Tree Density Calculator

Release 1.5.7

Dec 07, 2020

Contents

The *Tree Density Calculator* is a QGIS plugin and command line interface package designed to calculate tree densities based on brightness images, using the local maximum of a sliding window.

The original Tree Density Calculator was written in C++ and ported to PyQGIS in 2018/2019. It has been developed in an open source environment to encourage further development of the tool.

The **instruction pages** can be found at <https://treedensitycalculator.readthedocs.io>.

The code **repository** can be found at https://bitbucket.org/kul-reseco/localmaxfilter.

PLEASE GIVE US CREDIT

When using the Tree Density Calculator, please use the following citation:

Crabbé, A. H., Cahy, T., Somers, B., Verbeke, L.P., Van Coillie, F. (2020). Tree Density Calculator Software (Version x.x) [Software]. Available from https://bitbucket.org/kul-reseco/localmaxfilter.

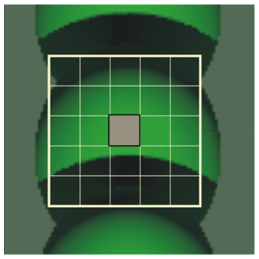
ACKNOWLEDGEMENTS

The software and user guide are based on the Local Maximum Filter [C++ software]: Ghent University, Laboratory of Forest Management and Spatial Information Techniques, Lieven P.C. Verbeke

The revision into a QGIS plugin is funded primarily through BELSPO (the Belgian Science Policy Office) in the framework of the STEREO III Programme – Project LUMOS - SR/01/321.

The LUMOS logo was created for free at https://logomakr.com.

For issues, bugs, proposals or remarks, visit the issue tracker.



CHAPTER 1

Context

Tree density is defined as the number of trees per hectare. Tree density and the evolution thereof can be used to evaluate forest management or as parameter in estimations of volume of wood per hectare. This forest inventory parameter usual is determined in the field by estimating the number of trees in selected sample circles. This can be done by counting the trees (intensive) or by using specialised equipment like the bitterlich.

Remote sensing offers an alternative for field measurements. Obviously, these techniques will work best on high resolution imagery, provided by sensors like IKONOS or Quickbird.

The assumpiton in the **Tree Density Calculator** is that a treetop is the brightest part of a tree in remote sensing imagery. The Tree Density Calculator will use a sliding window to move over the image. For every position of the window, the Tree Density Calculator checks whether the central pixel is the brightest of the window. If so, the pixel is marked as a *local maximum*.

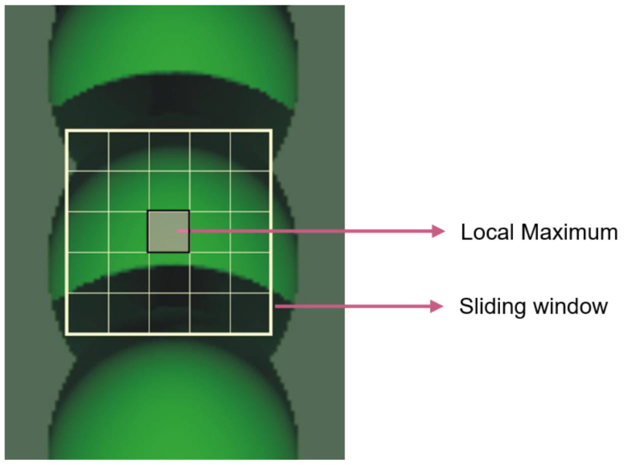


Fig. 1: The Tree Density Calculator recoginzes the treetop as the local maximum in brightmess within the window of 5x5 pixels.

We expect the best results for homogeneous coniferous forests. Conifers (especially spruce trees) have a very distinct conical treetop, of which the top is indeed the brightest part.

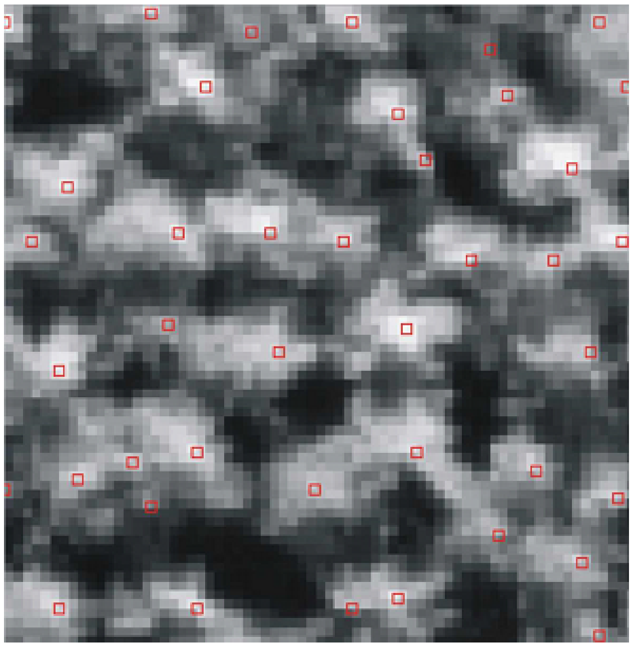


Fig. 2: An example of the result of applying the Tree Density Calculator on a 50 cm resolution image.

In it's most basic form, the *Tree Density Calculator* has only one parameter: the sliding window size. Depending on the resolution of the images and the size of the treetops, it is a process of trial and error to find the correct window size.

CHAPTER 2

Installation Instructions

For issues, bugs, proposals or remarks, visit the issue tracker.

2.1 Installation of QGIS Plugin

The plugin is available in the QGIS Python Plugins Repository:

```
Plugins menu > Manage and install plugins... > All > Search for 'Tree Density␣
↪Calculator'
```

Alternatively, you can download the latest stable distribution (*treedensitycalculator-x-qgis.zip*) and install the plugin manually:

```
Plugins menu > Manage and install plugins... > Install from ZIP > Browse to the zip␣
↪file > Click *Install plugin*.
```

Note: The plugin is build for QGIS Version 3.6 and up. We recommend to use QGIS version 3.14 or higher. The plugin has been tested on Windows 10.0, Ubuntu 16.04 and Raspbian GNU/Linux 10 (buster)

2.2 Installation of the python package

Open a shell by running the following batch script (adapt to match with your installation):

```
::QGIS installation folder
set OSGEO4W_ROOT=C:\OSGeo4W64

::set defaults, clean path, load OSGeo4W modules (incrementally)
call %OSGEO4W_ROOT%\bin\o4w_env.bat
```

(continues on next page)

(continued from previous page)

```
call qt5_env.bat
call py3_env.bat

::lines taken from python-qgis.bat
set QGIS_PREFIX_PATH=%OSGEO4W_ROOT%\apps\qgis
set PATH=%QGIS_PREFIX_PATH%\bin;%PATH%

::make PyQGIS packages available to Python
set PYTHONPATH=%QGIS_PREFIX_PATH%\python;%PYTHONPATH%

:: GDAL Configuration (https://trac.osgeo.org/gdal/wiki/ConfigOptions)
:: Set VSI cache to be used as buffer, see #6448 and
set GDAL_FILENAME_IS_UTF8=YES
set VSI_CACHE=TRUE
set VSI_CACHE_SIZE=1000000
set QT_PLUGIN_PATH=%QGIS_PREFIX_PATH%\qtplugins;%OSGEO4W_ROOT%\apps\qt5\plugins

::enable/disable QGIS debug messages
set QGIS_DEBUG=1

::open the OSGeo4W Shell
@echo on
@if [%1]==[] (echo run o-help for a list of available commands & cmd.exe /k) else␣
↪(cmd /c "%*")
```

For command-line interface and stand-alone usage, install the python package with pip:

```
pip install treedensitycalculator
```

For offline installation, you can download the latest stable distribution (*treedensitycalculator-x.tar.gz*) and:

```
C:\WINDOWS\system32>cd C:\Users\UserName\Downloads
C:\Users\UserName\Downloads>pip install treedensitycalculator-x.tar.gz
```

CHAPTER 3

User Guide

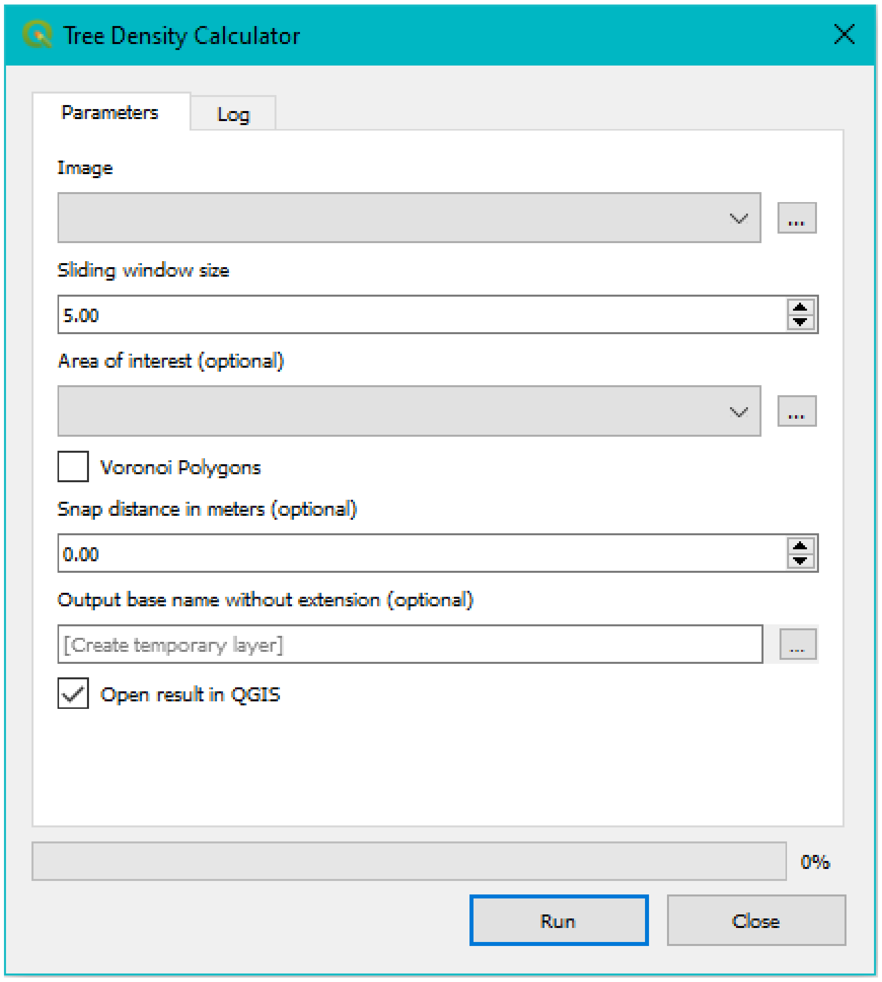
The Tree Density Calculator can be used in several ways:

1. As a plugin in QGIS
2. From the QGIS processing toolbox
3. As a commandline interface
4. Adapting the code to fulfil very specific needs

For the last option, we refer the user to the code repository and the API at the end of this document.

For issues, bugs, proposals or remarks, visit the issue tracker.

3.1 QGIS Plugin



Image

Sliding window size

5.00

Area of interest (optional)

☐ Voronoi Polygons

Snap distance in meters (optional)

0.00

Output base name without extension (optional)

[Create temporary layer]

☑ Open result in QGIS

0%

1. Select the remote sensing image of the forest.
2. Determine the sliding window size.
3. Optionally, choose a polygon vector layer representing the area(s) of interest. Trees will only be counted inside these polygons. The polygons **must have an ID field**.

   Note: For output, you will get an extra polygon layer with the following attributes: the polygon area in ha, the number of trees and the tree density (trees/ha).

4. Optionally, choose a snap distance. This distance should be maximum half of the window size.
5. Optionally, you can have an output layer with Voronoi polygons.
6. Optionally, choose a filename for the output layer(s).

3.2 Command Line Interface

The main command is:

```
>treedensity
```

Use `-h` or `--help` to list all possible arguments:

```
>treedensity -h
```

The **image** and **sliding window size** are required arguments. An example:

```
>treedensity data\input\image.tif 3
```

By default, the output shapefiles are stored in the same folder as the image file, with the extensions '_point.shp', '_mask.shp' and '_voronoi.shp'. To select another file name base (no extension) or another location, use the argument `-o` or `--output`:

```
>treedensity data\input\image.tif 3 -o data\other\new_name
```

To select a mask for counting the tree density only in a given area, use the argument `-m` or `--mask`:

```
>treedensity data\input\image.tif 3 -m data\input\mask.shp
```

To select a snap distance, use the argument `-s` or `--snap`. This distance should be maximum half of the window size:

```
>treedensity data\input\image.tif 7 -s 3
```

To create an output layer with voronoi polygons, use the argument `-v` or `--voronoi`:

```
>treedensity data\input\image.tif 3 -v
```

CHAPTER 4

Exercises

For issues, bugs, proposals or remarks, visit the issue tracker.

4.1 Tutorial Data Set

You can find the data (*data_qgis.zip*) on the bitbucket download page.

We will use five areas in *Heverlee Forest*, situated near *Leuven, BE*.

For each forest patch (with numbers 15, 38, 64, 84 and 103) you will find:

1. a false colour composite *polyg_x.tif* with 20 cm resolution
2. an image *polyg_x_100cm.tif* with the infrared band at 100 cm resolution
3. a vector file *polyg_x.shp* with the edges of the forest patch

Where *x* indicates the forest patch number.

4.2 Goal

We want to test if the **Tree Density Calculator** is a reliable estimator of tree densities.

It is important to have sufficient reference material (ground truth) to compare the results of the algorithm with. This reference material can be acquired for example in the field. In this exercise, we will use a remote sensing false colour composite image of 20 cm spatial resolution, to naively locate tree tops by hand.

In general, you will go through these steps for each forest patch:

1. Acquire a reference data set and count the number of trees in the data set.
2. Apply the Tree Density Calculator on a 100 cm and 200 cm resolution image, using two different sliding windows and again count the number of trees in the data set.
3. Determine the relation between the reference data set and the Tree Density Calculator data set.

If there is a strong relationship between the two, we can prove that the Tree Density Calculator can be used as a measure of determining tree densities.

4.3 Step 1. Reference data set

Manually digitize the tree tops in QGIS. For this we need to create a new, empty vector file.

1. Load the image patch in QGIS (drag and drop or *Add Raster Layer…*).

   Note: Coordinate reference system information should be *EPSG:31370 - Belge 1972 / Belgian Lambert 72*.

2. Load the forest patch boundaries (drag and drop or *Add Vector Layer…*). Again the CRS should be *EPSG:31370*.

   Change the symbology: make sure there is no fill color and the boundaries are clearly visible.

3. Click *Create Layer* > *New Shapefile Layer* in the *Layer* menu. Give the file a name (e.g. *polyg_x_reference*), make sure it is a point layer and has *EPSG:31370* as CRS. We won't be saving attribute data so you don't have to add any fields.

4. To start digitizing, select the new point layer in the table of contents and click *Toggle editing* in the *Layers* menu. Then click *Add Feature: Capture Point* in the *Edit* menu.

5. Now you can click on all the tree tops you see. QGIS will ask you for an *ID*. You can ignore this en simply click *OK* or press Enter. To delete a point, you can either undo an edit or use the *Select Object* tool and press Delete.

6. Don't forget to save your edits on regular basis! You find the save button in the *Layers* menu.

Counting the number of trees is easy now. Just open the attribute table and count the number of items. Alternatively, right click on the layer in the table of contents and select *Show number of objects*.

4.4 Step 2. Forest patch area

To get the area of the forest patch. Use the *Identify* tool in QGIS. Click on the shapefile with the boundaries. A window with identification results should open. One of the derived statistics is the area, based on the cartesian coordinates.

4.5 Step 3. Test the Tree Density Calculator

Before we can do this, we still need a 200 cm resolution image. We will create it by degrading each 100 cm resolution image with a factor of 2.

In QGIS, do the following:

1. Load the 100 cm resolution image.
2. Use the *Raster Alignment* tool in the *Raster* menu. Use *average* as resampling method, leave the CRS at it's *invalid* state (otherwise you get an error) and double the cell size (don't just add '1').

Now that you have both the 100 cm and 200 cm resolution images, you can use them in the *Tree Density Calculator*:

1. Open either the 100 cm or 200 cm resolution image.
2. Choose a window size. First try 5 pixels, next try 3 pixels. *What is that in meters?*
3. Open the corresponding vector layer representing the area of interest.

As a result, we will have a point vector layer representing the tree canopies detected with the given window size and a copy of the input polygon vector layer with the following attributes:

- “Area_ha” the polygon area in ha,
- “TreeCount” the number of trees and
- “TreeDens” the tree density as number of trees per ha.

4.6 Step 4. Determine the relation between the reference data set and the Tree Density Calculator data set

Now you have for each forest patch the tree density counted by hand, and the tree density counted by the QGIS plugin. Use a program like Excel to create a scatterplot per window size and add a trend line. Display the trend’s equation and R^2.

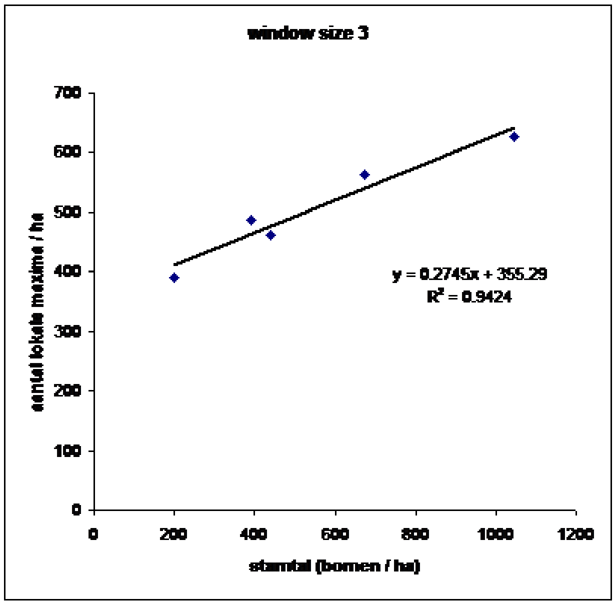


Fig. 1: An example of a scatterplot of the tree densities (reference numbers on the x-axis and measurements on the y-axis)

CHAPTER 5

Tree Density Calculator API

Source code: https://bitbucket.org/kul-reseco/localmaxfilter/src.

For issues, bugs, proposals or remarks, visit the issue tracker.

5.1 Core

class `localmaxfilter.core.local_max_filter.`**`LocalMaxFilter`**(*window_length*)

Bases: `object`

A window is used to slide over an image and look for maximum reflectance values in the center of that window.

`execute`(*image_array*, *area_of_interest=None*, *snap=None*, *geo_transform=None*, *set_progress: callable = None*)

The function will return an image containing the reflectance values of the local max of a sliding window going over the input image. It is possible to include an area of interest

Parameters

- **`image_array`** (*`ndarray`*) – 2D Numpy array image
- **`area_of_interest`** (*`ndarray`*) – 2D Numpy array image indicating with 0 and 1 the area of interest
- **`snap`** – defines the snap distance
- **`geo_transform`** – contains the pixel size of the raster
- **`set_progress`** – communicate progress (refer to the progress bar in case of GUI; otherwise print to console)

Returns dict: Dictionary containing the reflectance values of the local max together with pixel locations

`localmaxfilter.core.local_max_filter.`**`printProgress`**(*value: int*)

Replacement for the GUI progress bar

5.2 Interfaces

`localmaxfilter.interfaces.`**`run_algorithm_local_max_filter`**(*image_path*, *window*, *log_function=<built-in function print>*, *update_progress_bar=None*, *mask_path=None*, *output_base_path=None*, *snap_distance=None*, *voronoi=None*, *feedback=None*, *context=None*)

General interface between the GUI/CLI and script.

Parameters

- **`image_path`** – the absolute path to the raster file
- **`window`** – window size in meters
- **`update_progress_bar`** – function to update the progress bar
- **`log_function`** – function to log
- **`mask_path`** – absolute path to the vector file (optional)
- **`output_base_path`** – base path for output files (optional)
- **`snap_distance`** – snap distance for output points (optional)
- **`voronoi`** – set to True if you want voronoi polygons as output (optional)
- **`feedback`** – necessary for the processing tool
- **`context`** – necessary for the processing tool

Returns

`localmaxfilter.interfaces.imports.`**`check_path`**(*path*)

Check if path exists. Skipp path which are in memory

Parameters **`path`** – the absolute path to the input file

`localmaxfilter.interfaces.imports.`**`import_image`**(*image_path*, *mask_path=None*, *window_size=None*, *reflectance=False*, *feedback=None*, *context=None*)

Get an input image as array from file path. If a vector layer path is given, the raster is clipped, using a 2*window_size buffer. Also return the the spatial reference system and the geo_transform.

Parameters

- **`image_path`** – the absolute path to the input image
- **`mask_path`** – the absolute path to the polygon file
- **`window_size`** – the size of the sliding window, only required if mask is given
- **`reflectance`** – return reflectance values instead of DN between 0 and 255
- **`feedback`** – necessary for the processing tool
- **`context`** – necessary for the processing tool

Returns numpy array [#bands x #rows x #columns], ogr srs and geo_transform

`localmaxfilter.interfaces.imports.`**`import_vector_as_image`**(*path*, *geo_transform*, *image_size*, *image_srs*, *window_size*, *feedback=None*, *context=None*)

Browse for a vector file and return it as a raster after buffering with window_size.

Parameters

- **`path`** – the absolute path to the vector file
- **`geo_transform`** – 6 geo transformation coefficients
- **`image_size`** – size of the image
- **`image_srs`** – OSR spatial reference system object
- **`window_size`** – the size of the sliding window
- **`feedback`** – necessary for the processing tool
- **`context`** – necessary for the processing tool

Returns numpy array

`localmaxfilter.interfaces.exports.`**`check_path`**(*path*)

Check if path exists. Skipp path which are in memory

Parameters **`path`** – the absolute path to the input file

`localmaxfilter.interfaces.exports.`**`write_mask_layer`**(*output_path*, *mask_path*, *trees_path*, *log=<built-in function print>*)

Copy the input mask to a new shapefile, and add an extra attributes:

- field 'Area_ha' contains the area of the polygon
- field 'TreeCount' contains the number of trees found in that polygon
- field 'TreeDens' contains the number of trees per ha

Parameters

- **`output_path`** – the absolute path to the output Shapefile
- **`mask_path`** – the absolute path to the input vector file
- **`trees_path`** – the absolute path to the input point vector file
- **`log`** – log function

`localmaxfilter.interfaces.exports.`**`write_point_layer`**(*output_path*, *points_dict*, *geo_transform*, *srs*, *mask_path=None*, *feedback=None*, *context=None*)

Convert the input point dictionary to a vector point layer using the image metadata (srs and geo_transform).

The points are clipped to fit the given mask. https://gis.stackexchange.com/questions/268395/converting-raster-tif-to-point-shapefile-using-python

Parameters

- **`output_path`** – the absolute path to the output vector file
- **`points_dict`** – a dictionary containing the raster values and pixel locations Keys: RasterVal, Pixel with keys x and y
- **`mask_path`** – the absolute path to the mask file for clipping
- **`geo_transform`** – geo transformation coefficients
- **`srs`** – spatial reference system
- **`feedback`** – necessary for the processing tool
- **`context`** – necessary for the processing tool

`localmaxfilter.interfaces.exports.`**`write_voronoi_layer`**(*output_path*, *point_path*, *mask_path=None*, *feedback=None*, *context=None*)

Convert the input point vector to a Voronoi polygon layer and clip it by the mask layer

Parameters

- **`output_path`** – the absolute path to the output voronoi vector file
- **`point_path`** – the absolute path to the input point vector file
- **`mask_path`** – the absolute path to the overlay vector file
- **`feedback`** – necessary for the processing tool
- **`context`** – necessary for the processing tool

—————————————————————————————————————-

—————————————————————————————————————-

5.3 CLI

A window is used to slide over an image and look for maximum reflectance values in the center of that window.

```
usage: treedensity [-h] [-m ] [-v] [-s ] [-o ] image sliding_window
```

5.3.1 Positional Arguments

image	input reflectance image path

sliding_window	length of the sliding window, in meters (default 5 m), should be a multiple of the image pixel size

5.3.2 Named Arguments

-m, --mask	overlapping vector mask path
-v, --voronoi	create a Voronoi shapefile based on the tree locations (default off) Default: False
-s, --snap	give distance [m] for the snap tool, should be max half of the window size, tree tops which are closer to each other than the given distance will be taken as one
-o, --output	path for output files, give base without extension (default: in same folder with extension _window_{}.shp)

Python Module Index

Index