donager, A. J. Sánchez Meador, and R. C. Blackburn, ‘Adjudicating Perspectives on Forest Structure: How Do Airborne, Terrestrial, and Mobile Lidar-Derived Estimates Compare?’, *Remote Sensing*, vol. 13, no. 12, 2021.

[8] D. H. Atkins, R. C. Blackburn, D. C. Laughlin, M. M. Moore, and A. J. Sánchez Meador, ‘Handheld lidar sensors can accurately measure aboveground biomass’, *Ecosphere*, vol. 16, no. 6, p. e70232, 2025.

[9] E. L. Loudermilk *et al.*, ‘Terrestrial Laser Scan Metrics Predict Surface Vegetation Biomass and Consumption in a Frequently Burned Southeastern U.S. Ecosystem’, *Fire*, vol. 6, no. 4, 2023.

[10] Z. Zhu, C. Kleinn, and N. Nölke, ‘Assessing tree crown volume—a review’, *Forestry: An International Journal of Forest Research*, vol. 94, no. 1, pp. 18–35, 10 2020.

[11] J. Cannon, N. E. Zampieri, A. W. Whelan, T. M. Shearman, and J. M. Varner, ‘Preprint: Terrestrial Lidar Scanning Reduces Subjectivity in Measurements of Tree Fire Injury’, *SSRN Electronic Journal - Preprint*, 2024.

[12] J. Metz, D. Seidel, P. Schall, D. Scheffer, E.-D. Schulze, and C. Ammer, ‘Crown modeling by terrestrial laser scanning as an approach to assess the effect of aboveground intra- and interspecific competition on tree growth’, *Forest Ecology and Management*, vol. 310, pp. 275–288, 2013.

[13] T. Hakala, O. Nevalainen, S. Kaasalainen, and R. Mäkipää, ‘Technical Note: Multispectral lidar time series of pine canopy chlorophyll content’, *Biogeosciences*, vol. 12, no. 5, pp. 1629–1634, 2015.

[14] A. Lau *et al.*, ‘Quantifying branch architecture of tropical trees using terrestrial LiDAR and 3D modelling’, *Trees*, vol. 32, no. 4, pp. 1219–1231, 2018.

[15] H. Jan, C. Kim, M. Demol, P. Raumonen, A. Piboule, and D. Mathias, ‘SimpleForest - a comprehensive tool for 3d reconstruction of trees from forest plot point clouds’, *bioRxiv*, 2021.

[16] ‘AdQSM: A New Method for Estimating Above-Ground Biomass from TLS Point Clouds’, *Remote Sensing*, vol. 12, no. 18, p. 3089, 2020.

[17] L. B. Martin-Ducup M., ‘aRchi: a R package for the analysis of tree architecture’, *GitHub repository*. Github, 2022.

[18] T. J. Bubnik V., ‘3D Forest’, *GitHub repository*. Github, 2025.