UAV OS

Visualization and Analysis of Point Clouds Using Potree/Entwine

UAV OS Handout 04

Know how to install Potree for Point Cloud Visualization and Analysis. Also this short handout will explains how you can index and serve large point cloud for serving which can then be access by PDAL for analysis.

PS Singh

NESAC 2019

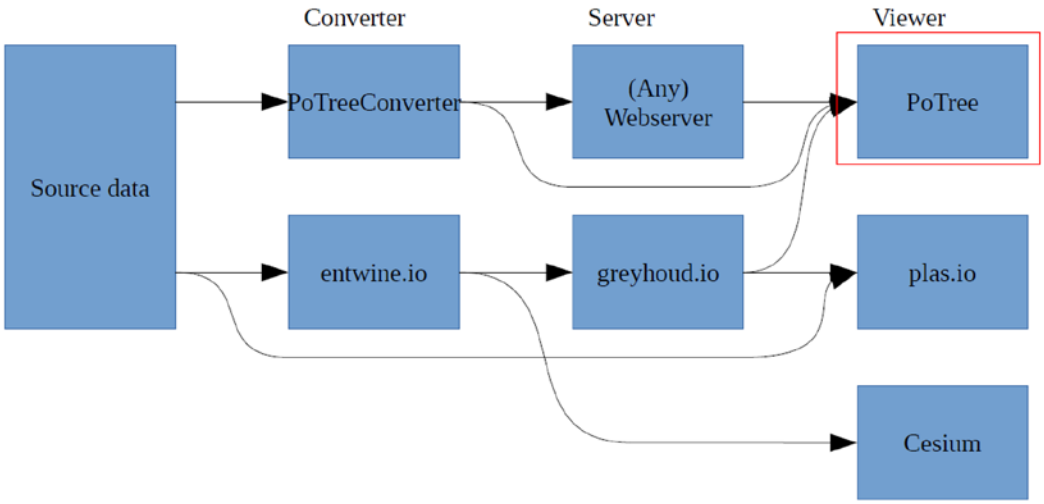
UAV OS

What you should be learning

- Install and use the powerful Potree for Point cloud visualization and analysis
- Build, Index and serve large scale point cloud as EPT datasets
- Use PDAL on entwine datasets for analysis

Learning Goal 1: Learn What Potree is and how to install one.

Potree is built based on WebGL and free and open source tool for rendering point cloud renderer in web based browsers which then can be visualize and analysed. In the figure given below, both Potree and plas.io are specifically meant for Point Cloud visualization, but PoTree has more functionalities for visualization and has tools for performing analysis in terms of measurements of area, height, length, angle, volume, profile etc and therefore is more useful and appropriate for point cloud analysis.



First clone repository from https://github.com/potree/potree.git

Build and Run

Make sure you have node.js pre-installed

Install all dependencies, as specified in package.json, then, install the gulp build tool:

```
cd <potree_directory>
npm install
npm install -g gulp
npm install -g rollup
```

Use the gulp watch command to

- create ./build/potree

- watch for changes to the source code and automatically create a new build on change
- start a web server at localhost:1234.

```
gulp watch
```

Go to http://localhost:1234/examples/ to test the examples.

Converter

The input data (las in this example) must be converted to a multi-resolution octree data structure where each node of the octree is stored in a file structure. This file is given as input to Potree viewer for rendering the point cloud. This step is only performed once (per input point cloud). This is performed by the Converter called as PotreeConverter. The PoTreeConverter can read LAZ/LAS, PTX, PLY and ASCII (XYZ) and then converts it to an Octree structure of an internal binary format, LAS or LAZ. These output files structure can be then used by the Potree viewer.

You can download the Converter from https://github.com/potree/PotreeConverter

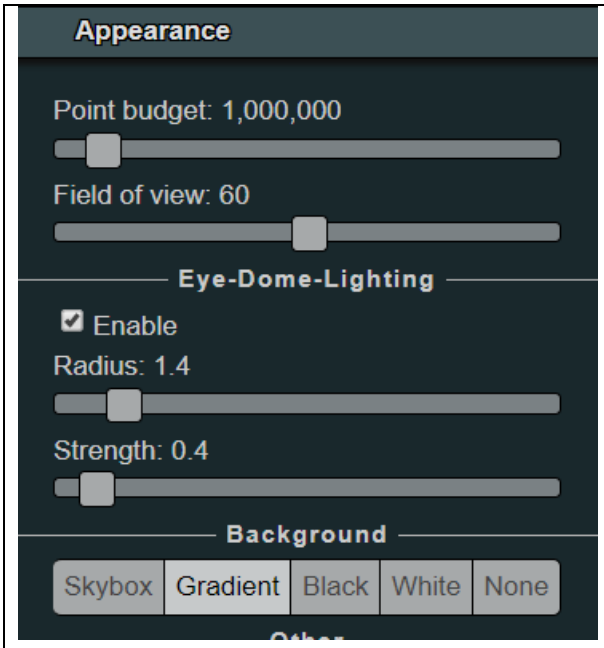
After downloaded, run the command () as below:

```
./PotreeConverter.exe C:/pointclouds/data.las -o
C:/pointclouds/data_converted
```

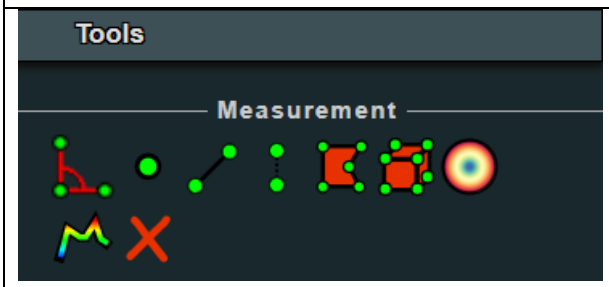
Copy the converted directory into <potreeDirectory>/pointclouds/data_converted. Then, duplicate and rename one of the examples and modify the path in the html file to your own point cloud.

See, how you can visualize your own point cloud and also you have couple of analysis tools you can try out on your point cloud.

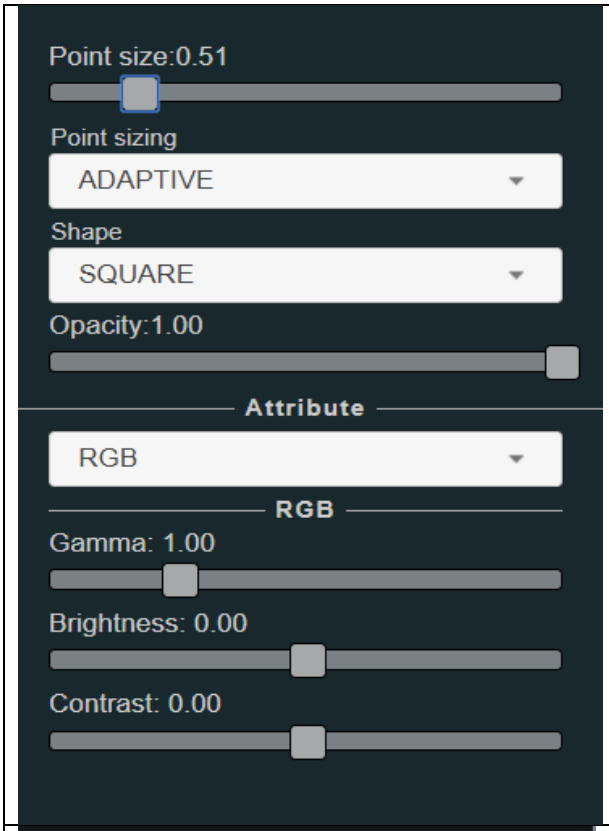
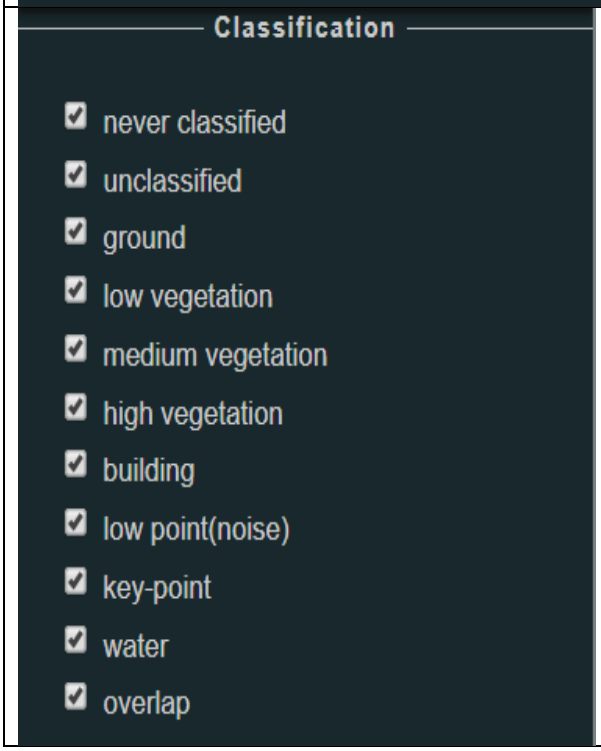
Below I describe different features of the Potree viewer



Point budget (default: 1 000 000)
How many points will be in memory (maximum), also known as *high-water mark*. Cells of the Octree will be loaded until this maximum is reached.
Field of view (default: 60)
Camera setting, determining the field of view of the virtual camera. Lower values will give a telescopic (binocular) effect. Extreme values may distort the image at the edges as the 3D world is projected on a flat screen.
Eye-Dome-Lighting
It is a non-photorealistic, image-based shading technique designed to improve depth perception in scientific visualization images
Background
Background colour behind the point cloud. Skybox (a cloud like pattern) and gradient will allow for the distinction between up and down.



Measurement tools available in PoTree are
Angles in a triangle | Point information (height, coordinates, RGB) | Distance between two or more points | Height difference (between two points) | Surface area | Volume, show a cube of specified dimensions in the point cloud | Profile | Volume, mark points in a specified volume in the point cloud.

	**Scene, point cloud settings** Settings can be set per point cloud. •Point size (default depends on point cloud, 0.5 – 1.0) Determines the size of each individual point in view. Taken to high (large) points will overlap, taken to low (small) the cohesion between points (eg. points on the same wall) is no longer apparent. •Point sizing (default: adaptive) Adaptive scaling makes points close to the viewer smaller compared to points further away. This prevents points in close proximity to the virtual camera from 'saturating' the view. •Shape (default: square) Circular points are more computational intensive. The paraboloid shape will 'melt' adjacent points to a 3D surface/volume. •Attribute (default depends on point cloud) Select the attribute to be shown, eg. RGB, elevation, level of detail, etc.. •RGBSome colouring settings for the RGB viewer.
	When ASPRS classification data is present in the point clouds and this information is retained during conversion filters can be applied here. When no classification was provided all data is marked as 'never classified'. In case of photogrammetry derived point clouds, initially it will be marked as 'never classified'.

Learning Goal 2: Learn how to install Entwine and use it for as Point Cloud Data Organisation Library and how one can run processing of point cloud in conjunction with PDAL.

Entwine is also a kind convertor similar to PotreeConverter but its more powerful in a sense that, it can index much larger point clouds with lacs of points. Further, it PDAL readers, filters and writers can be used for powerful processing of the entwine point tiles.

Installation

1. First, install Miniconda (or full Anaconda if you prefer) by downloading and running the install script for your platform. Then, run a shell and create an Entwine environment:
2. Install PDAL
3. Install Entwine : `conda create -n entwine -c conda-forge entwine`

This will create a new Conda environment, add the Conda Forge catalog to it, and install the Entwine package from the Conda Forge catalog.

Activate the entwine environment to use it:

4. `conda activate entwine`

Building the data

5. Host laz file using web-server. Assume that, laz file is hosted and available as `http://localhost:9090/laz/wkhsource2019.laz`
6. Build entwine - From a drive say, d:\. The following command builds an EPT dataset of it:

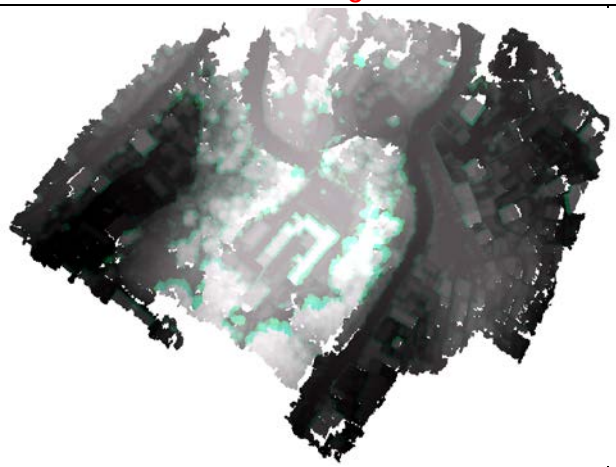
```
entwine build -i http://localhost:9090/laz/wkhsource2019.laz -o
~/entwine/wkh
```

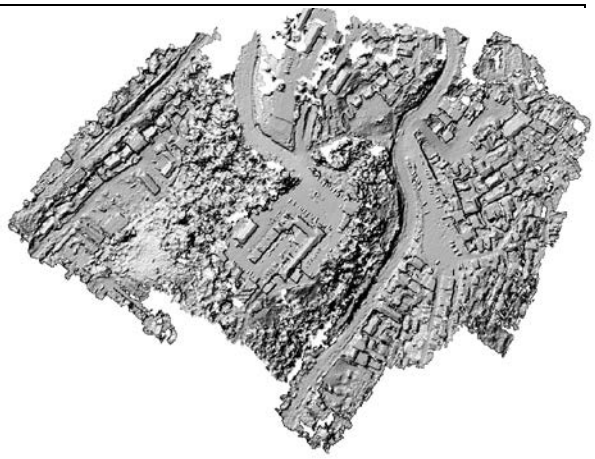
7. Install nodejs and http-server
   a. conda install nodejs –y
   b. npm install http-server –g
8. cd to entwine and run the command : http-server .
9. Open another Conda/Miniconda terminal

Processing with PDAL

We can also use the PDAL EPT reader to create an elevation model of the data. This can be done over HTTP or the local filesystem. Use PDAL to translate the service to a GeoTIFF using the GDAL writer driver:

10. Go to entwine drive, say, d:\entwine and run any PDAL readable command such as
    a. `pdal translate ept://wkh2 f:/wkh-dtm1.tif --writers.gdal.resolution=1.0`





create a hillshade using gdaldem:
`gdaldem hillshade f:/wkh-`

	`dtm.tif f:/wkh-hillshade.png`

Further details on Entwine config: https://entwine.io/configuration.html

Entwine and Entwine Point Tiles are a powerful tool for organising point data for visualisation and exploitation using the PDAL toolbox. While there are overheads in processing, this system represents a huge leap in how we can think about managing massive point cloud datasets as fundamental infrastructure.