**TLS2trees**

**URL:** <https://github.com/philwilkes/TLS2trees>

**Date:** 30 November 2022

**Author:** Phil Wilkes

**Description:** This tool is developed as a forked version of FSCT (Forest Structure Complexity Tool) for the purpose of TLS point cloud semantic classification and individual tree segmentation. It also explains the methods to extract individual trees from TLS point clouds. This tool can be run in python as well as docker environment. Here, I have explained the usage of this tool in python environment.

**Installation:**

cd PATH\_TO\_TLS2trees-MAIN\_DIRECTORY

conda create --name YOUR\_ENVIRONMENT\_NAME\_HERE python==3.9

conda activate YOUR\_ENVIRONMENT\_NAME\_HERE

conda install pip

pip install -r requirements.txt

Note: when you will download TLS2trees.zip file. Extract it and you will find a folder named “fsct”. Locate this folder to the anaconda lib folder in site packages folder, like: **C:\Users\singha\.conda\envs\TLS2trees\Lib\site-packages.**

After the successful installation. This can be done in 3 steps. The first step is for RIEGL VZ TLS data pre-processing. There will be some subsidiary files generate in the step 1 which will be given as input to the step 2 while running FSCT such as tile-index.

1. Rxp-pipeline: This step is for the preprocessing of RIEGL VZ TLS data. The detailed processing step is explained in the repository. It also gives you opportunity to pre-process data acquire with RIEGL VZ TLS. The workflow of the code is mentioned below.

20XX-XX-08.XXX.riproject

 ├── ScanPos001

 ├── ScanPos002

 ├── ScanPosXXX

 ├── matrix

 | ├── ScanPos001.DAT

 | ├── ScanPos002.DAT

 | └── ScanPosXXX.DAT

 ├── extraction

 | ├── rxp2ply

 | | └── <tiles created by rxp2ply.py

 | ├── downsample

 | | └── <tiles created by downsample.py

 | ├── fsct

 | | └── output from FSCT

 | └── tile\_index.dat

 ├── clouds

 | └── <trees extracted with FSCT or other>

 └── models

 └── <QSMs from either TreeQSM or treegraph>

1. **FSCT *lite***

If the installation is done. You can run the following command using Anaconda command prompt. This will only runs the **semantic segmentation (ground, wood, leaf).**

**python run.py -p <point\_cloud> --tile-index <path\_to\_index> --buffer <buffer> --verbose**

 Forexample:

C:\TLS2trees\fsct>python run.py -p "C:\Users\singha\OneDrive - CZU v Praze\arunima\virtual\_mobility\data\1.plot2a-CS.las" --tile-index "" --buffer 0 –verbose

 NOTE:

1. If you have processed point cloud, then put 0 for buffer and it will avoid the parameter tile-index.
2. The current version of this tool is only compatible with. ply point cloud data format.

optional arguments:

 -h, --help show this help message and exit

 --point-cloud POINT\_CLOUD, -p POINT\_CLOUD

 path to point cloud

 --params PARAMS path to pickled parameter file

 --odir ODIR output directory

 --step STEP which process to run to

 --redo REDO which process to run to

 --tile-index TILE\_INDEX

 path to tile index in space delimited format "TILE X Y"

 --buffer BUFFER included data from neighbouring tiles

 --batch\_size BATCH\_SIZE

 If you get CUDA errors, try lowering this.

 --num\_procs NUM\_PROCS

 Number of CPU cores you want to use. If you run out of RAM, lower this.

 --keep-npy Keeps .npy files used for segmentation after inference is finished.

 --output\_fmt OUTPUT\_FMT

 file type of output

 --verbose print stuff

1. **Instance segmentation to extract individual trees:**

For instance segmentation to separate individual trees, you need to run **point2trees.py** file. Typical usage is below:

python points2trees.py -t 001.downsample.segmented.ply --tindex ../tile\_index.dat -o ../tmp/ --n-tiles 5 --slice-thickness .5 --find-stems-height 2 --find-stems-thickness .5 --pandarallel --verbose --add-leaves --add-leaves-voxel-length .5 --graph-maximum-cumulative-gap 3 --save-diameter-class --ignore-missing-tiles

the detail of all the parameters use is shown below:

optional arguments:

 -h, --help show this help message and exit

 --tile TILE, -t TILE fsct directory

 --odir ODIR, -o ODIR output directory

 --tindex TINDEX path to tile index

 --n-tiles N\_TILES enlarges the number of tiles i.e. 3x3 or tiles or 5 x 5 tiles

 --overlap OVERLAP buffer to crop adjacent tiles

 --slice-thickness SLICE\_THICKNESS

 slice thickness for constructing graph

 --find-stems-height FIND\_STEMS\_HEIGHT

 height for identifying stems

 --find-stems-thickness FIND\_STEMS\_THICKNESS

 thickness of slice used for identifying stems

 --find-stems-min-radius FIND\_STEMS\_MIN\_RADIUS

 minimum radius of found stems

 --find-stems-min-points FIND\_STEMS\_MIN\_POINTS

 minimum number of points for found stems

 --graph-edge-length GRAPH\_EDGE\_LENGTH

 maximum distance used to connect points in graph

 --graph-maximum-cumulative-gap GRAPH\_MAXIMUM\_CUMULATIVE\_GAP

 maximum cumulative distance between a base and a cluster

 --min-points-per-tree MIN\_POINTS\_PER\_TREE

 minimum number of points for a identified tree

 --add-leaves add leaf points

 --add-leaves-voxel-length ADD\_LEAVES\_VOXEL\_LENGTH

 voxel sixe when add leaves

 --add-leaves-edge-length ADD\_LEAVES\_EDGE\_LENGTH

 maximum distance used to connect points in leaf graph

 --save-diameter-class

 save into dimeter class directories

 --ignore-missing-tiles

 ignore missing neighbouring tiles

 --pandarallel use pandarallel

 --verbose print something